Welcome to Deep Learning Online Bootcamp

Day 6 - Binary Classification

dφ

Democratizing Data Science Learning

Learning Objectives

Classification and Sigmoid function

Activation Functions

Methods of building Deep Learning models

Error Functions and Optimizers

In the previous unit, you learned about the working of a Neural Network on a conceptual level.

In this module, you'll understand how all of those concepts are brought into practice by understanding some crucial components of a Neural Network.

Finally, you'll build a model to detect whether a person has heart disease or not.

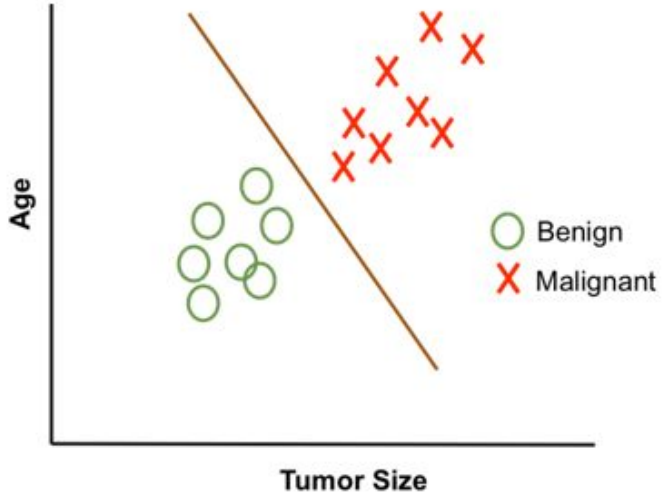
Classification
and
Importance of Sigmoid Function + Recap of Logistic Regression

What is Classification?

Let's learn with some examples:

- In **Classification we** classify the outcome into two classes (Eg: yes or no)
- **Examples:**
  - Predict whether a transaction is fraud or not fraud
  - Predict whether to give loan or not
  - Predict whether to give college admission or not
  - Note: There can be classification problems with more than 2 classes and it is called as multi-classification

Feature	Tumor Age and Tumor Size
Label	Tumor (Benign or Malignant)
Goal/ Aim	We want to predict whether a tumor is benign or malignant from the given age and tumor size



What is Multi-Classification?

It is as simple as dividing waste into 4 categories - plastic, glass, metal, paper (we will talk about multi-classification later units)

Logistic Regression

- Logistic Regression is one of the basic and popular algorithms to solve a binary classification problems

- For each input, logistic regression outputs a probability that this input belongs to the 2 classes
  - Set a probability threshold boundary and that determines which class the input belongs to

- Binary classification problems (2 classes):
  - Emails (Spam / Not Spam)
  - Credit Card Transactions (Fraudulent / Not Fraudulent)
  - Loan Default (Yes / No)

Logistic Regression

Now, you may ask why don't we use Linear Regression? Why do we need a new algorithm?

Well, you would find all the answers in the video in the next slides.

The video in the next slide is a must watch, the instructor has brilliantly explained about logistic regression!

Must Watch Understanding Logistic Regression

Linear Regression vs Logistic

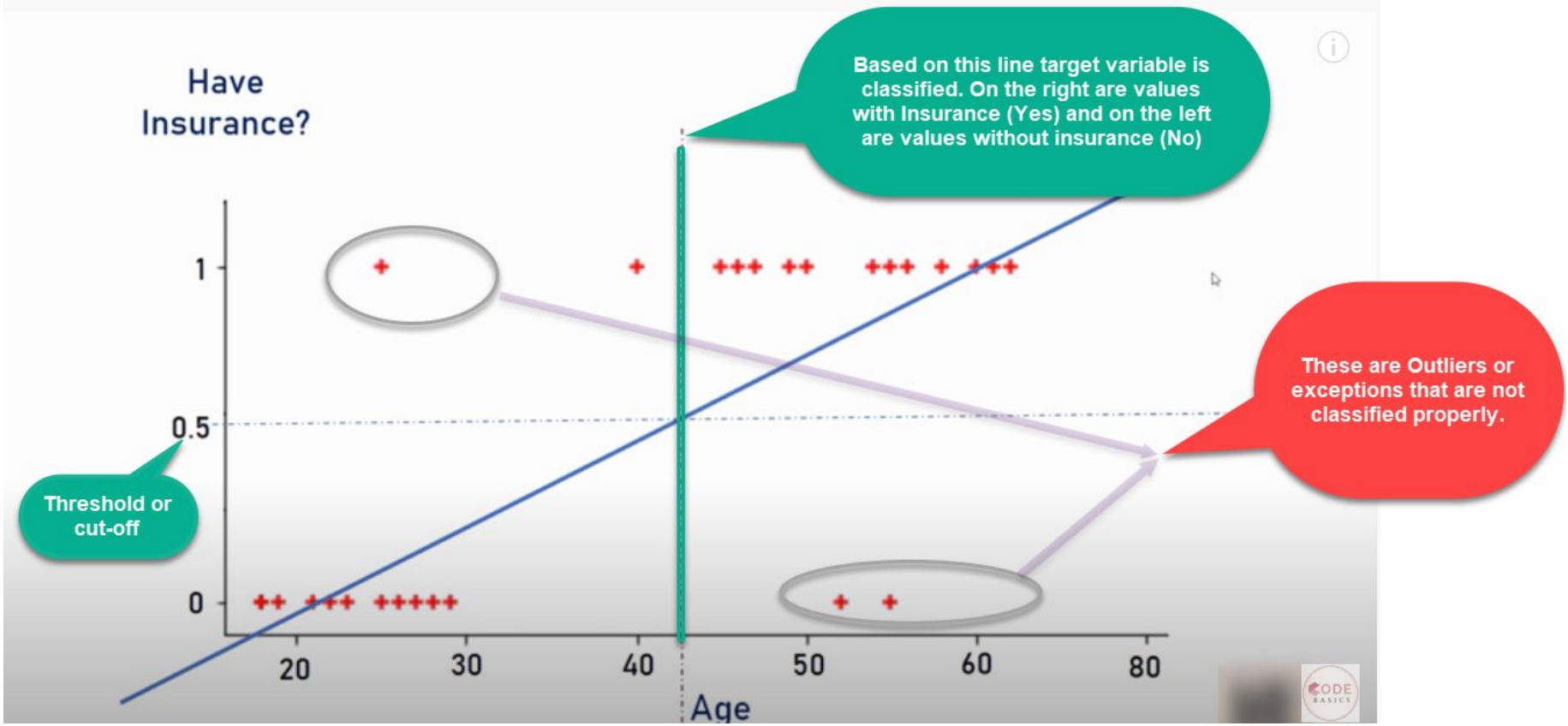
- Linear regression is used to solve regression problems with continuous values
- Logistic regression is used to solve classification problems with discrete categories
  - Binary classification (Classes 0 and 1)
  - Examples:
    - Emails (Spam / Not Spam)
    - Credit Card Transactions (Fraudulent / Not Fraudulent)
    - Loan Default (Yes / No)

