# Backpropagation learning by Cross Entropy Loss Function

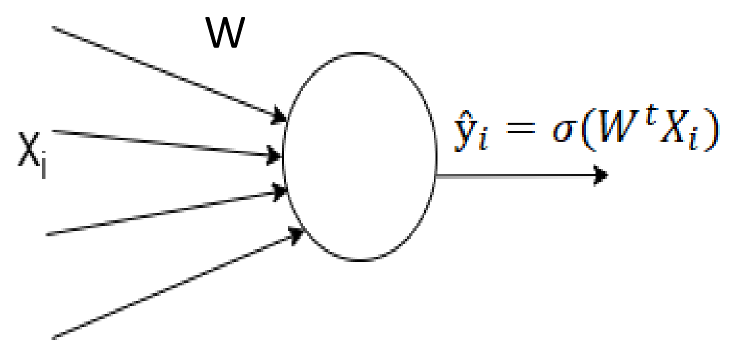
- We have discussed about Backpropagation learning in Multi Layer Feed Forward Neural Network.
- The cost or error function to be minimized was sum of square error or quadratic error.
- The weight updation rule that we have derived was based on the sigmoid activation function, which is

$W_{ij}{}^{K} = W_{ij}{}^{K} - \eta\, \delta_j{}^{K}\, O_i{}^{K-1}$ where $\delta_j{}^{K} = (O_j{}^{K} - t_j)\, O_j{}^{K}(1 - O_j{}^{K})$

- For small or large values of sigmoid activation function, $\delta_j{}^{K} \rightarrow 0$, which gives very small gradient. Which means either the learning process is very slow or it stops when gradient vanishes.
- This is one of the major problem of using sum of square error or quadratic error function.

Cross Entropy Loss Function for binary class problem

- Let us consider Cross Entropy Loss for 2 class problem, i.e., y =1 and y=0.
- Let, for a training sample $X_i$, actual class is y=1 and the predicted value is $\hat{y}$ , so $\hat{y}$ is the likelihood that the actual class of $X_i$ is y=1.




- So 1- $\hat{y}$ is the likelihood that the actual class of $X_i$ is y=0.

# Cross Entropy Loss Function for binary class problem

- Combining these two, we get likelihood to be maximized is equal to $\hat{y}^{y}\,(1-\hat{y})^{1-y}$
- Loglikelihood to be maximized to train the network is:

$$y \log \hat{y} + \ (1-y)\log(1-\hat{y}), where\ \hat{y} = \ \sigma(\theta) and\ \theta = W^{t}X_i$$

- If $X_i$ is in class y=1, we have to maximize $y \log \hat{y}$ and if X is in class y=0, we have to maximize $(1-y)\log(1-\hat{y})$
- From the likelihood we can define the cross entropy loss function to be minimized, as follows:

$$C = -\frac{1}{N}\sum_{\forall X_i} y \log \hat{y} + (1-y)\log(1-\hat{y})$$

- This is called binary cross entropy as there are two classes.

- So we have taken the summation over all N training samples and take the average, which gives us the cross entropy loss function C which is to be minimized as we consider negative sign.

# Backpropagation Learning using Cross Entropy Loss Function for binary class problem

- We will minimize $C = -\frac{1}{N}\sum_{\forall X_i} y\log\hat{y} + (1-y)\log(1-\hat{y})$ using gradient descent approach, where,

  $\hat{y} = \sigma(\theta) and\ \theta = W^t X_i$

- The gradient is: $\frac{\partial C}{\partial W} = \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial \theta}\frac{\partial \theta}{\partial W}$
- $\frac{\partial C}{\partial \hat{y}} = -\frac{1}{N}\sum_{\forall X_i}\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} = -\frac{1}{N}\sum_{\forall X_i}\frac{y}{\sigma(\theta)} - \frac{1-y}{1-\sigma(\theta)} = -\frac{1}{N}\sum_{\forall X_i}\frac{y-\sigma(\theta)}{\sigma(\theta)(1-\sigma(\theta))}$
- $\frac{\partial \hat{y}}{\partial \theta} = \sigma(\theta)(1-\sigma(\theta))$ and $\frac{\partial \theta}{\partial W} = X_i$

$$\frac{\partial C}{\partial W} = \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial \theta}\frac{\partial \theta}{\partial W} = -\frac{1}{N}\sum_{\forall X_i}\frac{y-\sigma(\theta)}{\sigma(\theta)(1-\sigma(\theta))}(\sigma(\theta)(1-\sigma(\theta)))X_i$$

$$= \frac{1}{N}\sum_{\forall X_i} X_i(\sigma(\theta)-y) = \frac{1}{N}\sum_{\forall X_i} X_i(\hat{y}-y)$$

$$So, the\ weight\ updation\ rule\ is: W = W - \boldsymbol{\eta}\frac{\partial C}{\partial W} = W - \boldsymbol{\eta}\frac{1}{N}\sum_{\forall X_i} X_i(\hat{y}-y)$$

