Assignment1_proj

January 19, 2024

1 Deep Learning Course

1.1 Assignment 1

Assignment Goals:

- Start with PyTorch.
- Implement and apply logistic regression and multi-layer feed-forward neural network classifiers.
- Understand the differences and trade-offs between linear regression, logistic regression, and multi-layer feed-forward neural network.

In this assignment, you will be asked to install PyTorch and Jupyter Notebook. (TA's environment to run your code is Python 3.11 + Torch 2.1.2). In addition, you are required to design several models to classify a Toy Dataset (Figure 1).

Dataset: We provide a toy dataset, which has 200 instances and 2 features. See below "Toy Data and Helper Functions" section for toy data generation code.

You do not need to generate separated training dataset and test dataset for this assignment. Both training and prediction will both be on one dataset. Directly use the "sample, target" variables we provide as the dataset for your assignment.

In the following accuracy is defined as the empirical accuracy on the training set, that is, accuracy = {number of correctly predicted instances}/{number of all instances in dataset}).

Requirements

1. Install Pytorch() and Jupyter Notebook. (10 points)
2. Implement a logistic regression to classify the Toy Dataset. (20 points) We have provided a very simple linear regression example, please refer to the example and implement your logistic regression model.
    - You should determine: what loss function and what optimization algorithm do you plan to use? (4 points)
    - Try to reach $> 72\%$ accuracy. (4 points)
    - We have provided a visualize() helper function, you can visualize the model's decision boundary using that function. What's more, you are asked to compute and visualize **the equation of the decision boundary** of your trained **logistic regression**. Fill in the 'equation of decision boundary' column in the following table. Then you can modify the visualize() function or implement a new visualization function to draw the linear decision

boundary (Hint: should be a straight line aligned with the decision boundary plotted in visualize()). (5 points)

3. Implement a multi-layer linear neural network (>= 2 hidden layers) to classify the Toy Dataset. (20 points) A deep linear neural network is a deep feed-forward neural network without activation functions (See here, page 11-13 for detail introduction of linear neural networks).
   - You should determine: what loss function and what optimization algorithm do you plan to use, what is your network structure? (4 points)
   - Try to reach $> 72\%$ accuracy. (4 points)
   - Compute and visualize **the equation of the decision boundary** of your trained **linear neural network**. Fill in the 'equation of decision boundary' column in the following table. Then you can modify the visualize() function or implement a new visualization function to draw the linear decision boundary. (5 points)
4. Implement a multi-layer feed-forward neural network (>= 2 hidden layers). (20 points)
   - You should determine: what loss function and what optimization algorithm do you plan to use? what is your network structure? what activation function do you use? (5 points)
   - Try to reach 100% accuracy. (5 points)
5. Add L2-regularization to your implemented nonlinear neural network in (4.). Set the coefficient of L2-regularization to be 0.01, 2, 100, respectively. How do different values of coefficient of L2-regularization affect the model (i.e., model parameters, loss value, accuracy, decision boundary)? You can use a table to compare models trained without regularization, with different coefficients of regularization. (20 points)
   - Please draw your table and analysis in the '**Answers and Analysis**' section.

You should:

- Train each of your models to its best accuracy. Then fill in the following table in the '**Answers and Analysis**' section.
- Complete the '**Answers and Analysis**' section.

Answers and Analysis

- First, fill in the following table. The '-' indicates a cell that does not need to be filled in.

Model	Loss	Accuracy	Equation of Decision Boundary	NN Structure	Activation Function	Loss Function
Linear Regression	0.15	74%	$0.1817x_1 + 0.5237x_2 + 0.4758 = 0$	-	-	Mean Square Error
Logistic Regression				-	-	

Model	Loss	Accuracy	Equation of Decision Boundary	NN Structure	Activation Function	Loss Function
Linear Neural Network						
Feedforward Neural Network			-			

- Then, compare and analyze the classification results of your models. In particular, are there any differences between the performance (i.e., accuracy, loss value) of linear regression, logistic regression, linear neural network and deep nonlinear neural network? What do you think is the reason for the difference? (10 points)
- Your table and analysis of (5. Add L2-regularization) here.

Submission Notes: Please use Jupyter Notebook. The notebook should include the final code, results and your answers. You should submit your Notebook in both .pdf and .ipynb format.

Instructions: The university policy on academic dishonesty and plagiarism (cheating) will be taken very seriously in this course. Everything submitted should be your own writing or coding. You must not let other students copy your work. Spelling and grammar count.

Your assignments will be marked based on correctness, originality (the implementations and ideas are from yourself), clarity and performance. Clarity means whether the logic of your code is easy to follow. This includes 1) comments to explain the logic of your code 2) meaningful variable names. Performance includes loss value and accuracy after training.

1.2 Your Implementation

1.2.1 Toy Data and Helper Functions

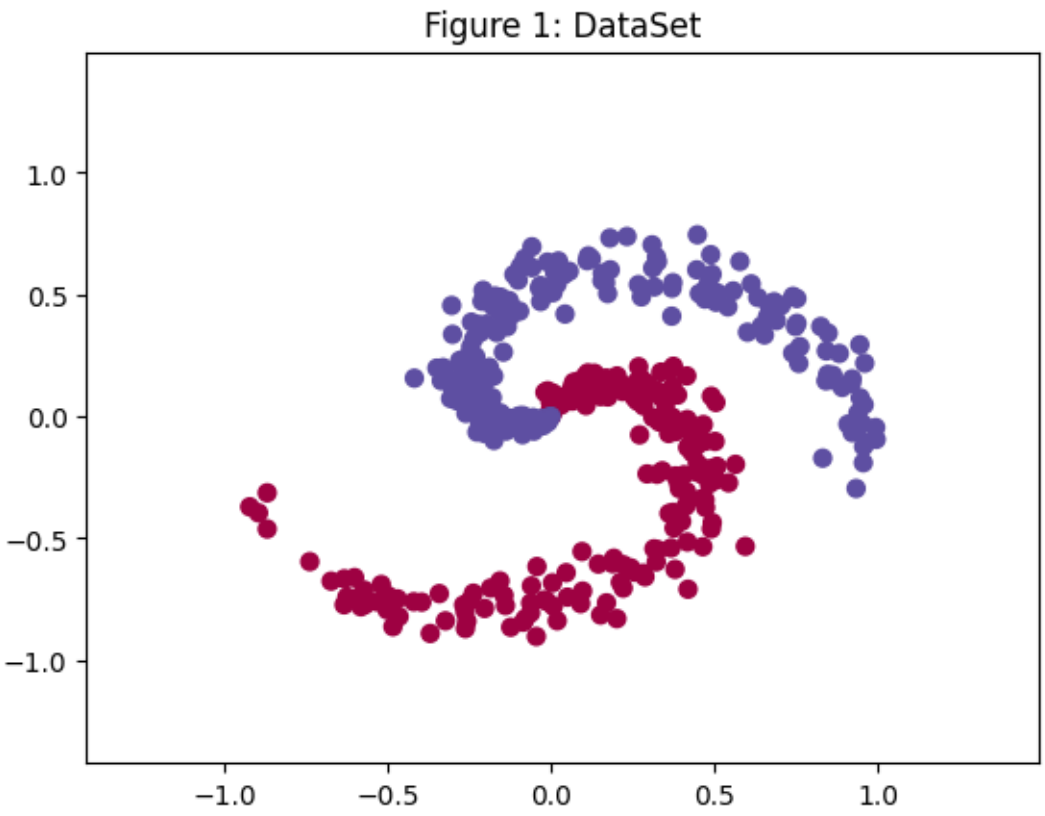
[1]:
```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
```