Linear Regression vs Logistic

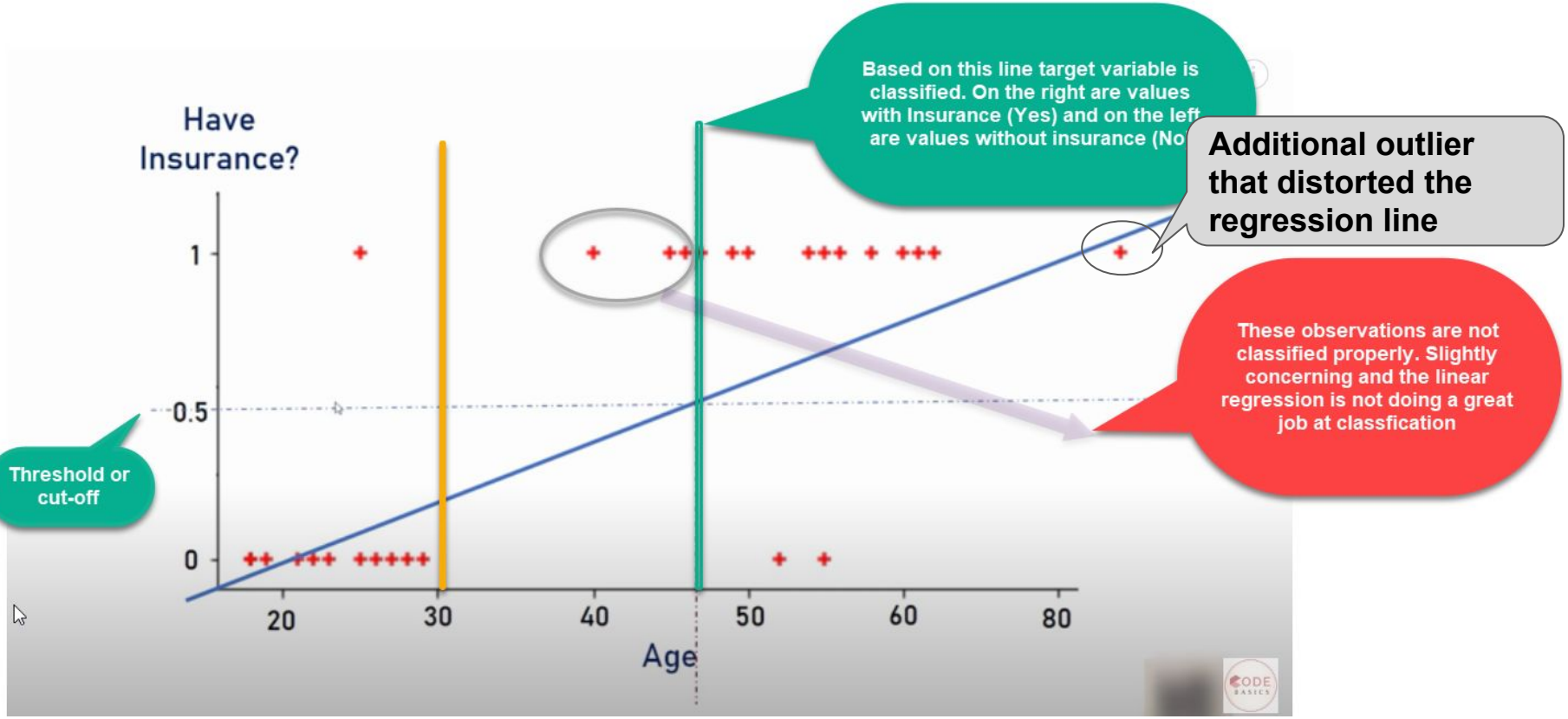
- Let's say a data scientist named John want to predict that whether a customer will buy insurance or not
- Remember that linear regression is used to predict a continuous value where the output (y) may vary between +∞ (positive infinity) to -∞ (negative infinity) whereas in this case, the target variable (y) takes only two discrete values, 0 (No insurance) and 1 (Yes, got the insurance).
- John's decides to extend the concepts of linear regression to fulfil his requirement. One approach is to take the output of linear regression and map it between 0 and 1, if the resultant output is below a certain threshold (say 0.5), classify it as No (didn't buy the insurance) whereas if the resultant output is above a certain threshold, classify it as bought the insurance (yes)

