DEEP LEARNING LIBRARY FROM SCRATCH

EKLAVYA MENTORSHIP PROGRAMME
at

SOCIETY OF ROBOTICS AND AUTOMATION,
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
MUMBAI
OCTOBER 2021

ACKNOWLEDGMENT

We would like to express our gratitude and appreciation towards our seniors at SRA, VJTI, who were our mentors.and guides for this project for giving us the opportunity to work under their guidance. It was their help that made it possible for us to complete this project. It was really amazing to work with all of them, and due to this we have gained a lot of knowledge in this domain

Special thanks to our mentors:
- Kush Kothari
- Aman Chhaparia

TEAM :
RISHABH BALI
+91 9821872320

ADITYA MHATRE
+91 9137392261

AAYUSH MEHTA
+91 9967760001

TABLE OF CONTENTS

1. PROJECT OVERVIEW

1.1 DESCRIPTION OF USE CASE AND PROJECT :

Deep learning can be considered as a subset of machine learning. It is a field that is based on learning and improving on its own by examining computer algorithms. Deep learning works with artificial neural networks consisting of many layers.
This project, which is creating a Deep Learning Library from scratch, can be further implemented in various kinds of projects that involve Deep Learning. Which include, but are not limited to applications in Image, Natural Language and Speech processing, among others.

1.2 TECHNOLOGY USED

1. **NUMPY LIBRARY USING PYTHON**
   In this project, the numpy library of Python was widely used. It is used to work with arrays, and it also has its applications in the field of computational science. We apply this library to formulate each and every equation and operation that is required to run a deep neural network for this deep learning library

2. **GOOGLE COLABORATORY**

1.3 BRIEF IDEA

The aim of the project is to create a deep learning library, from scratch, which is able to perform various simple deep learning tasks, which is dependant of the learning rate and other parameters that are provided to define the Deep Neural Network that is hence generated.

2. INTRODUCTION

2.1 GENERAL :

Our general idea behind this project was to build a deep learning library in python, similar to other widely used libraries for deep learning, like Tensorflow, or PyTorch, and learn the basic algorithms that drive it. For this, we have used different concepts like forward propagation, backward propagation and optimization by gradient descent, to achieve a simple deep neural network.

2.2 BASIC PROJECT DOMAINS :

- Linear Algebra
- Deep Learning

2.3 THEORY

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes.
So, in a Neural Network, there are a lot of Input Layers, Output Layers, and Hidden Layers (Which are actually neurons, but we can call them layers as they're collection of multiple neurons working together) Also, the hidden layers and input layers are interdependent on each other. This forms the basis for Deep Learning.

There are different types of Neural Networks

- Standard Neural Networks
  The normal type of Neural Networks with groups of multiple neurons at each layer, it's only processed in the forward direction. It consists of 3 layers, Input, Hidden and Output It is used for Simpler problems.
- Convolutional Neural Networks
  Convolutional Neural Networks work on the principle of Convolution Kernels and is most commonly used for Image processing
  The process of convolution is applied in these types of networks, which states that for a mathematical operation on two functions (f and g) that produces a third function ($f*g$) that expresses how the shape of one is modified by the other.

- Recurring Neural Networks
  RNNs are Neural Networks which are a derivative of Standard Neural Networks in which a looping constraint on the hidden layer of the SNN turns it into an RNN. This type of NN is used for one-dimensional temporal sequence or multidimensional sequence data, like Audio, Video.

  In Deep Learning, a neural network with multiple layers, or a deep neural network is applied. A deep learning process is gauged by both the performance of the neural network, as well as the amount of data involved in the process.
  With the same amount of data used for training, the performance of the Neural Network rises with Learning Algorithm or type of NN used

3. METHODS AND STAGES OF PROGRESS

3.1 APPROACH

The approach of the project is to basically create a deep learning library, as stated before. The aim of the project was to implement various deep learning algorithms, in order to drive a deep neural network and hence,create a deep learning library, which is modular,and driven on user input so that it can be applied for various deep learning processes, and to train and test it against a model.

