G H Raisoni College of Engineering and Management, Wagholi, Pune

Department of AI & AIML Engineering

LAB MANUAL

Subject: COMPUTER VISION AND DEEP LEARNING

Class: TY (Computer – 2020 Course)

Examination Scheme:
Practical (INT): 25 Marks
Practical (EXT): 25 Marks
Total: 50 Marks

Institute Vision & Mission

Vision

Our vision is to achieve excellent standards of quality education by keeping pace with rapidly changing technologies. Our continuous endeavor is to create manpower of global standards with capabilities of accepting new challenges.

Mission

Our efforts are dedicated towards imparting quality and value based education to raise the satisfaction level of our students. Ourstrength is directed towards creating competent professionals. Our endeavor is to provide all possible and sustainable support to promote research & development activities

Department Vision & Mission

VISION:

To produce global standard ethical professionals, innovators, andentrepreneurs having strong knowledge and urge to learn latest technologies in the field of computer engineering.

MISSION:

The department continuously strives

M1: Pursue excellence in Computer Engineering, able to adapt changing technologies through effective Teaching- Learning Process.

M2: Develop competent professionals for global market with the spirit of self-study, team work, innovation and ethics.

M3: Promote continuous learning, entrepreneurial skills and research.

ASSIGNMENT NO. 1:

TITLE:

Write a program Logistic Regression with Neural Network mindset

PROBLEM STATEMENT:

You are given a dataset (”data.h5”) containing:

- a training set of m_train images labeled as cat (y=1) or non-cat (y=0)
- a test set of m_test images labeled as cat or non-cat
- each image is of shape (num_px, num_px, 3) where 3 is for the 3 channels (RGB).Thus, each image is square (height = num_px) and (width = num_px).

OBJECTIVE:

- Build the general architecture of a learning algorithm, including:
  - Initializing parameters
  - Calculating the cost function and its gradient
  - Using an optimization algorithm (gradient descent)
- Gather all three functions above into a main model function, in the right order.

THEORY:

Welcome to the first (required) programming exercise of the deep learning specializa- tion. In this notebook you will build your first image recognition algorithm. You will build a cat classifier that recognizes cats with 70% accuracy!

Figure 1.2.1 cat

As you keep learning new techniques you will increase it to 80+ % accuracy on cat vs. non-cat datasets. By completing this assignment you will:

- Work with logistic regression in a way that builds intuition relevant to

neural net-works.

- Learn how to minimize the cost function.
- Understand how derivatives of the cost are used to update parameters.

Take your time to complete this assignment and make sure you get the expected outputs when working through the different exercises. In some code blocks, you will find a "#GRADED FUNCTION: functionName" comment. Please do not modify these comments. After you are done, submit your work and check your results. You need to score 70% to pass. Good luck :) !

Packages:

- numpy is the fundamental package for scientific computing with Python.
- h5py is a common package to interact with a dataset that is stored on an H5 file.
- matplotlib is a famous library to plot graphs in Python.
- PIL are used here to test your model with your own picture at the end.

General Architecture of the learning algorithm:

It's time to design a simple algorithm to distinguish cat images from non-cat images. You will build a Logistic Regression, using a Neural Network mindset. The following Figure explains why **Logistic Regression is actually a very simple Neural Net-**

work!

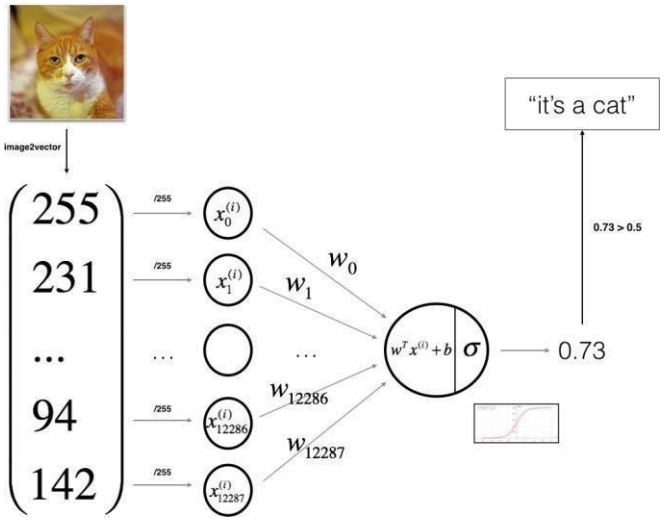


Figure 1.2.2 Principle of Logistic Regression

Mathematical expression of the algorithm:For one example$x_{(i)}$:

$$z_{(i)} = w_T x_{(i)} + b \tag{1.2.1}$$

$$\hat{y}^{(i)} = a^{(i)} = sigmoid(z^{(i)}) \quad (1.2.2)$$

$$L(a^{(i)}, y^{(i)}) = -y^{(i)} \log(a^{(i)}) - (1 - y^{(i)}) \log(1 - a^{(i)}) \quad (1.2.3)$$

The cost is then computed by summing over all training examples:

$$J = \frac{1}{m} \sum_{i=1}^{m} L(a_{(i)}, y_{(i)}) \quad (1.2.4)$$

Key steps: In this exercise, you will carry out the following steps:

- Initialize the parameters of the model
- Learn the parameters for the model by minimizing the cost
- Use the learned parameters to make predictions (on the test set)
- Analyse the results and conclude

Building the parts of our algorithm:

The main steps for building a Neural Network are:

- Define the model structure (such as number of input features)
- Initialize the model's parameters
- Loop:
  - Calculate current loss (forward propagation)
  - Calculate current gradient (backward propagation)
  - Update parameters (gradient descent)

You often build 1-3 separately and integrate them into one function we call model().

