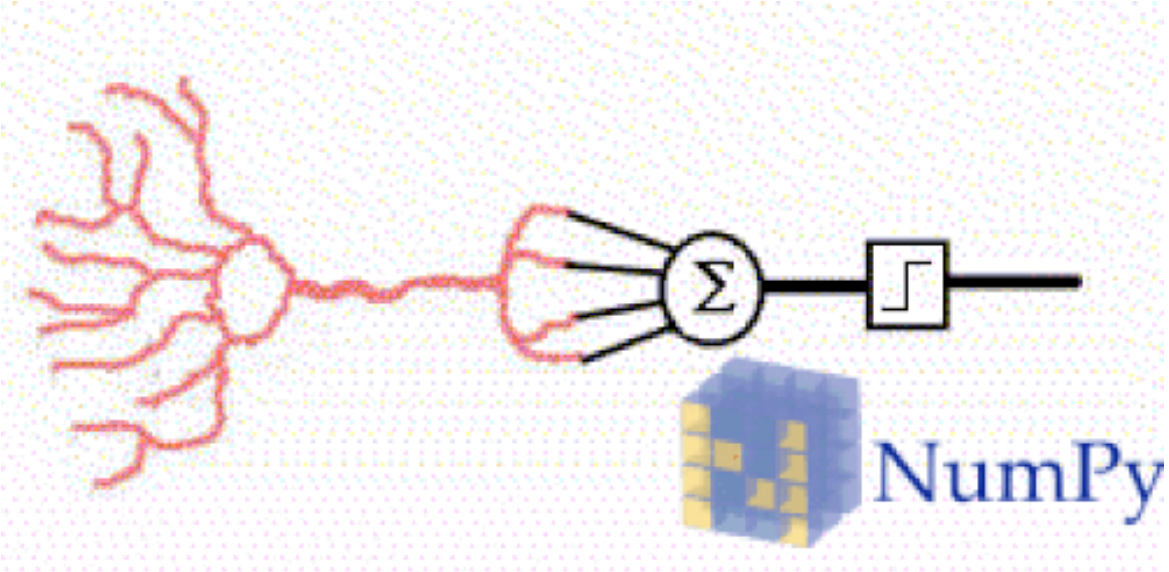
Deep Learning

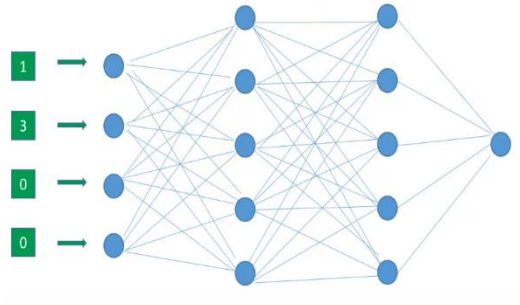
"Deep learning is a particular kind of machine learning that is inspired by the functionality of our brain cells called neurons which led to the concept of artificial neural network"

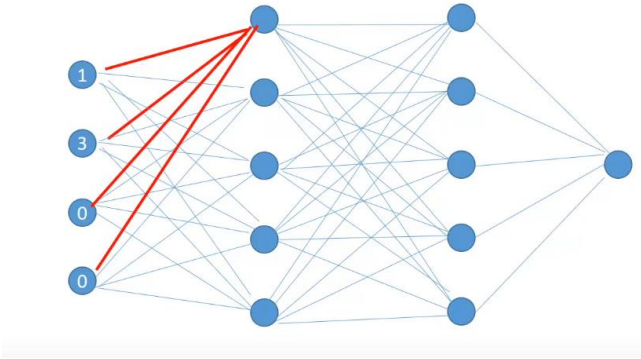
Learning in an Artificial Neural Network –

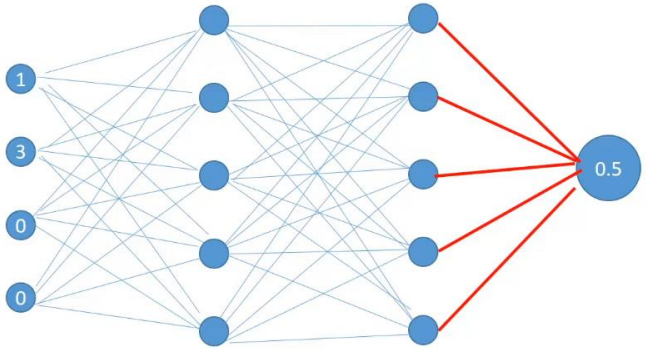
* *forward propagation,*
* *backward propagation*

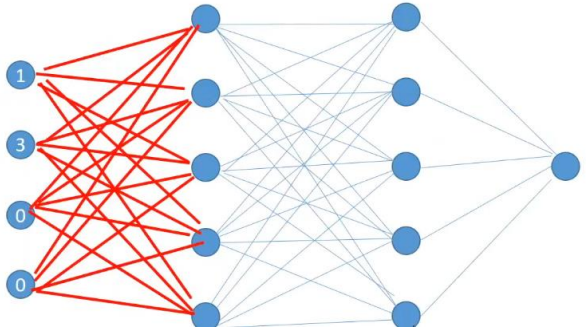


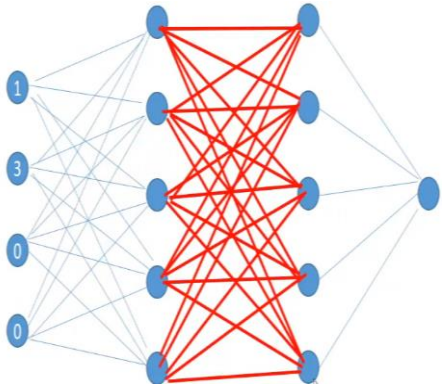
Forward Propogation:
Series of calculations from the input to output

