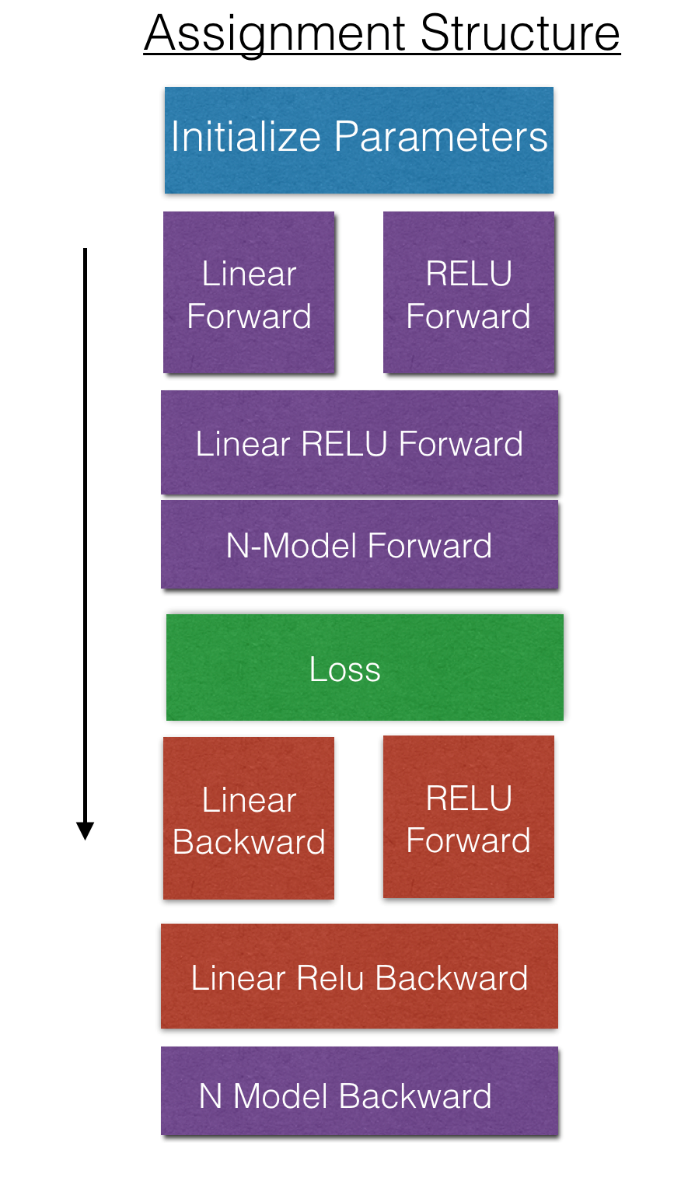
|  |  |
| --- | --- |
| **Ex No: 3**  **Date: 20/08/24** | **Building a Deep Neural Network: Step by Step** |

**Objective:**

To implement the essential functions required to build a deep neural network with multiple layers and nonlinear activation units like ReLU.

**Descriptions:**

In this exercise, we will implement the building blocks of a deep neural network and combine them to form a complete model. The process involves initializing parameters, performing forward and backward propagation, computing the cost function, and updating the parameters through gradient descent. The objective is to enable the construction of deep neural networks that can perform complex tasks such as image classification.



**Model:**

1. **Initialization:**

* 2-layer Neural Network: We begin by initializing the parameters for a simple 2-layer neural network. This involves setting up the weights and biases for the network.

- initialize\_parameters(): Initializes parameters for a 2-layer neural network.

* L-layer Neural Network: Extending the initialization process to a deeper network, where multiple layers are initialized using a loop.

- initialize\_parameters\_deep(): Initializes parameters for a deep neural network with L layers.

2. **Forward Propagation Module:**

* Linear Forward: Performs the linear part of the forward propagation for a single layer.

- linear\_forward(): Computes the linear transformation for a given layer.

* Linear-Activation Forward: Combines the linear transformation with an activation function (ReLU or Sigmoid) for a given layer.

- linear\_activation\_forward(): Performs the forward pass with an activation function.

* L-Layer Model: Implements forward propagation for the entire network, iterating through all layers.

- L\_model\_forward(): Executes the forward pass across the entire deep network.

3. **Cost Function:**

* Computes the cost (loss) of the network after forward propagation.

- compute\_cost(): Calculates the cross-entropy loss for the output.

4. **Backward Propagation Module:**

* Linear Backward: Computes the gradient of the loss with respect to the linear transformation for a single layer.

- linear\_backward(): Computes gradients for the linear part of a layer.

* Linear-Activation Backward: Combines the linear backward step with the backward pass of the activation function.

- linear\_activation\_backward(): Performs the backward pass with an activation function.

* L-Model Backward: Executes backward propagation for the entire network.

- L\_model\_backward(): Computes gradients for the entire network.

* Update Parameters: Updates the model's parameters using the gradients computed during backpropagation.

- update\_parameters(): Applies gradient descent to update the weights and biases.

**GitHub Link:**

* [**https://github.com/LohithR22/Fundamentals\_of\_DeepLearning**](https://github.com/LohithR22/Fundamentals_of_DeepLearning)