Optimization:

- You have initialized your parameters.
- You are also able to compute a cost function and its gradient.
- Now, you want to update the parameters using gradient descent.

Merge all functions into a model:

You will now see how the overall model is structured by putting together all thebuilding blocks (functions implemented in the previous parts) together, in the right order.

Exercise: Implement the model function. Use the following notation:

- Y_prediction for your predictions on the test set

- Y_prediction_train for your predictions on the train set

w, costs, grads for the outputs of optimize()

What to remember from this assignment:

1. Preprocessing the dataset is important.
2. You implemented each function separately: initialize(), propagate(), optimize().Then you built a model().
3. Tuning the learning rate (which is an example of a ”hyperparameter”) can make abig difference to the algorithm. You will see more examples of this later in this course!

SAMPLE CODE:

PLATFORM REQUIRED:

Operating System: Windows

Software or Tools: GOOGLE COLAP

CONCLUSION:

Hence we studied the concept of Logistic Regression with Neural Network mindset along with accuracy is 70% for the dataset CatNoncat.

ASSIGNMENT NO. 2:

TITLE:

Implement Planner data classification with one hidden layer

PROBLEM STATEMENT:

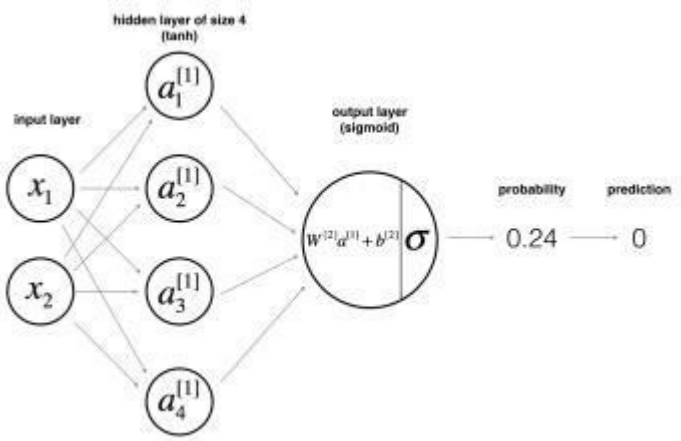
- Develop an intuition of back-propagation and see it work on data
- Recognize that the more hidden layers you have the more complex structure you could capture.
- Build all the helper functions to implement a full model with one hidden layer.
- Implement a 2-class classification neural network with a single hidden layer
- Use units with a non-linear activation function, such as tanh
- Compute the cross entropy loss
- Implement forward and backward propagation

OBJECTIVE:

1. Define the neural network structure (# of input units, # of hidden units, etc).
2. Initialize the model's parameters
3. Loop: – Implement forward propagation – Compute loss – Implement backward propagation to get the gradients – Update parameters (gradient descent)

THEORY:

Neural Network model Logistic regression did not work well on the "flower dataset". You are going to train a Neural Network with a single hidden layer. Here is our model: Figure Neural Network Model



Packages:

- numpy is the fundamental package for scientific computing with Python.
- sklearn provides simple and efficient tools for data mining and data analysis.
- matplotlib is a famous library to plot graphs in Python.
- testCases provides some test examples to assess the correctness of your functions

- planar_utils provide various useful functions used in this assignment

Dataset:

Dataset First, let's get the dataset you will work on. The following code will load a "flower" 2-class dataset into variables X and Y. X, Y = load_planar_dataset() Visualize the dataset using matplotlib. The data looks like a "flower" with some red (label y=0) and some blue (y=1) points. Your goal is to build a model to fit this data. # Visualize the data: plt.scatter(X[0, :], X[1, :], c=Y, s=40, cmap=plt.cm.Spectral);

Defining the neural network structure:

- Define three variables: • n_x: the size of the input layer
- n_h: the size of the hidden layer (set this to 4)
- n_y: the size of the output layer

Initialize the model's parameters:

- Make sure your parameters' sizes are right. Refer to the neural network figure above if needed.
- You will initialize the weights matrices with random values. – Use: np.random.randn(a,b) * 0.01 to randomly initialize a matrix of shape (a,b).
- You will initialize the bias vectors as zeros. – Use: np.zeros((a,b)) to initialize a matrix of shape (a,b) with zeros

The Loop:

- Look above at the mathematical representation of your classifier.
- You can use the function 'sigmoid()'. It is built-in (imported) in the notebook.
- You can use the function 'np.tanh()'. It is part of the numpy library. • The steps you have to implement are: 1 Retrieve each parameter from the dictionary "parameters" (which is the output of 'initialize_parameters()') by using 'parameters[".."]'. 2 Implement Forward Propagation. Compute Z [1], A[1], Z[2] and A[2] (the vector of all your predictions on all the examples in the training set).
- Values needed in the backpropagation are store

SAMPLE CODE:

PLATFORM REQUIRED:

Operating System: Windows

Software or Tools: GOOGLE COLAP

CONCLUSION:

Hence we studied the concept of Planner data classification with one hidden layer with 91% accuracy.

ASSIGNMENT NO. 3:

TITLE:

Implement Neural Network with one hidden layer

PROBLEM STATEMENT:

- Develop an intuition of forward-propagation and see it work on data
- Recognize that the one hidden layers you have the more complex structure you could capture.
- Build all the helper functions to implement a full model with one hidden layer.
- Use units with a non-linear activation function, such as tanh

OBJECTIVE:

1. Define the neural network structure (# of input units, # of hidden units, etc).
2. Initialize the model's parameters.

THEORY:

Different activation function used in neural network.

