exp4.py

```
'''4) Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the
same using appropriate data sets.
'''
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons
from sklearn.preprocessing import OneHotEncoder

# Activation function and its derivative
sigmoid = lambda x: 1 / (1 + np.exp(-x))
sigmoid_derivative = lambda x: x * (1 - x)

# ANN class
class NeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        self.W1 = np.random.randn(input_size, hidden_size)
        self.b1 = np.zeros((1, hidden_size))
        self.W2 = np.random.randn(hidden_size, output_size)
        self.b2 = np.zeros((1, output_size))

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

    def backward(self, X, y, output):
        d_output = (y - output) * sigmoid_derivative(output)
        d_hidden = d_output.dot(self.W2.T) * sigmoid_derivative(self.a1)
        self.W2 += self.a1.T.dot(d_output)
        self.b2 += np.sum(d_output, axis=0, keepdims=True)
        self.W1 += X.T.dot(d_hidden)
        self.b1 += np.sum(d_hidden, axis=0, keepdims=True)

    def train(self, X, y, epochs=10000):
        for epoch in range(epochs):
            output = self.forward(X)
            self.backward(X, y, output)
            if epoch % 1000 == 0:
                loss = np.mean(np.square(y - output))
                print(f'Epoch {epoch}, Loss: {loss:.4f}')

# Create and preprocess dataset
X, y = make_moons(n_samples=1000, noise=0.2)
y = OneHotEncoder().fit_transform(y.reshape(-1, 1)).toarray()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Initialize and train network
nn = NeuralNetwork(X_train.shape[1], 10, y_train.shape[1])
nn.train(X_train, y_train)

# Test network
output = nn.forward(X_test)
predictions = np.argmax(output, axis=1)
accuracy = np.mean(predictions == np.argmax(y_test, axis=1))
print(f'Accuracy: {accuracy * 100:.2f}%')
'''Output
Epoch 0, Loss: 0.2774
```

```
Epoch 1000, Loss: 0.5112
Epoch 2000, Loss: 0.5112
Epoch 3000, Loss: 0.5112
Epoch 4000, Loss: 0.5112
Epoch 5000, Loss: 0.5112
Epoch 6000, Loss: 0.5112
Epoch 7000, Loss: 0.5112
Epoch 8000, Loss: 0.5112
Epoch 9000, Loss: 0.5112
Accuracy: 54.50%
'''
```