[2]:
```
# helper functions

# helper function for geterating the data
def data_generator(N = 200,D = 2,K = 2):
    """
    N: number of points per class;
```

```
    D: dimensionality;
    K: number of classes
    """

    np.random.seed(10)
    X = np.zeros((N*K,D))
    y = np.zeros((N*K), dtype='uint8')

    for j in range(K):
      ix = range(N*j,N*(j+1))
      r = np.linspace(0.0,1,N) # radius
      t = np.linspace(j*4,(j+1)*4,N) + np.random.randn(N)*0.3 # theta
      X[ix] = np.c_[r*np.sin(t), r*np.cos(t)]
      y[ix] = j

    fig = plt.figure()
    plt.title('Figure 1: DataSet')
    plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap=plt.cm.Spectral)

    plt.xlim(X.min()-.5, X.max()+.5)
    plt.ylim(X.min()-.5, X.max()+.5)

    return X,y


# helper function for visualizing the decision boundaries
def visualize(sample, target, model):
    """
    Function for visualizing the classifier boundaries on the TOY dataset.

    sample: Training data features (PyTorch tensor)
    target: Target (PyTorch tensor)
    model: The PyTorch model
    """
    h = 0.02  # Step size in the meshgrid
    x_min, x_max = sample[:, 0].min() - 1, sample[:, 0].max() + 1
    y_min, y_max = sample[:, 1].min() - 1, sample[:, 1].max() + 1

    # Create a meshgrid for visualization
    xx, yy = torch.meshgrid(torch.arange(x_min, x_max, h), torch.arange(y_min,␣
 ↪y_max, h))

    # Flatten and concatenate the meshgrid for prediction
    grid_tensor = torch.cat((xx.reshape(-1, 1), yy.reshape(-1, 1)), dim=1)

    # Predict the class labels for each point in the meshgrid
    with torch.no_grad():
```

```
        model.eval()  # Set the model to evaluation mode
        predictions = model(grid_tensor)

    #Binary Classification
    Z = torch.where(predictions>0.5,1.0,0.0)
    Z = Z.reshape(xx.shape)

    # Create a contour plot to visualize the decision boundaries
    fig = plt.figure()
    plt.contourf(xx, yy, Z, cmap=plt.cm.Spectral, alpha=0.8)

    # Scatter plot the training data points
    plt.scatter(sample[:, 0], sample[:, 1], c=target, s=40, cmap=plt.cm.Spectral)

    # Set plot limits
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())

    plt.show()
```

[3]:
```
#  TOY DataSet
sample, target = data_generator(N = 200)
# print(target.shape)
```



1.2.2 Given Example: Linear Regression

Note that linear regression is usually used for regression tasks, not classification tasks. However, it can be used for binary classification problems (be labeled 0, 1) with a threshold classifier. That is, when linear regression outputs > 0.5, the prediction is 1; otherwise, the prediction is 0.