Activation Function :

The main objective of the activation function is to perform a mapping of a weighted sum upon the output. The transformation function comprises of activation functions such as tanh, ReLU, sigmoid, etc.

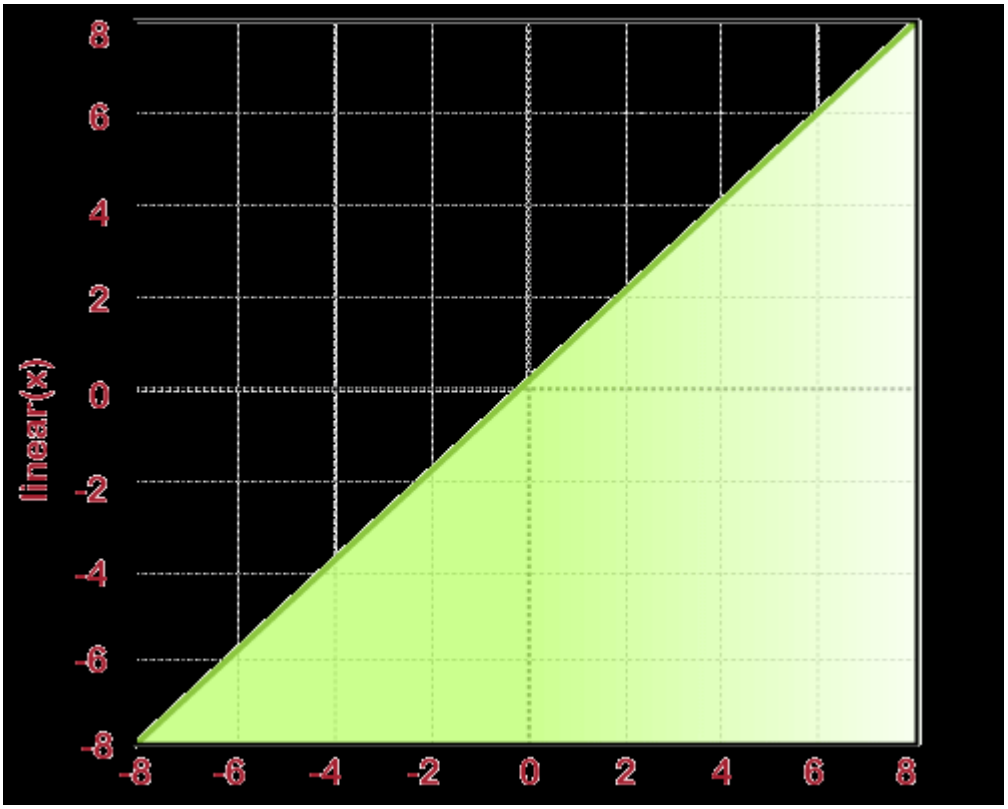
The activation function is categorized into two main parts:

Linear Activation Function

Non-Linear Activation Function

Linear Activation Function

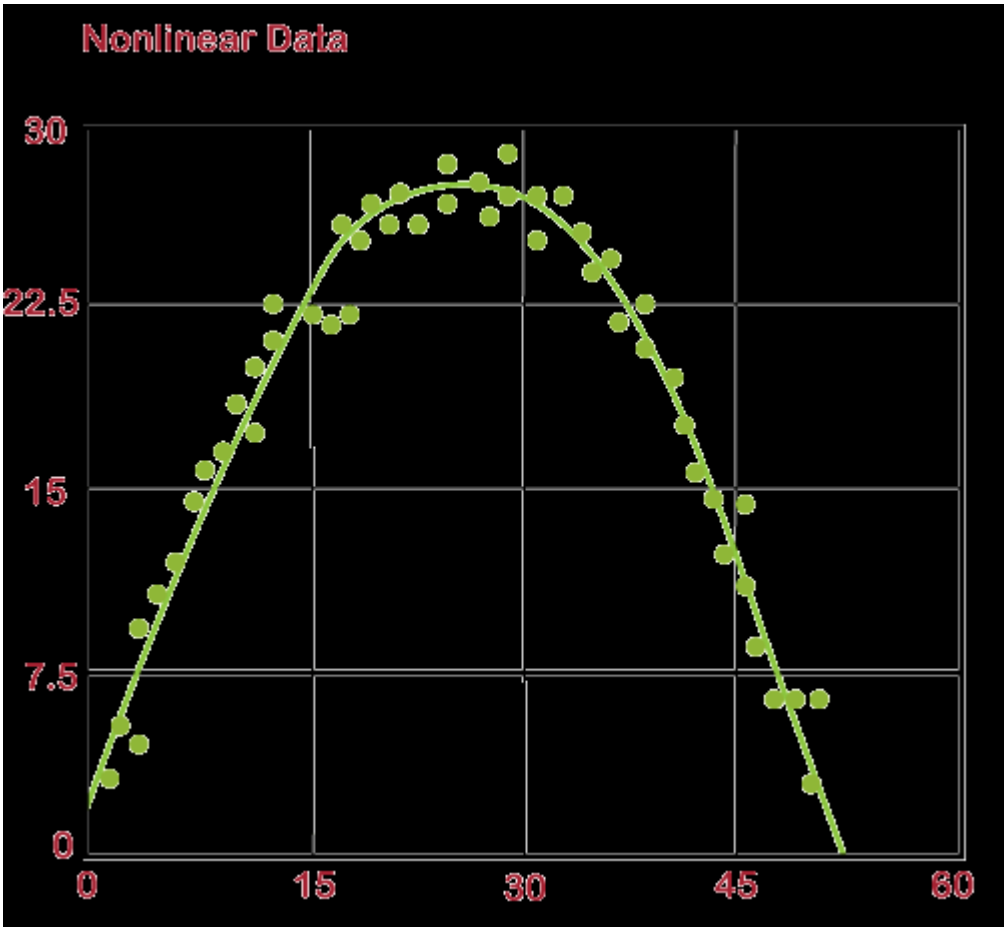
In the linear activation function, the output of functions is not restricted in between any range. Its range is specified from -infinity to infinity. For each individual neuron, the inputs get multiplied with the weight of each respective neuron, which in turn leads to the creation of output signal proportional to the input. If all the input layers are linear in nature, then the final activation of the last layer will actually be the linear function of the initial layer's input.



