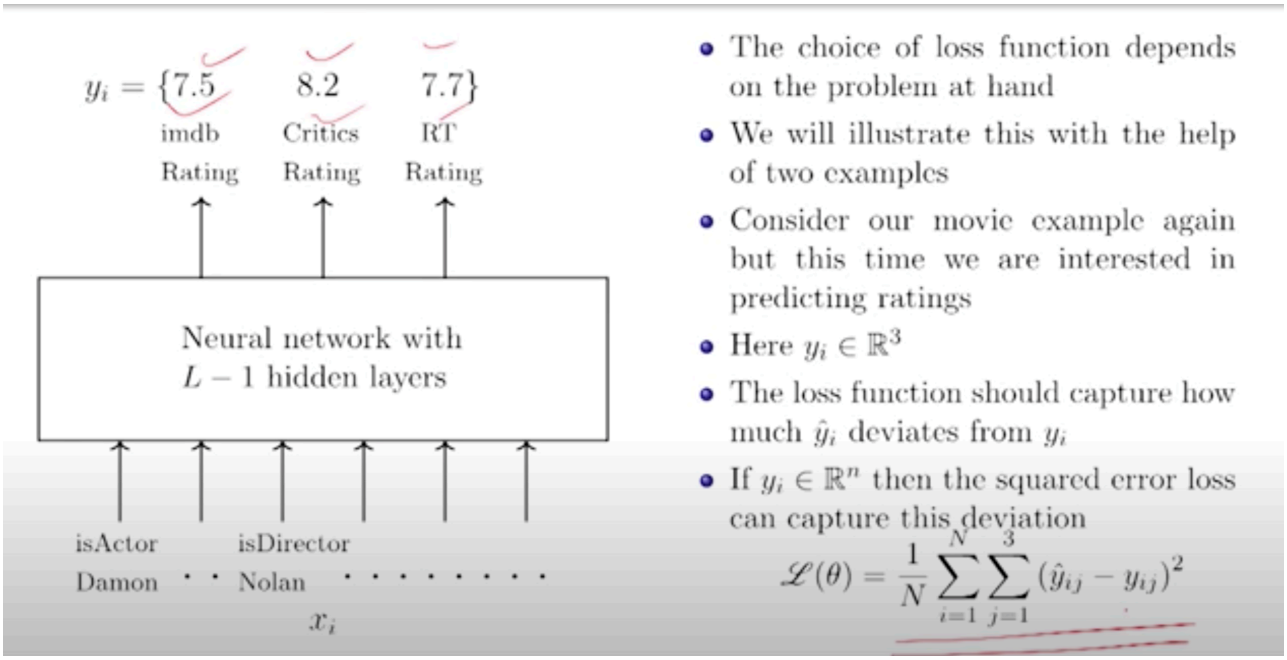
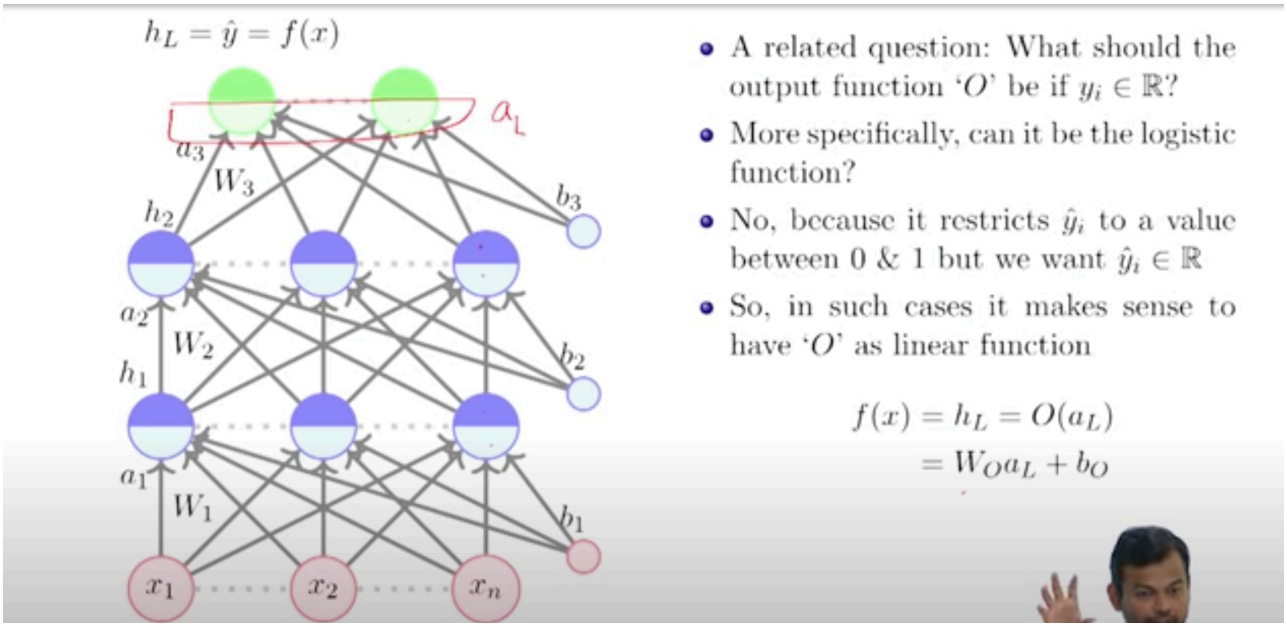
DL Basics

