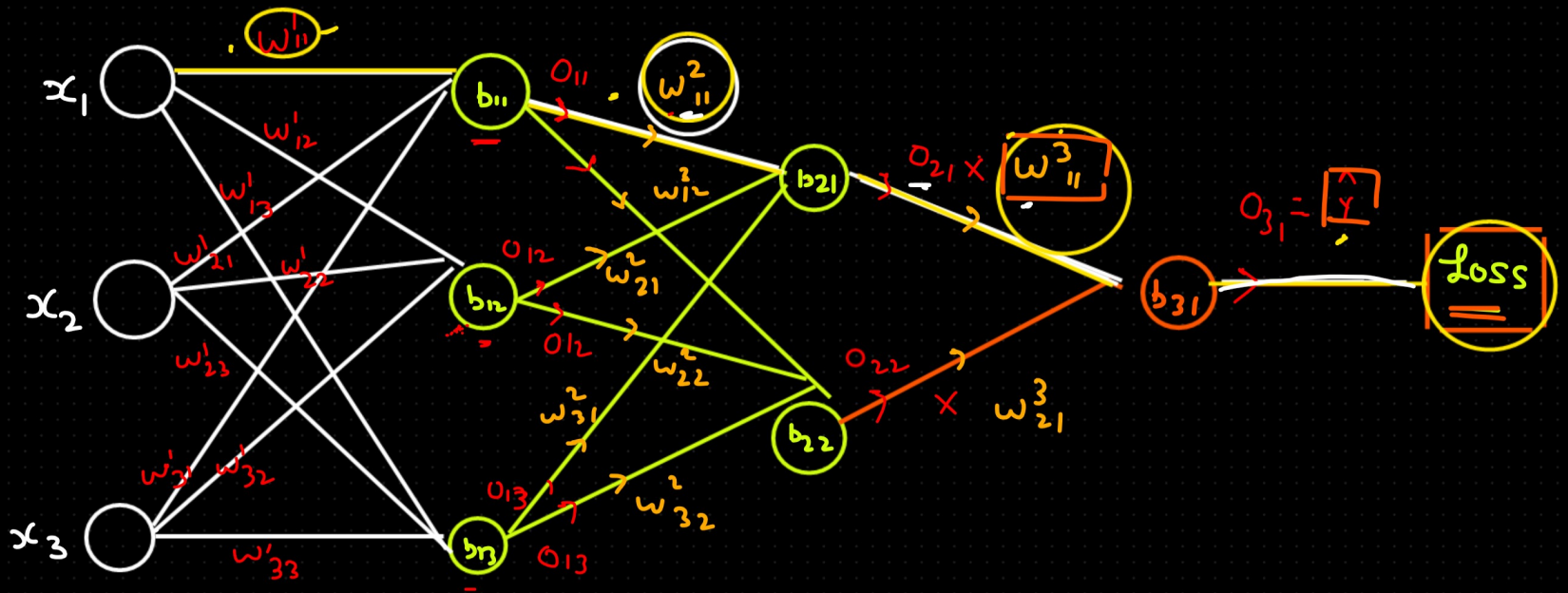
Plese note that:
Technical name of neural network is called Logical Architecture of Neural Network. The same is shown below:

--> Purpose of Back Propagation is to update the trainable parameters i.e. weights & bias

1

Starting with back propagation quick revision



I/P

Hidden layer1

Hidden layer2

O/P

Trainable Parameter $3 \times 3 + 3 = 12$ $(3 \times 2) + 2 = 8$ $(2 \times 1) + 1 = 3 = 23$

Now let's revise the BP => to update the (trainable Parameter) (weight + Biases)

Optimizer = (Gradient- Decent) =

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial w}$$

Wnew and Wold are weights. Row (η) represents the learning parameter. Where L represents the Loss which is dependent on weight trainable parameter. That's why partial derivation of L is done wrt W.