Artificial Neural Networks

Non- linear function

These are one of the most widely used activation function. It helps the model in generalizing and adapting any sort of data in order to perform correct differentiation among the output. It solves the following problems faced by linear activation functions:



Since the non-linear function comes up with derivative functions, so the problems related to backpropagation has been successfully solved.

For the creation of deep neural networks, it permits the stacking up of several layers of the neurons.

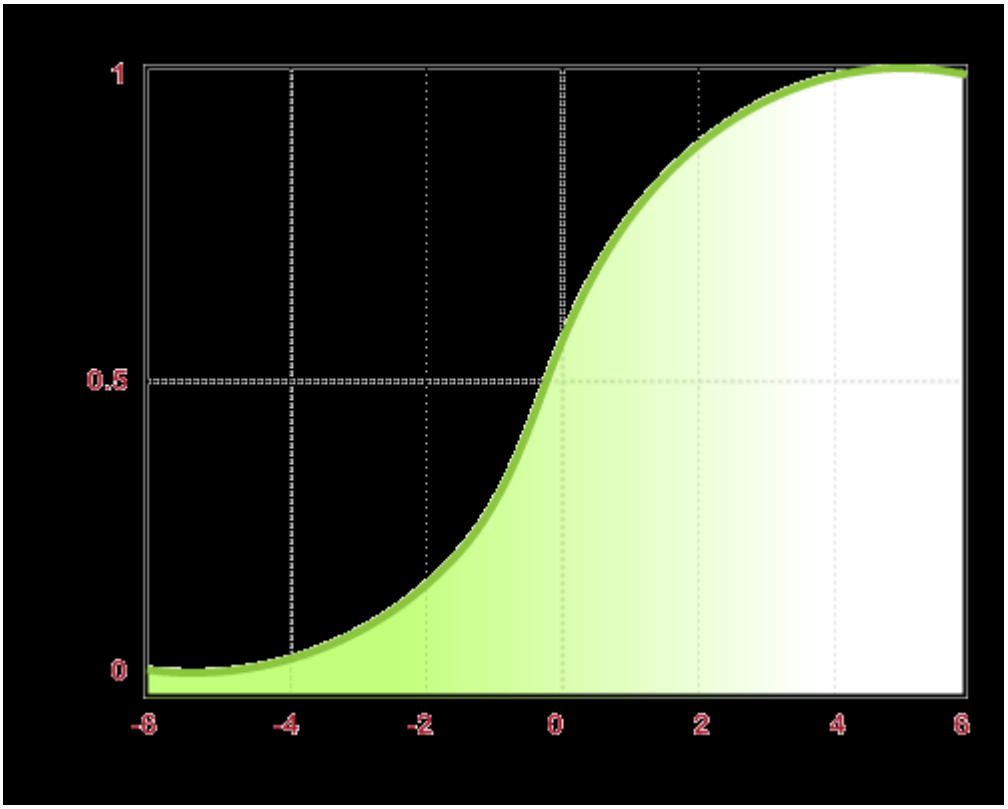
Artificial Neural Networks

The non-linear activation function is further divided into the following parts:

Sigmoid or Logistic Activation Function

It provides a smooth gradient by preventing sudden jumps in the output values. It has an output value range between 0 and 1 that helps in the normalization of each neuron's output. For X, if it has a value above 2 or below -2, then the values of y will be much steeper. In simple language, it means that even a small change in the X can bring a lot of change in Y.