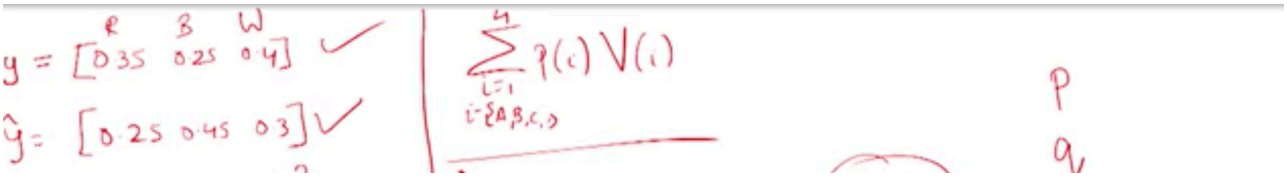
Tuesday, 28 December 2021 2:32 PM

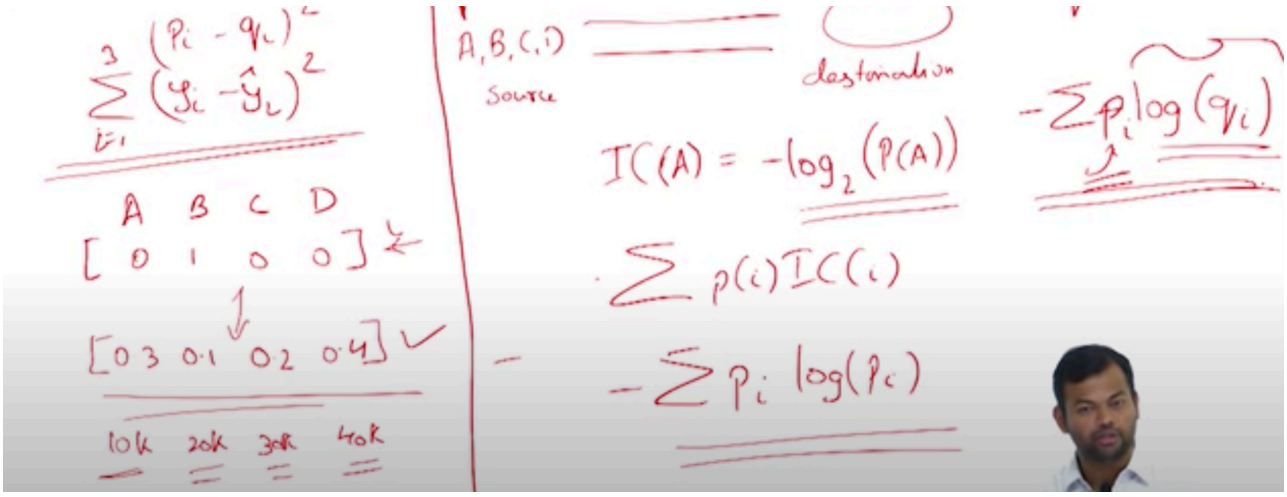
Output functions and Loss functions in FFNN:

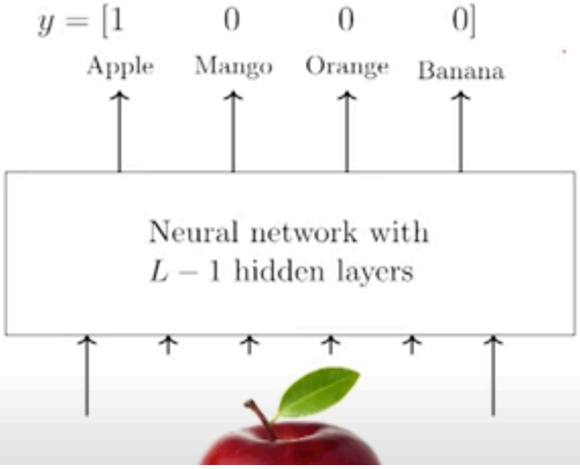


Q	1

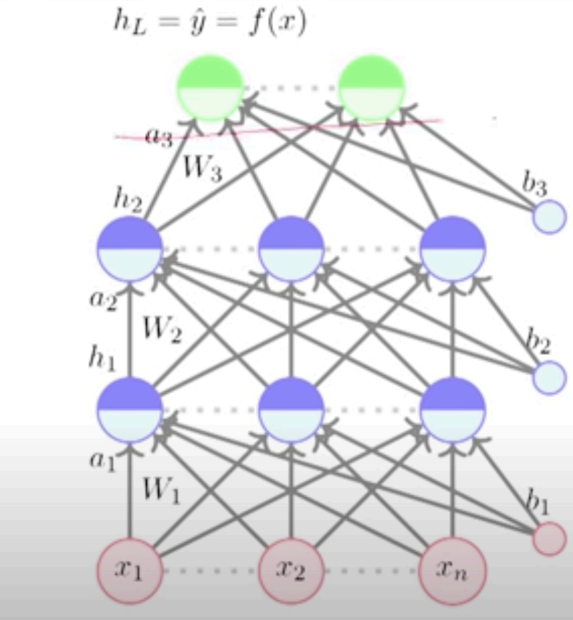








- Now let us consider another problem for which a different loss function would be appropriate
- Suppose we want to classify an image into 1 of k classes
- Here again we could use the squared error loss to capture the deviation
- But can you think of a better function?



- Notice that y is a probability distribution
- Therefore we should also ensure that $\hat{y}$ is a probability distribution
- What choice of the output activation 'O' will ensure this ?

$$a_L = W_L h_{L-1} + b_L$$

$$\hat{y}_j = O(a_L)_j = \frac{e^{a_{L,j}}}{\sum_{i=1}^{k} e^{a_{L,i}}}$$

$O(a_L)_j$ is the j^{th} element of $\hat{y}$ and $a_{L,j}$ is the j^{th} element of the vector a_L.