Linear Regression vs Logistic

- We then plot a simple linear regression line and set the threshold as 0.5
  - Negative class (Insurance = No)– Age on the left side
  - Positive class (Insurance = Yes) – Age on the right side



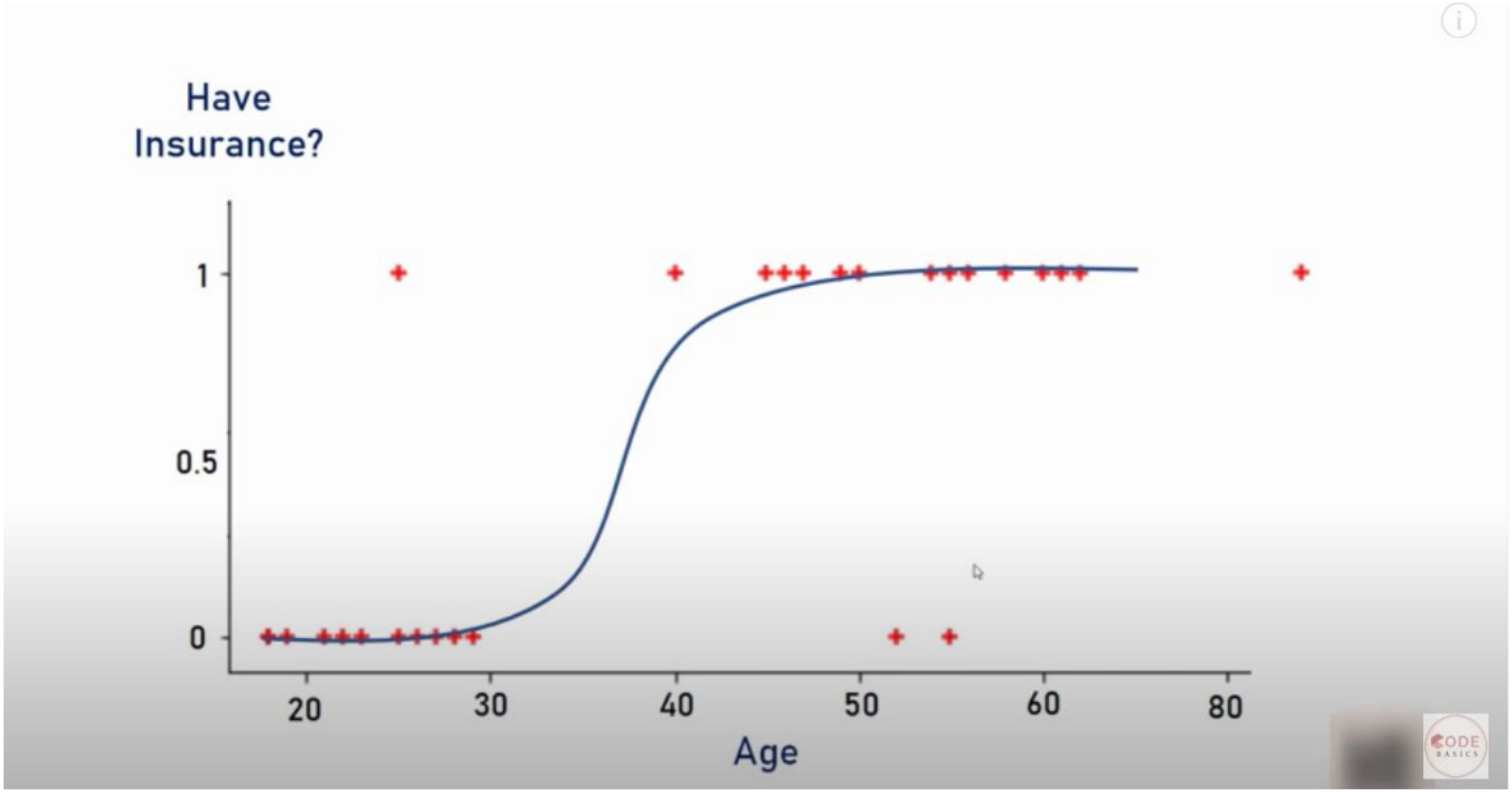
Imagine there is an outlier to towards right



- As we can see outlier in the data and will distort the whole linear regression line.
- Clearly the line is unable to differentiate the classes with the linear line fit
- The line should have been at the vertical yellow line which is able to divide the positive and negative classes i.e yes or no for insurance