In the same way we will construct optimizer or gradient descent formula for bias trainable parameter. In the above formula just replace W with b(bias)

Loss is dependent on the predicted Output y^ that is represented by the Output O31 which is turn is dependent on some trainable parameters like:
(W 3 11) , (W 3 21) and b31

$$\frac{\partial L}{\partial w} \Rightarrow \frac{\partial L}{\partial w_{11}^3}$$

$$\text{loss} \rightarrow \hat{y} = O_{31} \rightarrow w_{11}^3$$

Here just for easier demonstration only 1 trainable parameter is taken

$$\frac{\partial L}{\partial w_{11}^3} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_{11}^3} \Rightarrow ?$$

Here for instance we can clearly observe that the chain dependency is formed which mathematically is represented using chain rule differentiation

$$\text{loss} \rightarrow \hat{y}(O_{31}) \rightarrow w_{11}^3 \rightarrow O_{21} \rightarrow w_{11}^2$$

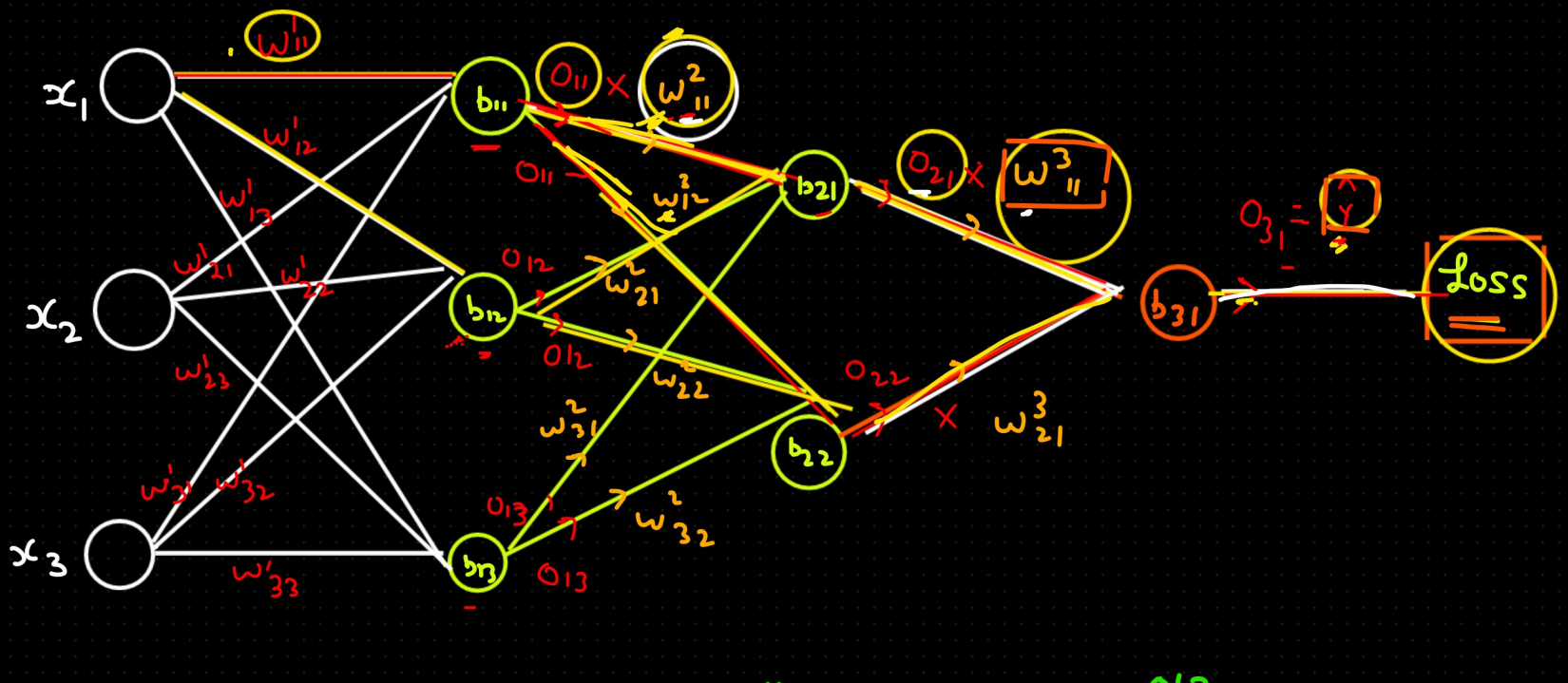
$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial w_{11}^2}$$

