USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

Ex No: 3 **Date: 20/08/2024**	**Building your Deep Neural Network**

Objective:

To implement a 2-class classification neural network with a single hidden layer, utilizing units with a non-linear activation function (such as tanh). The implementation will involve computing the cross entropy loss and developing both forward and backward propagation mechanisms to optimize the model.

Descriptions:

This project involves building a deep neural network using numpy for scientific computing, matplotlib for visualization, and custom utility functions for operations like activation and gradient calculations. The network will be constructed by stacking layers with linear transformations followed by non-linear activation functions, such as ReLU or Sigmoid. The goal is to implement a modular and reusable codebase that allows for easy construction and training of deep neural networks with varying architectures. The process starts with initializing parameters for a two-layer network and then extends this initialization to a deeper network with multiple layers. After parameter initialization, students will implement forward propagation, which includes calculating linear transformations and applying activation functions like ReLU and Sigmoid. Following forward propagation, we will proceed to backward propagation, where we compute gradients and update the parameters to minimize the loss function.

Model:

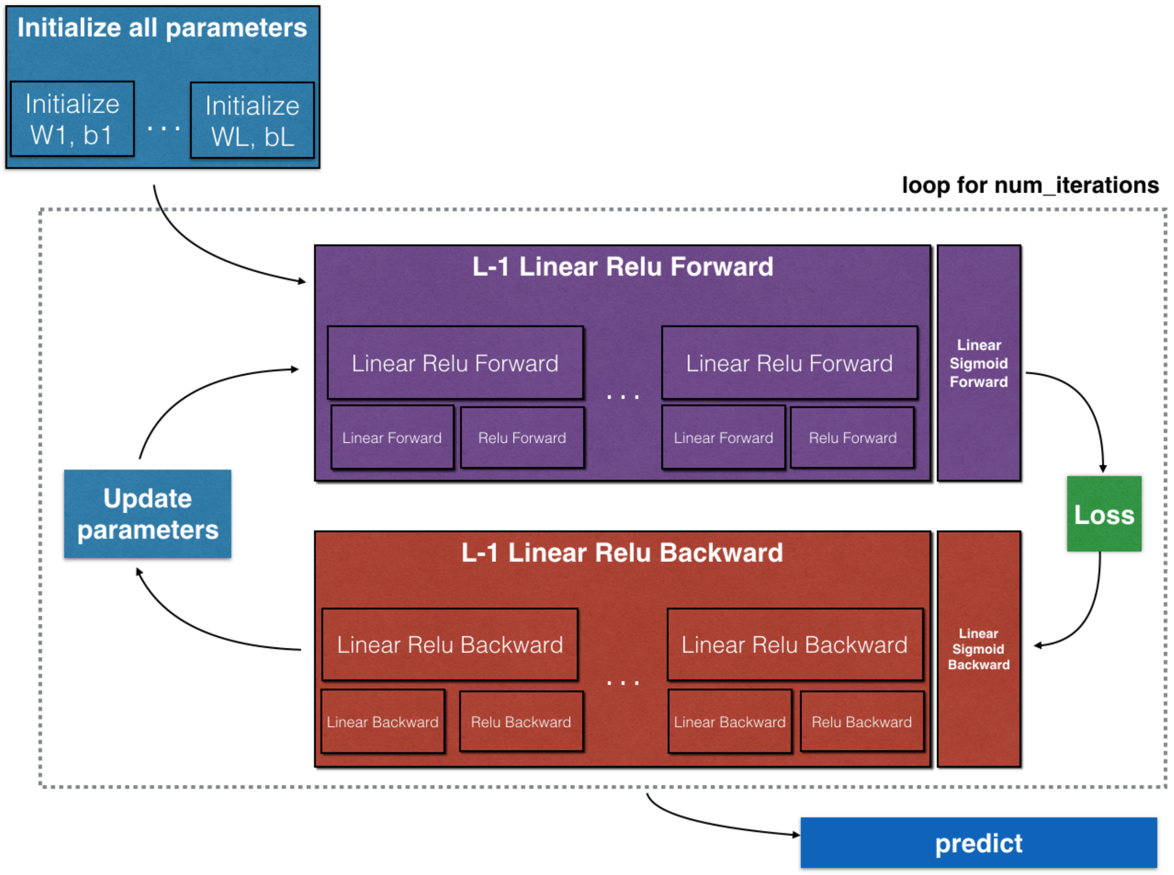
- Layer-wise structure: The network will consist of multiple layers, including input, hidden, and output layers.

Input layer of size 5 Hidden layer of size 4 Output layer of size 3

I

- Activation functions: The hidden layers will use the ReLU activation function, while the output layer will use the Sigmoid activation function, suitable for binary classification tasks.

- Parameter initialization: Weights will be initialized randomly with small values, and biases will be initialized to zero.
- Forward propagation: This process involves calculating the linear combination of inputs and weights, adding biases, and applying activation functions to obtain the activations for the next layer.
- Backward propagation: This step involves calculating the gradients of the loss function with respect to the weights and biases using the chain rule, which will then be used to update the parameters in the direction opposite to the gradient to minimize the loss.
- Cost function: The cross-entropy loss will be used as the cost function, commonly used in classification tasks.



USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

Building the parts of algorithm

1. Packages

- Objective: Import the necessary libraries.
- Details: Libraries like numpy for numerical operations, matplotlib for plotting, and custom utility functions (like sigmoid, relu, etc.) are imported to facilitate building and training the neural network.

2. Initialization of Parameters

- Objective: Initialize the network's weights and biases.
- Details:For each layer in the network, initialize the weights (W) with small random values to break symmetry, and the biases (b) with zeros.

This step ensures that the network has a good starting point for learning.

3. Forward Propagation

- Objective: Compute the predictions based on the input features.
- Details:The network performs a series of matrix multiplications and additions, followed by non-linear activations (e.g., ReLU, Sigmoid) to generate the output.

Forward propagation involves calculating the linear transformation Z and applying an activation function to get A.

4. Compute Cost

- Objective: Calculate the cost to measure how well the network's predictions match the actual labels.
- Details:Use a cost function, typically cross-entropy, to quantify the difference between the predicted values and actual outcomes.

5. Backward Propagation

- Objective: Compute the gradients of the cost function with respect to the parameters.
- Details:Backward propagation involves calculating the gradients of the loss with respect to the weights and biases, using the chain rule.

This step updates the parameters in the direction that minimizes the cost.

6. Update Parameters

- Objective: Update the network's weights and biases using the gradients computed during backpropagation.

- Details:The parameters are updated using gradient descent or other optimization algorithms to minimize the cost function and improve the model's accuracy.

7. Model Training

- Objective: Train the model on the training dataset.
- Details:The training process involves iterating over the dataset multiple times (epochs) and repeating the forward propagation, cost computation, backward propagation, and parameter update steps until the model converges to a solution.

8. Prediction and Evaluation

- Objective: Use the trained model to make predictions on new data and evaluate its performance.
- Details:After training, the model is tested on unseen data to assess its accuracy and generalization ability.

**GitHub Link:
https://github.com/joselynrianaaa/DeepLearning_Labs/blob/main/JoselynRiana_Building_your_Deep_Neural_Network_Step_by_Step_v8a.ipynb**