TECHNEEDS

WEEK-3

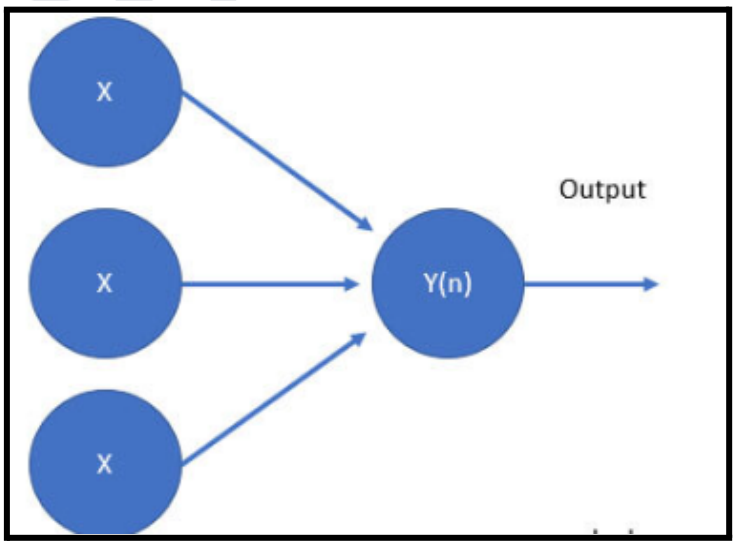
Notes

Topics Covered:

- **Singlelayer Perceptron**
- **Multilayer Perceptron**
- **Back Propagation**
- **Forward Propagation**
- **Loss Functions**

SINGLE LAYER PERCEPTRON:

A single-layer neural network represents the most simple form of neural network, in which there is one layer of input nodes that send weighted inputs to a subsequent layer of only one hidden nodes (neurons),and the output is obtained.



Input Layer and Hidden layer example use case:

The input layer can be compared to the human eye, which is responsible for taking the input in the form of the images, and features of all those images. These features are processed inside via neurons to identify the object and give the output result.

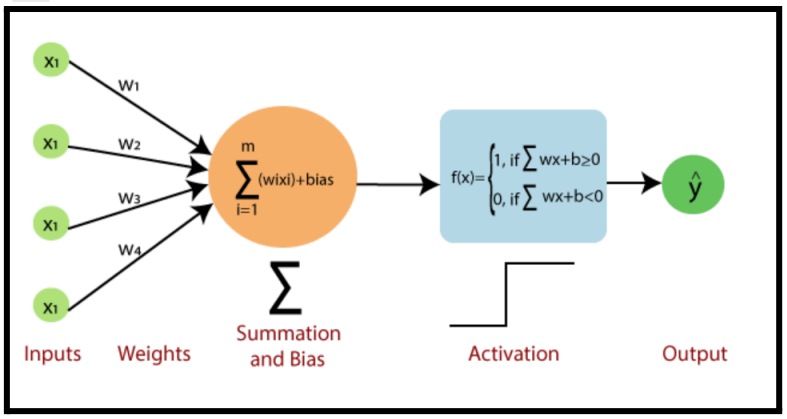
Human Eye

IMPORTANCE OF WEIGHTS IN THE COMPUTATION OF THE 'Z'(Refer notes of week2)

Weights plays a significant role in deciding that which neurons needs to be activated. So that neurons is only activated ignoring the other neurons.

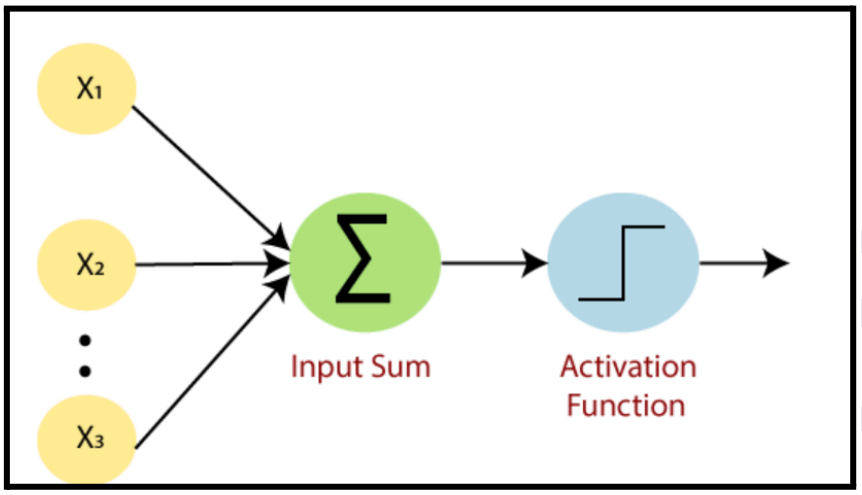
IMPORTANCE OF BIAS IN THE COMPUTATION OF THE 'Z'(Refer notes of week2)

