Path to Deep Learning

data is good

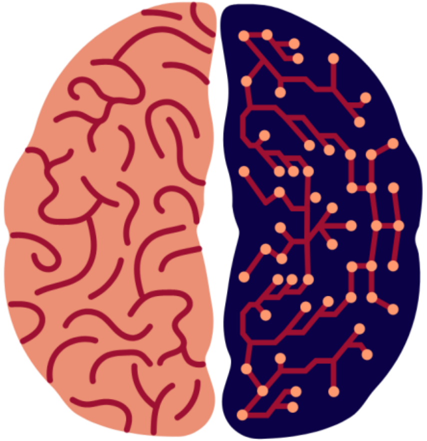
Topics to be Covered….

- Introduction to Neural Networks
- Activation Functions used in Neural Networks
- Optimizers used in Neural Networks
- The Dropout Layer for Neural Networks
- Hyper Parameter Tuning in Neural Networks
- Batch Normalization in Neural Networks
- Implementation of Neural Networks

Introduction to Neural Networks

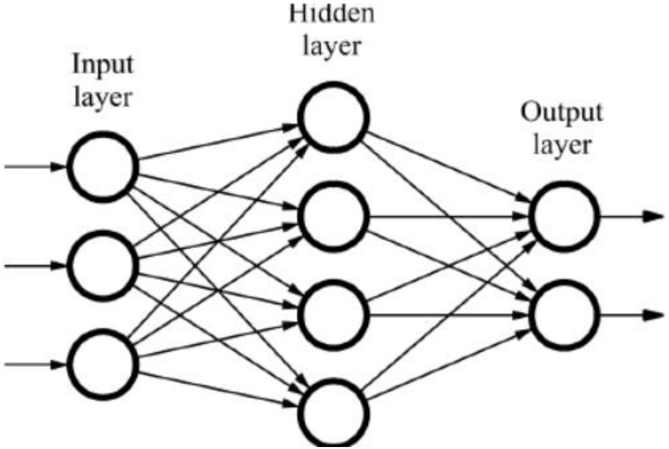
What is a Neural Network?

- A neural network is a series of algorithms that helps us to recognize relationships in a dataset through a process which mimics the way Human brain works.
- It can adapt to Changing Input and generate the Best Results for us.



What are the Components of a Neural Network?

- There are 3 Major Components in a Neural Network:
    - The Input Layer
    - The Hidden Layer
    - The Output Layer

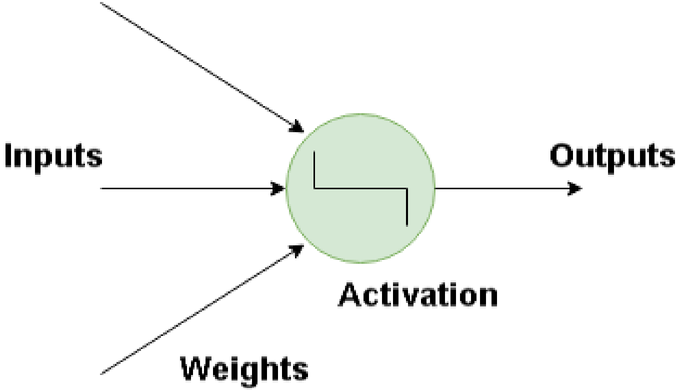


Introduction to Activation Functions

data is good

What is an Activation Function?

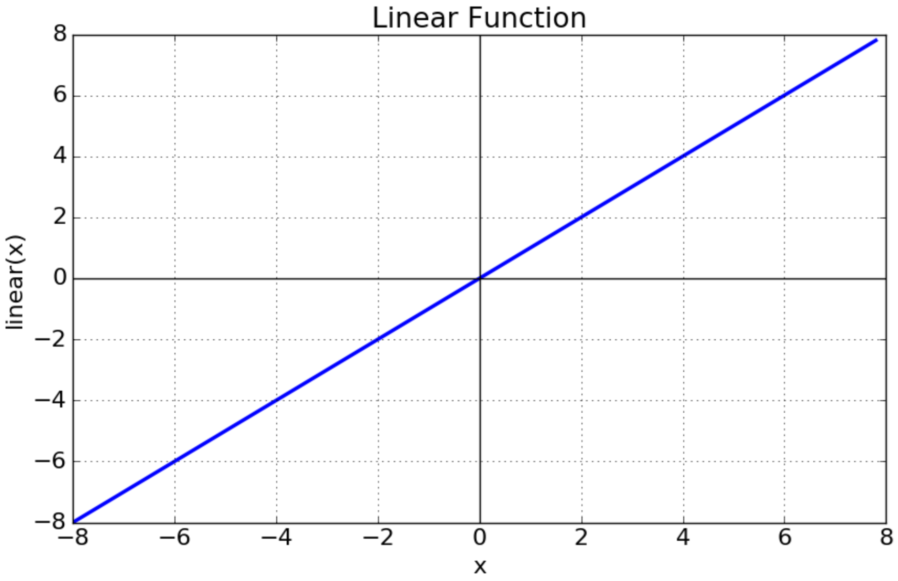
- Activation functions are mathematical equations that determine the output of a neural network.
- It help us to determine the output of the program by normalizing the output values in a range of 0 to 1 or -1 to 1.