3.2 CONCEPTS USED

a) **Logistic regression :-**

In statistics, a Logistic Model is used to model the probability of an event being either pass/fail , true/false, right/wrong, basically two opposite conditions. Logistic regression is used to estimate the parameters of a logistic model, which finally gives an output as 1 or 0, as a binary dependent variable.

In this project, we apply the Linear Regression model.
Where, if output prediction ŷ = P(y=1 | x) , and x is given $x \in R^{nx}$ also, by Parameters, $w \in R^{nx}$ and $b \in R$, final equation is given by

$$\hat{y} = w^T x + b$$

b) **Loss Function and Loss** : -
In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the objective function.
We may seek to maximize or minimize the objective function, meaning that we are searching for a candidate solution that has the highest or lowest score respectively.
Typically, with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply "loss."

Loss function is defined so as to see how good the output ŷ compared to output label y.
Given by,

$$L(\hat{y}, y) = -(y log \hat{y} + (1 - y) log(1 - \hat{y}))$$

c) **Cost Function** : -
It is a function that measures the performance of a Machine Learning model for given data. Cost Function quantifies the error between predicted values and expected values.
To train a single example, given by

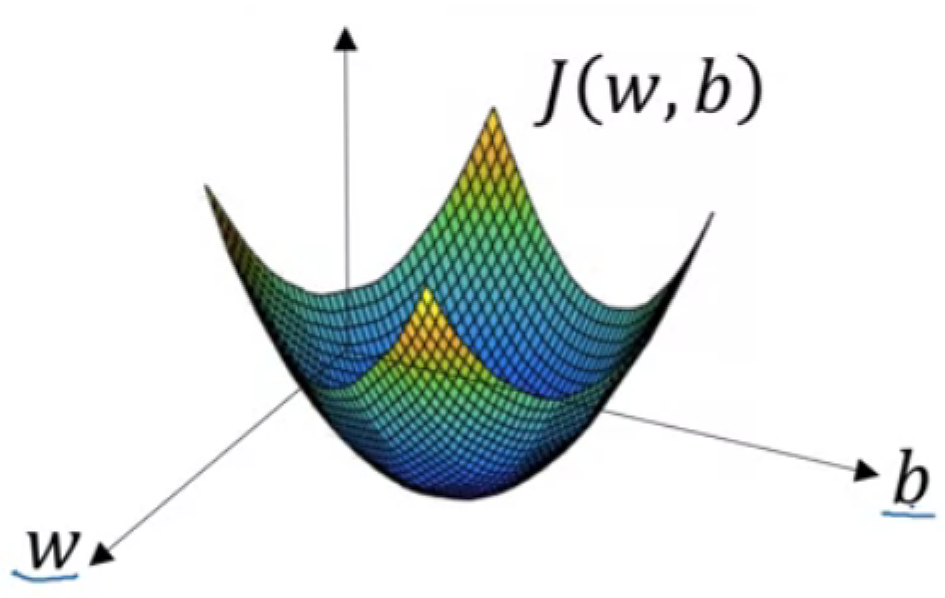
$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} log(\hat{y}^{(i)}) + (1 - y^{(i)} log(1 - \hat{y}^{(i)}))$$

d) **Gradient Descent** : -

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point, because this is the direction of steepest descent.

So,in this case, the loss function J(w,b) is minimized

$$L(\hat{y}, y) = -(y log \hat{y} + (1-y) log(1-\hat{y}))$$



So, values of w , b that minimize the Loss Function J(w,b)

$$w : = w - \alpha(\frac{\partial J(w, b)}{dw})$$

$$b : = b - \alpha(\frac{\partial J(w, b)}{dw})$$

where α is the learning rate.