Whenever we assign weights, there are situations when weights are assigned 0 and so the compute value of Z becomes 0. So to avoid this we introduce a very small value i.e. bias.This will help us to compute the value Z perfectly.



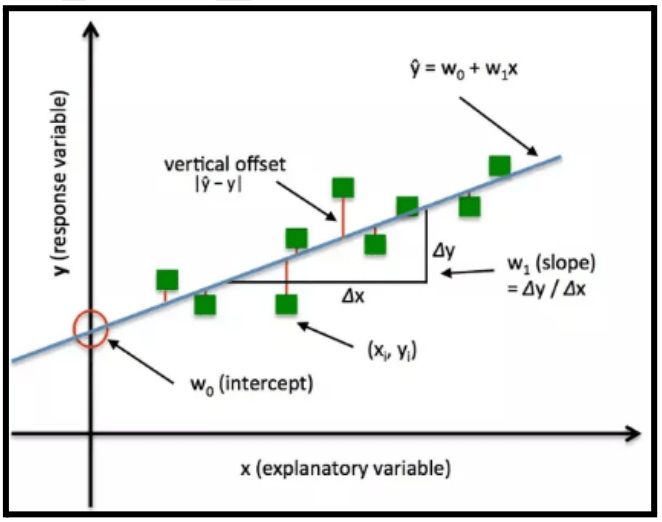
ACTIVATION FUNCTION:

An activation function determines whether a neuron can be activated or not. The activation function calculates a weighted sum and further adds bias with it to give the result.



LOSS FUNCTIONS:

After applying the activation function, we get the output as y_hat and let's say our actual output is y.
So loss function can be defined as the difference between the two and we have to try to reduce the loss function. This will help us to reduce the errors.



Types of Loss Functions:

We have different types of loss function which can be used in different scenarios for calculating the errors and minimizing it.

1.Mean Squared Error (LOSS 2 FUNCTION) :

Mean Squared Error (MSE) is a common loss function used in regression problems. It measures the average of the squares of the errors, where the error is the difference between the predicted value and the actual value.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where:

- n is the number of data points.
- y_i is the actual value for the i-th data point.
- $\hat{y}_i$ is the predicted value for the i-th data point.

***Note:We have to use linear activation function in the output layer if we use MSE**

2.Mean Absolute Error (LOSS1 FUNCTION):

Mean Absolute Error (MAE) is another common loss function used in regression problems. Unlike Mean Squared Error (MSE), which squares the errors, MAE calculates the average of the absolute differences between the predicted values and the actual values.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where:

- n is the number of data points.
- y_i is the actual value for the i-th data point.
- $\hat{y}_i$ is the predicted value for the i-th data point.
- $|y_i - \hat{y}_i|$ represents the absolute difference between the actual and predicted values.

3.Huber loss:

Huber Loss, also known as Huber error, is a loss function used in regression tasks that combines the advantages of both Mean Squared Error (MSE) and Mean Absolute Error (MAE). It is less sensitive to outliers compared to MSE while avoiding the issues of MAE's non-differentiability at zero. The Huber Loss is quadratic for small errors and linear for large errors, controlled by a parameter δ\deltaδ.

The Huber Loss function is defined as follows:

$$\begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{for } |a| > \delta \end{cases}$$

Where:

- $a = y - \hat{y}$ is the residual (difference between the actual value y and the predicted value $\hat{y}$).
- δ is a threshold parameter that determines the point where the loss function changes from quadratic to linear.