# Backpropagation Learning using Cross Entropy Loss Function for binary class problem

$$So, the\ weight\ updation\ rule\ is: W = W - \boldsymbol{\eta}\ \frac{\partial C}{\partial W} = W - \boldsymbol{\eta}\ \frac{1}{N}\sum_{\forall X_i} X_i(\hat{y} - y)$$

- Let us consider, stochastic gradient descend approach, then

$$W = W - \boldsymbol{\eta}\ \frac{1}{N} X_i(\hat{y} - y)$$

- If $X_i$ is of class y=1, and misclassified as ŷ = 0, then weight increases, and
- If $X_i$ is of class y=0, and misclassified as ŷ = 1, then weight decreases.
- But in case of sum of square or quadratic error function, the weight updation rule is: $W = W - \eta \hat{y}_i(1 - \hat{y}_i)(\hat{y}_i - y_i)X_i$
- Here, if activation function value, i.e., $\hat{y}_i$ is very small or very large, then the gradient almost vanishes, which provides slow learning to the network.

# Backpropagation Learning using Cross Entropy Loss Function for binary class problem

- The cross entropy loss function is : $W = W - \eta \frac{1}{N} X_i(\hat{y} - y)$

- If activation function value, i.e., the predicted value $\hat{y}_i$ is very large when y=0 then the difference ($\hat{y}$ - y) is very large, and when $\hat{y}$ is very small for y=1 then the absolute difference ($\hat{y}$ - y) is also very large.
- Therefore, the learning rate is proportional to the absolute value of the difference of actual and predicted value.
- So, if the error is more then the rate of learning is more.
- But in sum of square error, if error is more, i.e., $\hat{y}$ =0 for y=1; or $\hat{y}$=1 for y=0, then updation portion or step size of updation (contains $\hat{y}$ (1- $\hat{y}$)) is very less (since, $\hat{y}$ is the sigmoid function value). So the learning rate is very slow, even may be stopped when gradient vanishes.
- Thus the cross entropy loss is advantegious over quadratic error.

# Backpropagation Learning using Cross Entropy Loss Function for Multiclass problem

# Cross Entropy Loss Function for Multiclass problem

$$M_k = No.\,of\ neurons\ in\ the\ k-th\ layer, k = 0, 1, \dots, K$$

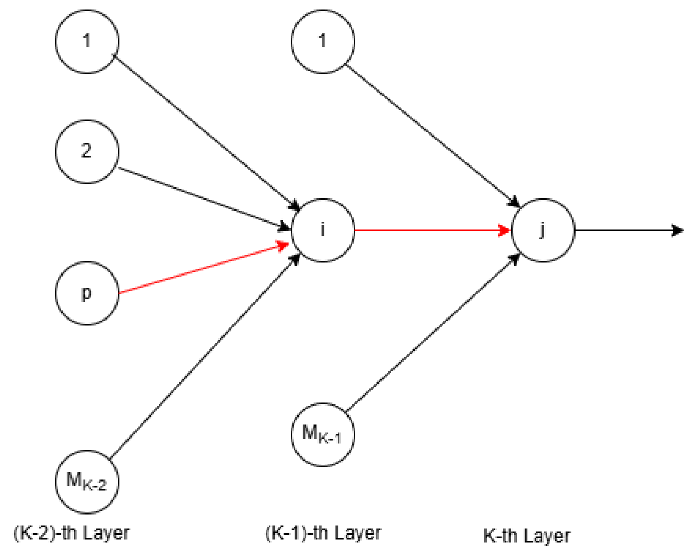
$$o_j^{(k)} = output\ of\ j-th\ neuron\ at\ k-th\ layer$$

$$w_{ij}^{k+1} = weight\ of\ the\ connection\ between\ i-th\ neuron\ at\ k-th\ layer$$
$$and\ j-th\ neuron\ at\ (k+1)-th\ layer$$

$$Each\ training\ data\ is\ of\ the\ form\ (X_i, y_i)\quad where\ y_i = any\ one\ of\ 1, 2, \dots, M_K$$

$$X_i = i-th\ feature\ vector, i = 1, 2, \dots, N$$

- In multiclass problem where we are using cross entropy, we assume that the outputs of the output layer neurons are obtained in a probabilistic manner.
- That is the outputs are softmax output, which we can obtain by using the softmax classifier at the output layer.
- It gives the normalized probability that input vector X belongs to class $t_j$.