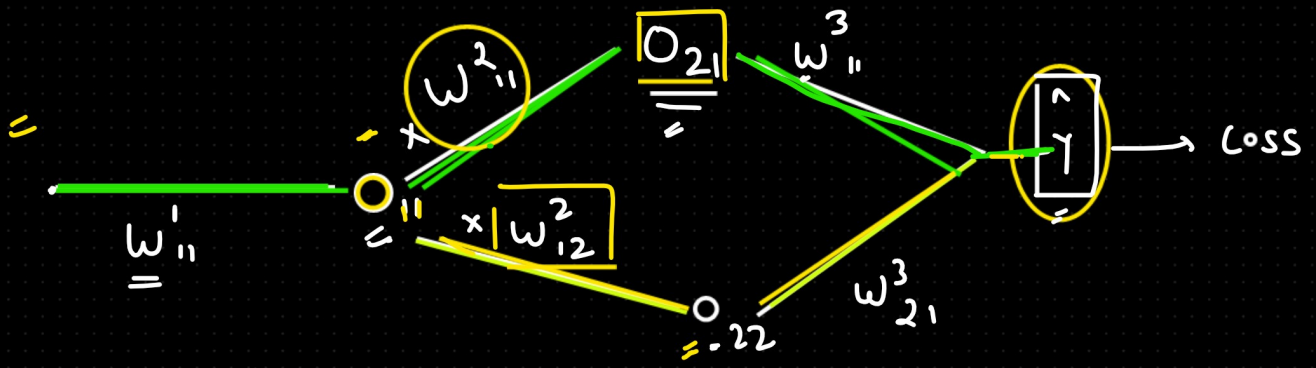
(Chain rule of differentiation)

Just visualize that how these dependencies are being calculated

$$\text{loss} \rightarrow \hat{y} \rightarrow w_{11}^3 \rightarrow O_{21} \rightarrow w2_{11} \rightarrow O_{11} \rightarrow w'_{11}$$

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial w^2_{11}} \times \frac{\partial w^2_{11}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w'_{11}}$$





$\frac{\partial L}{\partial w^1_{11}}$ = Will take both Path and for final out will add on both Path

Above for easier demonstration purpose we have only considered a Single path. In real time all possible path along with all the trainable parameters (weights and bias) will be taken into consideration.

$$= \frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \left[\frac{\partial o_{31}}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w'_{11}} + \frac{\partial o_{31}}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w'_{11}} \right]$$

{ loss fn
activation fn
optimizer }

Loss fn =>

Regression => MSE, MAE, huberloss (MSE and MAE already covered in the last lecture)

Classification => Binary, Categorical, Sparse Cross Entropy

MSE+MAE

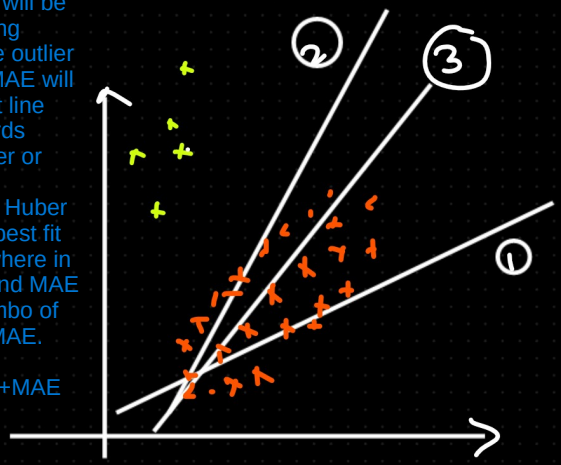
Huber Loss

$$= \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \leftarrow \text{this part in presence of outlier} \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2 & \leftarrow \text{if D.P. is outlier} \end{cases}$$