Types of Activation Function

Linear Activation Function

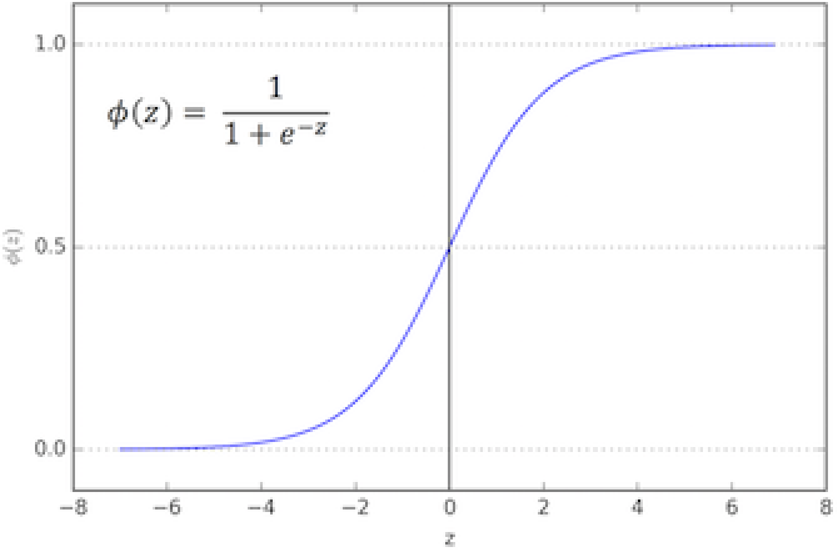
- The **Equation for this line would be** f(x) = x, and the range would be from -infinity to +infinity.



Types of Activation Function

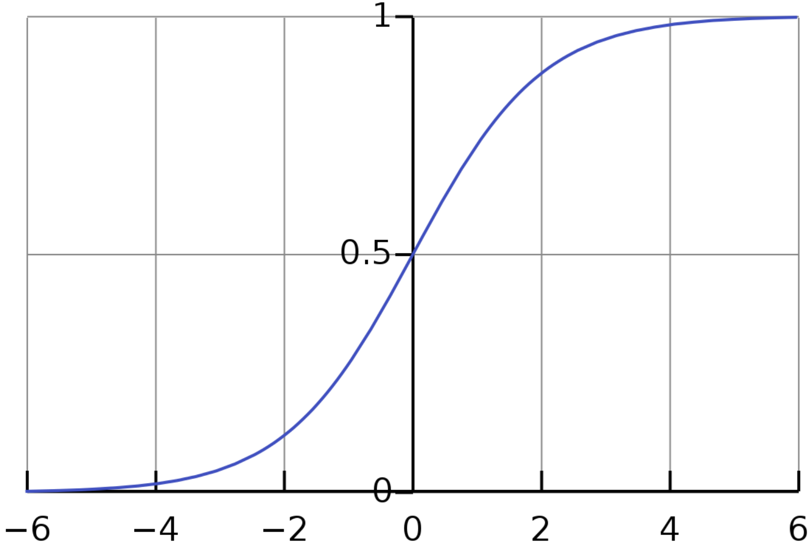
Non Linear Activation Function

- These Nonlinear Functions makes it easy for the model to **generalize or adapt with a variety of data** and to differentiate between the output.



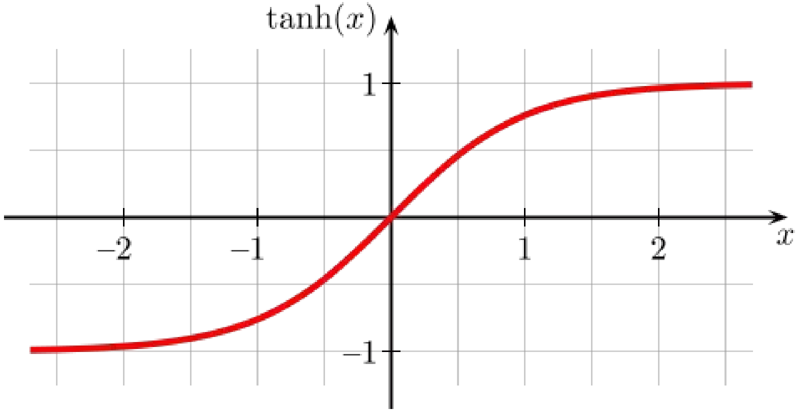
Sigmoid Activation Function

- A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve.
- The Main reason behind using the Sigmoid function is that this function exists between 0 to 1.