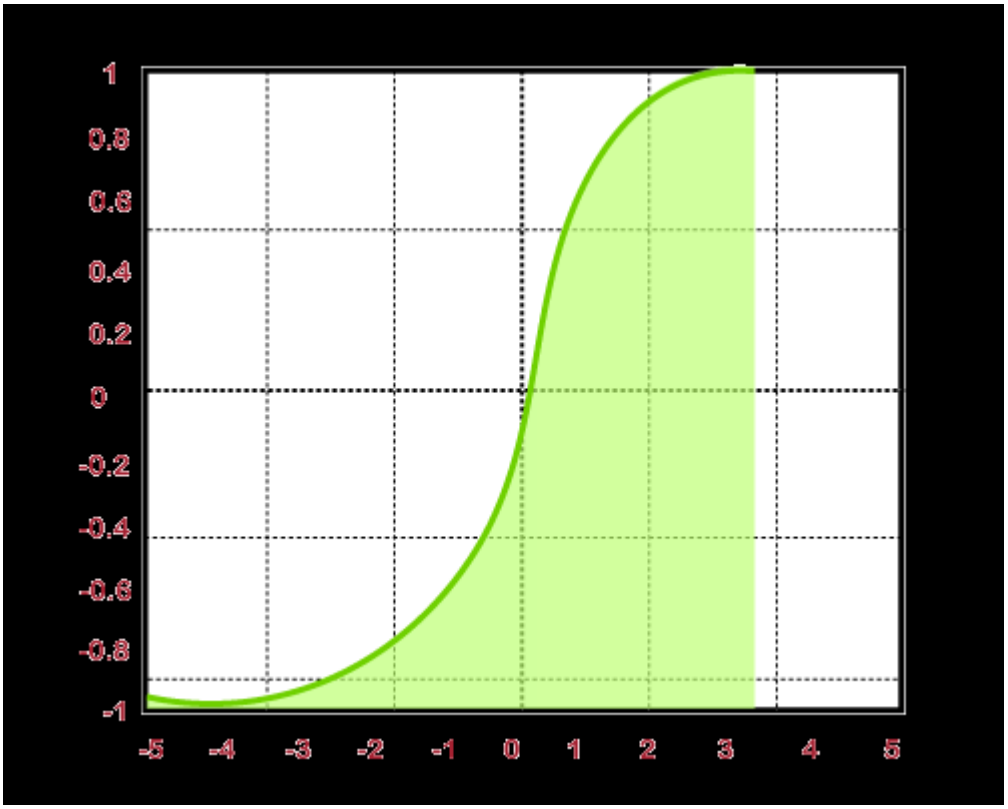
It's value ranges between 0 and 1 due to which it is highly preferred by binary classification whose result is either 0 or 1.



Artificial Neural Networks

Tanh or Hyperbolic Tangent Activation Function

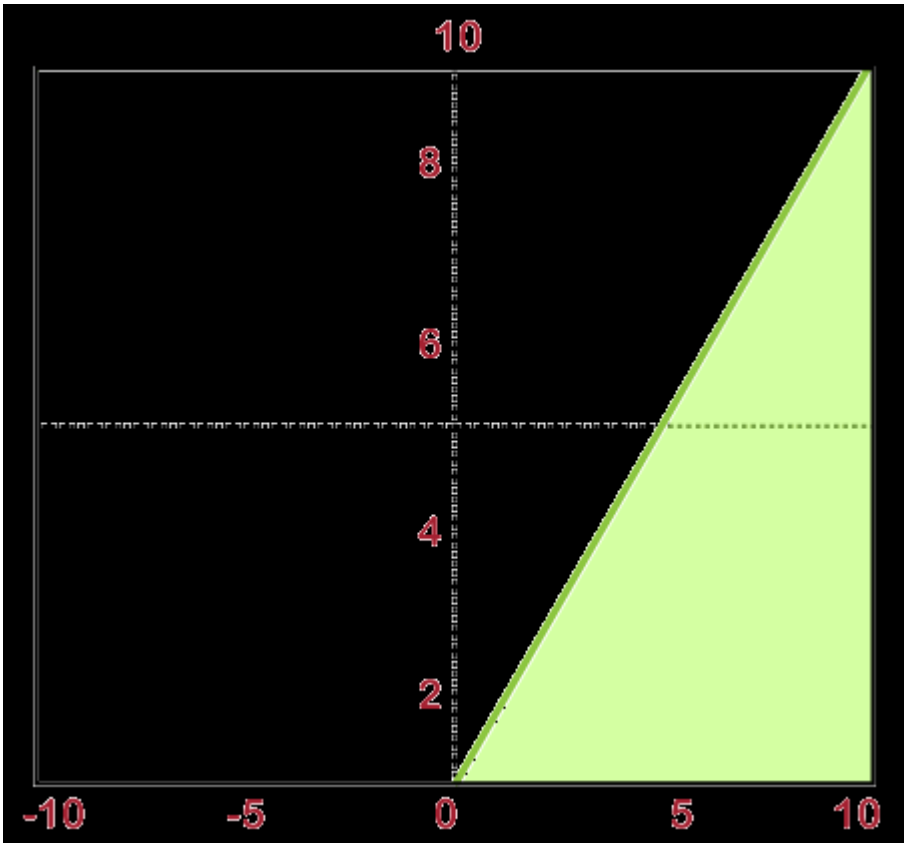
The tanh activation function works much better than that of the sigmoid function, or simply we can say it is an advanced version of the sigmoid activation function. Since it has a value range between -1 to 1, so it is utilized by the hidden layers in the neural network, and because of this reason, it has made the process of learning much easier.



Artificial Neural Networks

ReLU(Rectified Linear Unit) Activation Function

ReLU is one of the most widely used activation function by the hidden layer in the neural network. Its value ranges from 0 to infinity. It clearly helps in solving out the problem of backpropagation. It tends out to be more expensive than the sigmoid, as well as the tanh activation function. It allows only a few neurons to get activated at a particular instance that leads to effectual as well as easier computations.



Artificial Neural Networks

Softmax Function

It is one of a kind of sigmoid function whereby solving the problems of classifications. It is mainly used to handle multiple classes for which it squeezes the output of each class between 0 and 1, followed by dividing it by the sum of outputs. This kind of function is specially used by the classifier in the output layer.