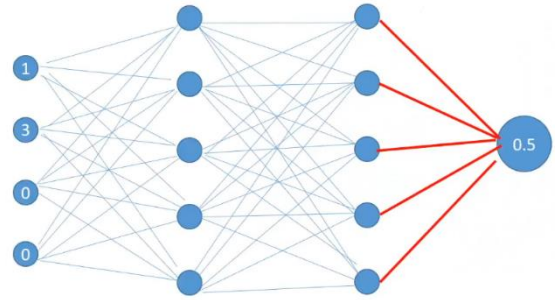










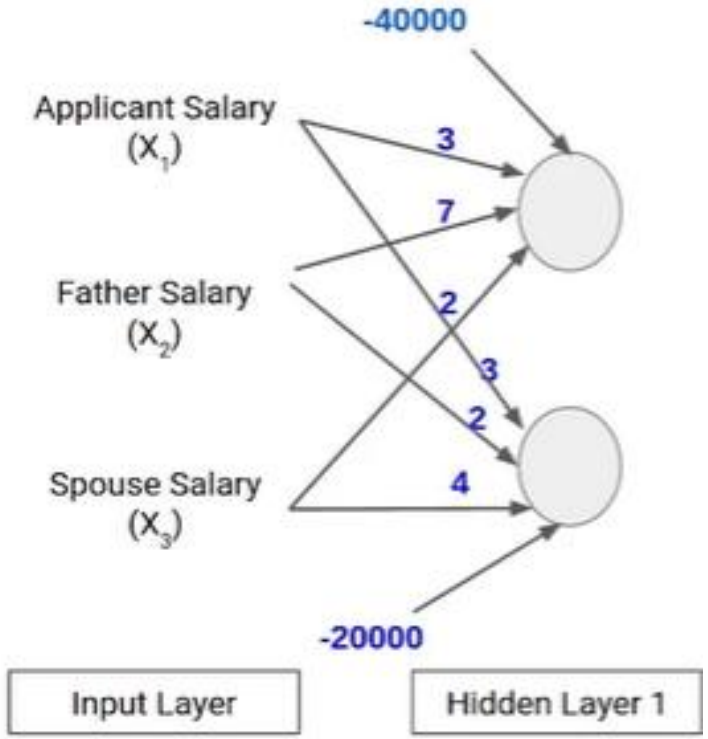


difference between Forward propagation and Backward Propagation

- **Forward propagation:** The input is fed into the network. In each layer, there is a specific activation function and between layers, there are weights that represent the connection strength of the neurons. The input runs through the individual layers of the network, which ultimately generates an output.
- **Backward propagation:** an error function measures how accurate the output of the network is. To improve the output, the weights have to be optimized. The backpropagation algorithm is used to determine how the individual weights have to be adjusted. The weights are adjusted during the gradient descent method.

Forward Propogation:

- Weights of the layers are constantly updated to improve the predictions of the future. Initially random values are assigned to weights for the connections and random values assigned to bias for the neurons.



$$Z_{11} = X_1 * w_1 + X_2 * w_2 + X_3 * w_3 + b_1$$

$$Z_{11} = X_1 * 3 + X_2 * 7 + X_3 * 2 - 40000$$

$$H_{11} = \sigma(Z_{11})$$

$$Z_{12} = X_1 * w_4 + X_2 * w_5 + X_3 * w_6 + b_2$$

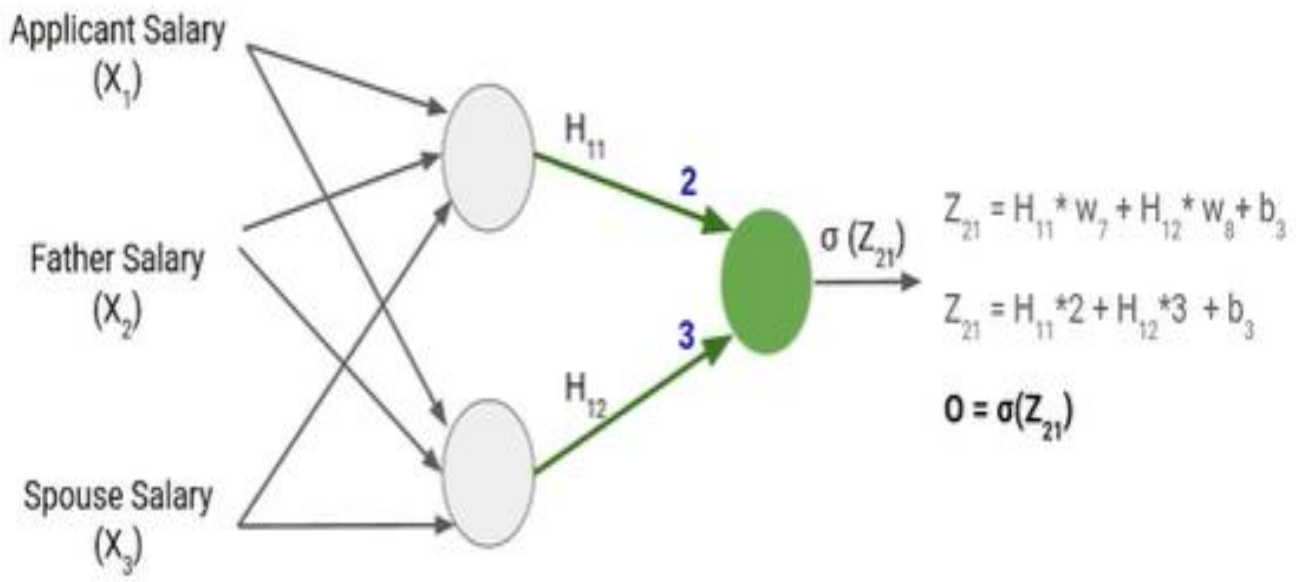
$$Z_{12} = X_1 * 3 + X_2 * 2 + X_3 * 4 - 20000$$