4.Binary Cross Entropy:

Binary Cross-Entropy (BCE), also known as log loss or logistic loss, is a loss function used for binary classification problems. It measures the performance of a classification model whose output is a probability value between 0 and 1. The BCE loss quantifies the difference between the predicted probability and the actual binary label (0 or 1).

The formula for Binary Cross-Entropy is:

$$\text{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

Where:

- n is the number of data points.
- y_i is the actual binary label for the i-th data point (0 or 1).
- $\hat{y}_i$ is the predicted probability for the i-th data point (ranging from 0 to 1).

***Note:We have to use sigmoid activation function in the output layer.**

5.Categorical Cross Entropy:

Categorical Cross-Entropy, also known as Softmax Loss or Multinomial Log Loss, is a loss function used for multi-class classification problems where each sample belongs to one of C classes. This loss function measures the performance of a classification model whose output is a probability distribution over C different classes.

The formula for Categorical Cross-Entropy is:

$$\text{CCE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})$$

Where:

- n is the number of data points.
- C is the number of classes.
- $y_{i,c}$ is a binary indicator (0 or 1) if class label c is the correct classification for the i-th data point.
- $\hat{y}_{i,c}$ is the predicted probability of the i-th data point belonging to class c.

Explanation

- **One-Hot Encoding**: The actual labels y_i are usually one-hot encoded vectors, meaning that for each data point, the vector has a value of 1 for the true class and 0 for all other classes.
- **Logarithmic Loss**: Categorical Cross-Entropy uses the logarithm of the predicted probabilities. It heavily penalizes incorrect predictions that are confident (i.e., a high predicted probability for an incorrect class).
- **Sum Over Classes**: For each data point, the loss is calculated by summing the log probabilities of the true classes. The negative log likelihood is then averaged over all data points.

***Note:We have to use softmax activation function in the output layer.**

6. Sparse categorical Entropy:

Same as Categorical cross entropy just one difference is there:

- The key difference is that Sparse Categorical Cross-Entropy is used when the target labels are provided as integers (class indices) rather than one-hot encoded vectors. This can save memory and computation, especially when dealing with a large number of classes.

Definition

The formula for Sparse Categorical Cross-Entropy is:

$$\text{Sparse CCE} = -\frac{1}{n}\sum_{i=1}^{n}\log(\hat{y}_{i,y_i})$$

Where:

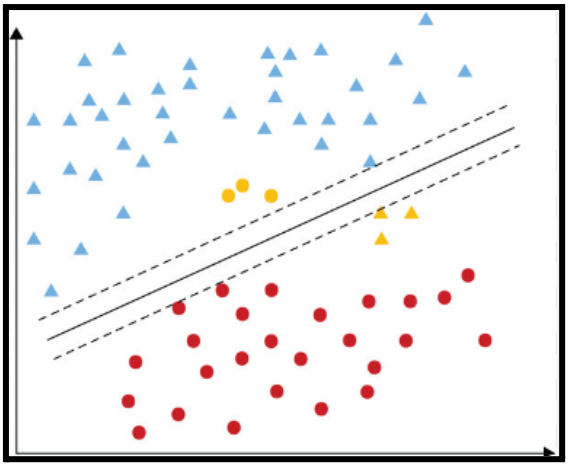
- n is the number of data points.
- C is the number of classes.
- y_i is the actual class index for the i-th data point (an integer from 0 to $C-1$).
- $\hat{y}_{i,c}$ is the predicted probability of the i-th data point belonging to class c.

Explanation

- **Class Index Labels**: Unlike Categorical Cross-Entropy, which uses one-hot encoded vectors, Sparse Categorical Cross-Entropy uses integer labels representing class indices.
- **Logarithmic Loss**: The loss is computed using the log of the predicted probability for the correct class index. This penalizes confident wrong predictions more heavily.
- **Sum Over Data Points**: The loss is summed over all data points and then averaged, making it a measure of the model's overall performance.