Packages:

- numpy is the fundamental package for scientific computing with Python.
- sklearn provides simple and efficient tools for data mining and data analysis.
- matplotlib is a famous library to plot graphs in Python.
- planar_utils provide various useful functions used in this assignment

Defining the neural network structure:

- Define three variables: • n_x: the size of the input layer
- n_h: the size of the hidden layer (set this to 1)
- n_y: the size of the output layer

SAMPLE CODE:

PLATFORM REQUIRED:

Operating System: Windows

Software or Tools: GOOGLE COLAP

CONCLUSION:

Hence we studied the concept of Planner data classification with one hidden layer with 67.5 % accuracy.

ASSIGNMENT NO. 4

TITLE:

To build deep neural network step by step.

PROBLEM STATEMENT:

- Develop an intuition of the over all structure of a neural network.
- Write functions (e.g. forward propagation, backward propagation, logistic loss, etc...) that would help you decompose your code and ease the process of building a neural network.
- Initialize/update parameters according to your desired structure

OBJECTIVE:

- Use non-linear units like ReLU to improve your model
- Build a deeper neural network (with more than 1 hidden layer)
- Implement an easy-to-use neural network class

THEORY:

1. Initialize the parameters for a two-layer network and for an L-layer neural network.
2. Implement the forward propagation module
- Complete the LINEAR part of a layer's forward propagation step (resulting in Z [l]).
- We give you the ACTIVATION function (relu/sigmoid).
- Combine the previous two steps into a new [LINEAR->ACTIVATION] forward function.
- Stack the [LINEAR->RELU] forward function L-1 time (for layers 1 through L-1) and add a [LINEAR->SIGMOID] at the end (for the final layer L). This gives you a new L_model_forward function.

3. Compute the loss.
4. Implement the backward propagation module (denoted in red in the figure below).
- Complete the LINEAR part of a layer's backward propagation step.
- We give you the gradient of the ACTIVATE function (relu_backward/sigmoid_backward)
- Combine the previous two steps into a new [LINEAR->ACTIVATION] backward function.
- Stack [LINEAR->RELU] backward L-1 times and add [LINEAR->SIGMOID] backward in a new L_model_backward function

Packages:

- numpy is the main package for scientific computing with Python.
- matplotlib is a library to plot graphs in Python.
- dnn_utils provides some necessary functions for this notebook.

- testCases provides some test cases to assess the correctness of your functions
- np.random.seed(1) is used to keep all the random function calls

PLATFORM REQUIRED:

Operating System: Windows

Software or Tools: GOOGLE COLAP

CONCLUSION:

Hence we studied the steps required to implement Deep neural network.

ASSIGNMENT NO. 5

TITLE:

Implement the concept of regularization, gradient checking and optimization in convolutional model: step by step

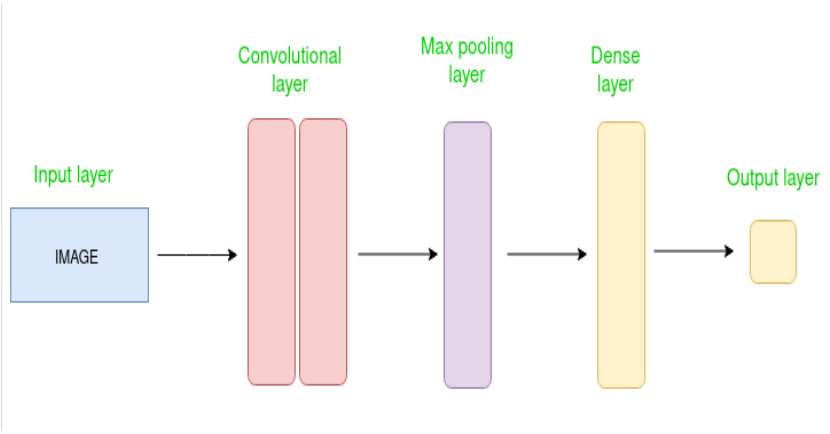
PROBLEM STATEMENT:

- Develop an intuition of the over all structure of a neural network.
- Write functions (e.g. forward propagation, backward propagation, logistic loss, etc...) that would help you decompose your code and ease the process of building a neural network.
- Initialize/update parameters according to your desired structure

OBJECTIVE:

- Takes an input volume
- Applies a filter at every position of the input
- Outputs another volume (usually of different size)

THEORY:



Input Layer:

- It's the layer in which we give input to our model.
- In CNN, Generally, the input will be an image or a sequence of images.
- This layer holds the raw input of the image with width 32, height 32, and depth 3.

Convolutional Layer:

- This is the layer, which is used to extract the feature from the input dataset.
- It applies a set of learnable filters known as the kernels to the input images.
- The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape.
- it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch.
- The output of this layer is referred ad feature maps.

Activation Layer:

- By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network.

- it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU: max(0, x), Tanh, Leaky RELU, etc.

Pooling layer:
- This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting.
- Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

Flattening:
The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

Fully connected layer:
It takes the input from the previous layer and computes the final classification or regression task.

Output Layer:
The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

Procedure:
1. import the necessary libraries
2. set the parameter
3. define the kernel
4. Load the image and plot it.
5. Reformat the image
6. Apply convolution layer operation and plot the output image.
7. Apply activation layer operation and plot the output image.
8. Apply pooling layer operation and plot the output image.

Packages:

- numpy is the fundamental package for scientific computing with Python.
- • matplotlib is a library to plot graphs in Python.
- • np.random.seed(1) is used to keep all the random function calls consistent. It will help us grade your work.

PLATFORM REQUIRED:

Operating System: Windows

Software or Tools: GOOGLE COLAP

CONCLUSION:

Hence we studied the steps required to implement CNN.