$$H_{12} = \sigma(Z_{12})$$

Z11 denotes the results of the first neuron of the first layer. Then activation function is applied to Z11.

The output of the hidden layer activation functions are denoted as H.
H11 is output for the first neuron in the first layer.
H12 is output for the second neuron in the first layer.

Forward Propogation

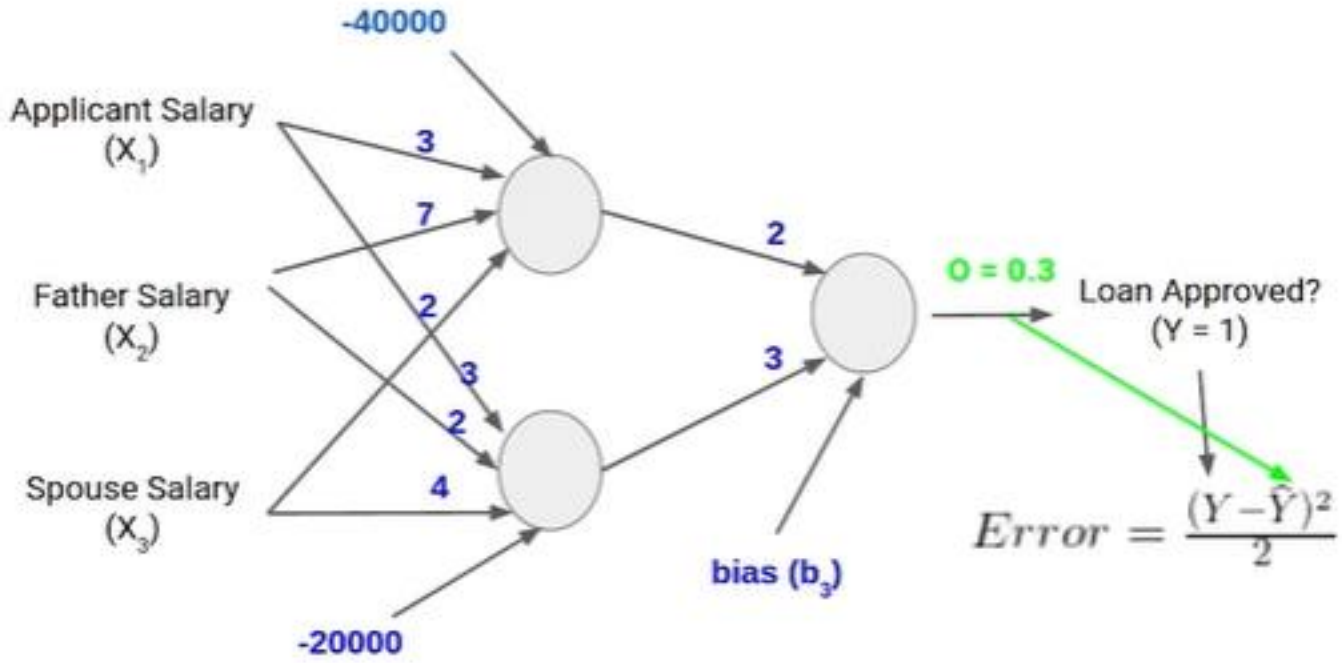


Z21 denotes the results of the second neuron of the first layer. Then activation function is applied to the result Z12.

These outputs are finally sent to the next layer. Again some random weight values are assigned for these new connections and the same process is repeated. Then activation function fired and output as O. The output at o may or may not be correct. Then we calculate the error.

Weights of the layers are constantly updated to improve the predictions. This compete process is called Forward Propagation.

Cost / Loss / Error Function



This error is for a single observation. The sum of the errors for all the observations is called cost.

