|  |  |
| --- | --- |
| **Ex No: 2**  **Date: 14-08-24** | **Planar Data Classification with One Hidden Layer** |

**Objective:**

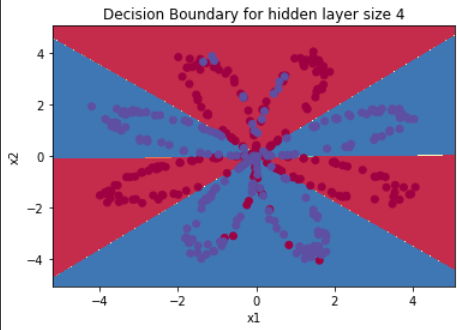
To build a neural network classifier with one hidden layer for classifying planar data into two classes.

**Descriptions:**

The problem of planar data classification such as the one shown in the given feature space is a learning problem whose task is to classify the data points between the two classes. This task is usually illustrated in a 2D space after a transformation which is useful in deciphering the position of the decision boundary of the classifier.

It is in such a binary classification problem that neural networks can be applied. Which is to conclude that there is a need to identify complex patterns in the data when training a neural network with one hidden layer, because it is suspected that logistic regression might have further difficulty in handling non-linear boundaries. They are a single input layer, which takes initial input, a single standard hidden layer, which we may define to contain a random number of neurons, and one output layer which should consist of one neuron that gives the binary result of the classification output.

In the training phase, there are the forward pass which computes the predictions, the backward pass which computes the loss, the difference between the prediction and the actual output and then there is the backward pass which makes the gradients of the weights of the model lower through the iteration of the Gradient Descent algorithm.

****

**MODAL**

Building the Parts of the Neural Network:

The following are the main steps to build a neural network with one hidden layer:

1. Define the Model Structure:
   * Input layer: Takes in the features from the planar data.
   * Hidden layer: Contains neurons that apply an activation function to capture non-linearities.
   * Output layer: Produces the final binary output (class 0 or 1).
2. Initialize the Model’s Parameters:
   * Initialize weights and biases for both the hidden and output layers.
3. Loop:
   * Forward Propagation:
     + Calculate the output of the hidden layer using the activation function.
     + Calculate the output of the network (prediction) using the activation function applied to the hidden layer output.
   * Loss Computation:
     + Measure the loss using a suitable loss function (e.g., cross-entropy).
   * Backward Propagation:
     + Compute the gradients of the loss with respect to the network’s parameters.
   * Parameter Update:
     + Update the weights and biases using Gradient Descent to minimize the loss.

**USN NUMBER:** 1RVU22BSC097

**NAME:** SHREYA SINHA

**GitHub Link:**

[**https://github.com/Shreyasinha7/lab2planar.git**](https://github.com/Shreyasinha7/lab2planar.git)