ASSIGNMENT NO. 6

TITLE:

Implementation and Demonstration of the usages of the YOLO algorithm for the detection of the surrounding cars and the objects.

PROBLEM STATEMENT:

- You are working on a self-driving car. As a critical component of this project, you'd
- like to first build a car detection system.
- To collect data, we have mounted a camera to
- the hood (meaning the front) of the car, which takes pictures of the road ahead every few
- seconds while you drive around.

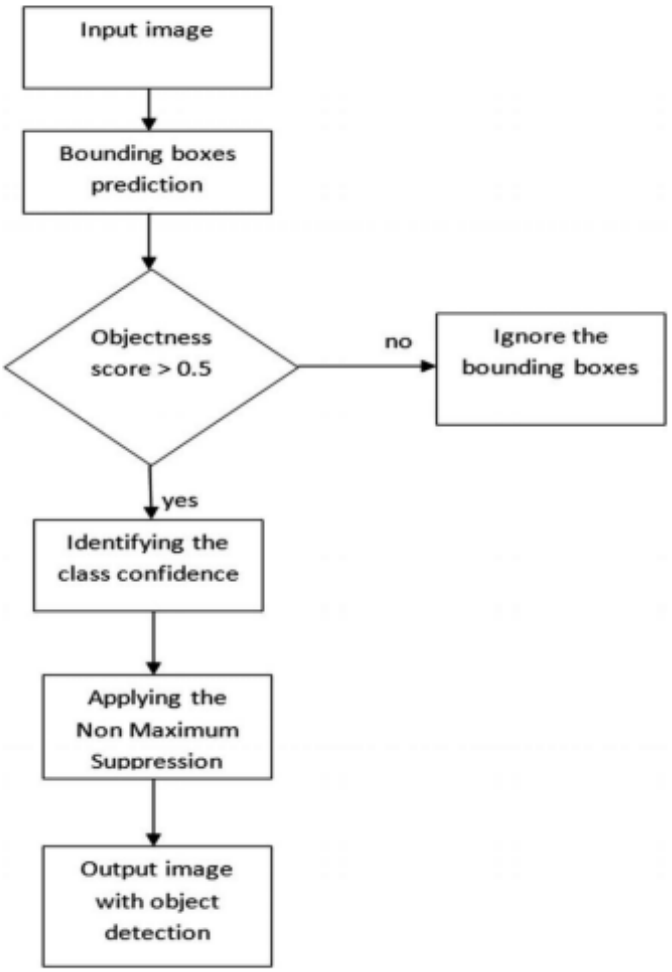
OBJECTIVE:

- YOLO algorithm Is capable of accurate vehicle detection with near real-time performance in the variety of different driving conditions I

THEORY:

YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region.

These bounding boxes are weighted by the predicted probabilities. YOLO Algorithm: The YOLO framework (You Only Look Once) on the other hand, deals with object detection in a different way. It takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The biggest advantage of using YOLO is its superb speed – it's incredibly fast and can process 45 frames per second.YOLO also understands generalized object representation. This is one of the best algorithms for object detection and has shown a comparatively similar performance to the R-CNN algorithms



Object Detection Flowchart: I

.

How the YOLO algorithm works

YOLO algorithm works using the following three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)

Residual blocks

First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids.

Image Source

In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

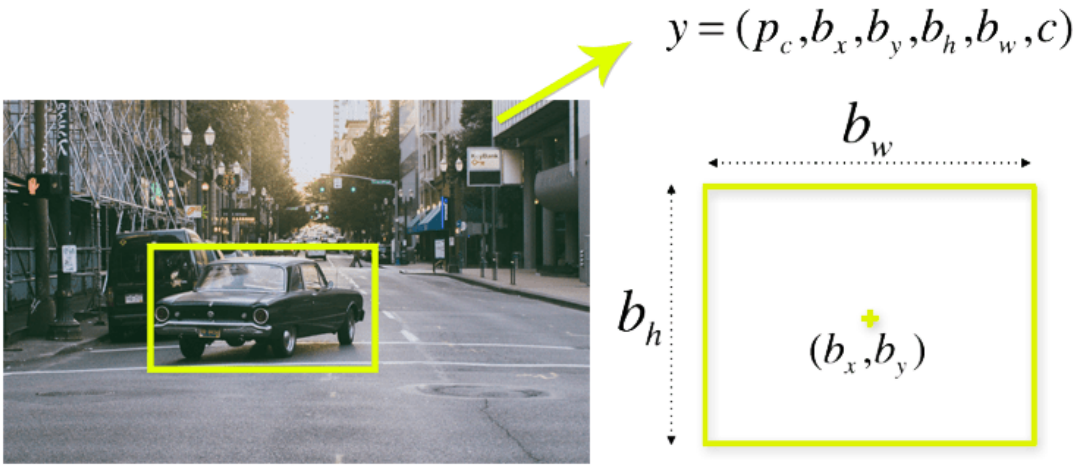
Bounding box regression

A bounding box is an outline that highlights an object in an image.

Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)
- Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
- Bounding box center (bx,by)

The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.