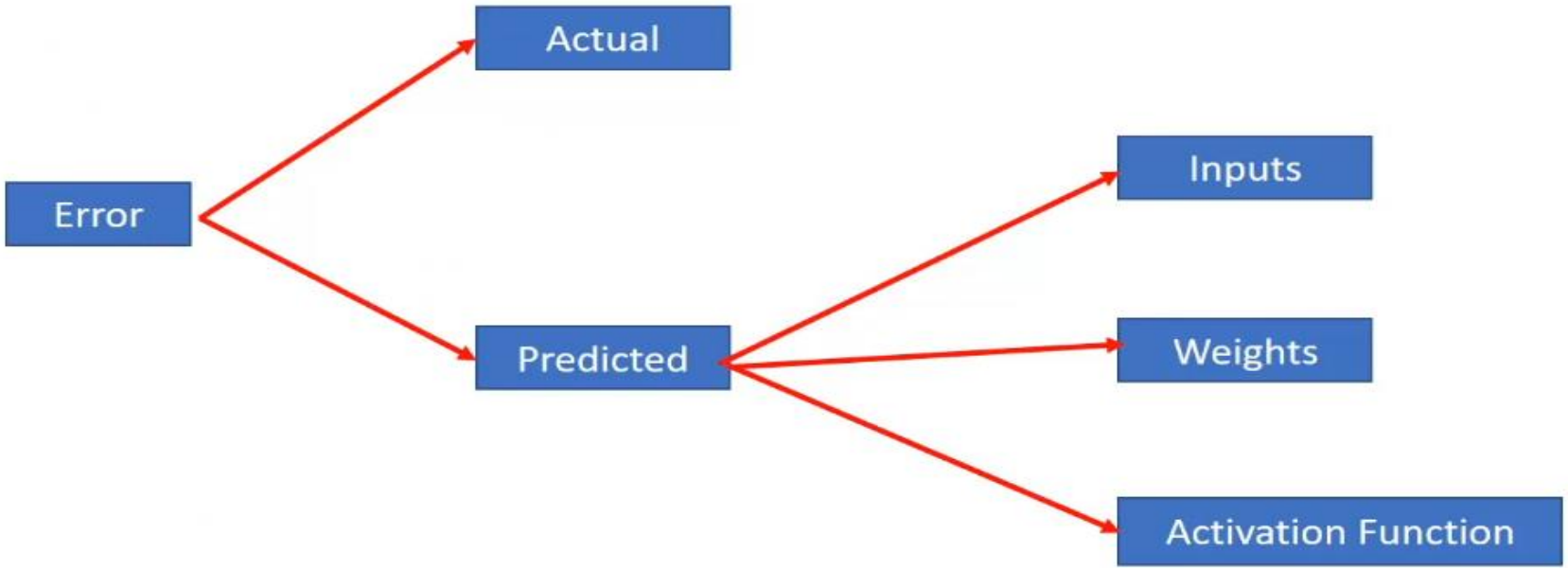
Our task is to minimize the error which is dependent on y and output (predicted y).

Errors in Neural Network

- Forward propagation helps us in calculating outputs.
- Let’s say for a particular row the actual target is 0 and the predicted target is 0.5.
- We can use this predicted value to calculate the error for a particular row. The kind of error that show is Mean Squared Error.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Errors in Neural Network



Use smart strategy to adjust the weights and reduce the errors

- First is the amount of change in error on changing the weight by a small amount
- And second is the direction of that change.

Change in Error on changing a Weight by a small amount

Direction of Change

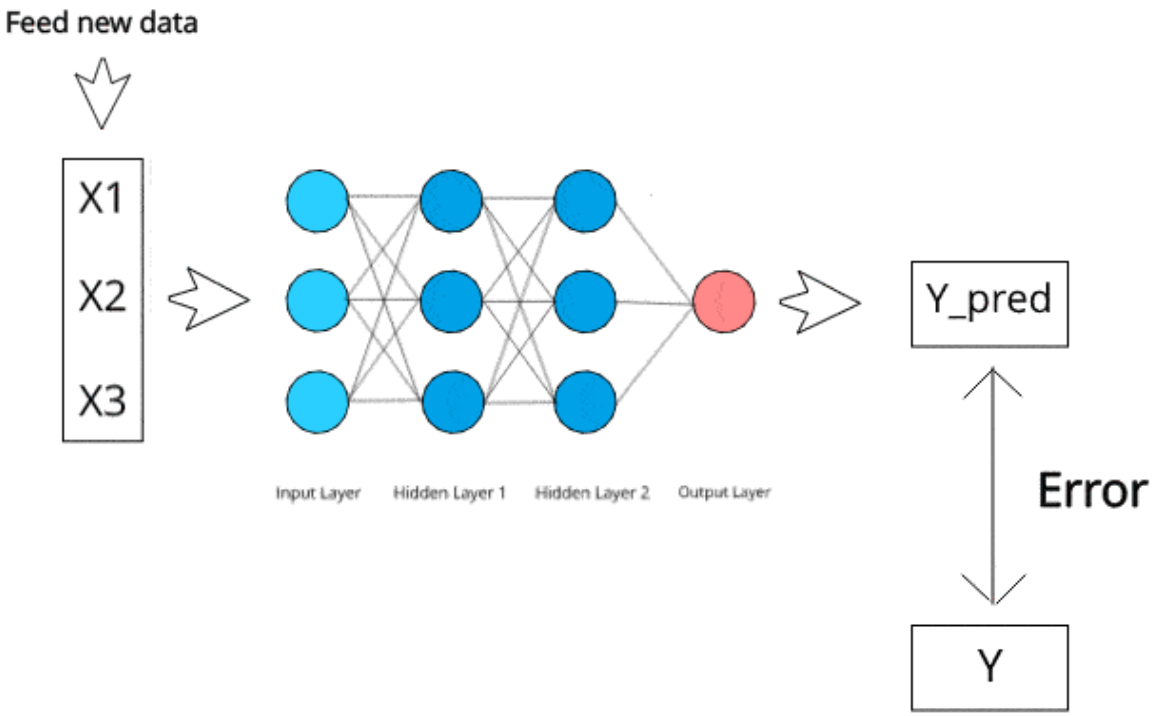
Minimize the error

- Error is dependent on :

  - **Value of Y :** Cant change as target values are fixed
  - **Output O :** Can change but it depends on input values, weights, bias and activation function
  - **Activation Function:** Cant be changed as it is to be defined before the training of the neural network starts. Once training started, AF cannot be changed as these values are hyper parameters.
  - **Weights and Bias :** Only these **c**an be changed during the training process and they are called parameters.