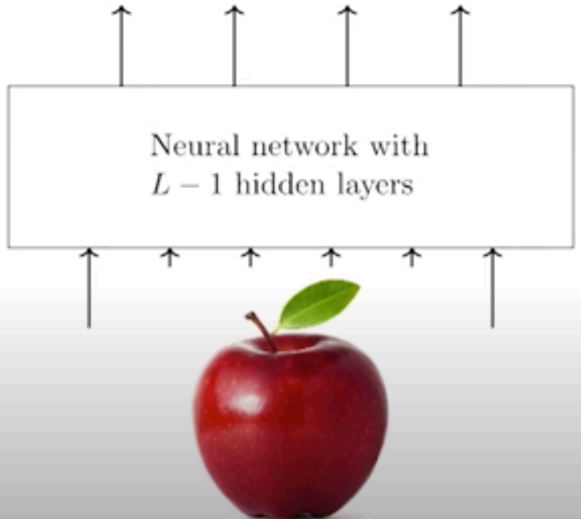
$a_L = [10 \quad -20 \quad 30]$ $\quad \hat{y}_1 = \frac{e^{10}}{e^{10} + e^{-20} + e^{30}}$

$\hat{y} = [\hat{y}_1 \quad \hat{y}_2 \quad \hat{y}_3]$

$y = [1 \quad 0 \quad 0 \quad 0]$

Apple Mango Orange Banana

- Now that we have ensured that both y & $\hat{y}$ are probability distributions can you think of a function which



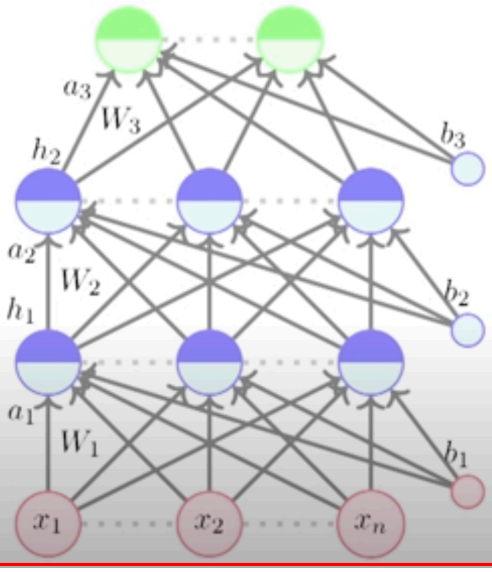
captures the difference between them?

- Cross-entropy

$$\mathscr{L}(\theta) = -\sum_{c=1}^{k} y_c \log \hat{y}_c$$

- Notice that

$$y_c = 1 \quad \text{if } c = \ell \text{ (the true class label)}$$
$$= 0 \quad \text{otherwise}$$
$$\therefore \ \mathscr{L}(\theta) = -\log \hat{y}_\ell$$



- So, for classification problem (where you have to choose 1 of K classes), we use the following objective function

$$\underset{\theta}{\text{minimize}} \quad \mathscr{L}(\theta) = -\log \hat{y}_\ell$$
$$\text{or} \quad \underset{\theta}{\text{maximize}} \quad -\mathscr{L}(\theta) = \log \hat{y}_\ell$$

- But wait!
  Is $\hat{y}_\ell$ a function of $\theta = [W_1, W_2, .., W_L, b_1, b_2, .., b_L]$?
- Yes, it is indeed a function of θ

$$\hat{y}_\ell = [O(W^3 g(W^2 g(W^1 x + b_1) + b_2) + b_3)]_\ell$$

- What does $\hat{y}_\ell$ encode?
- It is the probability that x belongs to the ℓ^{th} class (bring it as close to 1).
- $\log \hat{y}_\ell$ is called the *log-likelihood* of the data.

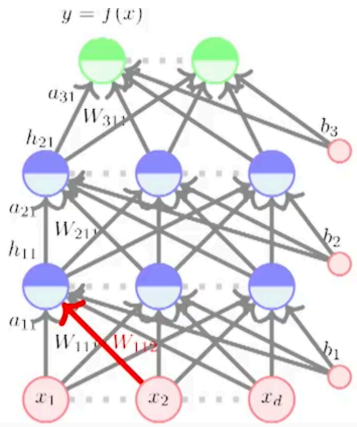
	Outputs	
	Real Values	Probabilities
Output Activation	Linear	Softmax
Loss Function	Squared Error	Cross Entropy

$\frac{\partial \mathscr{L}(\theta)}{\partial \theta}$

- Of course, there could be other loss functions depending on the problem at hand but the two loss functions that we just saw are encountered very often
- For the rest of this lecture we will focus on the case where the output activation is a softmax function and the loss function is cross entropy

4.1) BACK PROPAGATION:

- Let us focus on this one weight (W_{112}).
- To learn this weight using SGD we need a formula for $\frac{\partial \mathscr{L}(\theta)}{\partial W_{112}}$.