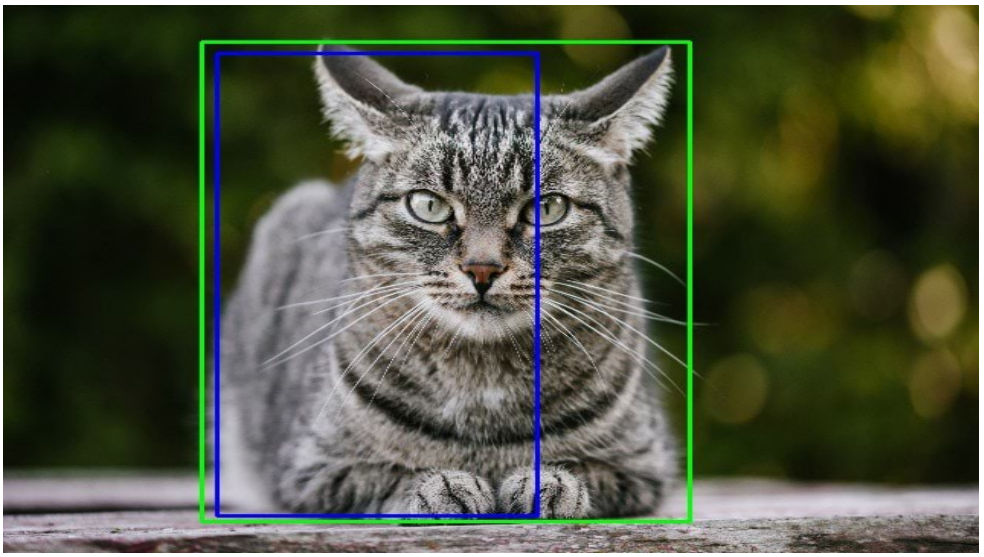
YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

Intersection over union (IOU)

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

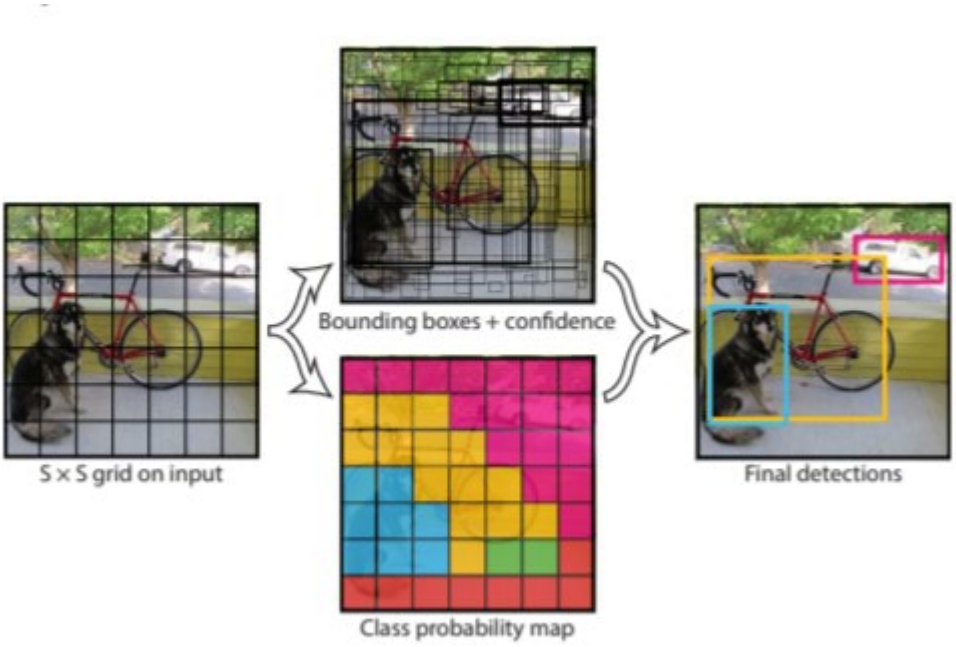
The following image provides a simple example of how IOU works.



In the image above, there are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal.

Combination of the three techniques

The following image shows how the three techniques are applied to produce the final detection results.



First, the image is divided into grid cells. Each grid cell forecasts B bounding boxes and provides their confidence scores. The cells predict the class probabilities to establish the class of each object.For example, we can notice at least three classes of objects: a car, a dog, and a bicycle. All the predictions are made simultaneously using a single convolutional neural network. Intersection over union ensures that the predicted bounding boxes are equal to the real boxes of the objects. This phenomenon eliminates unnecessary bounding boxes that do not meet the characteristics of the objects (like height and width). The final detection will consist of unique bounding boxes that fit the objects perfectly.For example, the car is surrounded by the pink bounding box while the bicycle is surrounded by the yellow bounding box. The dog has been highlighted using the blue bounding box.

Applications of YOLO

YOLO algorithm can be applied in the following fields:

- **Autonomous driving:** YOLO algorithm can be used in autonomous cars to detect objects around cars such as vehicles, people, and parking signals. Object detection in autonomous cars is done to avoid collision since no human driver is controlling the car.
- **Wildlife:** This algorithm is used to detect various types of animals in forests. This type of detection is used by wildlife rangers and journalists to identify animals in videos (both recorded and real-time) and images. Some of the animals that can be detected include giraffes, elephants, and bears.

- **Security:** YOLO can also be used in security systems to enforce security in an area. Let's assume that people have been restricted from passing through a certain area for security reasons. If someone passes through the restricted area, the YOLO algorithm will detect him/her, which will require the security personnel to take further action.

Packages:

- numpy is the fundamental package for scientific computing with Python.
- matplotlib is a library to plot graphs in Python.
- np.random.seed(1) is used to keep all the random function calls consistent. It will help us grade your work.

PLATFORM REQUIRED:

Operating System: Windows

Software or Tools: GOOGLE COLAP

CONCLUSION:

Hence we studied the steps required to implement CNN.