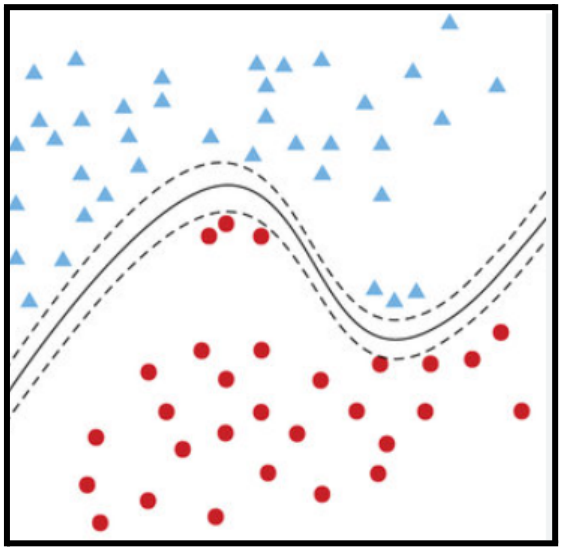
Why Single Layered Neural Network is not effective?

- We can use single layered neural network for the linearly separable data. This neural network gives a very good result when working with the linear data



Linearly Separable data

- We cannot use the single-layered neural network for nonlinear data. It doesn't perform efficiently when dealing with the non-linear data.

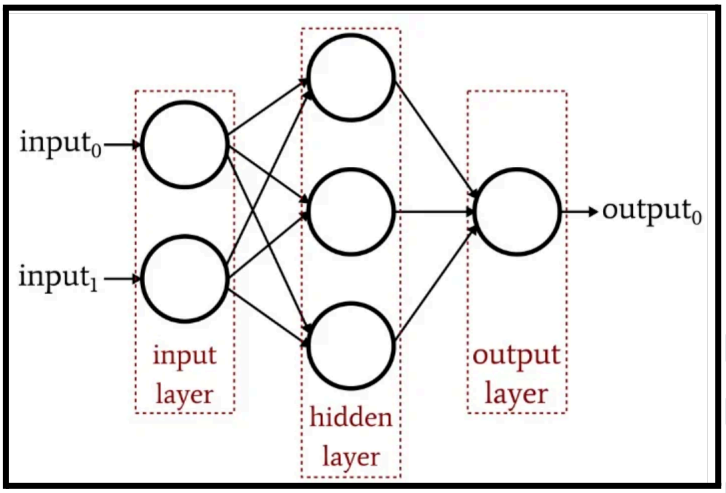


Non-Linearly Separable

MULTILAYERED PERCEPTRON:

An artificial neural network (ANN) is a machine learning model inspired by the structure and function of the human brain's interconnected network of neurons. It consists of interconnected nodes called artificial neurons, organized into layers. Information flows through the network,

with each neuron processing input signals and producing an output signal that influences other neurons in the network.

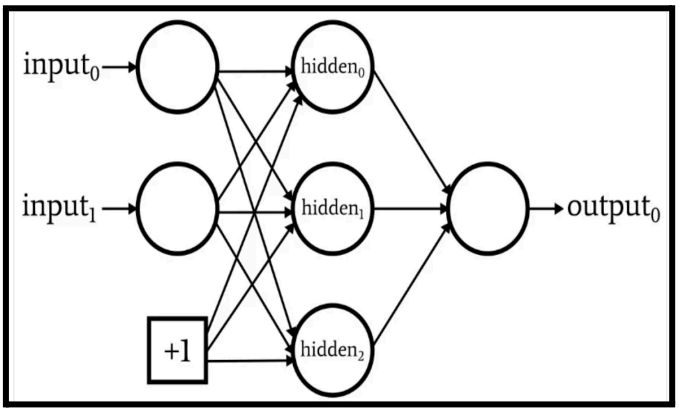


MULTILAYER PERCEPTRON

HOW CAN YOU CHANGE THE ARCHITECTURE OF NN?