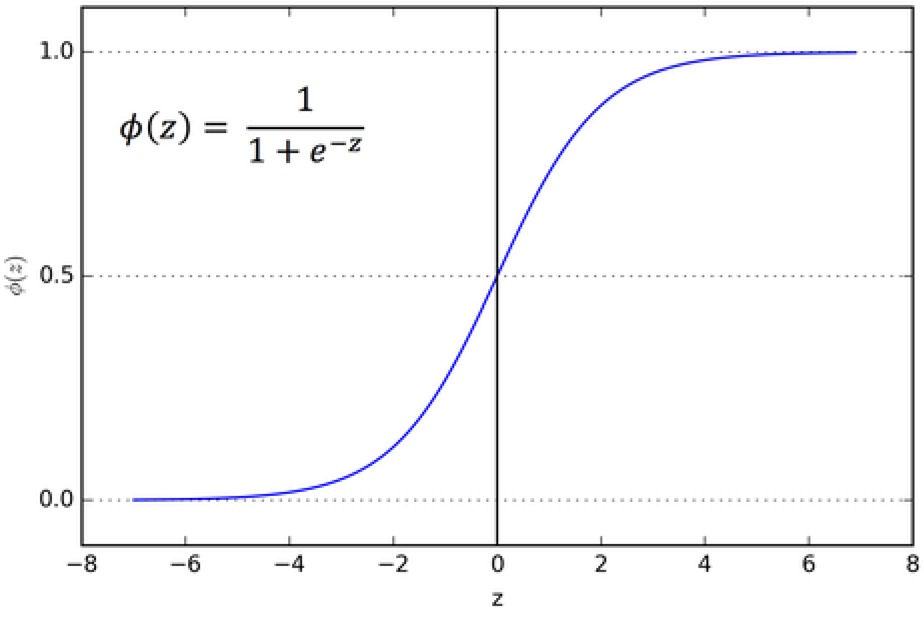
e) **Vectorization**

Vectorization in python is used to speed up the processing of a code, and also increases the efficiency of the code, while reducing the amount of code to be written. It is used in python codes which require a lot of processing power (in CPU and/or GPU) , hence reducing the time of processing.

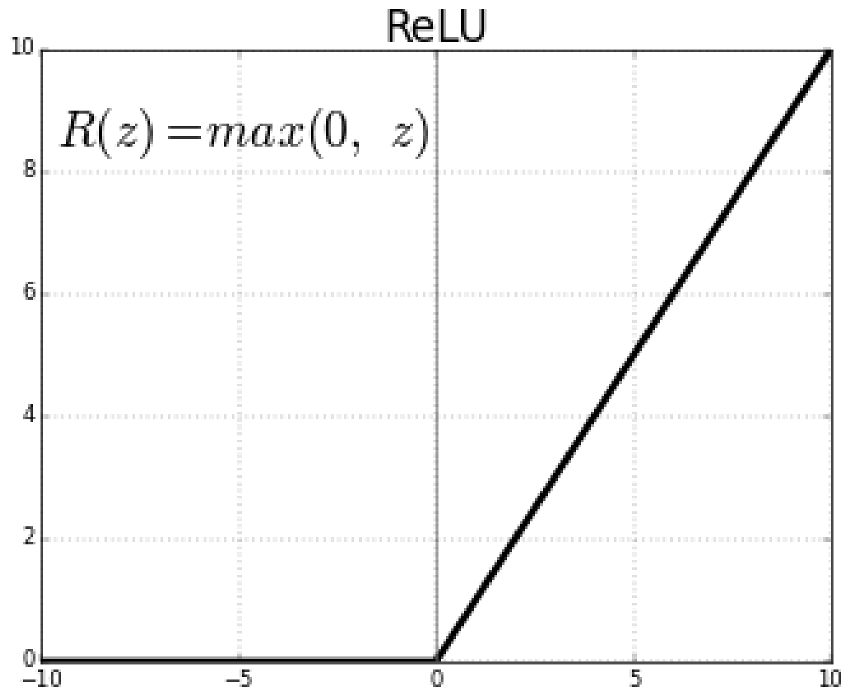
f) **Activation Functions**

Activation functions usually define the output of the node for a given set of inputs for the same. It takes the output signal from the previous neuron, and converts it into some form that can be taken to the next neuron.
We have used two Activation Functions