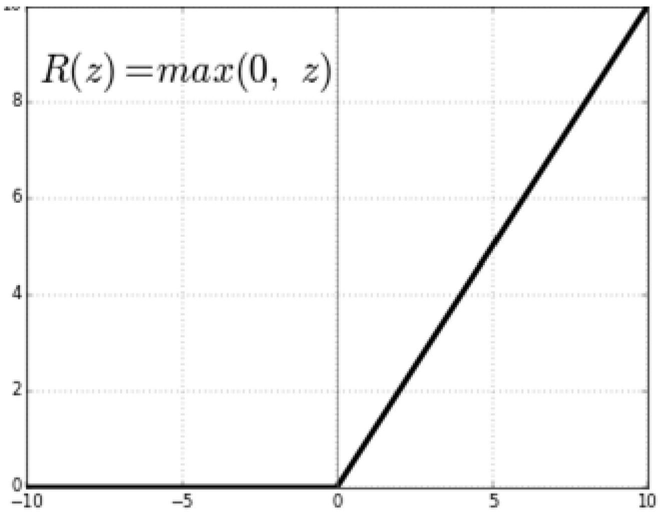
Tanh Activation Function

- The Range of the tanh Function is -1 to +1 as you can see in the Graph.
- Tanh activation has large area under better slope compared to sigmoid, this helps models using Tanh activation to learn better.



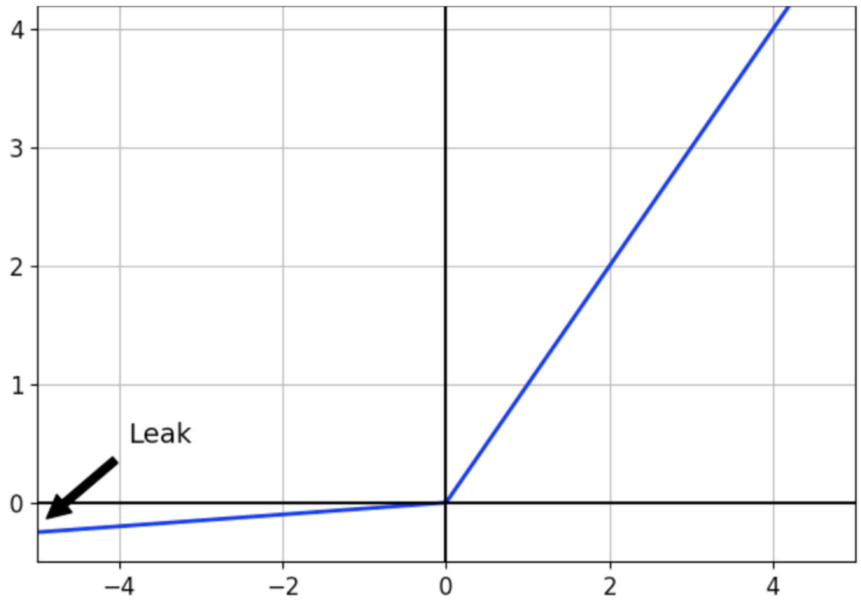
Relu Activation Function

- It is also known as Rectified Linear Unit Activation function.
- It is half rectified from bottom, which means If you feed any negative data to Relu, It will return you a Zero, and If you feed any positive data, It will return you the exact number



Leaky Relu Activation Function

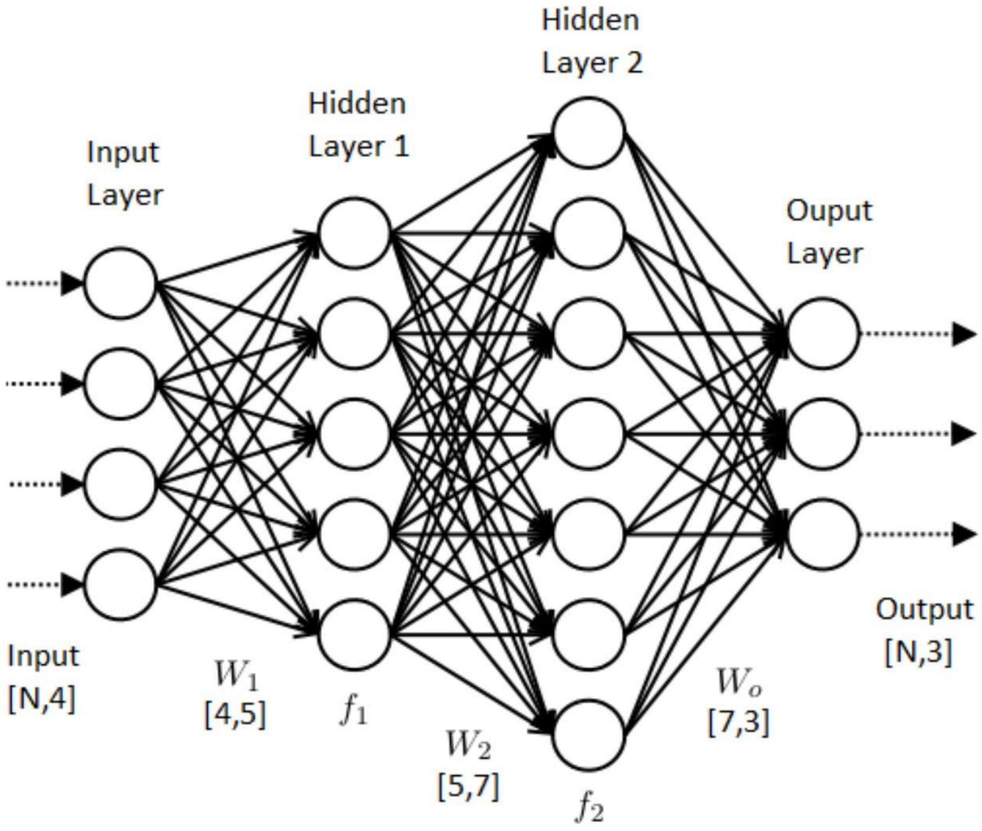
- Any negative input given to ReLU turns the value into zero which in turns affects the resulting graph by not mapping the negative values appropriately.
- This is the reason Relu was reintroduced with a Leak to Increase the Range of Learning.