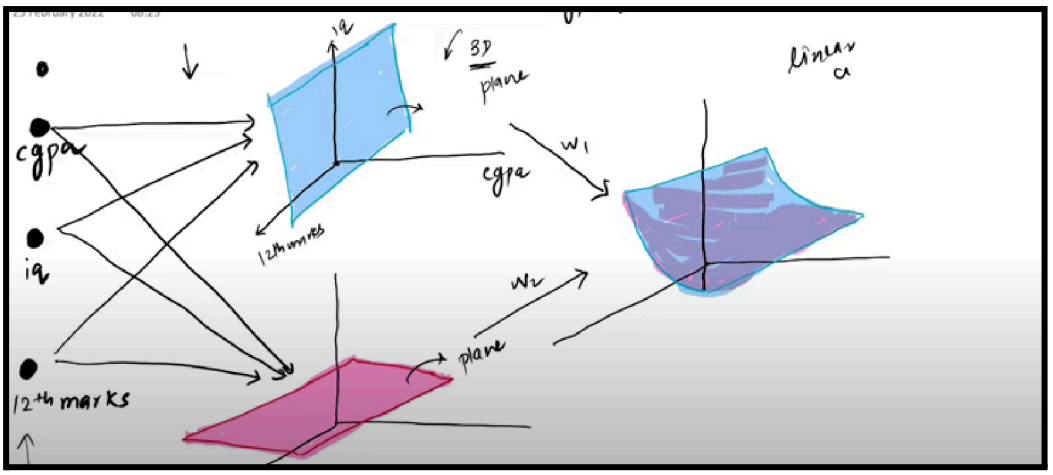
1.) Adding nodes in the hidden layers:

We can add an extra node in the hidden layer to handle the extra complexity /non-linearity of the data.



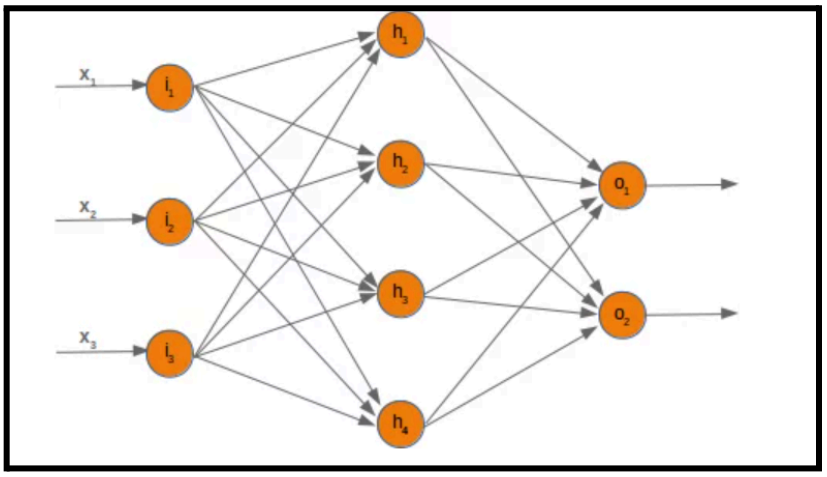
2. Adding nodes in the input layer:

Now adding the extra layer in the input layer can increase the dimension of the hidden layer from 2D ->3D ->4D. This can also help us to handle the complexity of the data.



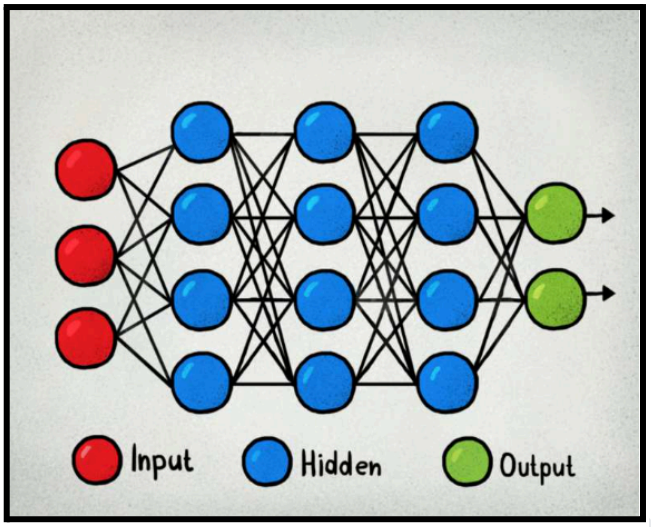
3. Adding nodes in the output node:

If we want to do multiclass classification then we have to add extra output nodes.