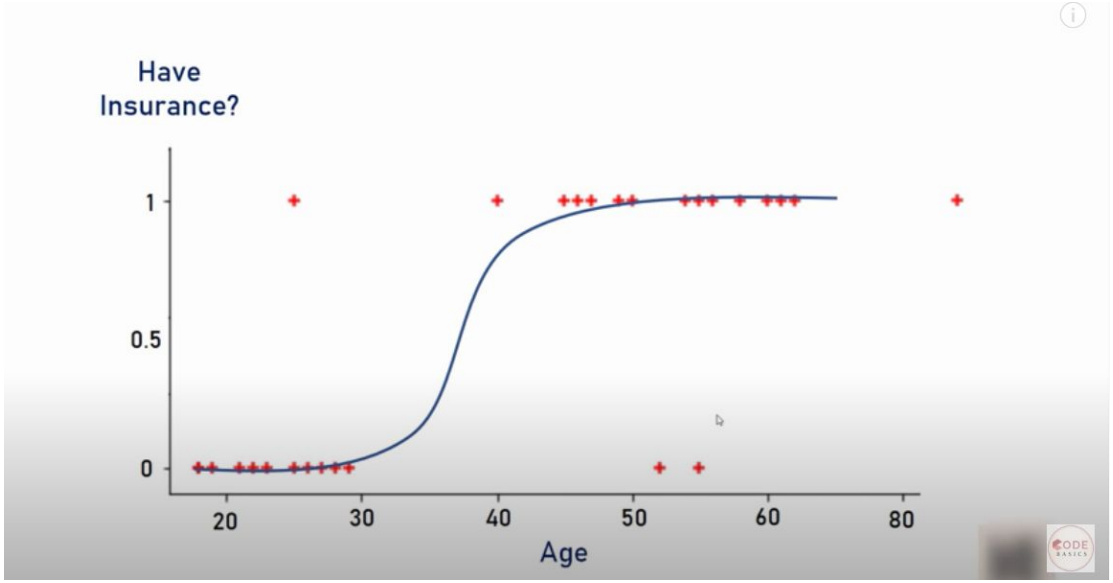
Happy John! (Data Scientist)

- **Well, life would be much simpler if we had a algorithm that would fit the points like below right? It is a much better fit compared to regression line!**



Solution

- Solution – Transform linear regression to a logistic regression curve
- Logistic regression is a Sigmoid function
- Now what does this sigmoid function do?
  - Sigmoid function takes in any real value and gives a output probability between 0 and 1



What are we doing in Logistic Regression?

We will use the real-valued output obtained from a linear regression model between 0 and 1 and classify a new example based on a threshold value. The function used to perform this mapping is the **sigmoid function**

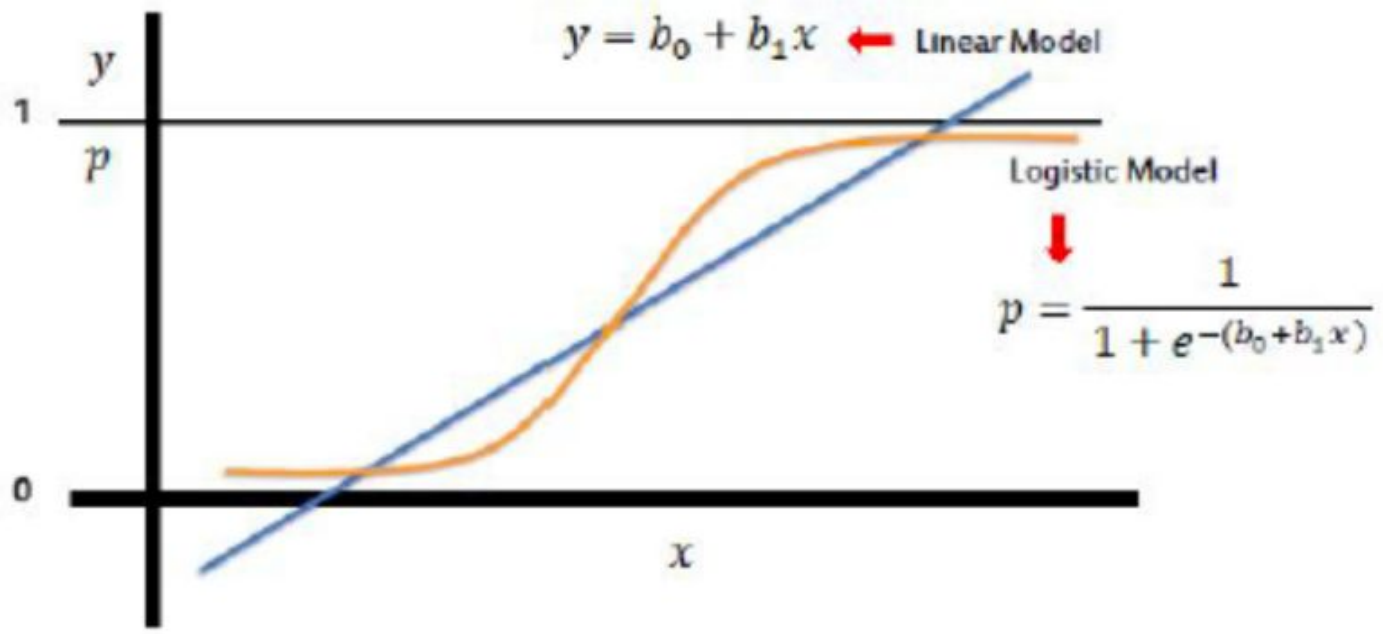
The Sigmoid Function is represented by the formula:

$$f_{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

There's no need to go into the depth of how we obtained this formula right now.