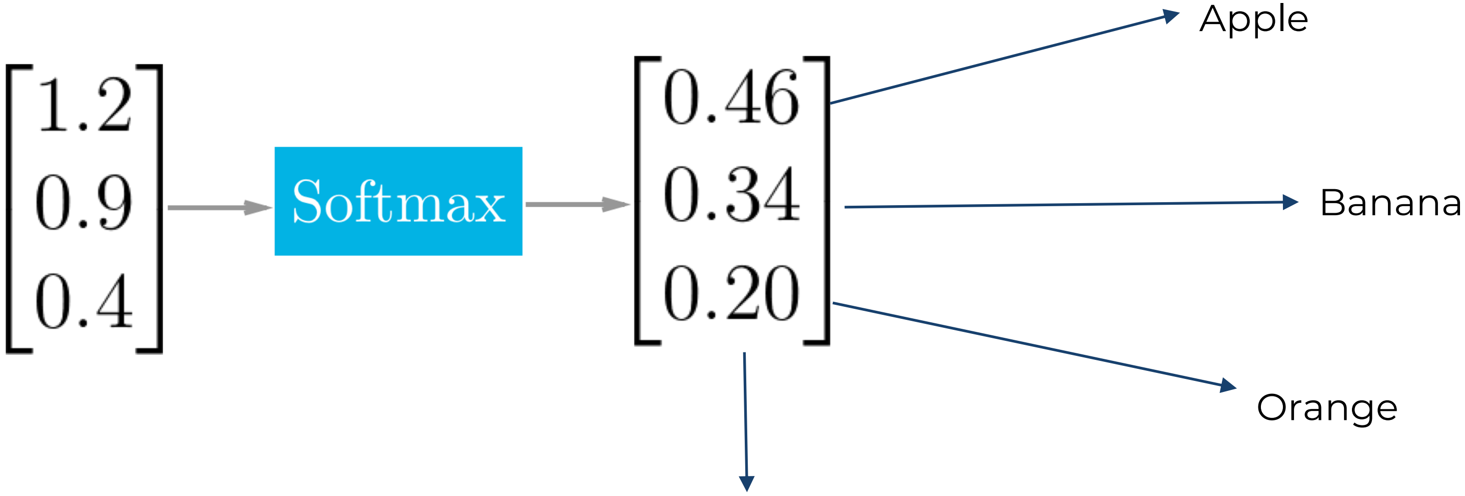
Softmax Activation

- The formula for Softmax Activation is : -

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$


Hidden
Layer 2
Hidden
Layer 1
Input
Layer
Ouput
Layer
Output
[N,3]
Input
[N,4]
W_1
[4,5]
f_1
W_2
[5,7]
f_2
W_o
[7,3]

data is good



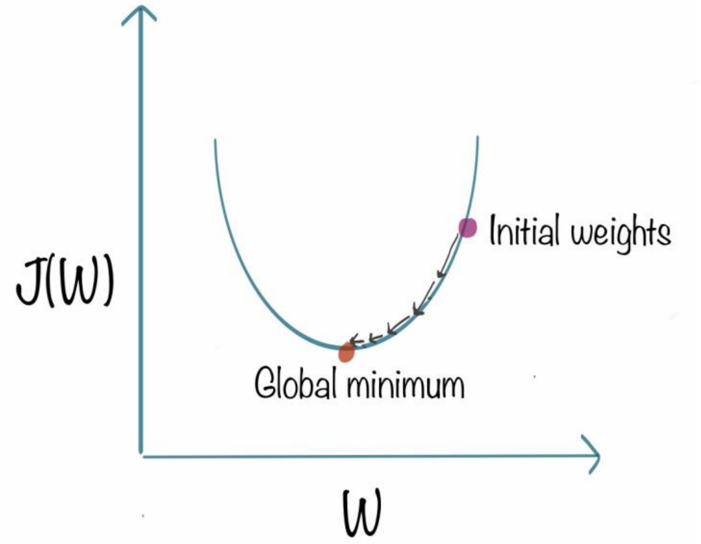
$$\begin{bmatrix}1.2\\0.9\\0.4\end{bmatrix} \rightarrow \boxed{\text{Softmax}} \rightarrow \begin{bmatrix}0.46\\0.34\\0.20\end{bmatrix}$$