4. Deep Neural Network:

You can increase the number of hidden layers also to change the architecture of the NN.



You can visit this website to see the visual representation of how you can change the architecture of the neural networks to handle complex data.

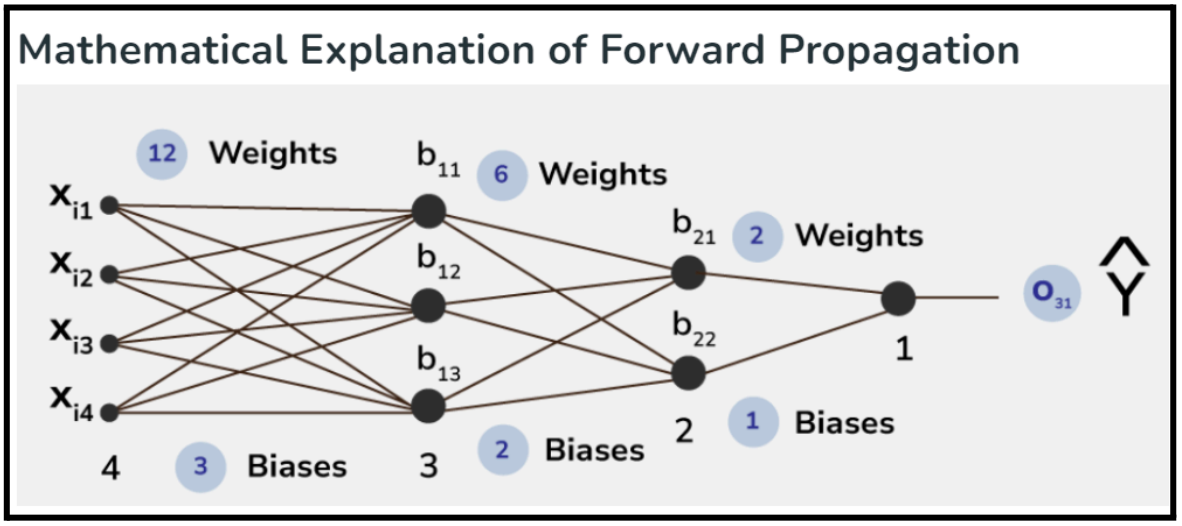
FORWARD PROPAGATION:

Forward propagation is the way data moves from left (input layer) to right (output layer) in the neural network.A neural network can be understood by a collection of connected input/output nodes. The accuracy of a node is expressed as a loss function or error rate.

It involves the following steps:

1. **Input Layer:** The input data is fed into the input layer of the neural network.
2. **Hidden Layers:** The input data is processed through one or more hidden layers. Each neuron in a hidden layer receives inputs from the previous layer, applies an activation function to the weighted sum of these inputs, and passes the result to the next layer.
3. **Output Layer:** The processed data moves through the output layer, where the final output of the network is generated. The output layer typically applies an activation function suitable for the task, such as softmax for classification or linear activation for regression.
4. **Prediction:** The final output of the network is the prediction or

classification result for the input data.

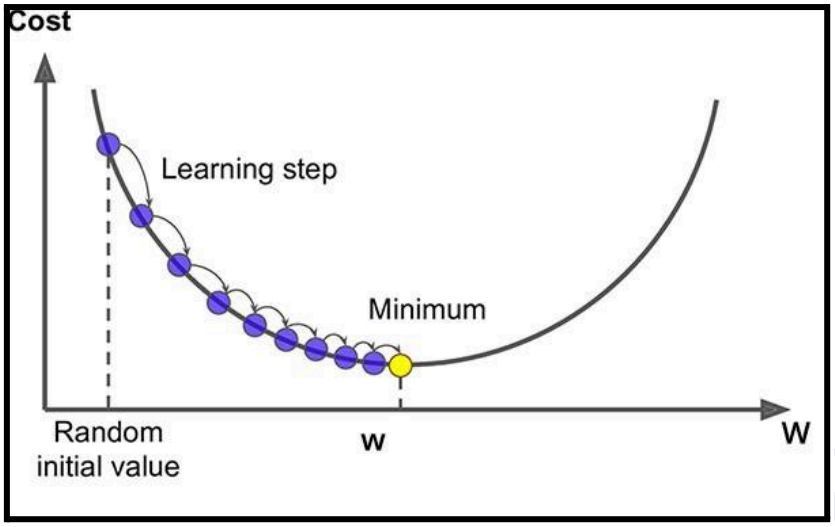


BACKWARD PROPAGATION:

Backpropagation is an algorithm that backpropagates the errors from the output nodes to the input nodes. Therefore, it is simply referred to as the backward propagation of errors.