δ → Hyperparameter

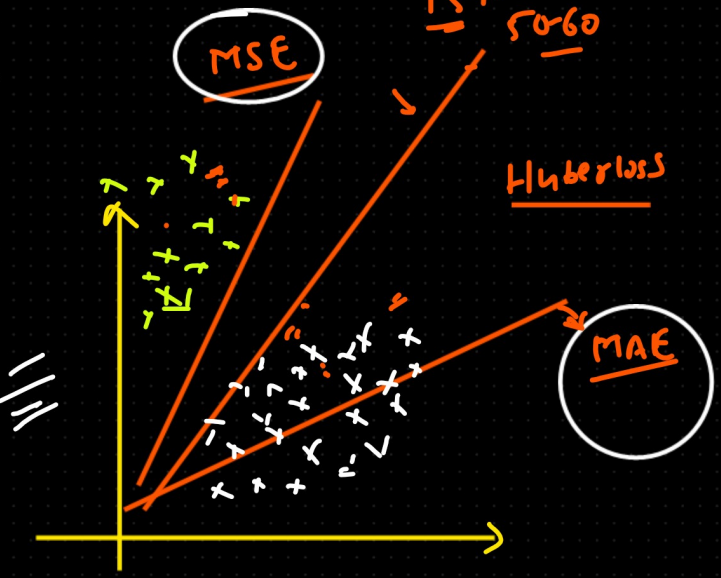
See below snip to understand the formula better

In case of MSE best fit line will be more tending towards the outlier whereas, MAE will give best fit line more towards more denser or non outlier datapoints. Huber loss gives best fit line somewhere in b/w MSE and MAE Kind of combo of MSE and MAE. Huber Loss=MSE+MAE



=> MSE => Sensitive to outliers

MAE => not Sensitive to outliers



defines the loss function piecewise

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

This function is quadratic for small values of a, and linear for large values, with equal values and slopes of the different sections at the two points where $|a| = \delta$. The variable a often refers to the residuals, that is to the difference between the observed and predicted values $a = y - f(x)$, so the former can be expanded to[2]

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \le \delta, \\ \delta \cdot \left(|y - f(x)| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

The Huber loss is the convolution of the absolute value function with the rectangular function, scaled and translated. Thus it "smoothens out" the former's corner at the origin.

alpha α represents the loss that is difference b/w actual value and predicted value

Dell δ Hyper-parameter is decided based on outlier

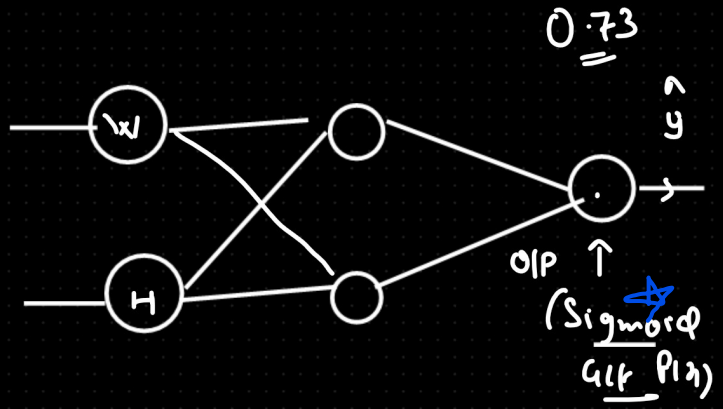
Loss fn for the classification

Please don't get confuse b/w loss and activation fun^. Sigmoid is the activation function that may be used in the case of Binary classification. Whereas, loss function used for binary classification is called Binary cross entropy

1. **Binary Cross entropy** ⇒ 2 classes **log loss** ⇒ logistic regression

Loss function for Binary class classification

Weight	height	ob/Not ob	
55	170	1	0.73
60	180	0	0.25
65	175	0	



Single value **Loss** $= -y \log(\hat{y}) - (1-y) \log(1-\hat{y})$

For the whole data ⇒ Cost fn $= \frac{1}{n} \sum_{i=1}^{n} \left[-y_i \log(\hat{y}_i) - (1-y_i) \log(1-\hat{y}_i) \right]$

