**Lab Assignment 3**

**Neural Network & Deep Learning**

Implementation of back propagation

PART B

|  |  |
| --- | --- |
| Roll No. C009 | Name: Samarth Borade |
| Class : BTI SEM 10 | Batch : EB1 |
| Date of Experiment: 05/01/24 | Date of Submission |
| Grade : |  |

**B.1 Software Code written by student:**

#Samarth Borade

#C009

#BTI SEM 10

#EXP 3:Implementation of back propagation

import pandas as pd

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error, accuracy\_score

import matplotlib.pyplot as plt

df = pd.read\_csv('Iris.csv')

df\_one\_hot = pd.get\_dummies(df, columns=['Species'])

df\_one\_hot = df\_one\_hot.drop(columns=['Id'])

iris\_features = df\_one\_hot.drop(columns=['Species\_Iris-setosa', 'Species\_Iris-versicolor', 'Species\_Iris-virginica'])

iris\_features\_scaled = StandardScaler().fit\_transform(iris\_features)

X = iris\_features\_scaled

y = df\_one\_hot[['Species\_Iris-setosa', 'Species\_Iris-versicolor', 'Species\_Iris-virginica']]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

num\_classes = y.shape[1]

model = tf.keras.Sequential([

    tf.keras.layers.Dense(2, activation='sigmoid', input\_dim=4),

    tf.keras.layers.Dense(num\_classes, activation='sigmoid')

])

custom\_optimizer = tf.keras.optimizers.SGD(learning\_rate=0.5)

model.compile(optimizer=custom\_optimizer, loss='mse', metrics=['accuracy'])

history = model.fit(X\_train, y\_train, epochs=500, batch\_size=len(X\_train), verbose=0)

y\_pred = model.predict(X\_test)

y\_pred\_classes = pd.DataFrame(y\_pred, columns=y\_test.columns).idxmax(axis=1)

y\_true\_classes = y\_test.idxmax(axis=1)

accuracy = accuracy\_score(y\_true\_classes, y\_pred\_classes)

print(f"Accuracy: {accuracy}")

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Mean Squared Error (MSE): {mse}")

**Plot :**

plt.figure(figsize=(12, 4))

# Plot Loss

plt.subplot(1, 2, 1)

plt.plot(history.history['loss'])

plt.title('Training Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

# Plot Accuracy

plt.subplot(1, 2, 2)

plt.plot(history.history['accuracy'])

plt.title('Training Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.tight\_layout()

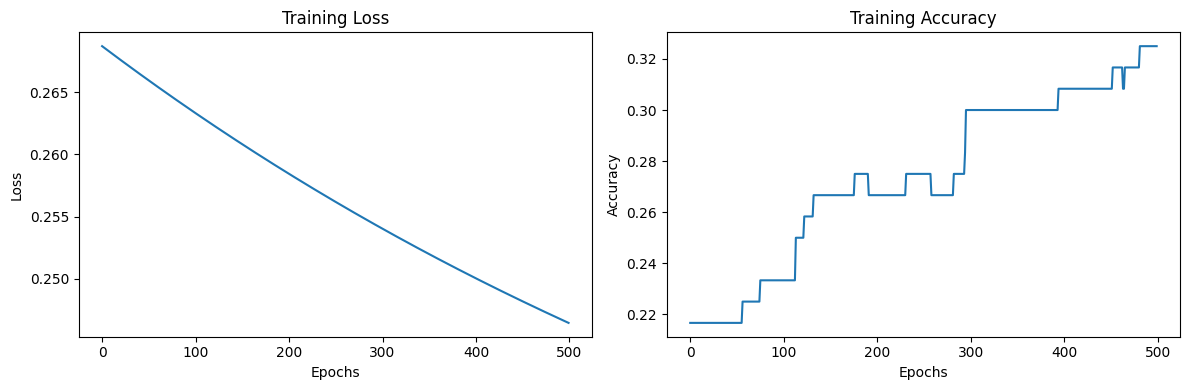
plt.show()

