|  |  |
| --- | --- |
| **Ex No: 2**  **Date: 14-08-2024** | **Planar data classification with one hidden layer** |

**Objective:**

The objective is to develop a 2-class classification neural network with a single hidden layer, utilizing non-linear activation functions, such as tanh. The project will involve calculating cross-entropy loss and implementing both forward and backward propagation to optimize the model.

**Descriptions:**

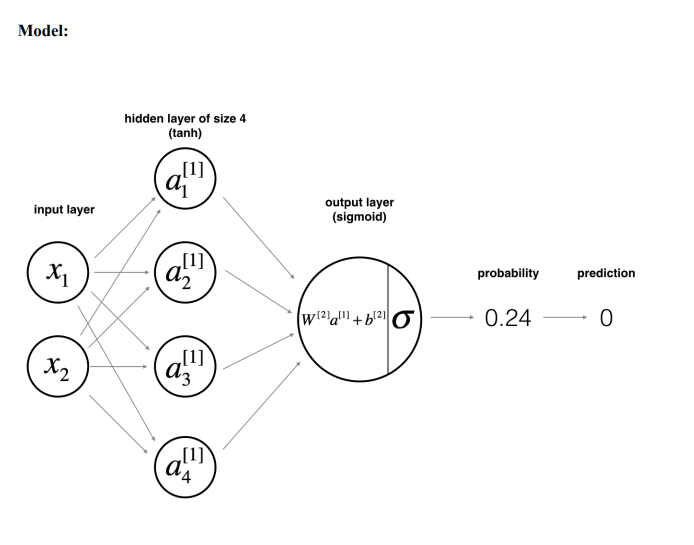
A 2-class classification neural network is a machine learning model designed to distinguish between two distinct categories. In this context, the model will classify data into two binary categories. The neural network we are constructing includes a single hidden layer, employing a non-linear activation function like tanh. This introduces non-linearity, allowing the model to better recognize complex patterns within the data.

We will use a "flower" dataset for this binary classification task, which will be stored in variables X (features) and Y (labels). The dataset contains images of flowers, and the aim is to classify these images into one of two categories, such as identifying between two different flower species.

Unlike logistic regression, which is a simpler binary classification method that directly computes outputs, our neural network introduces an additional hidden layer. This hidden layer enhances the model's ability to generalize and identify more intricate relationships in the input features. Each neuron in the hidden layer applies the tanh activation function, improving predictions by capturing non-linear relationships.

The model is trained using a loss function, specifically the cross-entropy loss. This loss function measures the discrepancy between the predicted probability distribution and the true distribution, guiding the network to learn accurate classifications over multiple iterations.

Forward propagation is used to calculate the predicted output by passing the input through the network layers, while backward propagation adjusts the network weights. During backward propagation, gradients with respect to the loss function are computed, and the weights are updated to minimize the cross-entropy loss, enhancing the model’s performance iteratively.

**Steps to Build the Model:**

Define the model structure by specifying the sizes of the input layer (n\_x), hidden layer (n\_h), and output layer (n\_y).

Initialize the model’s parameters.

Loop through the following:

* Perform forward propagation using the tanh function for the hidden layer and sigmoid for the output layer.
* Calculate the cost.
* Conduct backward propagation using the chain rule and derivatives of the activation functions.
* Update the parameters.
* Repeat the process for a set number of epochs or until convergence is achieved.

This process encompasses building a complete neural network with a hidden layer, effectively using a non-linear activation unit, implementing forward and backward propagation, training the model, and analyzing the effects of varying the hidden layer size, including potential overfitting.

**GitHub Link:**

[**https://github.com/abraaaar/RVU\_BtechHons/tree/main/Deep%20Learning/Lab%202**](https://github.com/abraaaar/RVU_BtechHons/tree/main/Deep%20Learning/Lab%202)