1) Sigmoid
   Sigmoid Activation Function is used for binary classification
   And is given by



2) ReLU
   ReLU or Rectified Linear Unit, is defined as f(x) = max(0,x)
   It is widely used, but the Disadvantage of ReLU is slope is 0 when x is -ve.
   ReLU is given by



g) **Forward Propagation**
   Forward propagation refers to the calculation and storage of intermediate variables (including outputs) for a neural network in order from the input layer to the output layer.
   In this, the input data is fed in the forward direction through the network.
   Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.
   At each neuron in a hidden or output layer, the processing happens in two steps
   1)Pre-activation : It is a weighted sum of inputs i.e., the linear transformation of weights w.r.t to inputs available. Based on this aggregated sum and activation function the neuron makes a decision whether to pass this information further or not.

2)Activation : this weighted sum of inputs is passed on to the Activation Functions

h) **Backward Propagation**
Backpropagation refers to the method of calculating the gradient of neural network parameters. In short, the method traverses the network in reverse order, from the output to the input layer, according to the chain rule from calculus. The algorithm stores any intermediate variables (partial derivatives) required while calculating the gradient with respect to some parameters. Back-propagation is the essence of neural net training. It is the practice of fine-tuning the weights of a neural net based on the loss obtained in the previous iteration. Proper tuning of the weights ensures lower error rates, making the model reliable by increasing its generalization.

3.3 DEEP LEARNING LIBRARY

A deep learning model in python usually consists of these major components in form of classes

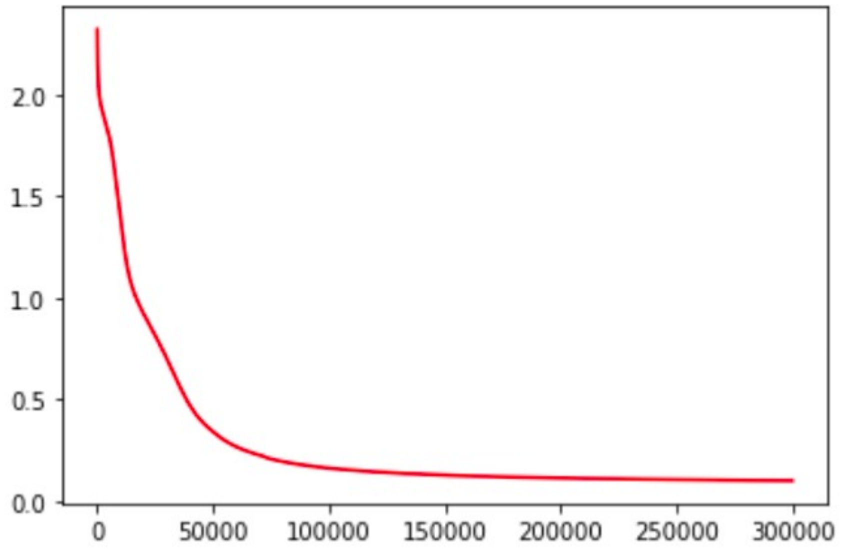
a) Initialization of Parameters
b) Initialization of Activation Functions
c) Forward Propagation
d) Derivative of Activation Functions
e) Backward Propagation

This is then run into a superclass which stores the entire model and is used to train the same with the help of datasets