**B.2 Input and Output:**

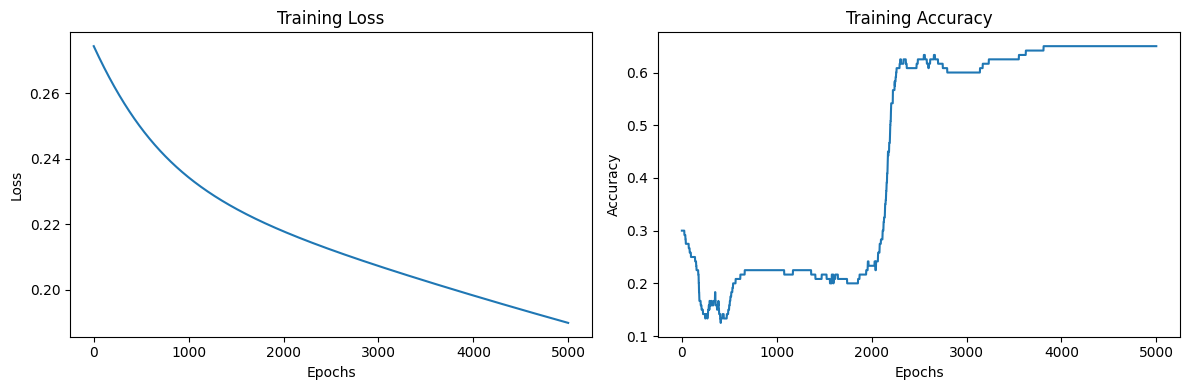
**Output:**

**LR: 0.01**

**Iterations: 500:**

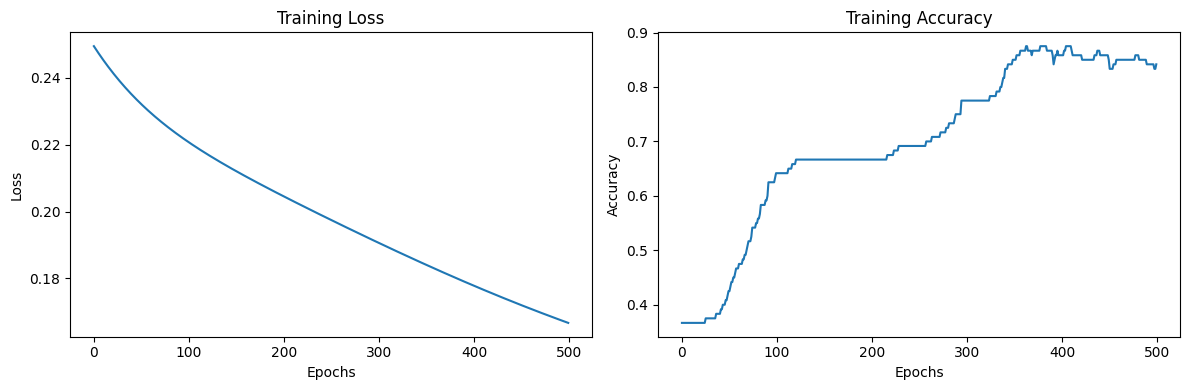
****

**Iterations: 5000:**

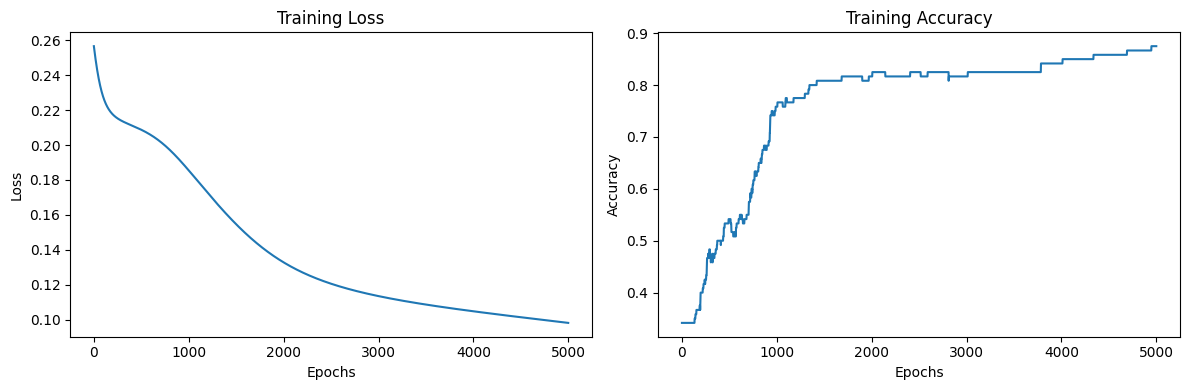
****

**LR: 0.1**

**Iterations: 500:**

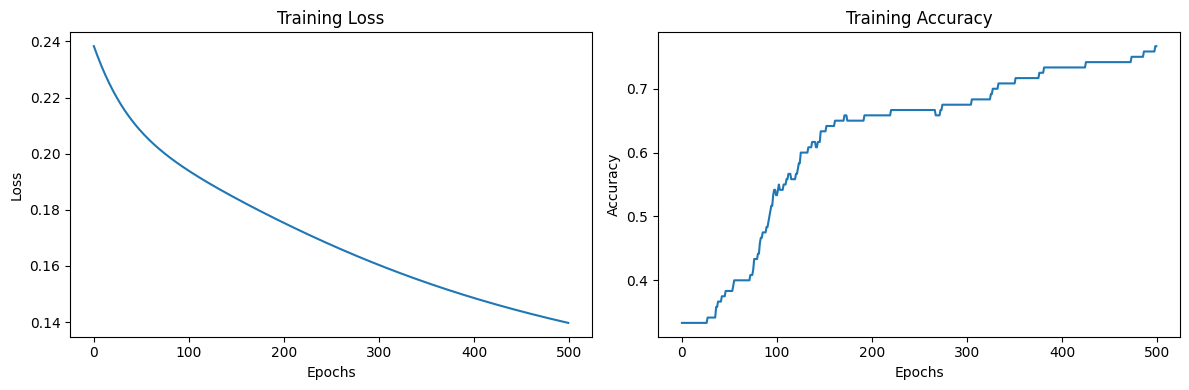
****

**Iterations: 5000:**

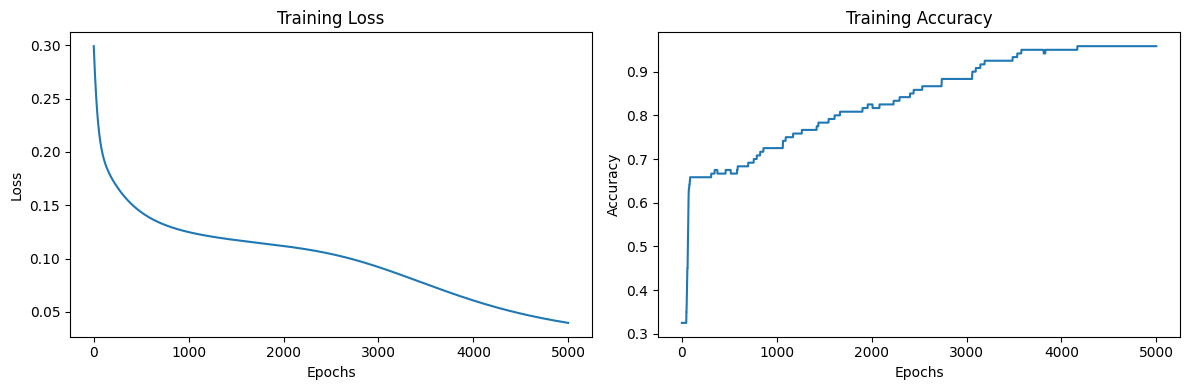