The Total Sum of Probabilities is 1, and as Apple has Highest Probability, **The answer will be Apple.**

Introduction to Gradient Descent

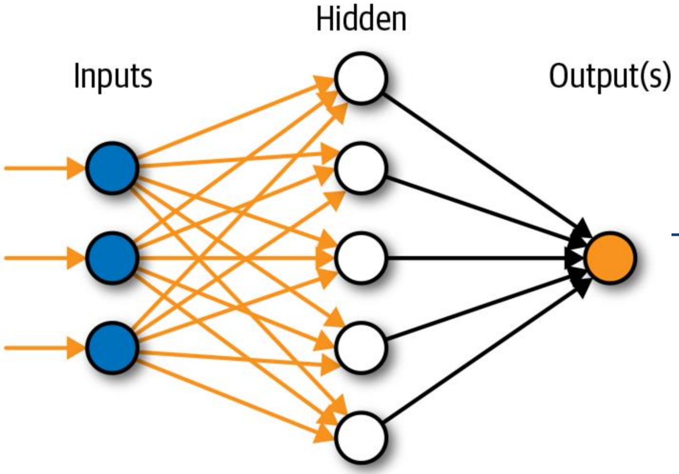
What is a Gradient Descent?

- Gradient Descent also known as “Back propagation” is an optimization technique that is used to improve deep learning and neural network-based models by minimizing the loss function.

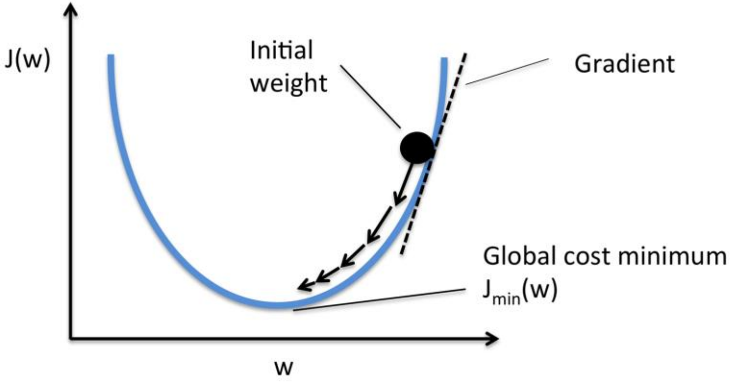


Loss Function

- Loss function quantifies the amount of error in your predictions, It is a metric that is directly related to the performance of the model.
- If Predictions are Good, then the Loss function would output a Lower Number and Vice Versa.
- Our Loss Function should always Justify two things that is
    - It should always be low.
    - It should be differentiable.



Artificial Neural Network



Initially, Our Neural Network has a High Loss,
Then slowly slowly Weights are Updated to Minimize the Loss.

Batch Gradient Descent

- We take the whole Training Data as a batch and pass it to the network. The network calculates the Gradient for whole Training Data and Updates the weights.
- **Advantage:** Speed
- **Disadvantage:** Poor Learning in Initial Phases, High Memory Consumption.

Stochastic Gradient Descent

Here we take each data point and calculate its gradient.

Next we update weights and repeat this process.

So if our dataset has 1000 data points, using stochastic gradient descent we will calculate gradients and update weight 1000 times in one epoch.

- **Advantage:** Low Memory Consumption
- **Disadvantage:** Low Speed

Mini Batch Gradient Descent

Here we internally batch the training data, we then calculate gradient and update weights for each batch.