[9]:
```
# Convert data to PyTorch tensors
X_tensor = torch.from_numpy(sample).float()
y_tensor = torch.from_numpy(target).float()

# Define the linear regression model
class LinearRegressionModel(nn.Module):
    def __init__(self, input_size):
        super(LinearRegressionModel, self).__init__()
        self.linear = nn.Linear(input_size,1)

    def forward(self, x):
        return self.linear(x)

# Instantiate the model, loss function, and optimizer
# input_size = 1  # Number of features in the input data
```

```
model = LinearRegressionModel(X_tensor.shape[1])
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Training loop
num_epochs = 500
for epoch in range(num_epochs):
    # Forward pass
    y_pred = model(X_tensor)
    y_pred = y_pred.reshape(y_tensor.shape)

    # Compute the loss
    loss = criterion(y_pred, y_tensor)

    #Calculate Accuracy

    output = torch.where(y_pred>0.5, 1.0,0.0)
    acc = accuracy_score(y_tensor, output)

    # Backward pass and optimization
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}, Accuracy:␣
 ↪{acc}')
```

```
Epoch [1/500], Loss: 0.3956, Accuracy: 0.285
Epoch [2/500], Loss: 0.3934, Accuracy: 0.285
Epoch [3/500], Loss: 0.3913, Accuracy: 0.285
Epoch [4/500], Loss: 0.3891, Accuracy: 0.2875
Epoch [5/500], Loss: 0.3871, Accuracy: 0.29
Epoch [6/500], Loss: 0.3850, Accuracy: 0.2925
Epoch [7/500], Loss: 0.3829, Accuracy: 0.295
Epoch [8/500], Loss: 0.3809, Accuracy: 0.2975
Epoch [9/500], Loss: 0.3789, Accuracy: 0.3025
Epoch [10/500], Loss: 0.3770, Accuracy: 0.3025
Epoch [11/500], Loss: 0.3750, Accuracy: 0.305
Epoch [12/500], Loss: 0.3731, Accuracy: 0.305
Epoch [13/500], Loss: 0.3712, Accuracy: 0.305
Epoch [14/500], Loss: 0.3693, Accuracy: 0.31
Epoch [15/500], Loss: 0.3674, Accuracy: 0.31
Epoch [16/500], Loss: 0.3656, Accuracy: 0.31
Epoch [17/500], Loss: 0.3638, Accuracy: 0.31
Epoch [18/500], Loss: 0.3620, Accuracy: 0.3125
Epoch [19/500], Loss: 0.3602, Accuracy: 0.315
Epoch [20/500], Loss: 0.3584, Accuracy: 0.32
Epoch [21/500], Loss: 0.3566, Accuracy: 0.325
```

```
Epoch [22/500], Loss: 0.3549, Accuracy: 0.325
Epoch [23/500], Loss: 0.3532, Accuracy: 0.33
Epoch [24/500], Loss: 0.3515, Accuracy: 0.3325
Epoch [25/500], Loss: 0.3498, Accuracy: 0.3325
Epoch [26/500], Loss: 0.3481, Accuracy: 0.3325
Epoch [27/500], Loss: 0.3465, Accuracy: 0.3325
Epoch [28/500], Loss: 0.3449, Accuracy: 0.3325
Epoch [29/500], Loss: 0.3432, Accuracy: 0.335
Epoch [30/500], Loss: 0.3416, Accuracy: 0.335
Epoch [31/500], Loss: 0.3400, Accuracy: 0.335
Epoch [32/500], Loss: 0.3385, Accuracy: 0.3375
Epoch [33/500], Loss: 0.3369, Accuracy: 0.3425
Epoch [34/500], Loss: 0.3354, Accuracy: 0.3425
Epoch [35/500], Loss: 0.3338, Accuracy: 0.3475
Epoch [36/500], Loss: 0.3323, Accuracy: 0.35
Epoch [37/500], Loss: 0.3308, Accuracy: 0.355
Epoch [38/500], Loss: 0.3293, Accuracy: 0.3625
Epoch [39/500], Loss: 0.3279, Accuracy: 0.3675
Epoch [40/500], Loss: 0.3264, Accuracy: 0.37
Epoch [41/500], Loss: 0.3250, Accuracy: 0.3775
Epoch [42/500], Loss: 0.3235, Accuracy: 0.38
Epoch [43/500], Loss: 0.3221, Accuracy: 0.38
Epoch [44/500], Loss: 0.3207, Accuracy: 0.385
Epoch [45/500], Loss: 0.3193, Accuracy: 0.39
Epoch [46/500], Loss: 0.3179, Accuracy: 0.395
Epoch [47/500], Loss: 0.3166, Accuracy: 0.3975
Epoch [48/500], Loss: 0.3152, Accuracy: 0.3975
Epoch [49/500], Loss: 0.3139, Accuracy: 0.3975
Epoch [50/500], Loss: 0.3125, Accuracy: 0.3975
Epoch [51/500], Loss: 0.3112, Accuracy: 0.3975
Epoch [52/500], Loss: 0.3099, Accuracy: 0.4
Epoch [53/500], Loss: 0.3086, Accuracy: 0.4075
Epoch [54/500], Loss: 0.3073, Accuracy: 0.4125
Epoch [55/500], Loss: 0.3060, Accuracy: 0.4125
Epoch [56/500], Loss: 0.3047, Accuracy: 0.42
Epoch [57/500], Loss: 0.3035, Accuracy: 0.425
Epoch [58/500], Loss: 0.3022, Accuracy: 0.43
Epoch [59/500], Loss: 0.3010, Accuracy: 0.435
Epoch [60/500], Loss: 0.2998, Accuracy: 0.4375
Epoch [61/500], Loss: 0.2986, Accuracy: 0.44
Epoch [62/500], Loss: 0.2974, Accuracy: 0.445
Epoch [63/500], Loss: 0.2962, Accuracy: 0.45
Epoch [64/500], Loss: 0.2950, Accuracy: 0.4525
Epoch [65/500], Loss: 0.2938, Accuracy: 0.4575
Epoch [66/500], Loss: 0.2926, Accuracy: 0.46
Epoch [67/500], Loss: 0.2915, Accuracy: 0.4625
Epoch [68/500], Loss: 0.2903, Accuracy: 0.465
Epoch [69/500], Loss: 0.2892, Accuracy: 0.4725
```

```
Epoch [70/500], Loss: 0.2881, Accuracy: 0.4725
Epoch [71/500], Loss: 0.2870, Accuracy: 0.4725
Epoch [72/500], Loss: 0.2858, Accuracy: 0.4725
Epoch [73/500], Loss: 0.2847, Accuracy: 0.4775