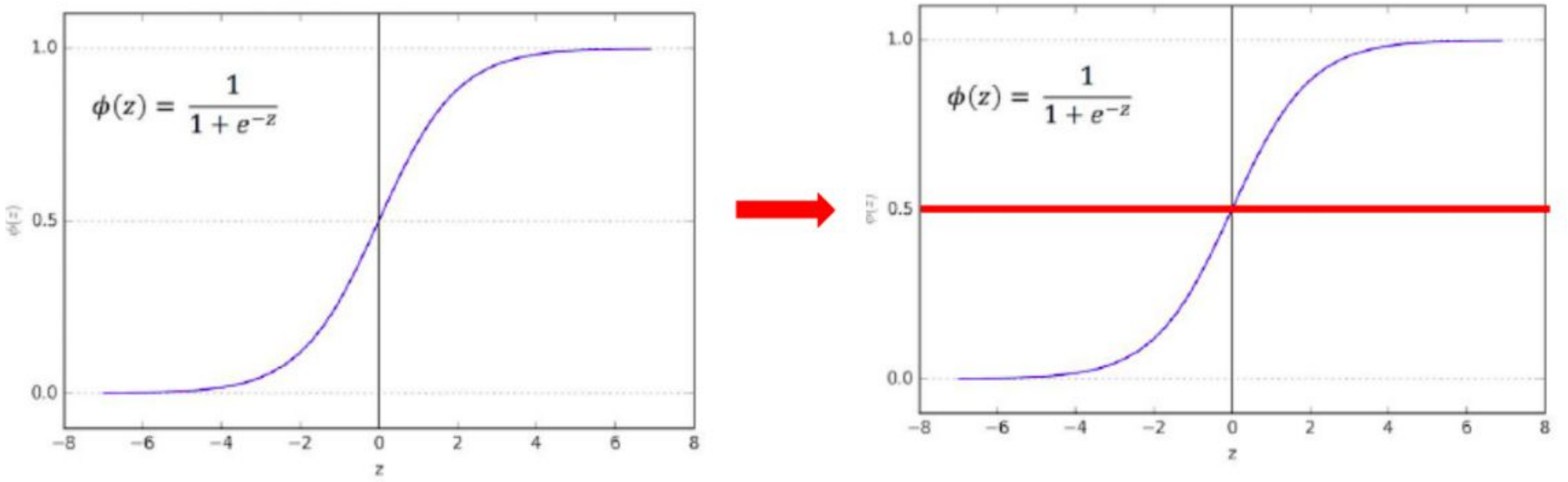
Sigmoid Function (Logistic Function/ Logit)

- Take linear regression function and put it into the Sigmoid function
- Sigmoid function outputs probability between 0 and 1



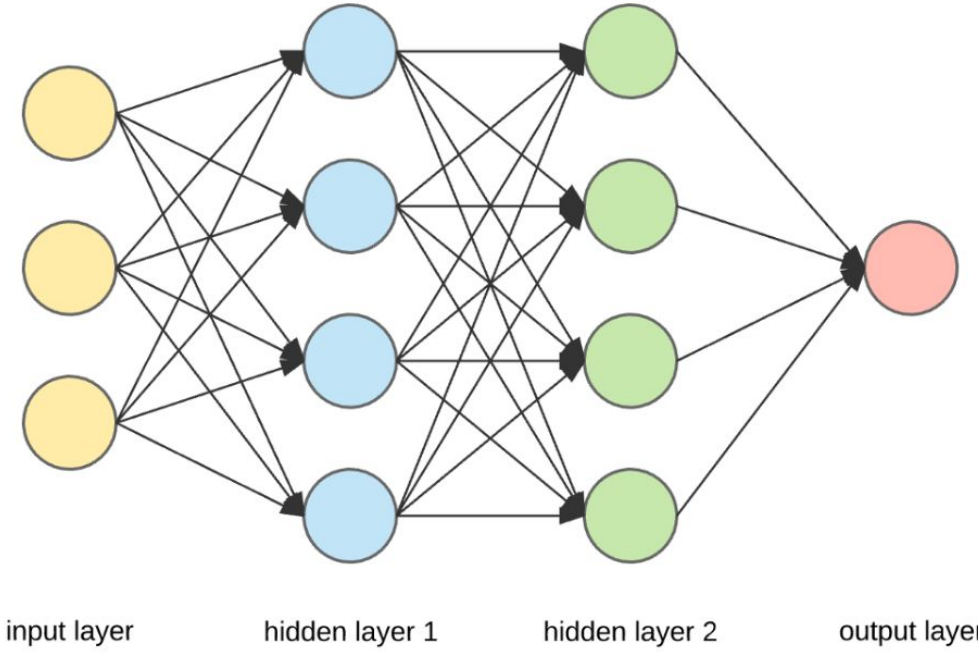
Sigmoid Function (Logistic Function/ Logit)

- Sigmoid function outputs probability between 0 and 1 (y axis)
- Default probability threshold is set at 0.5 typically
  - Class 0 – Below 0.5
  - Class 1 – Above 0.5



Activation Functions

Here’s what a Neural Network looks like. Let’s dive a bit into what exactly happens inside each neuron.



Artificial Neuron – A Quick Recap

Artificial neurons are the basic building blocks of a neural network. It can be considered as a computational unit that takes some inputs, applies some transformation on the input and fires the output. Below are typical steps for computation inside the neuron.

1. An artificial neuron takes the inputs and their respective weights.
2. It then applies dot products between input values & its weights and sums them up.
3. Finally, it applies activation function on above summation and fires the output