Algorithm: gradient descent()

$t \leftarrow 0$;
$max\_iterations \leftarrow$ 1000;
Initialize θ_0;
while $t{+}{+} < max\_iterations$ **do**
  $\theta_{t+1} \leftarrow \theta_t - \eta \nabla \theta_t$;
end

- First let us take the simple case when we have a deep but thin network.
- In this case it is easy to find the derivative by chain rule.

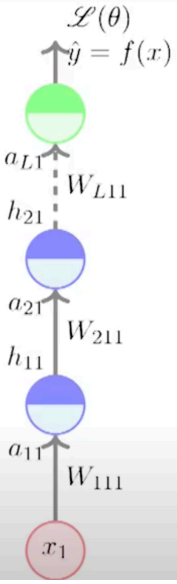
$$\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}} = \frac{\partial \mathscr{L}(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{L11}} \frac{\partial a_{L11}}{\partial h_{21}} \frac{\partial h_{21}}{\partial a_{21}} \frac{\partial a_{21}}{\partial h_{11}} \frac{\partial h_{11}}{\partial a_{11}} \frac{\partial a_{11}}{\partial W_{111}}$$

$$\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}} = \frac{\partial \mathscr{L}(\theta)}{\partial h_{11}} \frac{\partial h_{11}}{\partial W_{111}}$$

(just compressing the chain rule)

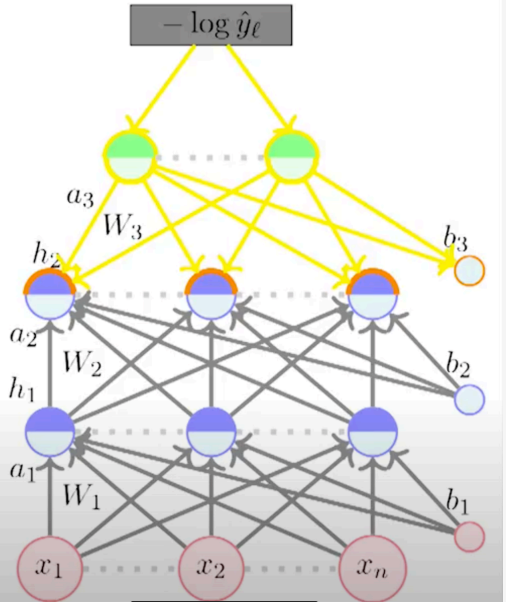
$$\frac{\partial \mathscr{L}(\theta)}{\partial W_{211}} = \frac{\partial \mathscr{L}(\theta)}{\partial h_{21}} \frac{\partial h_{21}}{\partial W_{211}}$$

$$\frac{\partial \mathscr{L}(\theta)}{\partial W_{L11}} = \frac{\partial \mathscr{L}(\theta)}{\partial a_{L1}} \frac{\partial a_{L1}}{\partial W_{L11}}$$



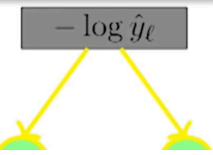
- We get a certain loss at the output and we try to figure out who is responsible for this loss
- So, we talk to the output layer and say "Hey! You are not producing the desired output, better take responsibility".
- The output layer says "Well, I take responsibility for my part but please understand that I am only as the good as the hidden layer and weights below me". After all ...

