In this video we will talk about how results will change if you use different function for loss.

Equations that we derived in last video, were based on assumption that loss in final layer is a squared error loss.

$$\left.\begin{aligned}
\delta^L &= (a^L - y) \odot \sigma'(z^L) \\
\delta^l &= (w^{l+1}\,\delta^{l+1}) \odot \sigma'(z^l) \\
\frac{\partial L}{\partial b^l} &= \delta^l \\
\frac{\partial L}{\partial w^l} &= a^{l-1}(\delta^l)^T
\end{aligned}\right\}$$

Assumption

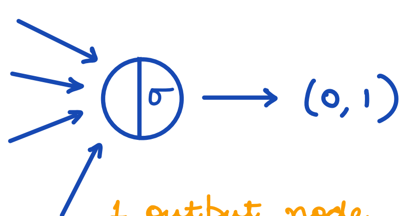
MSE loss = $\frac{1}{2}\sum_i (y_i - a_i)^2$

$$\boxed{\delta^L = \frac{\partial L}{\partial z} = (a^L - y) \cdot \sigma'(z^L)}$$

→ vectorization for 1 example

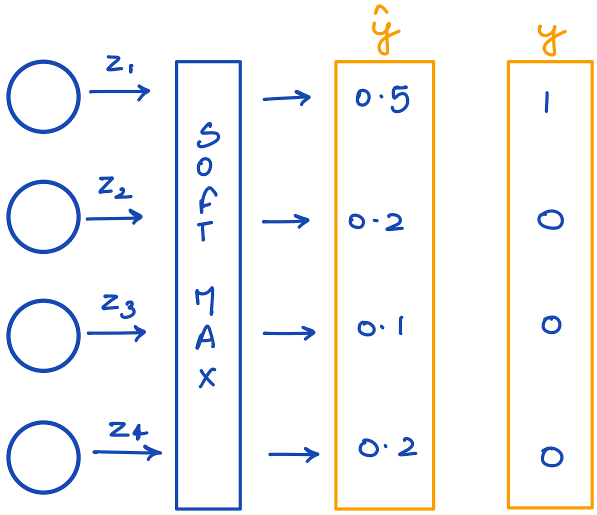
MSE loss is not good bcz it is non convex there will be lot of local minima and we will be stuck while optimizing.

In practice if you have a binary classification, like yes or no, male or female, pokemon or not, then you can have single output node in your network which will have sigmoid activation and this will predict your results in range (0,1). If probability is close to 0 then it is 0 class. and if probability is close to 1 or >0.5 then it is class 1.



1 output node

If you have to do classification in k categories then output layer will have k nodes and each node is going to produce some activation value z_1, z_2, z_3, z_4. In forward propagation we convert these z into probability by taking a softmax.



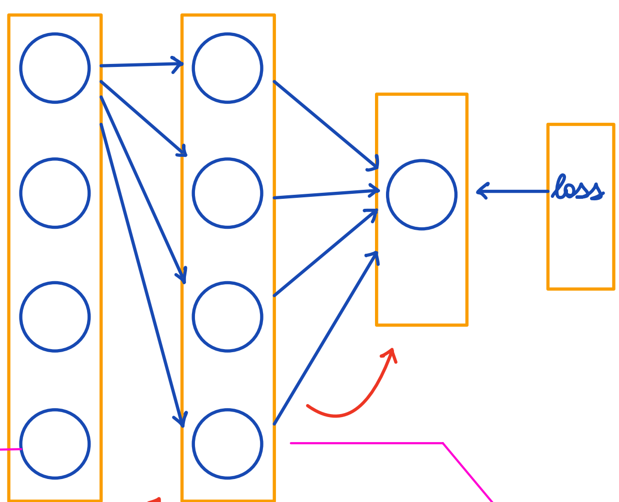
If you want to do binary classification then loss function which is used in practice is called cross entropy.