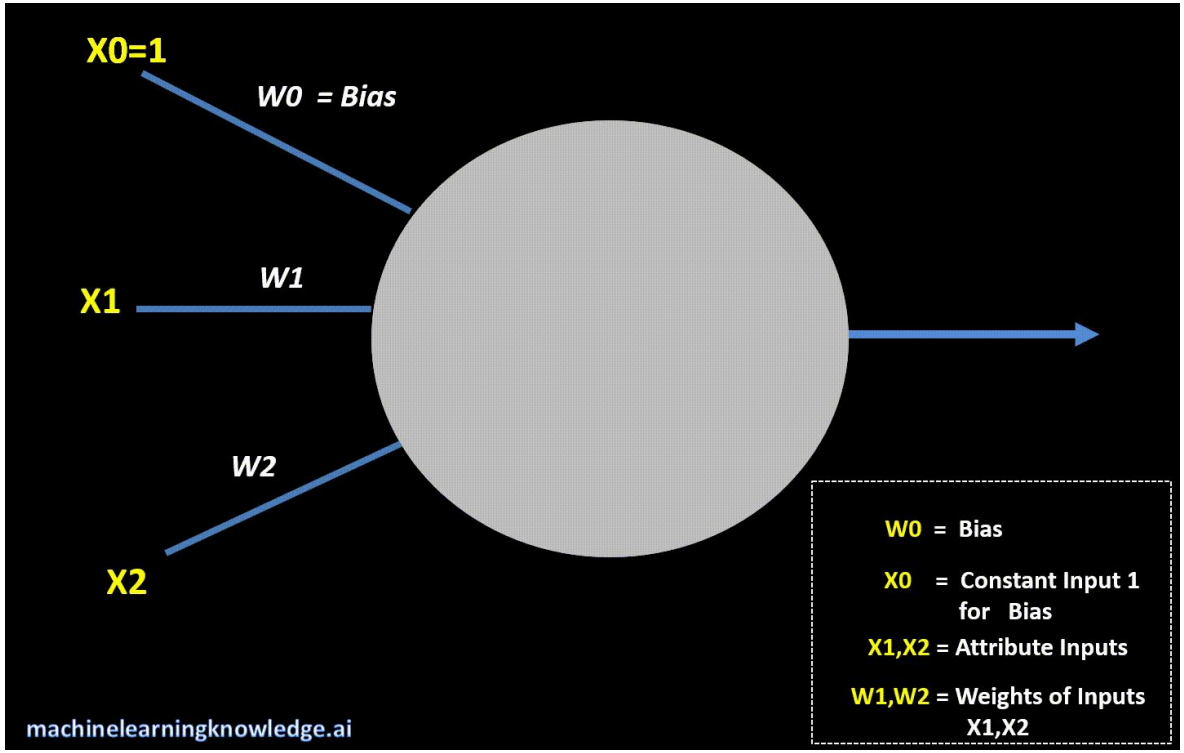
This can be written in a crude way as below –

Output = Activation(Summation(Inputs*Weights + bias))

Reference: machinelearningknowledge.ai

Inside a Neuron

- A neuron takes input from the previous layer's neurons (X0,X1,X2)
- It then multiplies each input with some weight (W0,W1,W2) and sums them
- Finally, it applies some activation function and sends/ fires an output.



Activation Functions

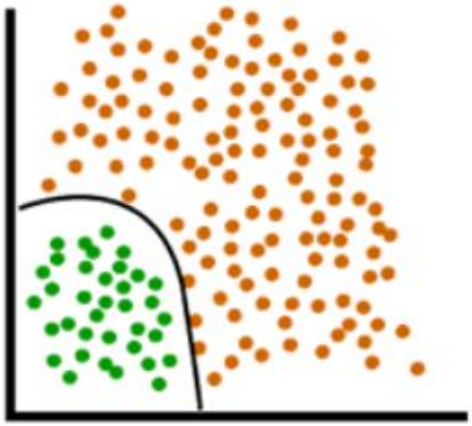
Activations functions are an important part of an artificial neural network.

They basically decide **whether a neuron should be activated(fired)** or not, based on whether each neuron's input is relevant for the model's prediction.

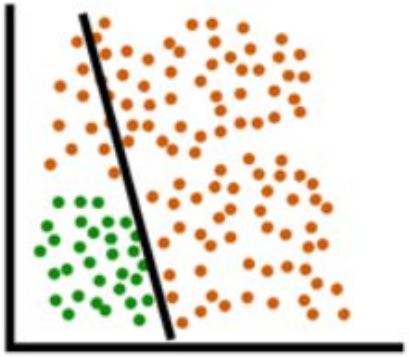
Activation Functions

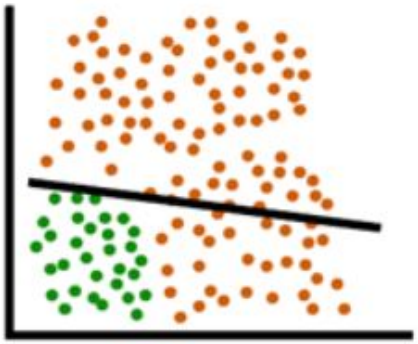
The purpose of the activation function is **to introduce non-linearity** into the output of a neuron.

Basically, it helps in creating a boundary like this:



Rather than the linear boundaries(straight lines) that are unable to divide data into 2 classes:





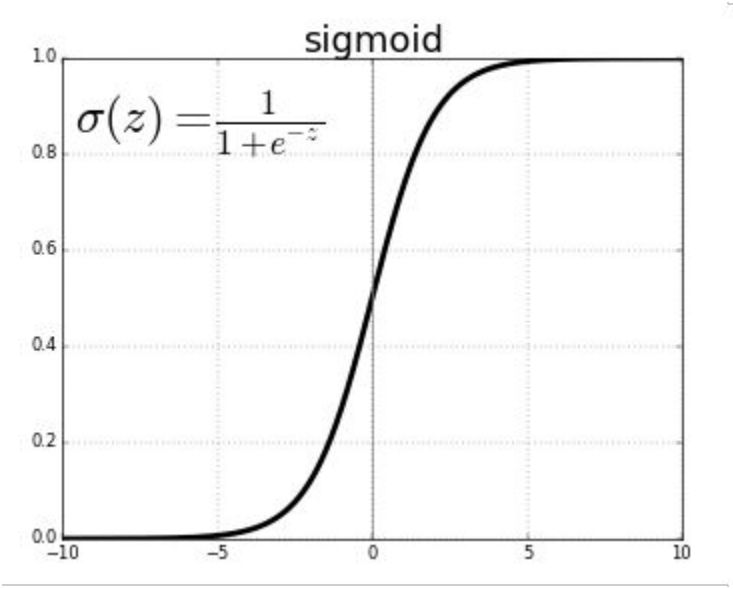
Types of Activation Functions

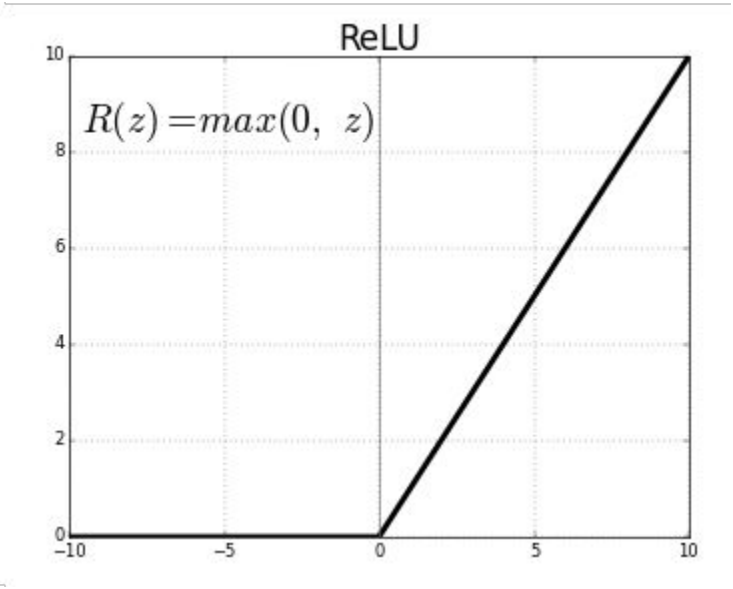
- Sigmoid Function
- ReLu
- tanH
- Leaky ReLU
- Softmax Function and more..

Activation Functions - ReLU

Similar to Sigmoid, we have a lot of other activation functions.

Let's have a look at **ReLU(Rectified Linear Unit)** , for example:



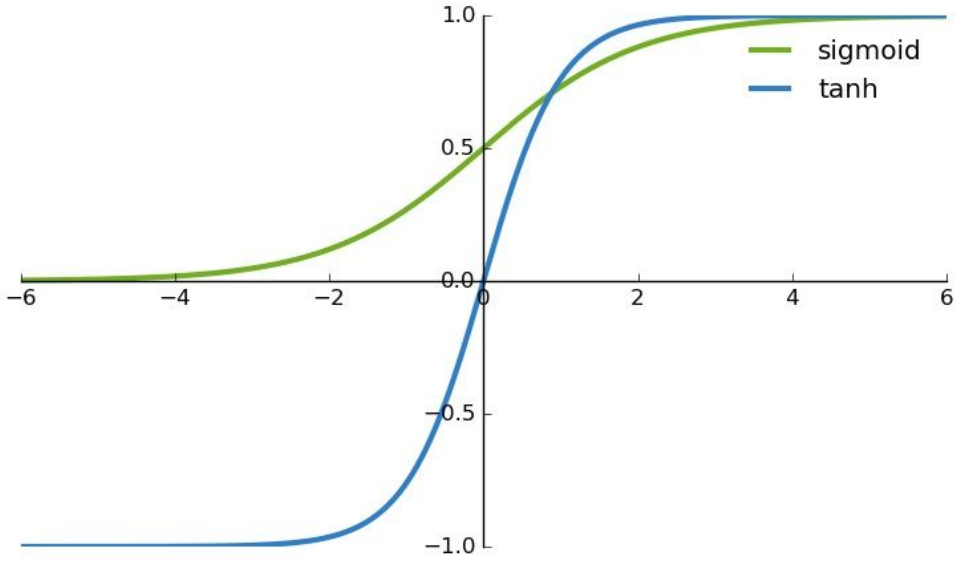


Look at the y axis of both the graphs. Just how Sigmoid had a range of 0 to 1, **ReLU** has a range of **0 to infinity**.

Activation Functions - tanH

TanH / Hyperbolic Tangent is another popular Activation Function.

It actually shares a few things in common with the sigmoid activation function. They both look very similar. But while a sigmoid function has a range of 0 and 1, **Tanh** has a range of **-1 and 1**.



There are a few more activation functions that we'll read about later.

General Guidelines

While choosing an activation function, you can keep the following guidelines in mind:

- Sigmoid is commonly used in the output layer. This is because it helps in giving a probability(value between 0 and 1) which is useful in Binary Classification.

- At places other than output layer, tanH usually performs better than Sigmoid.

- For Hidden layers, if you are not sure which activation function to use, just use ReLU as your default choice.

Note: While these guidelines are helpful, choosing an activation function also depends on trial and error. You should try out a few activation functions and see which one works the best for you.