Assignment: Perform the derivation (drive) of log loss

$-1 \log(0.73) - (1-1) \log(1-0.73)$

$-1 \times \log(0.73) - 0 \times \log(0.27)$

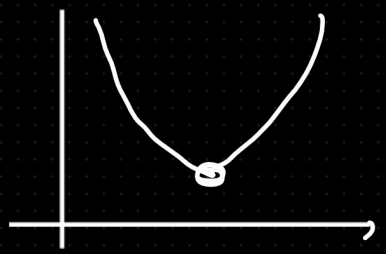
$-\log(0.73) = +(0.13667714)$

$-0 \times \log(0.25) - (1-0) \times \log(1-0.25)$

$0 \qquad -1 \times \log(0.75) = -\log(0.75) = 0.12$

Advantage

- We can use G.D. for this loss
  since it is differentiable because of convex curve



Disadvantage

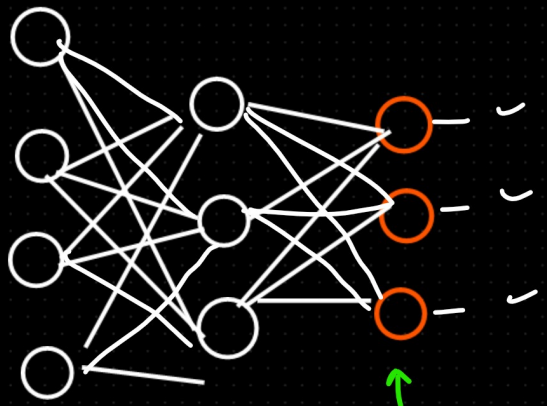
Categorical cross entropy ⇒ Multiclass classification ⇒ More than 2 clas

Iris dataset example is taken Sepal lenght, sepa; width, petal length and petal width

S_L	S_W	P_L	P_W	clas
				Setosa Verginica versicolor

one hot encoded value for the target clas

Since in multiclass Target will be classified into more than 2 classes so as a standard step we will require to apply One Hot Encoding on top of it



Softmax ⇒

SoftMax is the activation fun^ that may be used in the case of multiclass classification

$\hat{y} = a(r(\text{Summation}))$ ⇒ Softmax

Categorical cross entropy

$(y - \hat{y})$ ⇒

$$\frac{e^{z_i}}{\sum e^{z_i}} = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} \quad \frac{e^{z_2}}{\dots} \quad \frac{e^{z_3}}{\dots}$$

K represents the number of classes in which target will be classified

$$Loss = -\sum_{i=1}^{K} y_i \log(\hat{y}_i)$$

$$Loss = -\sum_{i=1}^{3} y_i \log(\hat{y}_i)$$

Please note that here only loss fun^ is being discussed. For calculating the Cost just take the average of the individual losses

$$-y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3)$$

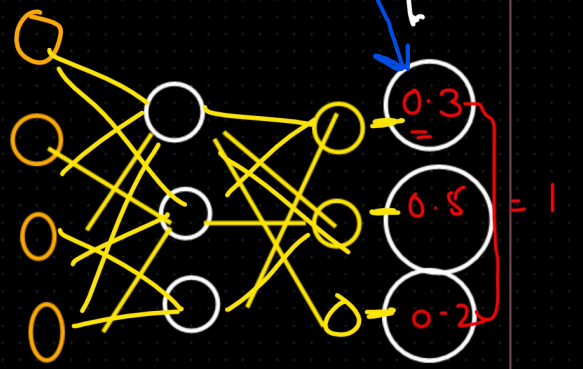
Since we have 3 classes that is why we are having 3 nodes in the Output layer

this is classification

One hot encoding

Input dataset:

