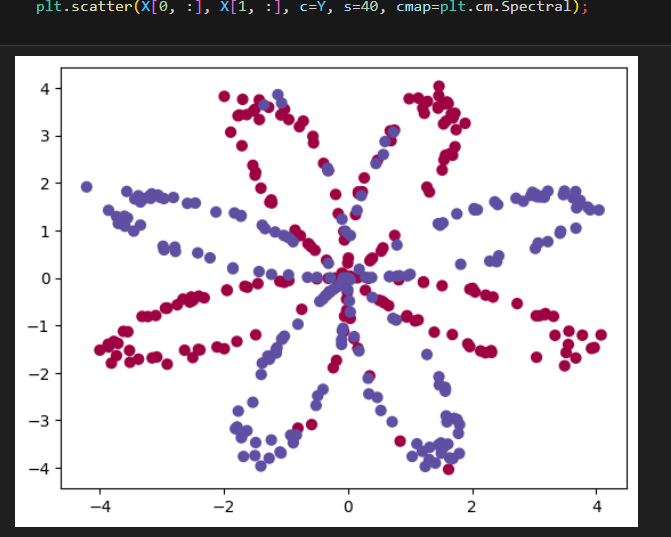
|  |  |
| --- | --- |
| **Ex No: 3**  **Date: 19th August 2024** | **Classify the Planar dataset using one hidden layer** |

**Objective:** The goal is to construct a binary classification neural network with a single hidden layer, utilizing the tanh activation function in the hidden layer and sigmoid in the output layer. The chosen loss function for this model is Cross-Entropy Loss.

**Description:**

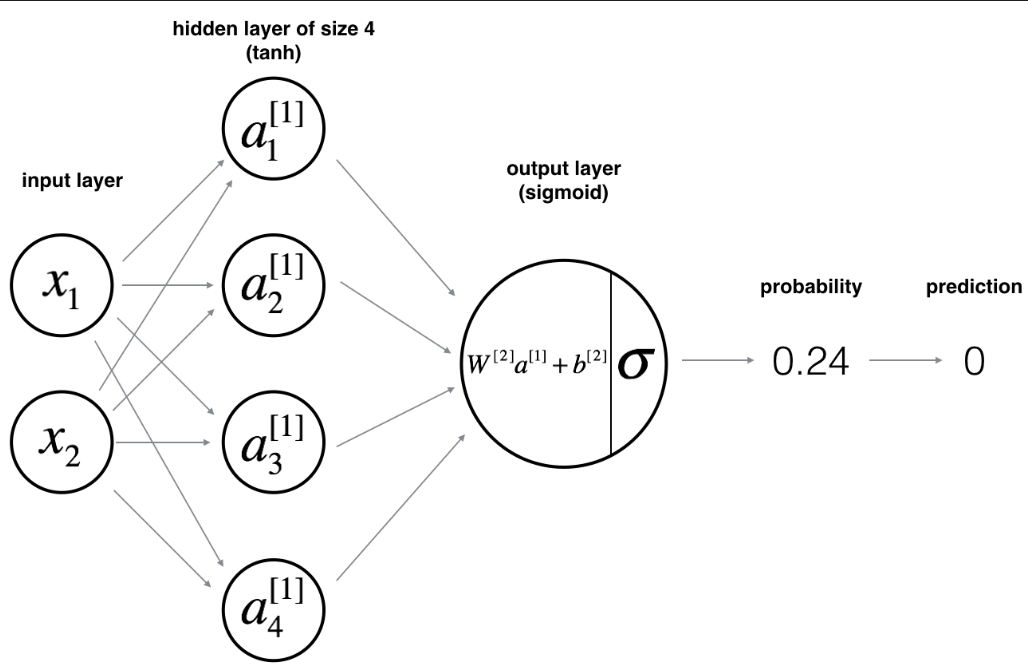
Binary classification refers to the task of categorizing data points into one of two distinct classes. In this case, we aim to classify data from a "flower" dataset, where each data point is associated with features *X* and an output label *Y*, where *Y = 0* or *Y = 1*.



**Model:**

In this experiment, we construct a neural network with a single hidden layer. This layer enables the model to identify more complex patterns within the data, as opposed to a logistic regression model, which lacks hidden layers.

* **Input Layer**: Accepts the feature set *X*.
* **Hidden Layer**: Applies a non-linear activation function (tanh) to capture intricate relationships within the data.
* **Output Layer**: Utilizes the sigmoid function to output probability predictions for each class.  
  Model parameters, including weights and biases, are initialized randomly and are fine-tuned through gradient descent during training.



.

**Building the parts of the algorithm:**

The primary steps to build this neural network are detailed as follows:

1. **Model Structure Definition**
   1. **Number of Input Features**: Defined by the dimensionality of the feature vector *X*.
   2. **Number of Hidden Units**: A hyperparameter that can be tuned to improve model performance.
   3. **Activation Functions**: tanh is applied in the hidden layer, while the output layer uses the sigmoid function.
2. **Parameter Initialization**
   1. Weights and biases are initialized randomly to facilitate effective learning.
   2. Proper initialization is key to ensuring the network learns efficiently during training.
3. **Forward Propagation**
   1. Compute the linear combination of inputs and weights.
   2. Apply activation functions to introduce non-linearity.
   3. Compute the predicted output.
4. **Loss Calculation**
   1. The cross-entropy loss function measures the difference between the predicted and actual labels.
   2. This loss function provides guidance for the model to adjust parameters.
5. **Backward Propagation**
   1. Calculate gradients of the loss function with respect to the model parameters.
   2. Use these gradients to adjust parameters in a direction that minimizes the loss.
6. **Parameter Update**
   1. Gradient descent is applied to iteratively improve model performance by minimizing the loss function.
7. **Building a Network using nn\_model()**
   1. Integrate the steps of forward propagation, cost computation, backward propagation, and parameter updating into a cohesive model. This model adjusts parameters iteratively to make accurate predictions.

**Key Observations:**

The neural network is trained on a planar dataset, and its performance is evaluated by visualizing the decision boundary and comparing predicted outputs with actual labels. With a well-trained model, we expect accurate classification, illustrating the benefit of incorporating a hidden layer in binary classification tasks. The trained model achieves a classification accuracy of 90%, highlighting the effectiveness of this shallow neural network.

**GitHub Link:** [**https://github.com/tulasigr/DeepLearning**](https://github.com/tulasigr/DeepLearning)