	Date
	Neural Networks
	TYEWIA TYCHOTIS
>	Newal Networks are based on the design of newsons in the human body
	A single neuron mode perceptron can be denoted as
	ay 6
9-1-1-	Node. output (22)
C	activation. Joutput = activation (\(\vec{z} \cdot \vec{\vec{\vec{a}} + \vec{b}} \)
No. of the	
in	put! output! - O1
	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ing	0 - 0 output 3 - \(\frac{\xi}{\xi} - 0_3\)
inf	output m - 0m
	→
	optional step.
	network (final result)
	So you can either stop before the softmax as your output or do a
	Softmax on the network outputs be the nesult which comes out of it
	can be your final output.
,20	· ,
5 9	Doing a softmax is encouraged since it squashes your outputs to [01]
	Softmax output: - Oi
	Zoi i-i

Activation functions: -Activation functions are added so that neural networks can approximate non-linear functions Let us say we never did the activation, then you can see that as the output progresses through the nodes, it will just be a linear sum of the inputs. How since linear functions are restricted to just linear decision making you cannot expect it to be universal approximators. You have to add non-linearity to it Some popular activation functions: -Sigmoid > 2) tanhoc 3) ReLU > max (0, x) Sigmoid > ReLU > 0 -00 (zero botos NEO Co x after x>0)

So there are a few problems with sigmoid Ee tanh. At the extremes of the graph you can see that the gradients at those points are almost zero. In the future we are going to extended learn to calculate the coeights as the gradient of the cost function. Now if our gradients start becoming zero after a point (Vanishing Gradients), we might have a problem. Therefore in practical cases we should go for ReLU.

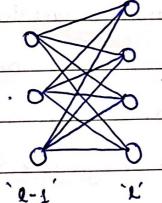
BACK PROPAGIATION:-

Let us first develop the intuition for newral networks. You have

input layer -> multiple hidden layers > output layer

Fach node in layer'l-1' is connected to each node in layer'l

as shown:



· and so on

al = 6 (& wikak + bi)

Ke no of nodes in layer L'i je no of nodes in layer L' we weights from layer l-1 to l'

be biases for nodes in layer &

ak -> output of kth news on in layer R-1 This is the input to as > output of it neuron in layer a.

our current layer &

Now as expected our weights & biases are unknown & we have to calculate them Here are a few more oquations which are important z = Ewkak-1+ bi & we just broke the previous two equations. a! = 6(z;2) Lets talk about the intuition for the back propagation now Lets say we started with handom weights and biases. We Feed forward the feed out input into the network network and get an output. Obviously the output is wrong. So what we do is backpropagate We fake a cost function & find the cost, and try to minimize it In the process, what we now do is do the same feed forward Back thing but backwards. We have the wrong output. We propagation calculate the evener at every node & update the weight so that the errors at each node is zero. We do this till we reach layer! Now we again do the feed forward with the new weights, cal culate the cost le dothe backpropagation again. We keep doing this for either some number of iterations or till we reach convergence or till we greach certain accuracy.

Mathematics of Back Propagation:	1
Since we had mentioned a cos	t hunction lots take MSF as
oux cost function (you can take any fur	nction as long as it is derivable)
the state of the s	term in the second
$\frac{C=1}{2n} \frac{\mathcal{E}\ y(n)-a_{(n)}\ ^2}{2n}$	6, acinput.
an 2	y(n) + actual output for x
	a(x) & predicted output for x
en e	Le last layer
	ne number of inputs.
And since we are going to minin	Mize. the cost to find over
weights le biases we are going to u	se the Gradient Descent Optimize
with the equations	,
ω; ω,-noc	Me loagning rate.
$\partial \omega_{\lambda}$	
	us can use twodifferent
bi: bi-ndc	learning rates of web
ن کان	optimization
Section 200 and a section of the sec	
We see that we have we by	The two tearms that are
missing are DC Ge DC dw; dbi	
And the Book of the second stages and	P.T.0

Let us define an error term that contributes to our overall			
cost function. Basically, sumember that we mentioned that			
in the intuition, when we were doing trackward pars & we			
wore updating the weights so that we could get the correct output.			
Another way to do this would be to see it in this way: -			
That the weights were correct but there was a change in the			
input that caused the crown in the output. We are doing this			
because use dont have ac.			
٥w			
We know over input to newson j'of layer'l' as Z?			
We know over input to newson j'aj layer's' as zj.			
Si = DC Zr ⇒ output of previous node			
$= \frac{\partial C}{\partial x} \times \frac{\partial a^2}{\partial x} \Rightarrow a^2 = \sigma(z^2)$			
$= \frac{\partial C}{\partial \alpha_{i}^{2}} \times \frac{\partial \alpha_{i}^{2}}{\partial z_{i}^{2}} \times \frac{\partial \alpha_{i}^{2}}{\partial z_{i}^{2}} = \sigma(z_{i}^{2})$			
82 - DC 6'(Z) (we did this since we			
$\frac{\delta^2}{\partial a_j^2} = \frac{\partial C}{\partial a_j^2} \left(\frac{\partial C}{\partial a_j^2} \right)$ (we did this since we have. C in terms of a_j^2)			
Remember, all these intuitions are so that we can go through			
this process in a computationally less intensive way			
We would always have calculated DC but that would			

sequire you to expand C from interms of at to ust and.
for that you will need to known the entire numerical expansion
from the input layer to the output which would have become a
huge harde.

Anyway, combi coming back to our explanation: -

$$\frac{\partial C}{\partial a_j} = \frac{\partial C}{\partial a_j$$

$$\frac{\delta_{j}^{1}}{\partial z_{j}^{2}} = \frac{\partial C}{\partial z_{k}^{2}} = \frac{\partial C}{\partial z_{k}^{2+1}} = \frac{\partial Z_{k}^{2+1}}{\partial z_{j}^{2}} = \frac{\partial Z_{k}^{2+1}}{\partial z_{j}^{2}}.$$

$$=\underbrace{\mathcal{E}\,\mathcal{E}_{k}^{Q+1}\,\partial_{\mathcal{Z}_{k}^{Q+1}}}_{|\mathcal{E}|}$$