In backward propagation, the network adjusts the weights and biases to minimize the loss. Here's how it works:

- **Compute Gradients:** Calculate the gradient of the loss function with respect to each weight by applying the chain rule of calculus. This involves:
  - Calculating the error at the output layer.
  - Propagating this error backward through the network, layer by layer, calculating the contribution of each weight to the error.



Gradient Descent Graph

- **Update Weights and Biases:** Adjust the weights and biases in the direction that reduces the loss, typically using an optimization algorithm like gradient descent.
- **Learning Rates:** It decides the speed of convergence.By default the value of convergence rates is 1.

$$W^{[2]} = W^{[2]} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial W^{[2]}}$$

$$W^{[2]} = \begin{bmatrix} w_1^{[2]} \\ w_2^{[2]} \\ w_3^{[2]} \\ w_4^{[2]} \end{bmatrix} - \alpha \cdot (a_2 - y) \cdot \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[2]} - \alpha \cdot (a_2 - y) \cdot a_1^{[1]} \\ w_2^{[2]} - \alpha \cdot (a_2 - y) \cdot a_2^{[1]} \\ w_3^{[2]} - \alpha \cdot (a_2 - y) \cdot a_3^{[1]} \\ w_4^{[2]} - \alpha \cdot (a_2 - y) \cdot a_4^{[1]} \end{bmatrix}$$

$W^{(2)}$ weights update computation

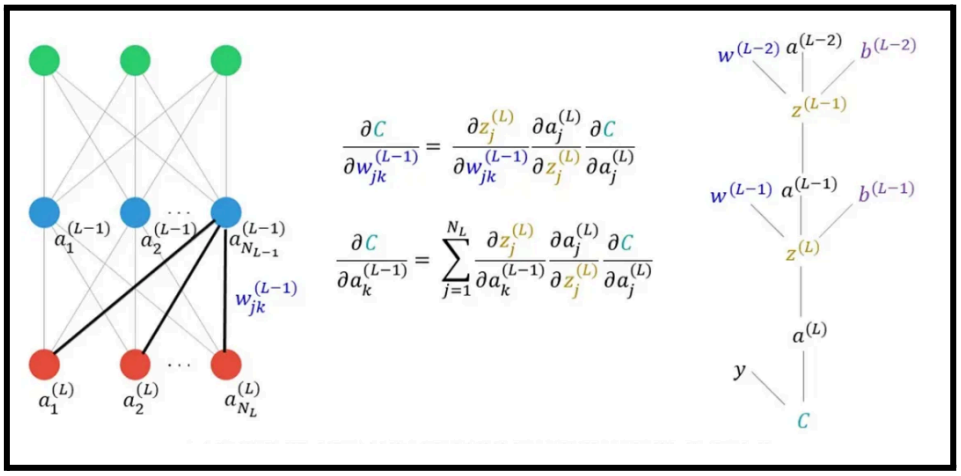
$$b_2 = b_2 - \alpha \cdot \frac{\partial \mathcal{L}}{\partial b_2}$$

$$b_2 = b_2 - \alpha \cdot (a_2 - y)$$

$b^{(2)}$ bias update computation

Chain Derivative of Back Propagation:

We start with the derivative of outermost function and then move towards derivatives of inner functions sequentially. In the end, all the derivatives are multiplied together to obtain the final result. Chain rule is used in backpropagation during training phase of neural networks.



Refer this for the above topics:

- **Single Layered Perceptron**:Single Layered
- **Multilayer Perceptron:** Multi Layered
- **Backward Propagation:**Backward Propagation
- **Forward Propagation:**Forward Propagation
- **Loss Functions:**Loss Functions