# Cross Entropy Loss Function for Multiclass problem

- Let, $O_j^K$ be the likelihood that the sample X belongs to class $t_j$
- (1- $O_j^K$) is the likelihood that X is in class $(1 - t_j)$
- So, for j-th output node, the cross entropy loss is:
  $-[t_j \log O_j^K + (1 - t_j) \log(1 - O_j^K)]$
- Considering all the output nodes, we get the overall cross entropy loss: $-\sum_{j=1}^{M_K} [t_j \log O_j^K + (1 - t_j) \log(1 - O_j^K)]$
- So, considering all the feature vectors, we get the final cross entropy loss function as:

$$C = -\frac{1}{N} \sum_{\forall X} \sum_{j=1}^{M_K} [t_j \log O_j^K + (1 - t_j) \log(1 - O_j^K)]$$

# Backpropagation Learning using Cross Entropy Loss Function for Multiclass problem

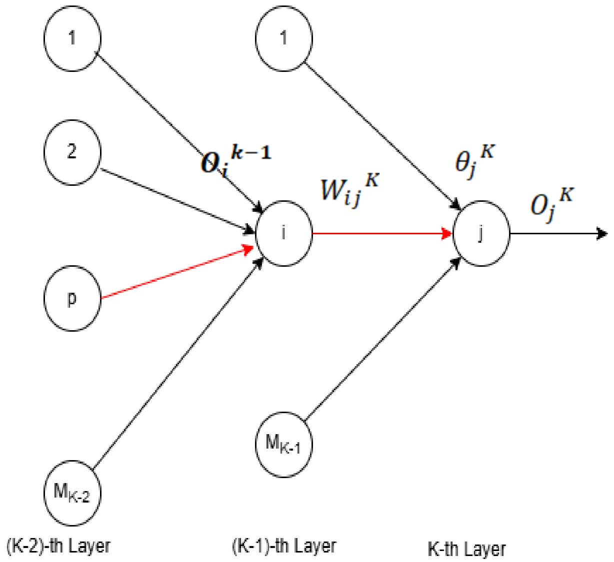
- We have to minimize $C = -\frac{1}{N}\sum_{\forall X}\sum_{j=1}^{M_K}[t_j \log O_j{}^K + (1-t_j)\log(1-O_j{}^K)]$

where, $O_j{}^K = \frac{1}{1+e^{-\theta_j{}^K}}$ and $\theta_j{}^K = \sum_{i=1}^{M_{K-1}} W_{ij}{}^K O_i{}^{K-1}$

$$\frac{\partial C}{\partial W_{ij}{}^K} = \frac{\partial C}{\partial O_j{}^K}\frac{\partial O_j{}^K}{\partial \theta_j{}^K}\frac{\partial \theta_j{}^K}{\partial W_{ij}{}^K}$$

$$= -\frac{1}{N}\sum_{\forall X}\left(\frac{t_j}{O_j{}^K} - \frac{1-t_j}{1-O_j{}^K}\right)O_j{}^K(1-O_j{}^K)O_i{}^{K-1}$$

$$= -\frac{1}{N}\sum_{\forall X}(t_j - O_j{}^K)O_i{}^{K-1} = \frac{1}{N}\sum_{\forall X}(O_j{}^K - t_j)O_i{}^{K-1}$$




- So, the weight updation rule is: $W_{ij}{}^K = W_{ij}{}^K - \eta\frac{1}{N}\sum_{\forall X}(O_j{}^K - t_j)O_i{}^{K-1}$

# Backpropagation Learning using Cross Entropy Loss Function for Multiclass problem

- Here, the weight updation rule is:

$$W_{ij}{}^{K} = W_{ij}{}^{K} - \eta \frac{1}{N} \sum_{\forall X} ( O_j{}^{K} - t_j ) O_i{}^{K-1}$$

- If $( O_j{}^{K} - t_j )$ is more, i.e., if error is more then the updation part or step size of updation is more.
- So, the learning rate is proportional to the error.
- This is the advantage of cross entropy over quadratic error.
- But it should be remember that to use the cross entropy loss, the output of the neural network must be a softmax output, because it has to be a probabilistic measure.

THANK YOU