Learn more about Activation Functions

- Animated guide to Activation Functions in Neural Network

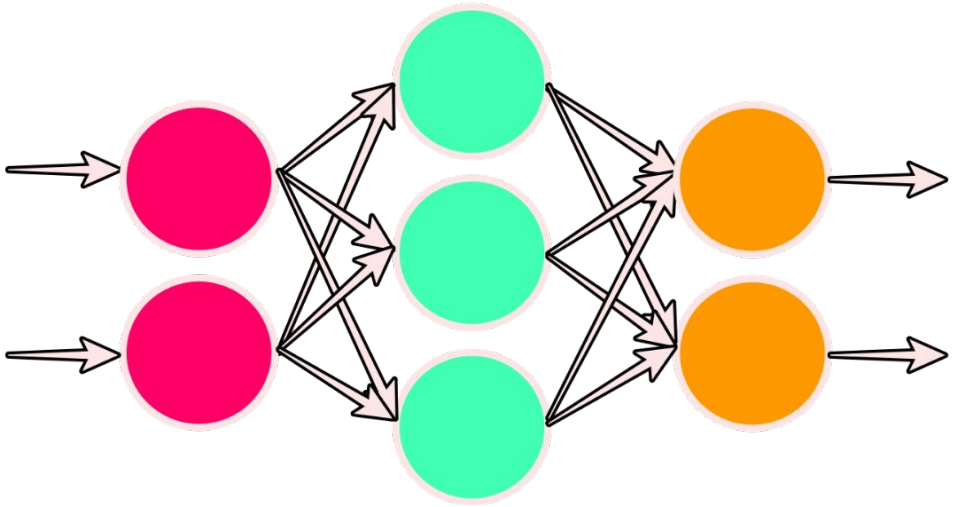
The above article explains the need of activation functions, the different types of activations as well as the advantages and disadvantages of each.

We would recommend glancing over it just to get and idea through the various gifs present.

There is no need to dive into the mathematics or worry about the technical terms used there, we'll get to it in some days. The process of learning is slow but surely steady :)

Layers

The neurons in a neural network are divided into layers.



While we know them with the names Input, Hidden and Output till now, Tensorflow doesn't go by those names. It wants the user to specify the type of that particular layer.

For utilising the different types of layers we have available, Tensorflow provides a submodule called layers that we can import as follows:

```
from tensorflow.keras import layers
```

Dense Layer

- A dense layer is just a regular layer of neurons in a neural network.
- It is the most common and frequently used layer.
- Look at the middle layer in the previous image. Each neuron receives input from all the neurons in the previous layer and is thus called **densely connected or dense**.
- Each output of a dense layer is computed using every input to the layer.
- Dense is one particular type of layer, but there are many other types that we will see as we continue our deep learning journey.

Creating models with Layers

Sequential Model API

There are two ways to create a model using the Layers API:

1. **A sequential model**
   The most common type of model is the Sequential model, which is a **linear stack of layers**. In short, it allows you to build a model layer by layer. Each layer has weights that correspond to the layer that follows it.

2. **A functional model**
   Unlike the stack of layers in Sequential API, the functional API is a way to build **graphs of layers**.

As a beginner, the sequential model is the recommended way to get started.

TIP

Learning by Doing

You **DON'T need to memorize code** given the below notebook. But you must understand what each line of code is doing and should be able to replicate it if required for solving other problems. We have provided explanation as much as possible, if you still don't get certain things, please don't hesitate to put it up on discuss forum!

Let's practice building Sequential & Functional Models!

https://github.com/dphi-official/Deep_Learning_Bootcamp/blob/master/DL%20For%20Classification/%20DL_Day6_Building_a_DL_Model.ipynb

- Download
- Extract zip file
- Open in Jupyter Notebook or Upload on Google Colab

Error functions and Optimizers

Error/Loss Functions

In most learning networks, error is calculated as the difference between the actual output and the predicted output.

The function that is used to compute this error is known as **Loss Function.**

Different loss functions will give different errors for the same prediction, and thus have a considerable effect on the performance of the model.

For Binary Classification problems, the loss function usually used is known as **Binary Cross Entropy Loss**. We don't need to go into the depths of this loss right now. This will be covered in later sessions.

Optimization Function

Error is a function of internal parameters of model i.e weights and bias. For eg. m and c in a straight line equation.

For accurate predictions, one needs to **minimize the calculated error**.

In a neural network, this is done using back propagation. The current error is typically propagated backwards to a previous layer, where it is used to modify the weights and bias in such a way that the error is minimized.

The **weights are modified using a function called Optimization Function**. Optimisation functions usually calculate the gradient.

There are a number of Optimizers available such as Adam, RMSProp, SGD etc. We don't need to get into the theory behind these optimizers right now as this will also be covered later.

For our problem, we'll be using RMSProp as the optimizer.

Thus, the components of a neural network model i.e the **activation function**, **loss function** and **optimization algorithm** play a very important role in efficiently and effectively training a Model to produce accurate results.

Different tasks require a different set of such functions to give the most optimum results.

TIP

Learning by Doing

You **DON'T need to memorize code** given the below notebook. But you must understand what each line of code is doing and should be able to replicate it if required for solving other problems. We have provided explanation as much as possible, if you still don't get certain things, please don't hesitate to put it up on discuss forum!

Your first Neural Network for Classification

In this notebook, we'll perform binary classification using a neural network to determine whether a person is suffering from a heart disease.

https://github.com/dphi-official/Deep_Learning_Bootcamp/blob/master/DL%20For%20Classification/DL_Day6_Binary_Classification.ipynb

- Download
- Extract Zip File
- Open in Jupyter Notebook or Upload on Google Colab

Slide Download Link

- You can download the slides here: https://docs.google.com/presentation/d/19GR6Tg2E06yyJO0b4KCthauhkQoasLZFICbNL6JZP38/edit?usp=sharing

That’s it for the day. Thank you!

Feel free to post any queries in the #help channel on Slack