Epoch [74/500], Loss: 0.2837, Accuracy: 0.485
Epoch [75/500], Loss: 0.2826, Accuracy: 0.4875
Epoch [76/500], Loss: 0.2815, Accuracy: 0.49
Epoch [77/500], Loss: 0.2804, Accuracy: 0.495
Epoch [78/500], Loss: 0.2794, Accuracy: 0.495
Epoch [79/500], Loss: 0.2783, Accuracy: 0.495
Epoch [80/500], Loss: 0.2773, Accuracy: 0.495
Epoch [81/500], Loss: 0.2763, Accuracy: 0.5
Epoch [82/500], Loss: 0.2752, Accuracy: 0.5025
Epoch [83/500], Loss: 0.2742, Accuracy: 0.5025
Epoch [84/500], Loss: 0.2732, Accuracy: 0.5025
Epoch [85/500], Loss: 0.2722, Accuracy: 0.505
Epoch [86/500], Loss: 0.2712, Accuracy: 0.51
Epoch [87/500], Loss: 0.2702, Accuracy: 0.515
Epoch [88/500], Loss: 0.2693, Accuracy: 0.515
Epoch [89/500], Loss: 0.2683, Accuracy: 0.5175
Epoch [90/500], Loss: 0.2673, Accuracy: 0.52
Epoch [91/500], Loss: 0.2664, Accuracy: 0.53
Epoch [92/500], Loss: 0.2654, Accuracy: 0.5275
Epoch [93/500], Loss: 0.2645, Accuracy: 0.53
Epoch [94/500], Loss: 0.2636, Accuracy: 0.535
Epoch [95/500], Loss: 0.2627, Accuracy: 0.5375
Epoch [96/500], Loss: 0.2617, Accuracy: 0.5375
Epoch [97/500], Loss: 0.2608, Accuracy: 0.5425
Epoch [98/500], Loss: 0.2599, Accuracy: 0.545
Epoch [99/500], Loss: 0.2590, Accuracy: 0.55
Epoch [100/500], Loss: 0.2582, Accuracy: 0.5525
Epoch [101/500], Loss: 0.2573, Accuracy: 0.5525
Epoch [102/500], Loss: 0.2564, Accuracy: 0.555
Epoch [103/500], Loss: 0.2555, Accuracy: 0.5575
Epoch [104/500], Loss: 0.2547, Accuracy: 0.5575
Epoch [105/500], Loss: 0.2538, Accuracy: 0.5625
Epoch [106/500], Loss: 0.2530, Accuracy: 0.5625
Epoch [107/500], Loss: 0.2521, Accuracy: 0.5675
Epoch [108/500], Loss: 0.2513, Accuracy: 0.5725
Epoch [109/500], Loss: 0.2505, Accuracy: 0.5775
Epoch [110/500], Loss: 0.2497, Accuracy: 0.58
Epoch [111/500], Loss: 0.2488, Accuracy: 0.5875
Epoch [112/500], Loss: 0.2480, Accuracy: 0.5875
Epoch [113/500], Loss: 0.2472, Accuracy: 0.5875
Epoch [114/500], Loss: 0.2464, Accuracy: 0.5925
Epoch [115/500], Loss: 0.2457, Accuracy: 0.595
Epoch [116/500], Loss: 0.2449, Accuracy: 0.6
Epoch [117/500], Loss: 0.2441, Accuracy: 0.6025
```

```
Epoch [118/500], Loss: 0.2433, Accuracy: 0.6075
Epoch [119/500], Loss: 0.2426, Accuracy: 0.615
Epoch [120/500], Loss: 0.2418, Accuracy: 0.6175
Epoch [121/500], Loss: 0.2411, Accuracy: 0.6225
Epoch [122/500], Loss: 0.2403, Accuracy: 0.6225
Epoch [123/500], Loss: 0.2396, Accuracy: 0.625
Epoch [124/500], Loss: 0.2388, Accuracy: 0.63
Epoch [125/500], Loss: 0.2381, Accuracy: 0.6325
Epoch [126/500], Loss: 0.2374, Accuracy: 0.635
Epoch [127/500], Loss: 0.2367, Accuracy: 0.635
Epoch [128/500], Loss: 0.2359, Accuracy: 0.635
Epoch [129/500], Loss: 0.2352, Accuracy: 0.6375
Epoch [130/500], Loss: 0.2345, Accuracy: 0.645
Epoch [131/500], Loss: 0.2338, Accuracy: 0.655
Epoch [132/500], Loss: 0.2331, Accuracy: 0.655
Epoch [133/500], Loss: 0.2325, Accuracy: 0.655
Epoch [134/500], Loss: 0.2318, Accuracy: 0.66
Epoch [135/500], Loss: 0.2311, Accuracy: 0.665
Epoch [136/500], Loss: 0.2304, Accuracy: 0.6675
Epoch [137/500], Loss: 0.2298, Accuracy: 0.6725
Epoch [138/500], Loss: 0.2291, Accuracy: 0.67
Epoch [139/500], Loss: 0.2285, Accuracy: 0.6725
Epoch [140/500], Loss: 0.2278, Accuracy: 0.6725
Epoch [141/500], Loss: 0.2272, Accuracy: 0.6725
Epoch [142/500], Loss: 0.2265, Accuracy: 0.6775
Epoch [143/500], Loss: 0.2259, Accuracy: 0.6775
Epoch [144/500], Loss: 0.2253, Accuracy: 0.68
Epoch [145/500], Loss: 0.2246, Accuracy: 0.685
Epoch [146/500], Loss: 0.2240, Accuracy: 0.6925
Epoch [147/500], Loss: 0.2234, Accuracy: 0.6925
Epoch [148/500], Loss: 0.2228, Accuracy: 0.6875
Epoch [149/500], Loss: 0.2222, Accuracy: 0.6875
Epoch [150/500], Loss: 0.2216, Accuracy: 0.6925
Epoch [151/500], Loss: 0.2210, Accuracy: 0.6925
Epoch [152/500], Loss: 0.2204, Accuracy: 0.6925
Epoch [153/500], Loss: 0.2198, Accuracy: 0.6925
Epoch [154/500], Loss: 0.2192, Accuracy: 0.695
Epoch [155/500], Loss: 0.2186, Accuracy: 0.6925
Epoch [156/500], Loss: 0.2181, Accuracy: 0.6975
Epoch [157/500], Loss: 0.2175, Accuracy: 0.6975
Epoch [158/500], Loss: 0.2169, Accuracy: 0.7
Epoch [159/500], Loss: 0.2164, Accuracy: 0.7025
Epoch [160/500], Loss: 0.2158, Accuracy: 0.71
Epoch [161/500], Loss: 0.2152, Accuracy: 0.7125
Epoch [162/500], Loss: 0.2147, Accuracy: 0.715
Epoch [163/500], Loss: 0.2142, Accuracy: 0.7175
Epoch [164/500], Loss: 0.2136, Accuracy: 0.72
Epoch [165/500], Loss: 0.2131, Accuracy: 0.7225
```

```
Epoch [166/500], Loss: 0.2125, Accuracy: 0.725
Epoch [167/500], Loss: 0.2120, Accuracy: 0.73
Epoch [168/500], Loss: 0.2115, Accuracy: 0.73