↳ refer logistic regression

$$\text{Binary Cross Entropy} = -\sum_{i=1}^{m} \left(y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i) \right)$$

y_i = True Label

$\hat{y}_i$ = Predict Label

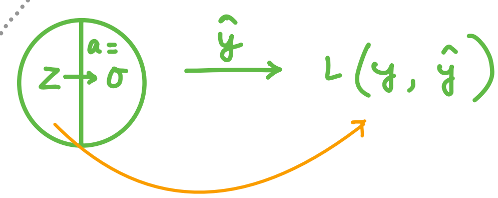


we expressed loss for this layer in terms of loss for next layer.

which will affect δ of this layer $\leftarrow$ which will affect δ of this layer $\leftarrow$ then δ of this layer will be affected $\leftarrow$ If you change loss fxn

For each layer δ was written in terms of δ of next layer. We need to figure out how does δ^L change if we use cross entropy as loss function.

$$\delta_L = \frac{\partial L}{\partial z^L} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z}$$



we want to analyze how change in z causes change in L.

$$\hat{y} = a$$

$$a = \sigma(z)$$

$$\frac{\partial a}{\partial z} = \sigma(z)\,(1-\sigma(z))$$

$$= a(1-a)$$

$$= \hat{y}\,(1-\hat{y})$$

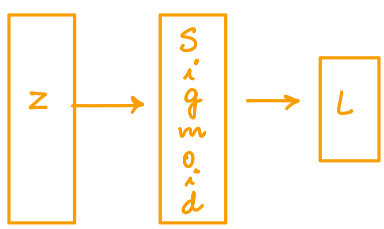
$$\delta_L = \left(\frac{-y_i}{\hat{y}_i} + \frac{1-y_i}{1-\hat{y}_i}\right) \cdot \left(\hat{y}_i\,(1-\hat{y}_i)\right)$$

$$\delta_L = \left(\frac{-y_i + \cancel{y_i \hat{y}_i} + \hat{y}_i - \cancel{y_i \hat{y}_i}}{\cancel{\hat{y}_i\,(1-\hat{y}_i)}}\right) \left(\cancel{\hat{y}_i\,(1-\hat{y}_i)}\right)$$

$$\boxed{\delta_L = (\hat{y}_i - y_i)}$$

δ_L in Cross Entropy loss

What if you have k classes, in this case loss function is called categorical cross entropy. In our example we had vector z it goes through sigmoid layer and it goes to binary cross entropy loss and we computed δ_L.



Binary Cross Entropy loss

This formula is going to remain same because Binary Cross Entropy is a special case of categorical entropy.

Categorical Cross Entropy:

$$L(y, \hat{y}) = \underbrace{\frac{-1}{N} \sum_{n \in N} \sum_{i \in C} y_{n,i} \log \hat{y}_{n,i}}_{\text{iterating over all the classes predicted for each example}}$$

If there are only 2 classes then it will be:

$$\frac{-1}{N} \sum_{n \in N} \underbrace{y_{n,1} \log \hat{y}_{n,1}}_{y_n \text{ wrt } 1^{st} \text{ class}} + \underbrace{y_{n,0} \log \hat{y}_{n,0}}_{y_n \text{ wrt } 0^{th} \text{ class}}$$

If you have only 2 classes and one class has a probability of $y_{n,1}$ then other class will have a probability of $1 - y_{n,1}$

Substitute $y_{n,0} = 1 - y_{n,1}$, we get binary cross entropy

Generalised loss for K classes.

Now, you take derivative like this:

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z}$$

Derivative of softmax

a = activation

$$= (\hat{y}_i - y_i)$$

Final output remains same

Prediction

True label

In code implementation we will be using above equations.
In our case we will say $\delta^L = a^L - y$

If you have a different loss function don't use these equations blindly. See what loss function is given to us in the problem and what should be the equation for back propagation.

You should know how to do forward propagation and back propagation is handled by pytorch and tensorflow.