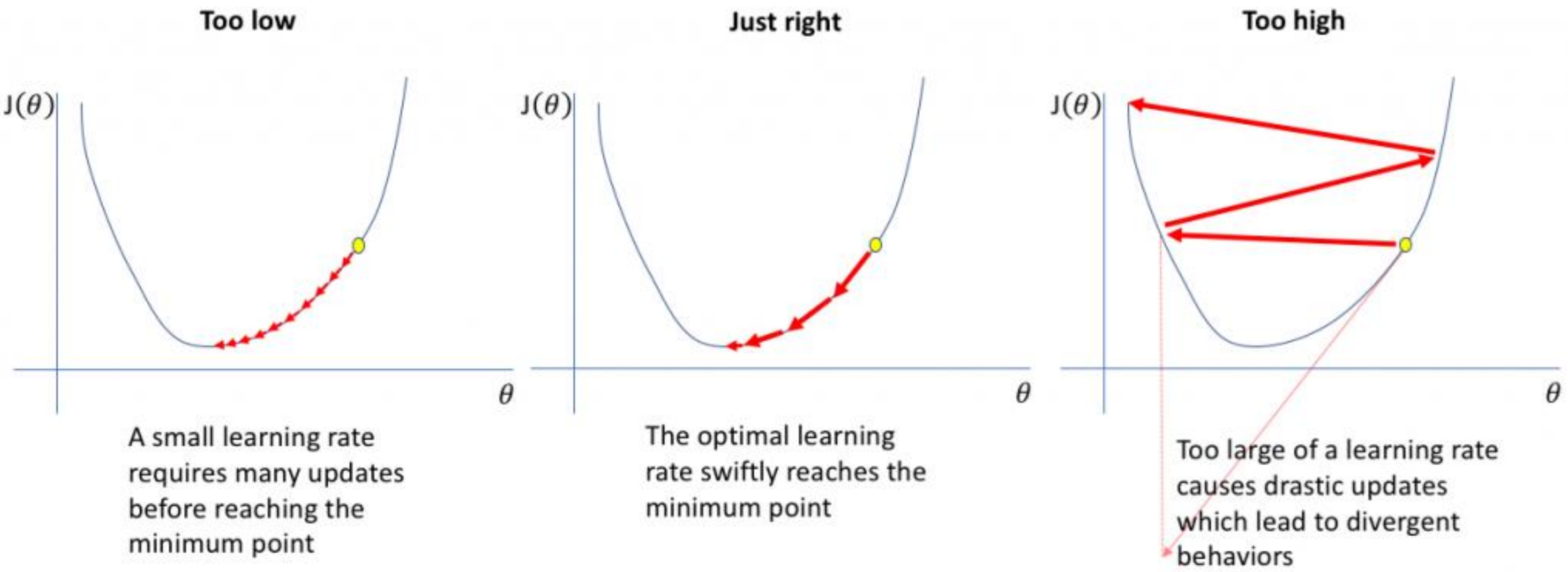
We repeat this process until gradients over whole training data is calculated and weights are updated.

So if our dataset has 1000 examples and our batch size is suppose 100, then we update weights 10 times for each epoch.

- **Advantage:** Low Memory Consumption, High Speed

Introduction to Optimizers

Why do we need Optimizers?



Too low

A small learning rate requires many updates before reaching the minimum point

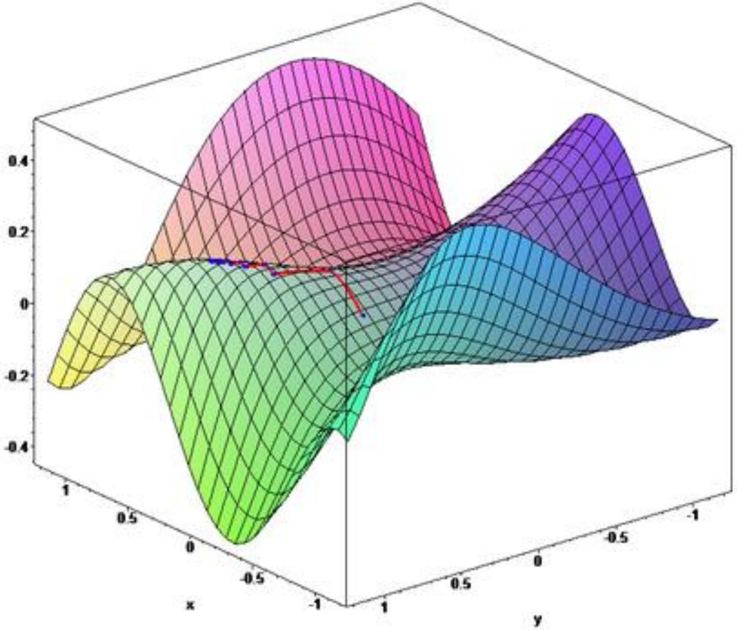
Just right

The optimal learning rate swiftly reaches the minimum point

Too high

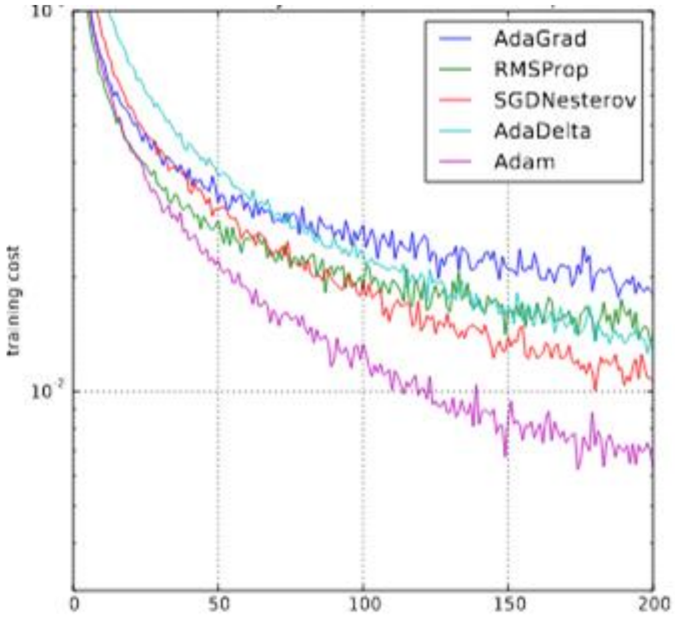
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Loss vs Weight Curves in 3 Dimensions