Epoch [169/500], Loss: 0.2110, Accuracy: 0.735
Epoch [170/500], Loss: 0.2105, Accuracy: 0.7375
Epoch [171/500], Loss: 0.2099, Accuracy: 0.74
Epoch [172/500], Loss: 0.2094, Accuracy: 0.74
Epoch [173/500], Loss: 0.2089, Accuracy: 0.74
Epoch [174/500], Loss: 0.2084, Accuracy: 0.74
Epoch [175/500], Loss: 0.2079, Accuracy: 0.74
Epoch [176/500], Loss: 0.2074, Accuracy: 0.7375
Epoch [177/500], Loss: 0.2069, Accuracy: 0.7375
Epoch [178/500], Loss: 0.2065, Accuracy: 0.735
Epoch [179/500], Loss: 0.2060, Accuracy: 0.735
Epoch [180/500], Loss: 0.2055, Accuracy: 0.735
Epoch [181/500], Loss: 0.2050, Accuracy: 0.735
Epoch [182/500], Loss: 0.2046, Accuracy: 0.735
Epoch [183/500], Loss: 0.2041, Accuracy: 0.7375
Epoch [184/500], Loss: 0.2036, Accuracy: 0.7375
Epoch [185/500], Loss: 0.2032, Accuracy: 0.7425
Epoch [186/500], Loss: 0.2027, Accuracy: 0.7425
Epoch [187/500], Loss: 0.2022, Accuracy: 0.7425
Epoch [188/500], Loss: 0.2018, Accuracy: 0.74
Epoch [189/500], Loss: 0.2013, Accuracy: 0.74
Epoch [190/500], Loss: 0.2009, Accuracy: 0.74
Epoch [191/500], Loss: 0.2005, Accuracy: 0.74
Epoch [192/500], Loss: 0.2000, Accuracy: 0.74
Epoch [193/500], Loss: 0.1996, Accuracy: 0.74
Epoch [194/500], Loss: 0.1992, Accuracy: 0.74
Epoch [195/500], Loss: 0.1987, Accuracy: 0.74
Epoch [196/500], Loss: 0.1983, Accuracy: 0.7425
Epoch [197/500], Loss: 0.1979, Accuracy: 0.7425
Epoch [198/500], Loss: 0.1975, Accuracy: 0.7425
Epoch [199/500], Loss: 0.1971, Accuracy: 0.7425
Epoch [200/500], Loss: 0.1966, Accuracy: 0.7425
Epoch [201/500], Loss: 0.1962, Accuracy: 0.7425
Epoch [202/500], Loss: 0.1958, Accuracy: 0.7425
Epoch [203/500], Loss: 0.1954, Accuracy: 0.7425
Epoch [204/500], Loss: 0.1950, Accuracy: 0.7425
Epoch [205/500], Loss: 0.1946, Accuracy: 0.745
Epoch [206/500], Loss: 0.1942, Accuracy: 0.745
Epoch [207/500], Loss: 0.1939, Accuracy: 0.745
Epoch [208/500], Loss: 0.1935, Accuracy: 0.745
Epoch [209/500], Loss: 0.1931, Accuracy: 0.7475
Epoch [210/500], Loss: 0.1927, Accuracy: 0.7475
Epoch [211/500], Loss: 0.1923, Accuracy: 0.7475
Epoch [212/500], Loss: 0.1919, Accuracy: 0.7475
Epoch [213/500], Loss: 0.1916, Accuracy: 0.7475
```

```
Epoch [214/500], Loss: 0.1912, Accuracy: 0.745
Epoch [215/500], Loss: 0.1908, Accuracy: 0.7475
Epoch [216/500], Loss: 0.1905, Accuracy: 0.7475
Epoch [217/500], Loss: 0.1901, Accuracy: 0.7475
Epoch [218/500], Loss: 0.1898, Accuracy: 0.7475
Epoch [219/500], Loss: 0.1894, Accuracy: 0.75
Epoch [220/500], Loss: 0.1890, Accuracy: 0.75
Epoch [221/500], Loss: 0.1887, Accuracy: 0.75
Epoch [222/500], Loss: 0.1883, Accuracy: 0.75
Epoch [223/500], Loss: 0.1880, Accuracy: 0.75
Epoch [224/500], Loss: 0.1877, Accuracy: 0.75
Epoch [225/500], Loss: 0.1873, Accuracy: 0.75
Epoch [226/500], Loss: 0.1870, Accuracy: 0.75
Epoch [227/500], Loss: 0.1866, Accuracy: 0.7475
Epoch [228/500], Loss: 0.1863, Accuracy: 0.7475
Epoch [229/500], Loss: 0.1860, Accuracy: 0.75
Epoch [230/500], Loss: 0.1856, Accuracy: 0.75
Epoch [231/500], Loss: 0.1853, Accuracy: 0.75
Epoch [232/500], Loss: 0.1850, Accuracy: 0.75
Epoch [233/500], Loss: 0.1847, Accuracy: 0.7475
Epoch [234/500], Loss: 0.1844, Accuracy: 0.7475
Epoch [235/500], Loss: 0.1840, Accuracy: 0.75
Epoch [236/500], Loss: 0.1837, Accuracy: 0.75
Epoch [237/500], Loss: 0.1834, Accuracy: 0.75
Epoch [238/500], Loss: 0.1831, Accuracy: 0.75
Epoch [239/500], Loss: 0.1828, Accuracy: 0.7525
Epoch [240/500], Loss: 0.1825, Accuracy: 0.7525
Epoch [241/500], Loss: 0.1822, Accuracy: 0.755
Epoch [242/500], Loss: 0.1819, Accuracy: 0.755
Epoch [243/500], Loss: 0.1816, Accuracy: 0.755
Epoch [244/500], Loss: 0.1813, Accuracy: 0.755
Epoch [245/500], Loss: 0.1810, Accuracy: 0.755
Epoch [246/500], Loss: 0.1807, Accuracy: 0.755
Epoch [247/500], Loss: 0.1804, Accuracy: 0.755
Epoch [248/500], Loss: 0.1801, Accuracy: 0.755
Epoch [249/500], Loss: 0.1799, Accuracy: 0.755
Epoch [250/500], Loss: 0.1796, Accuracy: 0.7575
Epoch [251/500], Loss: 0.1793, Accuracy: 0.7575
Epoch [252/500], Loss: 0.1790, Accuracy: 0.7575
Epoch [253/500], Loss: 0.1787, Accuracy: 0.7575
Epoch [254/500], Loss: 0.1785, Accuracy: 0.7575
Epoch [255/500], Loss: 0.1782, Accuracy: 0.7575
Epoch [256/500], Loss: 0.1779, Accuracy: 0.7575
Epoch [257/500], Loss: 0.1777, Accuracy: 0.7575
Epoch [258/500], Loss: 0.1774, Accuracy: 0.7575
Epoch [259/500], Loss: 0.1771, Accuracy: 0.7575
Epoch [260/500], Loss: 0.1769, Accuracy: 0.7575
Epoch [261/500], Loss: 0.1766, Accuracy: 0.7575
```

```
Epoch [262/500], Loss: 0.1764, Accuracy: 0.7575
Epoch [263/500], Loss: 0.1761, Accuracy: 0.7575
