Weight Initialization

- The weight initialization technique you choose for your neural network can determine how quickly the network converges or whether it converges at all. Although the initial values of these weights are just one parameter among many to tune, they are incredibly important. Their distribution affects the gradients and, therefore, the effectiveness of training.

Why is weight initialization important? ¶

- Improperly initialized weights can negatively affect the training process by contributing to the vanishing or exploding gradient problem.
- With the vanishing gradient problem, the weight update is minor and results in slower convergence — this makes the optimization of the loss function slow and in a worst case scenario, may stop the network from converging altogether.
- Conversely, initializing with weights that are too large may result in exploding gradient values during forward propagation or back-propagation.

1. Zero initialization :

- If all the weights are initialized with 0, the derivative with respect to loss function is the same for every weight(w), thus all weights have the same value in subsequent iterations.
- This makes hidden units symmetric and continues for all the n iterations i.e. setting weights to 0 does not make it better than a linear model.
- An important thing to keep in mind is that biases have no effect what so ever when initialized with 0.
- It also gives problems like vanishing gradient problem.

2. initialization With -ve Number :

- If all weigth can be negative then it affect Relu Activation Function.As in -ve Relu comes under dead activation problem. so we cant use this technique.
- Weights can’t be too high as gives problems like exploding Gradient problem(weights of the model explode to infinity), which means that a large space is made available to search for global minima hence convergence becomes slow.

To prevent the gradients of the network’s activations from vanishing or exploding, we need to have following rules:

```
        1. The mean of the activations should be zero.
        2. The variance of the activations should stay the same across e
very layer.
```

Idea 1 : Normal or Naïve Initialization:

- In normal distribution weights can be a part of normal or gaussian distribution with mean as zero and a unit standard deviation.

$$W \approx N(0, \sigma) \quad \mu = 0 \qquad \sigma = \text{small number}$$

- Random initialization is done so that convergence is not to a false minima.
- In Keras it can be simply written as hyperparameter as - kernel_initializer='random_normal'

Idea 2: Uniform Initialization:

- In uniform initialization of weights , weights belong to a uniform distribution in range a,b with values of a and b as below:

$$W \approx U(a,b) \qquad a = \frac{-1}{\sqrt{f\_in}} \quad , \quad b = \frac{1}{\sqrt{f\_in}}$$

- Whenever **sigmoid** activation function is used as , Uniform works well.
- In Keras it can be simply written as hyperparameter as - kernel_initializer='random_uniform'

Idea 3: Xavier/ Glorot Weight Initialization:

- The variance of weights in the case normal distribution was not taken care of which resulted in too large or too small activation values which again led to exploding gradient and vanishing gradient problems respectively, when back propagation was done.
- In order to overcome this problem Xavier Initialization was introduced. It keeps the variance the same across every layer. We will assume that our layer's activations are normally distributed around zero.
- Glorot or Xavier had a belief that if they maintain variance of activations in all the layers going forward and backward convergence will be fast as compared to using standard initialization where gap was larger.
- It have Two Varient

```
a. Normal Distribution - kernel_initializer='glorot_normal'
b. Uniform Distribution - kernel_initializer='glorot_uniform'
```

Point : Works well with **tanh , sigmoid** activation functions.

a. Normal Distribution:

- In Normal Distribution, weights belong to normal distribution where mean is zero and standard deviation is as below:

$$W \approx N(\mu, \sigma)$$

$$\mu = 0 \qquad \sigma = \sqrt{\frac{2}{f\_in + f\_out}}$$

b. Uniform Distribution:

- Uniform Distribution , weights belong to uniform distribution in range of a and b defined as below:

$$W \approx U(a,b)$$

$$a = -\sqrt{\frac{6}{f\_in + f\_out}}, \qquad b = \sqrt{\frac{6}{f\_in + f\_out}}$$

Idea 4: He-Initialization:

- When using activation functions that were zero centered and have output range between-1,1 for activation functions like tanh and softsign, activation outputs were having mean of 0 and standard deviation around 1 average wise.
- But if ReLu is used instead of tanh, it was observed that on average it has standard deviation very close to square root of 2 divided by input connections.
  - It have Two Varient

```
a. Normal Distribution - kernel_initializer='he_normal'
b. Uniform Distribution - kernel_initializer='he_uniform'
```

Point : Works well with **Relu And Leaky Relu** activation functions.

a. Normal Distribution:

- In He-Normal initialization method, weights belong to normal distribution where mean is zero and standard deviation is as below:

$$W \approx N(\mu, \sigma)$$

$$\mu = 0 \qquad \sigma = \sqrt{\frac{2}{f\_in}}$$

b. Uniform Initialization :

- In He Uniform Initialization weights belong to uniform distribution in range as shown below:

$$W \approx U(a,b) \qquad a = -\sqrt{\frac{6}{f\_in}} \quad , \quad b = \sqrt{\frac{6}{f\_in}}$$