**TUTORIAL - 2**

**Q1. Explain the reasons for vanishing and exploding gradients.**

Vanishing gradient and exploding gradients problem occurs when were training the neural network using the backpropagation algorithm. In backpropagation we work from the output layer to the input layer, changing the error gradient as we move along with the layers. In each layer parameter updation takes place, gradients are used to update each parameter.

As the algorithm progresses this gradient gets smaller and smaller due to which the weights of the lower layer remain unchanged, this is called vanishing gradient.

Exploding gradient problem occurs when the gradient grows bigger and bigger due to which there are very large weight updation.

**Q2. Discuss the solution proposed by Xavier Glorot and Youshua Bengio. Explain the solution logically.**

For vanishing and exploding gradient a solution was proposed which stated that we can use the combination of sigmoid function and weight initialization techniques, which means that the variance of the output of each layer is much greater than the variance of the input layers. As the training will progress in the network the variance will keep on increasing after each layer until the activation function saturates at the top layers i.e., the function will show 0 or 1 when the derivative is very close to 0 or 1.

For this problem Glorot and Bengio proposed a solution which stated that for the proper flow of signal in both forward and backward propagation we need the variance of the output of each layer to be equal to the variance of the input. Along with this the connection weights much be initialized randomly.

**Q3. Write a Python Code for any weight initialization approach.**

| **import numpy as np def glorot\_init(fanIn,fanOut):  limit = np.sqrt(6/(fanIn+fanOut))  return np.random.uniform(-limit, limit, (fanIn,fanOut))  def bengio\_init(fanIn,fanOut):  std = np.sqrt(2/(fanIn+fanOut))  return np.random.normal(0,std, (fanIn,fanOut))  nInput = 10 *# input neurons of the layer* nOutput = 5 *# output neurons of the layer*  glorot\_wts = glorot\_init(nInput, nOutput) bengio\_wts = bengio\_init(nInput, nOutput)** |
| --- |

**print(glorot\_wts)**

[[ 0.26640277 -0.5829163 0.4017685 0.22171642 -0.49860008]

[-0.43088413 -0.20522745 -0.60789111 -0.01119747 0.62348791]

[-0.59343952 -0.301156 0.14499118 0.13652263 -0.43700195]

[ 0.07221451 0.44567146 0.28976111 0.56314179 0.47848197]

[ 0.36634348 0.52377333 -0.61225797 0.13297402 -0.52363347]

[-0.11773296 0.57834973 0.5930481 -0.3037868 -0.22838053]

[ 0.2508548 0.38681998 -0.15359628 -0.08349468 0.20706978]

[ 0.0555894 -0.27952795 -0.6138133 -0.06647183 0.17411591]

[ 0.32380493 0.20319937 0.07452834 -0.29054958 -0.04507877]

[ 0.48263925 0.06671748 -0.31318962 0.40105043 -0.00088912]]

print(bengio\_wts)

[[ 0.28811706 -0.00880679 0.02134329 0.35068897 -0.2534258 ]

[ 0.28318611 0.41210322 0.13004117 -0.0543082 -0.09090712]

[-0.42743403 0.14433276 -0.36835141 -0.00646561 0.03830652]

[ 0.10686924 0.02680771 0.33761485 -0.39602094 0.1268665 ]

[ 0.74466015 -0.64561965 -0.25392097 0.73668992 0.16298868]

[ 0.17217357 0.24687328 0.04128906 -0.16657677 0.42071763]

[-0.41425832 0.00459657 0.44614565 0.27860447 0.69624667]

[-0.21042869 0.42490259 0.28357943 -0.07565457 0.24320256]

[-0.1917123 -0.21571808 0.46686724 -0.31376654 -0.4196143 ]

[ 0.13078148 -0.07136081 -0.24055556 -0.37306126 -0.29708132]]