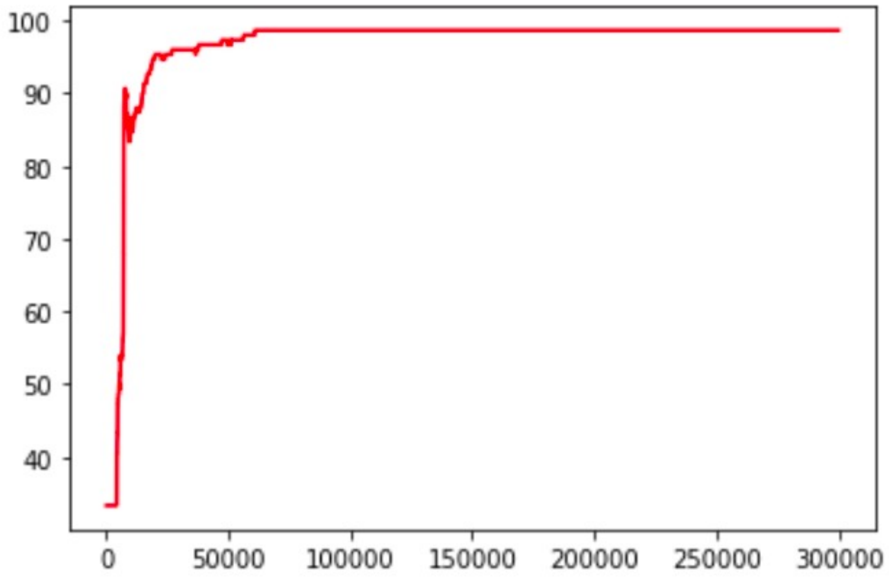
4. PROJECT CONCLUSION AND FUTURE WORK

4.1 RESULTS

After training it with the IRIS dataset, with 30000 iterations, and 6 layers, learning rate of 0.0005



This graph represents the cost (Y-Axis) vs the number of iterations.(X-Axis)
The cost value keeps on decreasing with the amount of iterations that take place.



This graph represents the accuracy (in%) (Y-axis) vs Number of Iterations (X-Axis).

4.2 ACHIEVEMENTS AND CONCLUSION

We have built a Deep Learning library which contains algorithms for Binary and Multi-class classification. We also trained a model on the iris dataset, which worked to an accuracy of over 90%.

4.3 FUTURE ASPECTS AND FURTHER DEVELOPMENT

A deep learning library has a lot of applications in various fields. This library can be further developed to add more functions and different activation functions, like softmax, leaky ReLU, tanh, among various others. The model can be further modified and enhanced to decrease loss, trained over a bigger dataset to increase the efficiency of the model.

More Future Implementations of this library can include ,
-In the short term, we can add classes for normalization and regularization. Addition of forward propagation and backward propagation classes to train RNN models.
- Addition of classes for LSTM and GRU blocks. (Near Future)
- Addition of algorithms to support Computer Vision models.
- Addition of more Machine Learning algorithms
- Addition of support for Linear Regression (Near Future)
- Addition of AdamProp and RMSProp Algorithms for optimization (Near Future)

In the future, our goal is to include algorithms to facilitate Image Recognition, Machine Translation and Natural Language Processing

5. REFERENCES

LINKS TO THE VARIOUS SOURCES AND COURSES FOLLOWED :

- Coursera course by Andrew NG on Deep Learning and Neural Networks
  https://www.coursera.org/learn/neural-networks-deep-learning/home/welcome
- Coursera course by Andrew NG on Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization
  https://www.coursera.org/learn/deep-neural-network?specialization=deep-learning
- GitHub Repository
  https://github.com/rishabh2002-lang/ARA
- On Gradient Descent Optimization
  https://ruder.io/optimizing-gradient-descent/
- Essence of Linear Algebra by 3Blue1Brown
  https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab
- Basics of Neural Networks by 3Blue1Brown
  https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
- Basics of Implementing Deep Learning Libraries
  https://towardsdatascience.com/on-implementing-deep-learning-library-from-scratch-in-python-c93c942710a8
- Deep Learning
  https://www.ibm.com/cloud/learn/deep-learning
  https://www.deeplearningbook.org/
- Deep Learning Libraries
  https://www.sabinasz.net/deep-learning-from-scratch-theory-and-implementation/
- Iris Dataset
  https://archive.ics.uci.edu/ml/datasets/iris
  https://www.kaggle.com/uciml/iris