$$f(x) = \hat{y} = O(W_L h_{L-1} + b_L).$$



Let us first consider the partial derivative w.r.t. i-th output

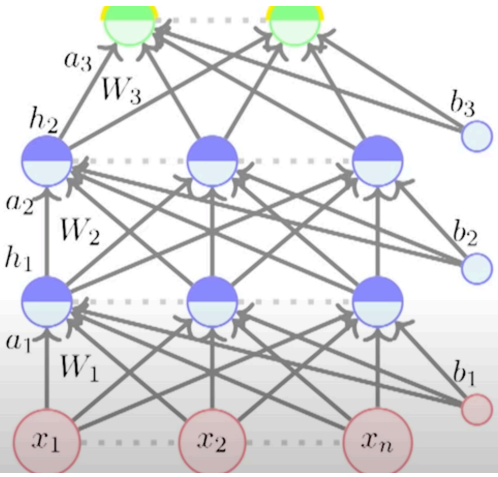
$$\mathscr{L}(\theta) = -\log \hat{y}_\ell \quad (\ell = \text{true class label})$$



$$\frac{\partial}{\partial \hat{y}_i}\left(\mathscr{L}(\theta)\right) = \frac{\partial}{\partial \hat{y}_i}\left(-\log \hat{y}_\ell\right)$$
$$= -\frac{1}{\hat{y}_\ell} \quad \text{if } i = \ell$$
$$= 0 \qquad otherwise$$

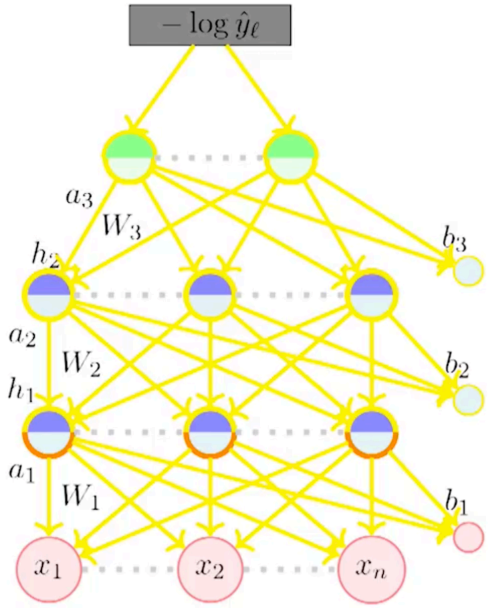
More compactly,

$$\frac{\partial}{\partial \hat{y}_i}\left(\mathscr{L}(\theta)\right) = -\frac{\mathbb{1}_{(i=\ell)}}{\hat{y}_\ell}$$



- So, we talk to W_L, b_L and h_L and ask them "What is wrong with you?"
- W_L and b_L take full responsibility but h_L says "Well, please understand that I am only as good as the pre-activation layer"
- The pre-activation layer in turn says that I am only as good as the hidden layer and weights below me.
- We continue in this manner and realize that the responsibility lies with all the weights and biases (i.e. all the parameters of the model)
- But instead of talking to them directly, it is easier to talk to them through the hidden layers and output layers (and this is exactly what the chain rule allows us to do)

$$\underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}}}_{\text{Talk to the weight directly}} = \underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial a_3}}_{\text{Talk to the output layer}} \underbrace{\frac{\partial a_3}{\partial h_2}\frac{\partial h_2}{\partial a_2}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_2}{\partial h_1}\frac{\partial h_1}{\partial a_1}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\text{and now talk to the weights}}$$



Quantities of interest (roadmap for the remaining part):

- Gradient w.r.t. output units
- Gradient w.r.t. hidden units
- Gradient w.r.t. weights and biases

$$\underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}}}_{\text{Talk to the weight directly}} = \underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial a_3}}_{\text{Talk to the output layer}} \underbrace{\frac{\partial a_3}{\partial h_2}\frac{\partial h_2}{\partial a_2}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_2}{\partial h_1}\frac{\partial h_1}{\partial a_1}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\text{and now talk to the weights}}$$

- Our focus is on *Cross entropy loss* and *Softmax* output.