S_L	S_W	P_L	P_W	(target)
-	-	-	-	Verginica
-	-	-	-	Setosa
-	-	-	-	versicolor



Applying one hot encoding to the Target column

Verginica	Setosa	versicolor
1	0	0
0	1	0
0	0	1

$$\begin{bmatrix} 0.3 & 0.5 & 0.2 \end{bmatrix} = \hat{y}$$

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} = y$$

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

wrt first records we are getting y^= [0.3 0.5 0.2] whereas actual y = [1 0 0]
Appying loss fun^ formula on y^

$$-y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3) = -1 \times \log(0.3) - \underbrace{0 \times \log(0.5)}_{0} - \underbrace{0 \times \log(0.2)}_{0}$$

These getting canceled out since respective label encoded values is 0

$$= -\log(0.3) = 0.52$$

$$\Rightarrow -0 \times \log(0.3) - 1 \times \log(0.5) - 0 \times \log(0.2)$$

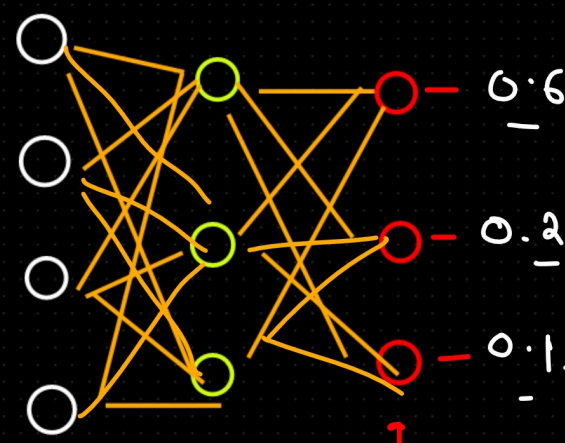
$$= -1 \log(0.5) \Rightarrow -\log(0.5) = 0.30$$

Sparse Categorical cross entropy ⇒ Fast

In above we can clearly observe that in the case of multi class classification there is a lot of sparsity (0s and 1s) which leads to overfitting. To counter that we have Sparse categorical cross entropy

S_L	S_W	P_L	P_W	Classes	Label | Integer
-	-	-	-	Setosa	1
-	-	-	-	Versinica	2
-	-	-	-	Versicolor	3

In Categorical cross entropy target is categorically label encoded based on one hot label encoding whereas in Sparse Categorical cross entropy target is Integral encoded



Softmax activation fun^ gives Probabilty value of the respective I/P value

Softmax ⇒ $\frac{e^{z_i}}{\sum e^{z_i}}$

Since we are having 3 classes in the target column so that is why we are having 3 nodes in the Output Layer

0.6 + 0.21 + 0.19 = 1

Please note for a specific record one hot encoded value (y or y^) sum of encoded value is equals to 1(Probability concept)

1 st row

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.6 & 0.21 & 0.19 \end{bmatrix} \Rightarrow \text{Cat cross entropy} \quad ?$$

When classes is less then we use Categorical cross entropy target whereas, when classes are more then we use Sparse Categorical cross entropy

$$\begin{bmatrix} 1 & 2 & 3 \\ 0.6 & 0.21 & 0.19 \end{bmatrix}$$

Actual Vs Predicted

$$-y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3)$$

For example here we are calculating the loss for the first record so we are directly neglecting the influence of other unnecessary encoded labels. Which earlier in case of Multiclass we neglecting by multiplying with Zero.

$$\Rightarrow -1 \times \log(0.6) \Rightarrow -\log(0.6)$$

Please note that here,
Labeling does not mean that we will multiply with 2 in case calculation is for 2nd label. We will still multiply by 1 representing actual value of respective label encoded value.
That means that in Sparse categorical cross entropy all the integral encoded target values are internally mapped to 1 when calculating the loss(this consideration is only for the actual value)

$$\begin{bmatrix} 1 & 2 & 3 \\ \times & & \times \\ 0.6 & 0.21 & 0.19 \end{bmatrix}$$

Both are same just that in CCE(Categorical Cross Entropy) we are multiplying the not needed labels with 0 which may seems computational expensive whereas, in SCCE (Sparse Categorical Cross Entropy) we are directly neglecting the effect of not needed labels