Note : Changing the bias and weights to reduce the error is called back propagation.

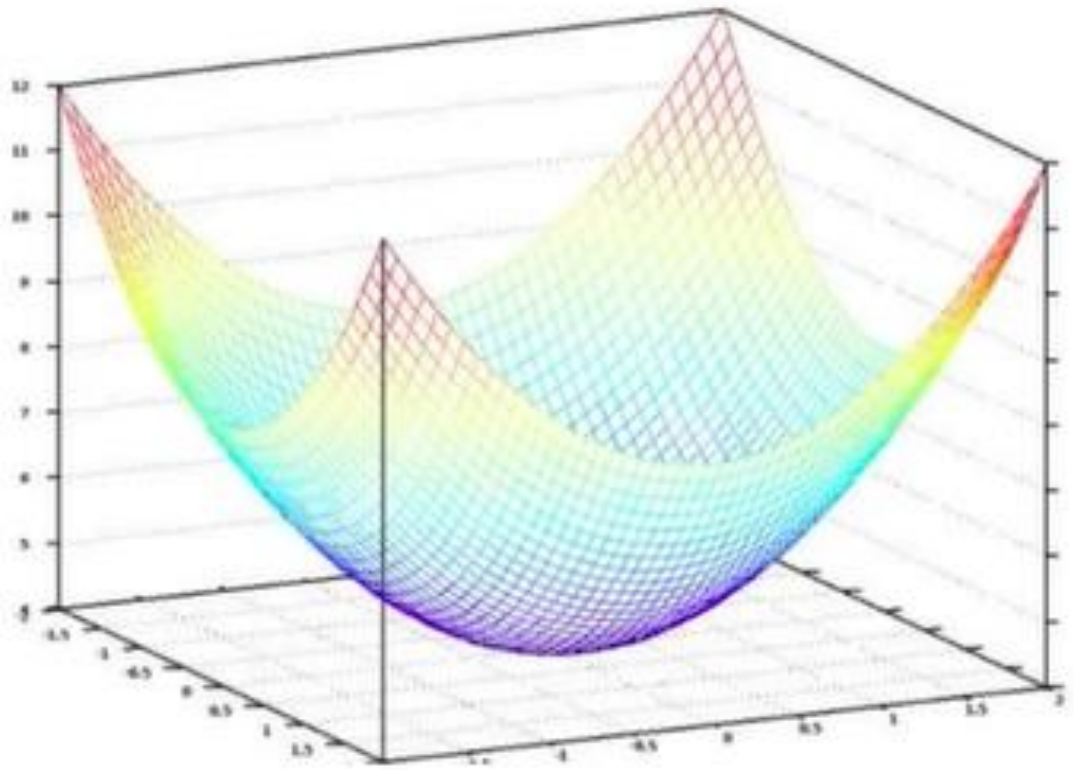
Back Propogation

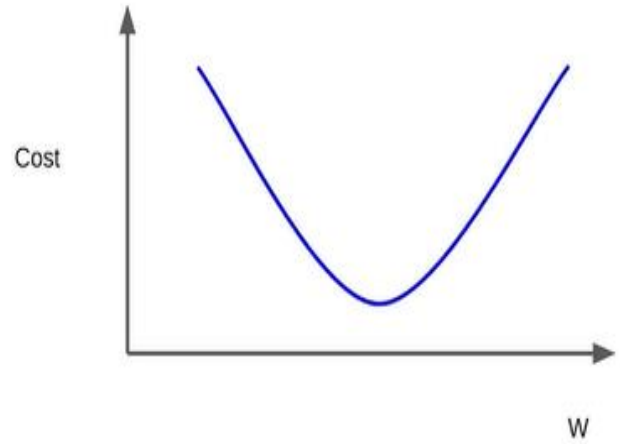


Backpropagation can be divided into the following steps:

- It can forward the propagation of training data through the network to generate output.
- It uses target value and output value to compute error derivatives by concerning the output activations.
- It can backpropagate to calculate the derivatives of the error concerning output activations in the previous layer and continue for all the hidden layers.
- It uses the previously computed derivatives for output and all hidden layers to calculate the error derivative concerning weights.
- It updates the weights and repeats until the cost function is minimized.

Updating Weights Randomly
(For simplicity taken single weight value)





- Continuous
- Differentiable

initializing Weights and Biases in Neural Networks

- Neural network initialization means initialized the values of the parameters i.e, weights and biases. Biases can be initialized to zero but we can’t initialize weights with zero.
- Weight initialization is one of the crucial factors in neural networks since bad weight initialization can prevent a neural network from learning the patterns.
- On the contrary, a good weight initialization helps in giving a quicker convergence to the global minimum. As a rule of thumb, the rule for initializing the weights is to be close to zero without being too small.

Gradient Descent

Gradient Descent update equation

$$w = w - \alpha * dE / dw$$

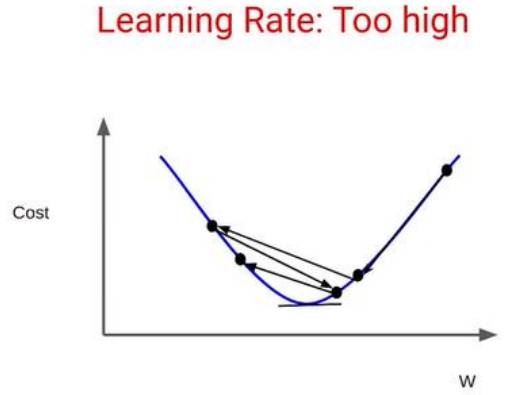
- In which direction should I move?
- How much should I move?