Optimizers in Tensorflow

- Momentum
- RMS Prop
- Adagrad
- AdaDelta
- Adam



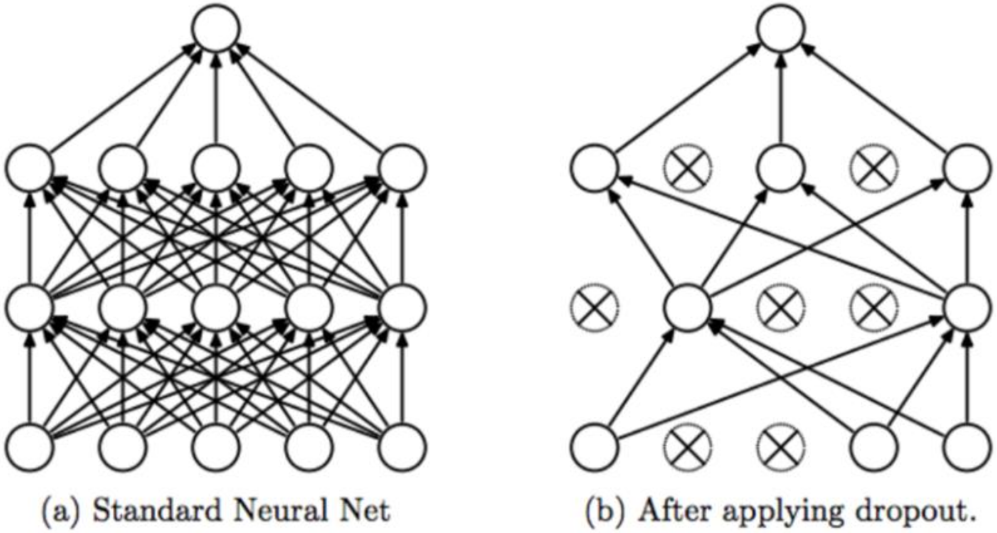
A chart that explains the loss to epochs comparison of different optimizers.

Adam Seems to be the Best One!

Why do we need Dropout

What is Dropout?

- Dropout is an approach to Regularization in Neural Networks which helps in Independent Learning of the Neurons.
- It is Highly effective in avoiding Overfitting



(a) Standard Neural Net

(b) After applying dropout.

Common Observations While using Dropout for Avoiding Overfitting in Neural Networks

1. Dropout Forces the Neural Network to learn more Robust Features that are Useful for Predictive analytics.
2. Dropout Helps us to Reduce the Training Time required for the Neural Network.
3. Choosing the Right value for Dropout is crucial to get Good Results.

Values for Dropout

- Values for Dropout Vary from 0 to 1.
- If zero is specified, it means no dropout is required.
- 0.2 means 20% of all the Neurons from Hidden Layer is removed.
- 0.5 means 50% of all the Neurons are removed at random from the Specified Hidden Layer.

Hyperparameter Tuning

Tuning our Neural Networks

Important Parameters to be considered while tuning a Neural Network

- Number of Layers in a Neural Network
- The Dropout Value
- Learning Rate of the Neural Network
- The Batch Size

Choosing the Right Number of Hidden Layers in a Neural Network

- Choose **3 to 5** Hidden Layers for a Medium Sized Dataset.
- Neural Networks Might Suffer from **Underfitting** if the hidden layers are less than 2.
- Similarly, the Neural Networks can suffer from **Overfitting** if the Number of Hidden layers are too high.
- For Bigger Problems, Slowly Increase the Hidden Layers and Check see if Increasing Hidden Layers optimizes the Neural Network.

How to Choose the Right Value for Dropout in Neural Networks?

- Most of the Times, the Value of **0.2 to 0.4** is the Most preferable for **Dropout**, which means deactivating 20 to 40% of the Neurons from the Hidden Layers.
- Always Try to Maintain a Dropout Value **lesser than 0.5**, As Higher than 0.5 Value for Dropout can lead to **Overfitting.**
- **Dropout values Less than 0.2** Value will have no or Little Significance.

How to choose the Learning rate for the Neural Networks?

- Optimizing the value of a Learning Rate is considered to be the **Most Important step in Optimization of a Neural Network.**
- **If the learning rate is low**, then training is more reliable, but optimization will take a **lot of time**, and If the **learning rate is high,** then training may **not converge or even diverge.**
- The Most common learning rates in Neural networks are **0.01 or 0.001.**

How to select the Batch Size in Neural Networks?