Epoch [264/500], Loss: 0.1758, Accuracy: 0.7575
Epoch [265/500], Loss: 0.1756, Accuracy: 0.7575
Epoch [266/500], Loss: 0.1753, Accuracy: 0.76
Epoch [267/500], Loss: 0.1751, Accuracy: 0.76
Epoch [268/500], Loss: 0.1748, Accuracy: 0.76
Epoch [269/500], Loss: 0.1746, Accuracy: 0.76
Epoch [270/500], Loss: 0.1744, Accuracy: 0.76
Epoch [271/500], Loss: 0.1741, Accuracy: 0.76
Epoch [272/500], Loss: 0.1739, Accuracy: 0.76
Epoch [273/500], Loss: 0.1736, Accuracy: 0.7575
Epoch [274/500], Loss: 0.1734, Accuracy: 0.7575
Epoch [275/500], Loss: 0.1732, Accuracy: 0.755
Epoch [276/500], Loss: 0.1729, Accuracy: 0.755
Epoch [277/500], Loss: 0.1727, Accuracy: 0.755
Epoch [278/500], Loss: 0.1725, Accuracy: 0.7575
Epoch [279/500], Loss: 0.1723, Accuracy: 0.7575
Epoch [280/500], Loss: 0.1720, Accuracy: 0.7575
Epoch [281/500], Loss: 0.1718, Accuracy: 0.7575
Epoch [282/500], Loss: 0.1716, Accuracy: 0.7575
Epoch [283/500], Loss: 0.1714, Accuracy: 0.7575
Epoch [284/500], Loss: 0.1711, Accuracy: 0.7575
Epoch [285/500], Loss: 0.1709, Accuracy: 0.755
Epoch [286/500], Loss: 0.1707, Accuracy: 0.755
Epoch [287/500], Loss: 0.1705, Accuracy: 0.7525
Epoch [288/500], Loss: 0.1703, Accuracy: 0.7525
Epoch [289/500], Loss: 0.1701, Accuracy: 0.7525
Epoch [290/500], Loss: 0.1699, Accuracy: 0.7525
Epoch [291/500], Loss: 0.1697, Accuracy: 0.755
Epoch [292/500], Loss: 0.1694, Accuracy: 0.755
Epoch [293/500], Loss: 0.1692, Accuracy: 0.755
Epoch [294/500], Loss: 0.1690, Accuracy: 0.755
Epoch [295/500], Loss: 0.1688, Accuracy: 0.755
Epoch [296/500], Loss: 0.1686, Accuracy: 0.755
Epoch [297/500], Loss: 0.1684, Accuracy: 0.755
Epoch [298/500], Loss: 0.1682, Accuracy: 0.755
Epoch [299/500], Loss: 0.1680, Accuracy: 0.7575
Epoch [300/500], Loss: 0.1679, Accuracy: 0.7575
Epoch [301/500], Loss: 0.1677, Accuracy: 0.7575
Epoch [302/500], Loss: 0.1675, Accuracy: 0.7575
Epoch [303/500], Loss: 0.1673, Accuracy: 0.7575
Epoch [304/500], Loss: 0.1671, Accuracy: 0.7575
Epoch [305/500], Loss: 0.1669, Accuracy: 0.7575
Epoch [306/500], Loss: 0.1667, Accuracy: 0.7575
Epoch [307/500], Loss: 0.1665, Accuracy: 0.7575
Epoch [308/500], Loss: 0.1664, Accuracy: 0.7575
Epoch [309/500], Loss: 0.1662, Accuracy: 0.7575
```

```
Epoch [310/500], Loss: 0.1660, Accuracy: 0.7575
Epoch [311/500], Loss: 0.1658, Accuracy: 0.7575
Epoch [312/500], Loss: 0.1656, Accuracy: 0.7575
Epoch [313/500], Loss: 0.1655, Accuracy: 0.7575
Epoch [314/500], Loss: 0.1653, Accuracy: 0.7575
Epoch [315/500], Loss: 0.1651, Accuracy: 0.7575
Epoch [316/500], Loss: 0.1649, Accuracy: 0.7575
Epoch [317/500], Loss: 0.1648, Accuracy: 0.7575
Epoch [318/500], Loss: 0.1646, Accuracy: 0.7575
Epoch [319/500], Loss: 0.1644, Accuracy: 0.7575
Epoch [320/500], Loss: 0.1643, Accuracy: 0.76
Epoch [321/500], Loss: 0.1641, Accuracy: 0.76
Epoch [322/500], Loss: 0.1639, Accuracy: 0.7625
Epoch [323/500], Loss: 0.1638, Accuracy: 0.7625
Epoch [324/500], Loss: 0.1636, Accuracy: 0.7625
Epoch [325/500], Loss: 0.1634, Accuracy: 0.7625
Epoch [326/500], Loss: 0.1633, Accuracy: 0.7625
Epoch [327/500], Loss: 0.1631, Accuracy: 0.7625
Epoch [328/500], Loss: 0.1630, Accuracy: 0.7625
Epoch [329/500], Loss: 0.1628, Accuracy: 0.7625
Epoch [330/500], Loss: 0.1626, Accuracy: 0.7625
Epoch [331/500], Loss: 0.1625, Accuracy: 0.7625
Epoch [332/500], Loss: 0.1623, Accuracy: 0.7625
Epoch [333/500], Loss: 0.1622, Accuracy: 0.765
Epoch [334/500], Loss: 0.1620, Accuracy: 0.765
Epoch [335/500], Loss: 0.1619, Accuracy: 0.765
Epoch [336/500], Loss: 0.1617, Accuracy: 0.765
Epoch [337/500], Loss: 0.1616, Accuracy: 0.765
Epoch [338/500], Loss: 0.1614, Accuracy: 0.765
Epoch [339/500], Loss: 0.1613, Accuracy: 0.765
Epoch [340/500], Loss: 0.1612, Accuracy: 0.765
Epoch [341/500], Loss: 0.1610, Accuracy: 0.765
Epoch [342/500], Loss: 0.1609, Accuracy: 0.765
Epoch [343/500], Loss: 0.1607, Accuracy: 0.765
Epoch [344/500], Loss: 0.1606, Accuracy: 0.765
Epoch [345/500], Loss: 0.1604, Accuracy: 0.765
Epoch [346/500], Loss: 0.1603, Accuracy: 0.765
Epoch [347/500], Loss: 0.1602, Accuracy: 0.765
Epoch [348/500], Loss: 0.1600, Accuracy: 0.765
Epoch [349/500], Loss: 0.1599, Accuracy: 0.765
Epoch [350/500], Loss: 0.1598, Accuracy: 0.765
Epoch [351/500], Loss: 0.1596, Accuracy: 0.765
Epoch [352/500], Loss: 0.1595, Accuracy: 0.765
Epoch [353/500], Loss: 0.1594, Accuracy: 0.7625
Epoch [354/500], Loss: 0.1592, Accuracy: 0.7625
Epoch [355/500], Loss: 0.1591, Accuracy: 0.76
Epoch [356/500], Loss: 0.1590, Accuracy: 0.76
Epoch [357/500], Loss: 0.1589, Accuracy: 0.76
```