In SCCE not needed labels are directly(computational inexpensive) getting rejected whereas in CCE not needed labels are rejected by multiplying with 0(computational expensive)

Height	Weight	Body Size	
170	66	M	1
175	60	S	2
180	75	L	3
160	70	M	1
155	50	L	3

One hot encoding

CCA

$$-\sum_{i=1}^{3} y_i \log(\hat{y}_i)$$

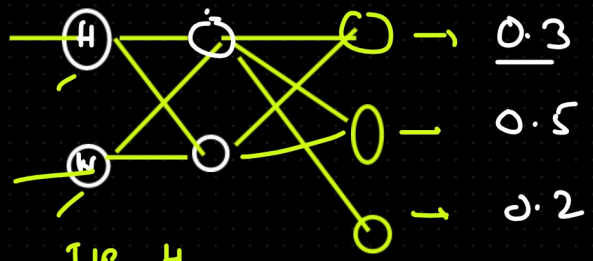
$$= -y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3)$$

$$= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0.3 & 0.5 & 0.2 \end{bmatrix}$$

$$= -1 \times \log(0.3) - 0 \times \log(0.5) - 0 \times \log(0.2)$$

$$= -\log(0.3) \Rightarrow 0.52$$



⇒ CCA

⇒ SCCA

SCCA ⇒ labelling

$$= \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 0.3 & 0.5 & 0.2 \end{bmatrix}$$