Gradient Descent

- Update Equation is used to update the parameters which are the weights and biases.
- The partial derivative of the error e w.r.t w is the rate of Change of error wrt change in weight.
- Sign of partial derivative tells the direction to move so error is minimized, that is why we need continuous and differentiable cost functions.
- And the magnitude of the partial derivative tells how much should we move in the particular direction.

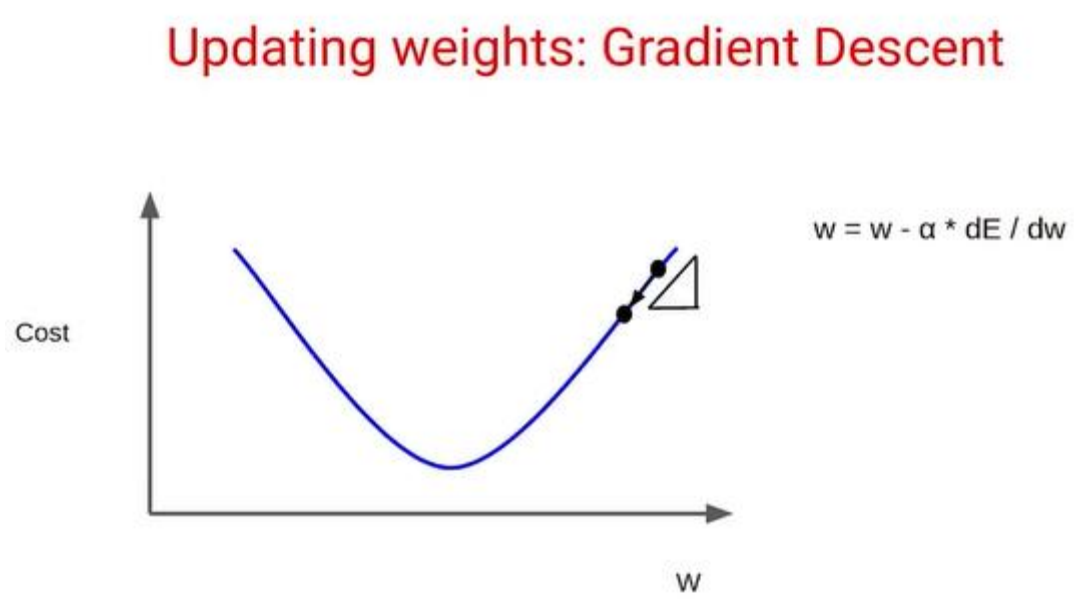
Alpha

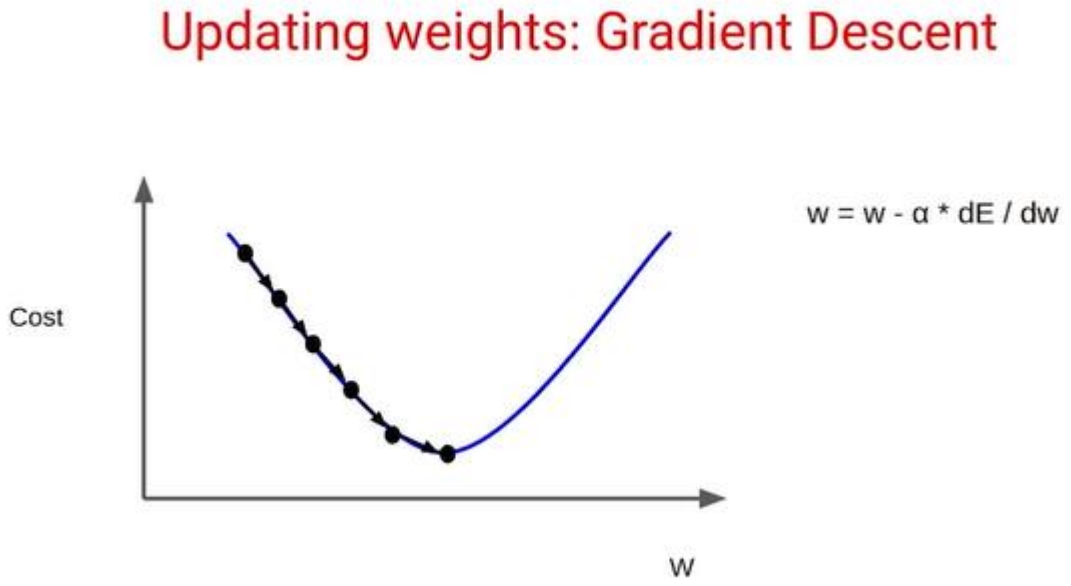
- Alpha is the learning rate which controls the updation of parameters in terms of magnitude.
- Should we update the value of w 0.1 times of de/dw or 10 times de/dw, that depends on value of alpha.
- If learning rate is high, error might fluctuate very high we may not reach minimum. If we set it too low, update will be slow and we need many iterations to reach minimum.
- Right value of alpha is normally 0.01 or 0.001.



Steps of Gradient Descent

- Take current value of w and b
- Take a step in the steepest downhill direction.
- Repeat the previous step until minima is achieved.





Stopping Criteria for Gradient Descent

- 1. We stop running the algorithm when the error is not updating in the consecutive epochs or the error has been minimized.

- 2. We stop the updation process when the no of iterations is achieved. Example if the no of epochs is achieved as 100 even if errors is not minimized we can stop updating the parameters.