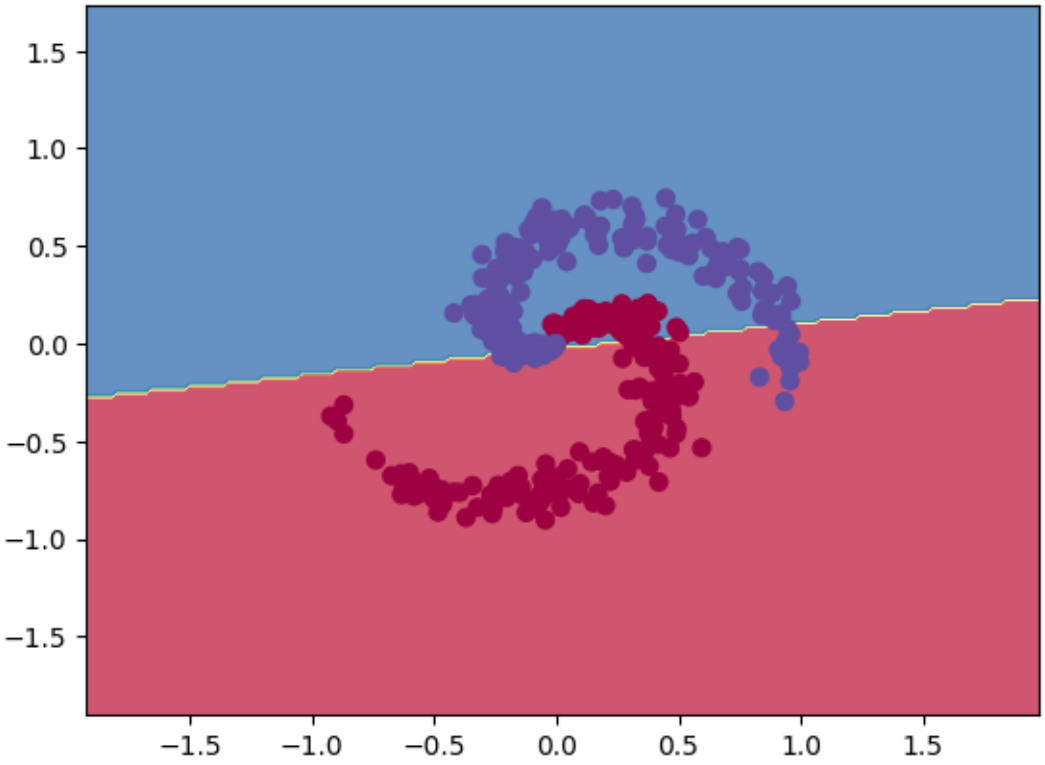
```
Epoch [358/500], Loss: 0.1587, Accuracy: 0.76
Epoch [359/500], Loss: 0.1586, Accuracy: 0.76
Epoch [360/500], Loss: 0.1585, Accuracy: 0.76
Epoch [361/500], Loss: 0.1584, Accuracy: 0.76
Epoch [362/500], Loss: 0.1582, Accuracy: 0.76
Epoch [363/500], Loss: 0.1581, Accuracy: 0.7575
Epoch [364/500], Loss: 0.1580, Accuracy: 0.76
Epoch [365/500], Loss: 0.1579, Accuracy: 0.76
Epoch [366/500], Loss: 0.1577, Accuracy: 0.76
Epoch [367/500], Loss: 0.1576, Accuracy: 0.76
Epoch [368/500], Loss: 0.1575, Accuracy: 0.76
Epoch [369/500], Loss: 0.1574, Accuracy: 0.76
Epoch [370/500], Loss: 0.1573, Accuracy: 0.76
Epoch [371/500], Loss: 0.1572, Accuracy: 0.76
Epoch [372/500], Loss: 0.1570, Accuracy: 0.76
Epoch [373/500], Loss: 0.1569, Accuracy: 0.7575
Epoch [374/500], Loss: 0.1568, Accuracy: 0.755
Epoch [375/500], Loss: 0.1567, Accuracy: 0.755
Epoch [376/500], Loss: 0.1566, Accuracy: 0.755
Epoch [377/500], Loss: 0.1565, Accuracy: 0.755
Epoch [378/500], Loss: 0.1564, Accuracy: 0.755
Epoch [379/500], Loss: 0.1563, Accuracy: 0.755
Epoch [380/500], Loss: 0.1562, Accuracy: 0.7525
Epoch [381/500], Loss: 0.1561, Accuracy: 0.7525
Epoch [382/500], Loss: 0.1559, Accuracy: 0.7525
Epoch [383/500], Loss: 0.1558, Accuracy: 0.7525
Epoch [384/500], Loss: 0.1557, Accuracy: 0.7525
Epoch [385/500], Loss: 0.1556, Accuracy: 0.7525
Epoch [386/500], Loss: 0.1555, Accuracy: 0.7525
Epoch [387/500], Loss: 0.1554, Accuracy: 0.7525
Epoch [388/500], Loss: 0.1553, Accuracy: 0.7525
Epoch [389/500], Loss: 0.1552, Accuracy: 0.755
Epoch [390/500], Loss: 0.1551, Accuracy: 0.755
Epoch [391/500], Loss: 0.1550, Accuracy: 0.755
Epoch [392/500], Loss: 0.1549, Accuracy: 0.755
Epoch [393/500], Loss: 0.1548, Accuracy: 0.755
Epoch [394/500], Loss: 0.1547, Accuracy: 0.755
Epoch [395/500], Loss: 0.1546, Accuracy: 0.755
Epoch [396/500], Loss: 0.1545, Accuracy: 0.755
Epoch [397/500], Loss: 0.1544, Accuracy: 0.7525
Epoch [398/500], Loss: 0.1543, Accuracy: 0.7525
Epoch [399/500], Loss: 0.1543, Accuracy: 0.7525
Epoch [400/500], Loss: 0.1542, Accuracy: 0.7525
Epoch [401/500], Loss: 0.1541, Accuracy: 0.7525
Epoch [402/500], Loss: 0.1540, Accuracy: 0.7525
Epoch [403/500], Loss: 0.1539, Accuracy: 0.7525
Epoch [404/500], Loss: 0.1538, Accuracy: 0.7525
Epoch [405/500], Loss: 0.1537, Accuracy: 0.7525
```

```
Epoch [406/500], Loss: 0.1536, Accuracy: 0.7525
Epoch [407/500], Loss: 0.1535, Accuracy: 0.7525
Epoch [408/500], Loss: 0.1534, Accuracy: 0.75
Epoch [409/500], Loss: 0.1534, Accuracy: 0.75
Epoch [410/500], Loss: 0.1533, Accuracy: 0.75
Epoch [411/500], Loss: 0.1532, Accuracy: 0.75
Epoch [412/500], Loss: 0.1531, Accuracy: 0.75
Epoch [413/500], Loss: 0.1530, Accuracy: 0.75
Epoch [414/500], Loss: 0.1529, Accuracy: 0.75
Epoch [415/500], Loss: 0.1528, Accuracy: 0.75
Epoch [416/500], Loss: 0.1528, Accuracy: 0.75
Epoch [417/500], Loss: 0.1527, Accuracy: 0.75
Epoch [418/500], Loss: 0.1526, Accuracy: 0.75
Epoch [419/500], Loss: 0.1525, Accuracy: 0.75
Epoch [420/500], Loss: 0.1524, Accuracy: 0.75
Epoch [421/500], Loss: 0.1524, Accuracy: 0.75
Epoch [422/500], Loss: 0.1523, Accuracy: 0.75
Epoch [423/500], Loss: 0.1522, Accuracy: 0.75
Epoch [424/500], Loss: 0.1521, Accuracy: 0.7475
Epoch [425/500], Loss: 0.1520, Accuracy: 0.7475
Epoch [426/500], Loss: 0.1520, Accuracy: 0.7475
Epoch [427/500], Loss: 0.1519, Accuracy: 0.7475
Epoch [428/500], Loss: 0.1518, Accuracy: 0.7475
Epoch [429/500], Loss: 0.1517, Accuracy: 0.7475
Epoch [430/500], Loss: 0.1517, Accuracy: 0.7475
Epoch [431/500], Loss: 0.1516, Accuracy: 0.7475
Epoch [432/500], Loss: 0.1515, Accuracy: 0.7475
Epoch [433/500], Loss: 0.1514, Accuracy: 0.7475
Epoch [434/500], Loss: 0.1514, Accuracy: 0.7475
Epoch [435/500], Loss: 0.1513, Accuracy: 0.7475
Epoch [436/500], Loss: 0.1512, Accuracy: 0.7475
Epoch [437/500], Loss: 0.1511, Accuracy: 0.7475
Epoch [438/500], Loss: 0.1511, Accuracy: 0.7475
Epoch [439/500], Loss: 0.1510, Accuracy: 0.7475
Epoch [440/500], Loss: 0.1509, Accuracy: 0.7475
Epoch [441/500], Loss: 0.1509, Accuracy: 0.745
Epoch [442/500], Loss: 0.1508, Accuracy: 0.745
Epoch [443/500], Loss: 0.1507, Accuracy: 0.745
Epoch [444/500], Loss: 0.1507, Accuracy: 0.745
Epoch [445/500], Loss: 0.1506, Accuracy: 0.745
Epoch [446/500], Loss: 0.1505, Accuracy: 0.745
Epoch [447/500], Loss: 0.1505, Accuracy: 0.745
Epoch [448/500], Loss: 0.1504, Accuracy: 0.745
Epoch [449/500], Loss: 0.1503, Accuracy: 0.7425
Epoch [450/500], Loss: 0.1503, Accuracy: 0.7425
Epoch [451/500], Loss: 0.1502, Accuracy: 0.7425
Epoch [452/500], Loss: 0.1501, Accuracy: 0.7425
Epoch [453/500], Loss: 0.1501, Accuracy: 0.7425
```

```
Epoch [454/500], Loss: 0.1500, Accuracy: 0.7425
Epoch [455/500], Loss: 0.1499, Accuracy: 0.7425
Epoch [456/500], Loss: 0.1499, Accuracy: 0.7425
Epoch [457/500], Loss: 0.1498, Accuracy: 0.7425
Epoch [458/500], Loss: 0.1498, Accuracy: 0.7425
Epoch [459/500], Loss: 0.1497, Accuracy: 0.7425
Epoch [460/500], Loss: 0.1496, Accuracy: 0.7425
Epoch [461/500], Loss: 0.1496, Accuracy: 0.7425
Epoch [462/500], Loss: 0.1495, Accuracy: 0.7425
Epoch [463/500], Loss: 0.1495, Accuracy: 0.7425
Epoch [464/500], Loss: 0.1494, Accuracy: 0.7425
Epoch [465/500], Loss: 0.1493, Accuracy: 0.7425
Epoch [466/500], Loss: 0.1493, Accuracy: 0.7425
Epoch [467/500], Loss: 0.1492, Accuracy: 0.7425
Epoch [468/500], Loss: 0.1492, Accuracy: 0.7425
Epoch [469/500], Loss: 0.1491, Accuracy: 0.7425
Epoch [470/500], Loss: 0.1491, Accuracy: 0.7425
Epoch [471/500], Loss: 0.1490, Accuracy: 0.7425
Epoch [472/500], Loss: 0.1489, Accuracy: 0.7425
Epoch [473/500], Loss: 0.1489, Accuracy: 0.7425
Epoch [474/500], Loss: 0.1488, Accuracy: 0.7425
Epoch [475/500], Loss: 0.1488, Accuracy: 0.7425
Epoch [476/500], Loss: 0.1487, Accuracy: 0.7425
Epoch [477/500], Loss: 0.1487, Accuracy: 0.7425
Epoch [478/500], Loss: 0.1486, Accuracy: 0.7425
Epoch [479/500], Loss: 0.1486, Accuracy: 0.7425
Epoch [480/500], Loss: 0.1485, Accuracy: 0.7425
Epoch [481/500], Loss: 0.1485, Accuracy: 0.7425
Epoch [482/500], Loss: 0.1484, Accuracy: 0.7425
Epoch [483/500], Loss: 0.1484, Accuracy: 0.7425
Epoch [484/500], Loss: 0.1483, Accuracy: 0.7425
Epoch [485/500], Loss: 0.1483, Accuracy: 0.74
Epoch [486/500], Loss: 0.1482, Accuracy: 0.74
Epoch [487/500], Loss: 0.1482, Accuracy: 0.74
Epoch [488/500], Loss: 0.1481, Accuracy: 0.74
Epoch [489/500], Loss: 0.1481, Accuracy: 0.74
Epoch [490/500], Loss: 0.1480, Accuracy: 0.74
Epoch [491/500], Loss: 0.1480, Accuracy: 0.74
Epoch [492/500], Loss: 0.1479, Accuracy: 0.74
Epoch [493/500], Loss: 0.1479, Accuracy: 0.74
Epoch [494/500], Loss: 0.1478, Accuracy: 0.74
Epoch [495/500], Loss: 0.1478, Accuracy: 0.74
Epoch [496/500], Loss: 0.1477, Accuracy: 0.74
Epoch [497/500], Loss: 0.1477, Accuracy: 0.74
Epoch [498/500], Loss: 0.1476, Accuracy: 0.74
Epoch [499/500], Loss: 0.1476, Accuracy: 0.74
Epoch [500/500], Loss: 0.1475, Accuracy: 0.74
```