SCCA v/s CCA

$-1 \times \log(0.5)$

$-\log(0.5)$

$= 0.69$

When the clause is low use CCA

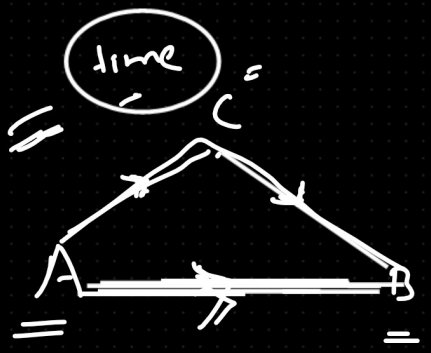
When the clause is high use SCCA

CCA reduce the performance

SCCA increase the performance

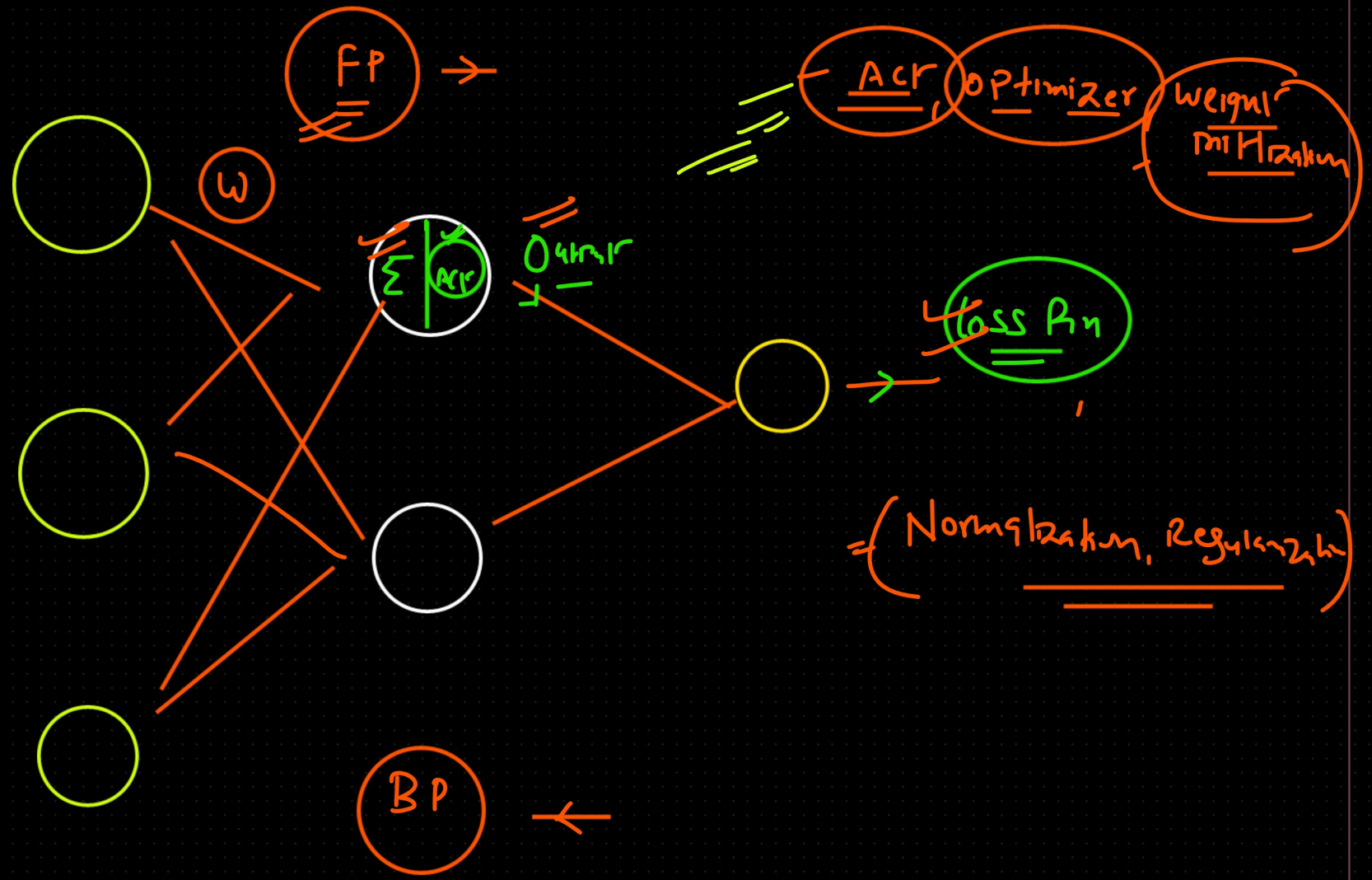
Low data + low class ⇒ CCA is fine

Noise data + high class ⇒ SSCA is fine



mapping

Simplified


FP
Act
Optimizer
Weight
Initialization
W
Σ
Act
Output
Loss Fn
Normalization, Regularization
BP

Act Fln

Activation function tells when to trigger the summation part of a neuron. It acts like gate that opens or closes when based on conditions. Please refer to different types as discussed below for better understanding

Activation fun^ has 2 parts Linear and Non Linear
Linear activation Fun^ is the activation fun^ that works on linear dataset or hyposthesis tha is linear in nature wheras, in case it is non linear then it is called Non Linear activation fun^

Linear activation fln

1 Binary step Activation

- x represents the summation part of a neuron
- f(x) = y represents the output of a neuron produced after applying activation function f(x)

$$y = f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

Activation function Logic

2 Linear activation fln

$$y = f(x) = x$$

$$f(1) = 1$$
$$f(2) = 2$$
$$f(100) = 100$$

Non Linear activation fln

This will be discussed in depth in the next lecture

1 Sigmoid fln = Logistic fln

$$y = f(x) = \frac{1}{1 + e^{-x}}$$

(2) tanh ⇒ $y = f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

(3) Relu ⇒ $\boxed{y = f(x) = \max(0, x)}$

(4) Prelu or Parametric relu ⇒ $f(x) = \max(ax, x)$

(5) Softmax ⇒ $\text{Softmax}(z_i) = \frac{\exp(z_i) \text{ or } e^{z_i}}{\sum \exp(z_i) \text{ or } e^{z_i}}$

Regression ⇒ Linear A/c fn or Sigmoid fn

Binary classification ⇒ Sigmoid, tanh ; (Relu)

= multiclass ⇒ Softmax

Why, where, how = ⇒ About activation fun^ to be discussed in next lec


Optimizer