Neural Networks

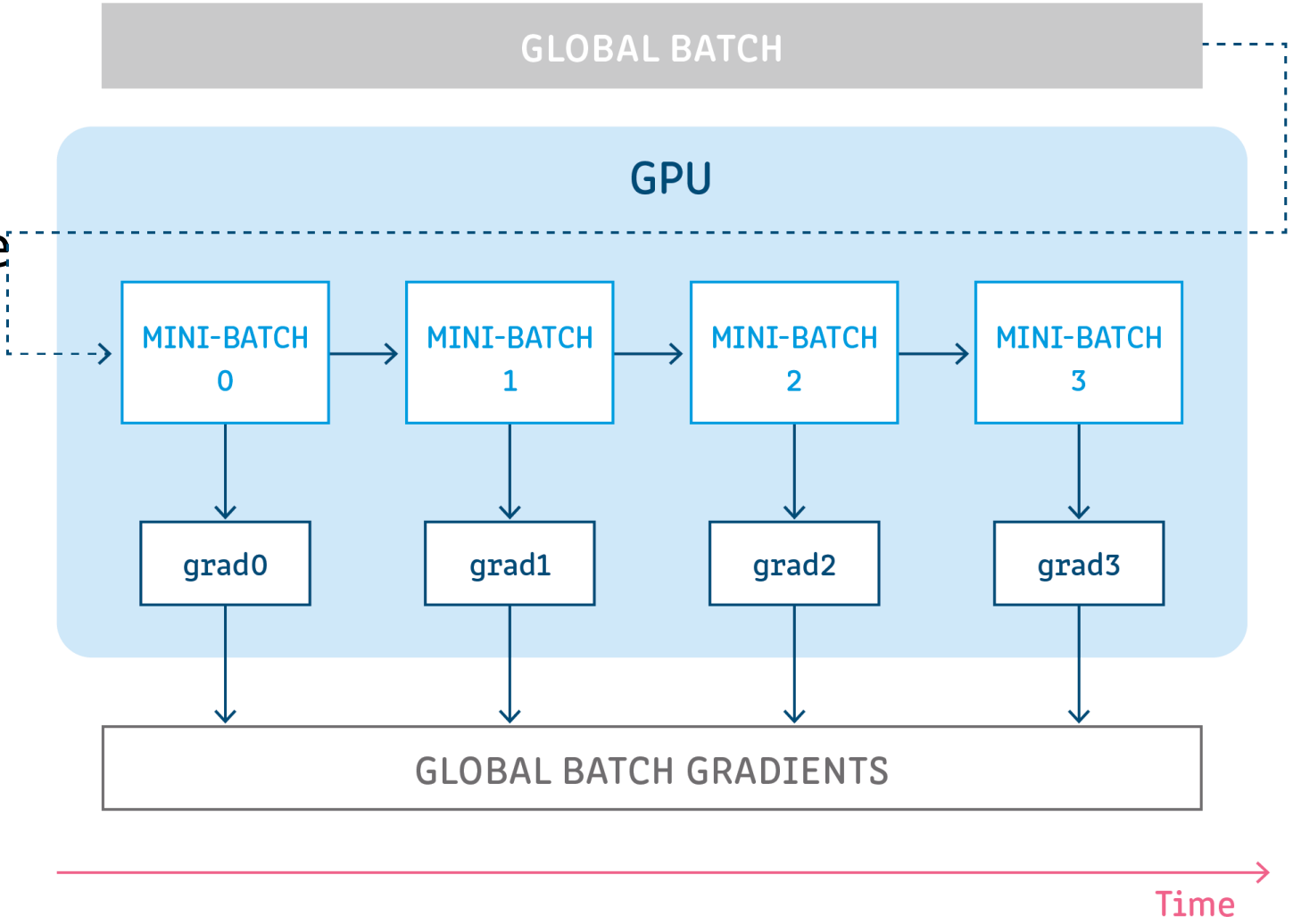
- Neural networks are trained iteratively using optimization techniques like gradient descent.
- After each cycle of training, an error metric is calculated based on the difference between prediction and target.
- The derivatives of this error metric are calculated and propagated back through the network using a technique called backpropagation.
- Each neuron’s coefficients (weights) are then adjusted relative to how much they contributed to the total error.
- This process is repeated iteratively until the network error drops below an acceptable threshold.

Gradient accumulation

- Gradient accumulation is a mechanism to split the batch of samples—used for training a neural network—into several mini-batches of samples that will be run sequentially.
- This is used to enable using large batch sizes that require more GPU memory than available. Gradient accumulation helps in doing so by using mini-batches that require an amount of GPU memory that can be satisfied.
- Gradient accumulation means running all mini-batches sequentially (generally on the same GPU) while accumulating their calculated gradients and not updating the model variables - the weights and biases of the model.

Gradient accumulation

- The model variables must not be updated during the accumulation in order to ensure all mini-batches use the same model variable values to calculate their gradients.
- Only after accumulating the gradients of all those mini-batches will we generate and apply the updates for the model variables.



How to Create and Train Deep Learning Models

- The three most common ways people use deep learning to perform object classification are:
  1. Training from Scratch
     - To train a deep network from scratch, you gather a very large labeled data set and design a network architecture that will learn the features and model.
     - good for new applications, or applications that will have a large number of output categories.
     - Takes lot of time - days or weeks to train

How to Create and Train Deep Learning Models

- The three most common ways people use deep learning to perform object classification are:
  2. Transfer Learning
     - a process involves fine-tuning a pretrained model.
     - start with an existing network, such as AlexNet or GoogLeNet, and feed in new data containing previously unknown classes
     - After making some tweaks to the network, you can now perform a new task, such as categorizing only dogs or cats instead of 1000 different objects.
     - Has advantage of needing much less data (processing thousands of images, rather than millions), so computation time drops to minutes or hours.