[10]:
```
visualize(sample,target, model)
```

```
/Users/erfanehmahmoudzadeh/miniconda3/envs/deeplearning/lib/python3.11/site-
packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcoming
release, it will be required to pass the indexing argument. (Triggered
internally at /Users/runner/work/pytorch/pytorch/pytorch/aten/src/ATen/native/Te
nsorShape.cpp:3527.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
```



Here is an example: the green line is the line of the decision boundary. You should draw the linear decision boundary like this.

Given Example: Weights and Bias You can use weight and bias attributes of your model to find the equation of the decision boundary.

[73]:
```
print("Weights: \n",model.linear.weight)
print("Bias: \n",model.linear.bias)
```

```
Weights:
 Parameter containing:
tensor([[-0.1152,  0.5983]], requires_grad=True)
Bias:
 Parameter containing:
```

```
tensor([0.5201], requires_grad=True)
```

1.2.3 Logistic Regression

[ ]:
```
# implement your logistic regression here
```

1.2.4 Deep Linear Neural Network

[ ]:
```
# # implement your nonlinear feed forward neural network here
```

1.2.5 Deep Neural Network

[ ]:
```
# implement your nonlinear feed forward neural network here
```

1.2.6 Deep Neural Network with L2-regularization

$\lambda = 0.01, 2, 100$

[ ]:
```

```