****

**LR: 0.2**

**Iterations: 500:**

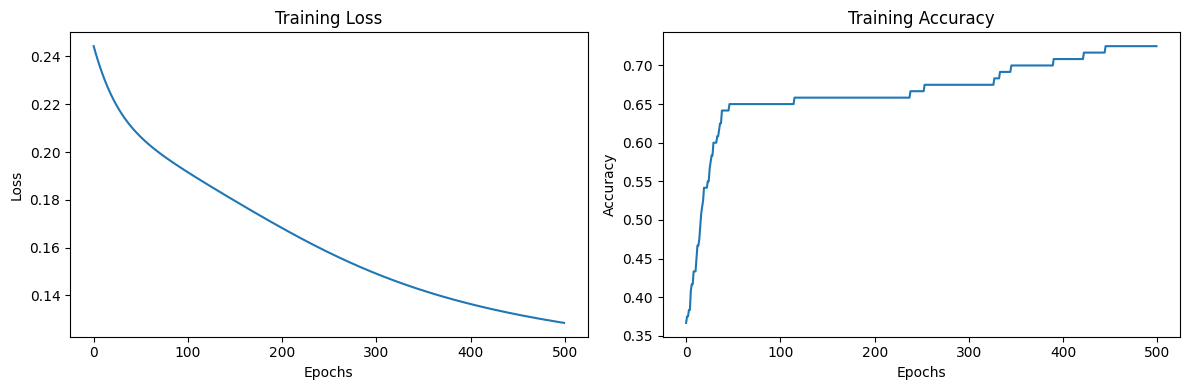
****

**Iterations: 5000:**

****

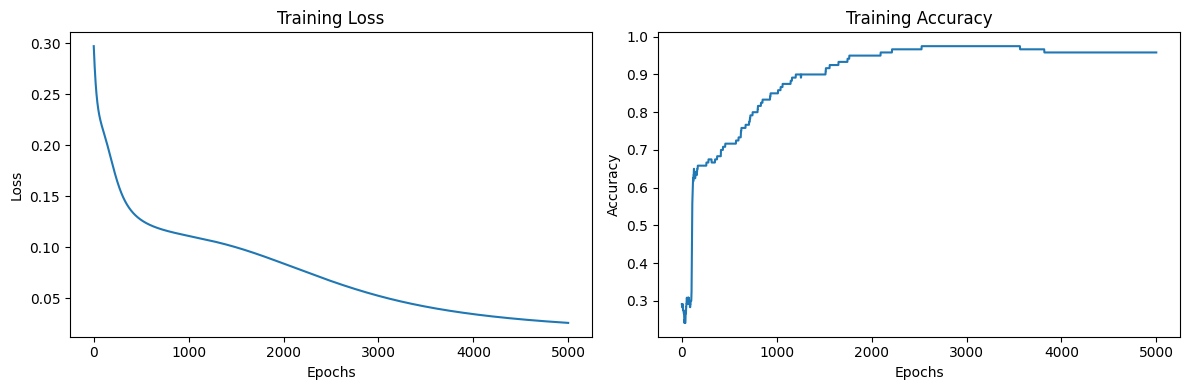
**LR: 0.3**

**Iterations: 500:**

****

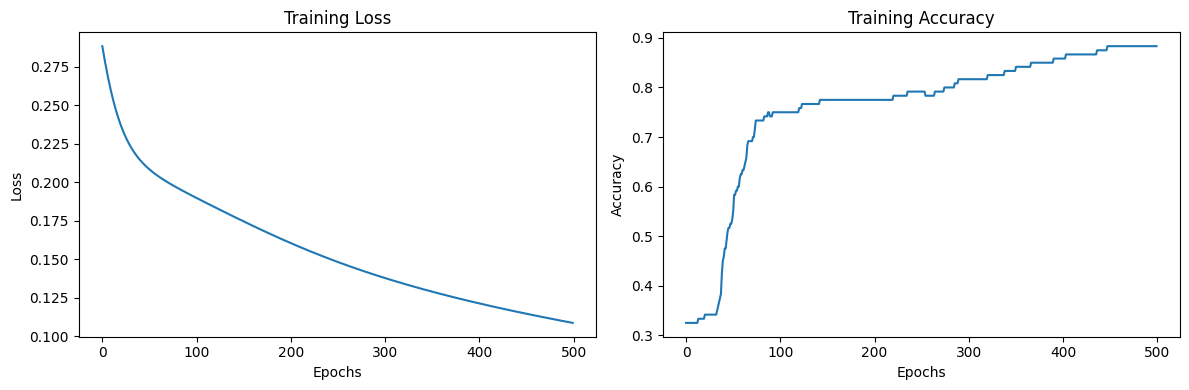
**LR: 0.3**

**Iterations: 5000:**

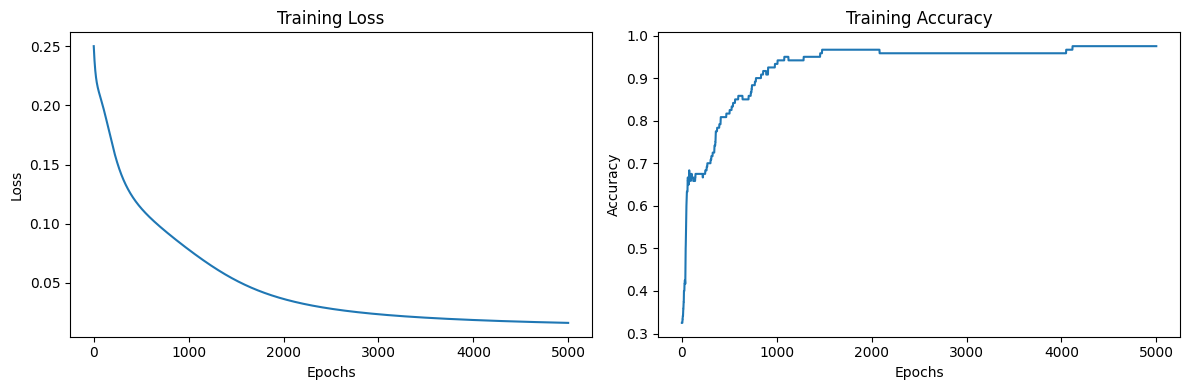
****

**LR: 0.4**

**Iterations: 500:**

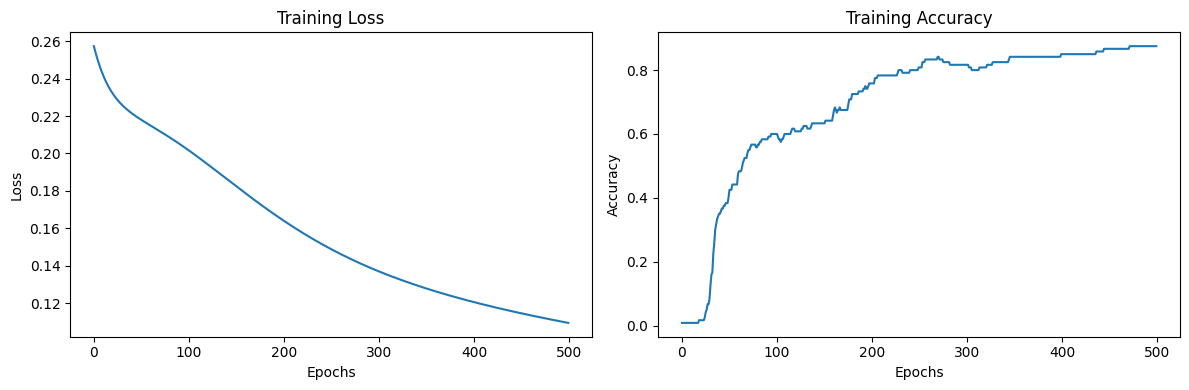