- Generally **The Batch Size of 32 is Good,** But we should also try the Batch Sizes of 64, 128, and 256.
- If your **problem statement is complex,** having **low batch size works better** as low batch size and mini batch gradient descent we will have more weight updates which ensures our weights are better tuned.
- If your **problem statement is simple,** then try to have the **maximum batch size** that can fit your memory.

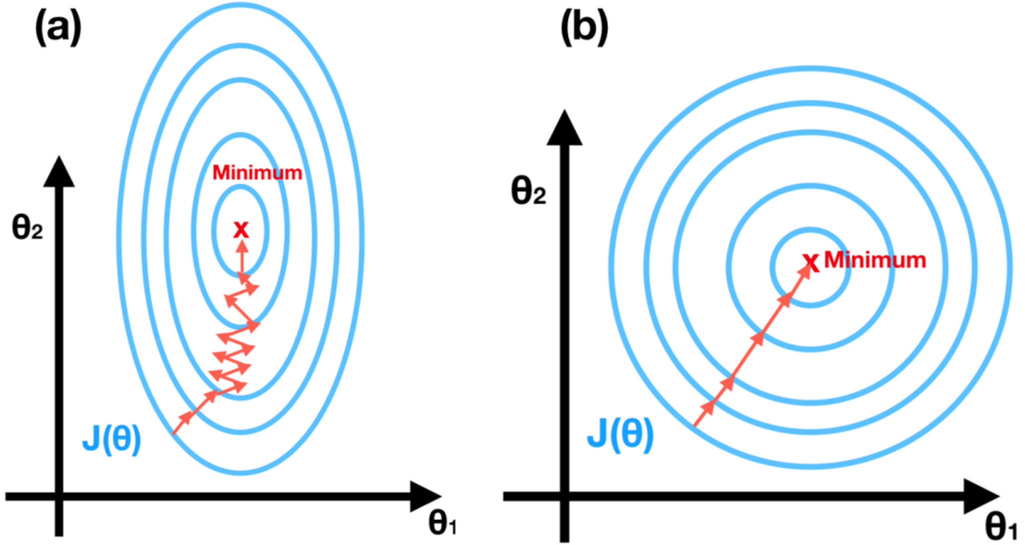
Introduction to Batch Normalization

What is Batch Normalization?

- Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch.
- This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

Here, the Features are **Non-Normalized**

As the Values of Both Age and Salary are not Normalized.



Here, the Features are **Normalized.**

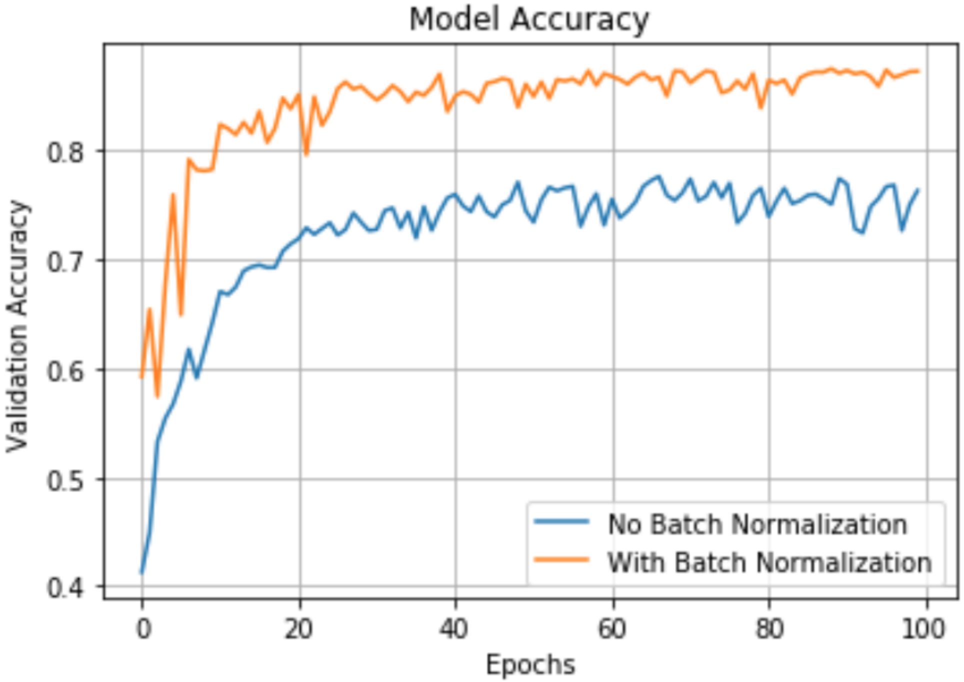
As the Values of Both Age and Salary lie from 0 to 1.

On the left side we have a loss curve using Non normalized features,
On the right we have a loss curve with normalized features.

Let’s Suppose We have Two Features **Age and Salary.**
Their scales would be 1 to 100 and 10000 to a big number respectively.

Advantages of Batch Normalization

- Faster Training
- Better Accuracy


Model Accuracy
Validation Accuracy
Epochs
No Batch Normalization
With Batch Normalization