How to Create and Train Deep Learning Models

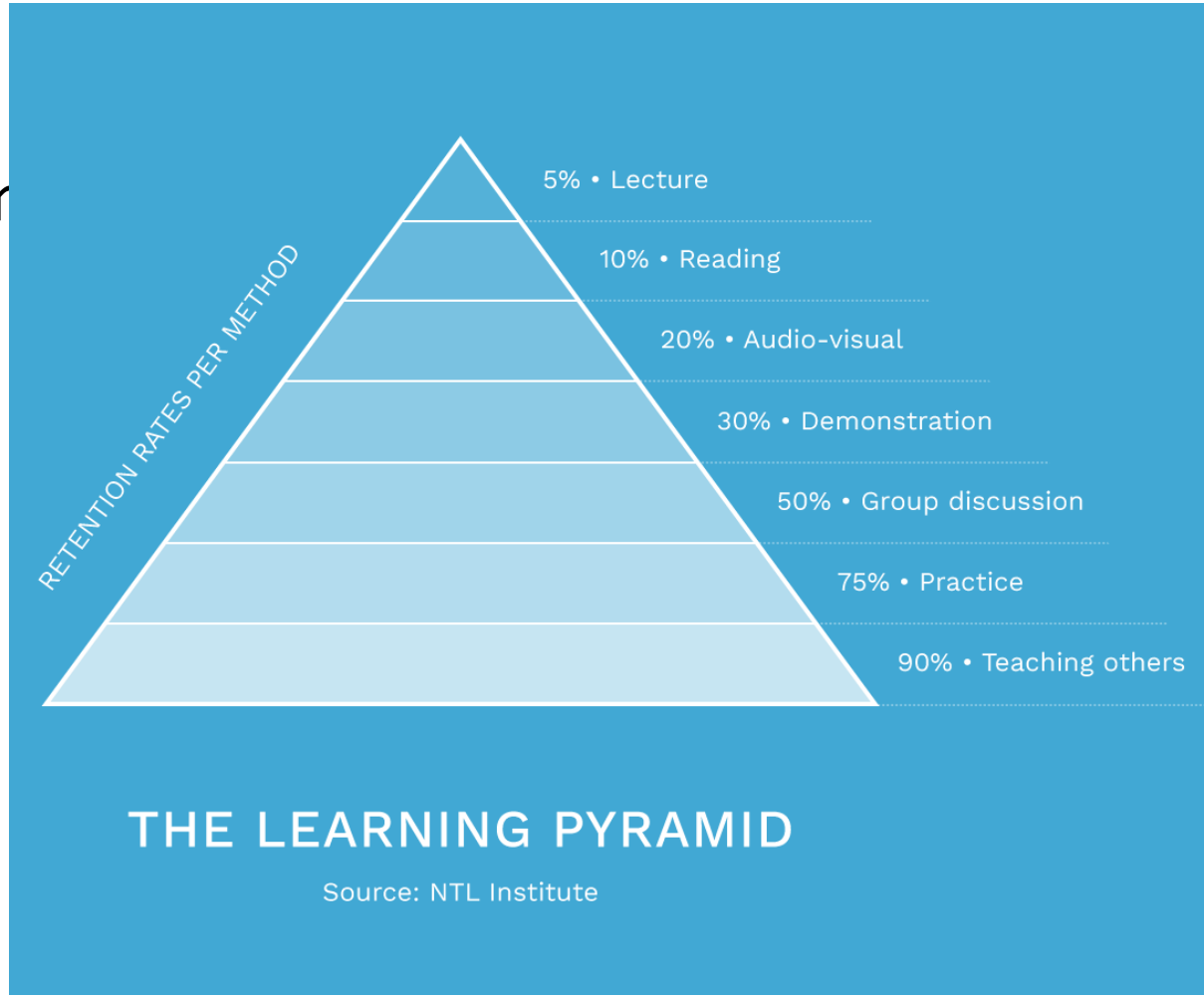
- The three most common ways people use deep learning to perform object classification are:
  3. Feature Extraction
     - slightly less common, more specialized approach to deep learning is to use the network as a feature extractor.
     - Since all the layers are tasked with learning certain features from images, we can pull these features out of the network at any time during the training process.
     - These features can then be used as input to a machine learning model such as support vector machines (SVM).

What do you mean by Hyperparameters in ANN?

- Once the data is formatted correctly, we are usually working with hyperparameters in neural networks. A hyperparameter is a kind of parameter whose values are fixed before the learning process begins.
- It decides how a neural network is trained and also the structure of the network which includes:
- The number of hidden units
- The learning rate
- The number of epochs, etc.

References

- Medium.com - Deeplearning
- neuralnetworksanddeeplearning.com
- Fundamentals of Deep Learning - *Designing Next-Generation Machine Intelligence Algorithms* … Nikhil B
- Website : Wikipedia , kdnuggets, youtube, simplilearn, mathworks
- Dive into Deep Learning {d2l.ai}



Single layer Artificial Neural Network

Steps to follow:

1. Define independent variables and dependent variable
2. Define Hyperparameters
3. Define Activation Function and its derivative
4. Train the model
5. Make predictions