****

**Iterations: 5000:**

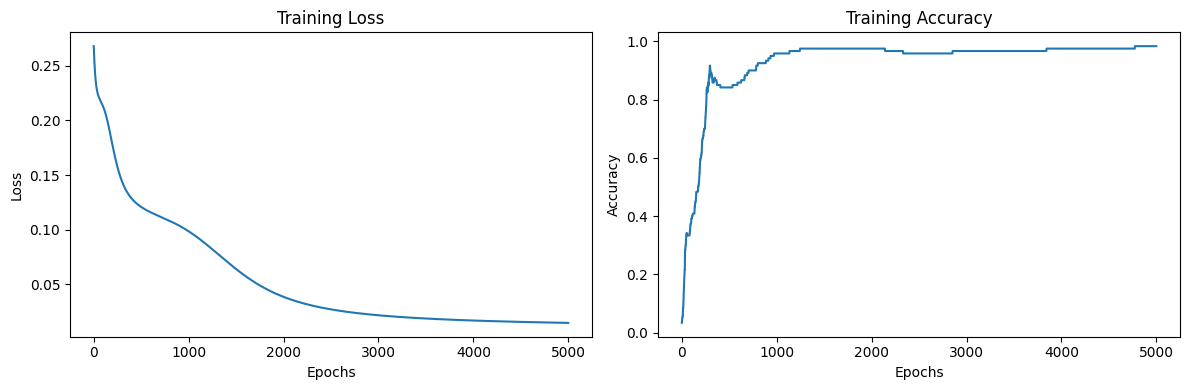
****

**LR: 0.5**

**Iterations: 500:**

****

**Iterations: 5000:**

****

**B.3 Observations and learning:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No. | Learning rate | Iterations | MSE | Accuracy |
| 1 | 0.01 | 500 | 0.260 | 0.233 |
| 2 | 5000 | 0.1100 | 0.9 |
| 3 | 0.1 | 500 | 0.195 | 0.7 |
| 4 | 5000 | 0.077 | 0.966 |
| 5 | 0.2 | 500 | 0.19 | 0.86 |
| 6 | 5000 | 0.036 | 1.0 |
| 7 | 0.3 | 500 | 0.135 | 0.8 |
| 8 | 5000 | 0.016 | 0.96 |
| 9 | 0.4 | 500 | 0.108 | 0.8 |
| 10 | 5000 | 0.0133 | 1.0 |
| 11 | 0.5 | 500 | 0.103 | 0.9 |
| 12 | 5000 | 0.010 | 1.0 |

**B.4 Conclusion:**

I carried out the implementation of back propagation using the Iris dataset and plotted graphs for multiple learning rate and iterations(Epochs).