0 – 255
black white

m training images

784 px – training img
28x28

784 px => 0, 1, 2, ... 9
28x28
10 classes

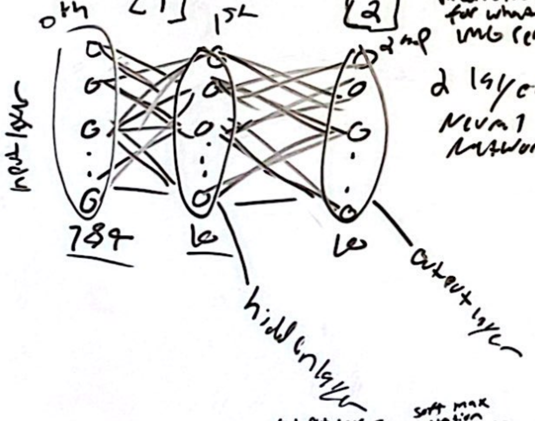
representation as MATRIX

$$X = \begin{bmatrix} — x^{(1)} — \\ — x^{(2)} — \\ \vdots \\ — x^{(m)} — \end{bmatrix} = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

each row 784 columns long

forward propogation
is take a picture and get probability out but we need good weights and biases to make these predictions & the whole point of machine learning is that we'll LEARN these weights and biases so we run an Algorithm to optimize these weights and biases and that's called backwards propogation

TAKE IMG do processing & give a prediction for what that IMG represents



2 layer Neural Network

unactivated first layer = we get z_1 by applying weight and bias

input layer = x no processing yet

The weights is going to be the matrix, we're going to take the dot product between that matrix and A^0 our input layer matrix. We're really just doing is multiplying by a bunch of weights that correspond to each of these connections & then we add constant bias term to each of the nodes

forward propogation

$$A^{[0]} = X \quad (784 \times m)$$

$$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$$

$10 \times m$, 10×784, $784 \times m$, $10 \times 1 \to 10 \times m$

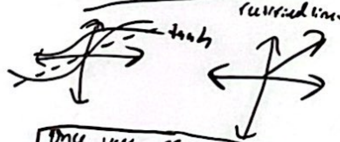
$$A^{[1]} = g(Z^{[1]}) = ReLU(Z^{[1]})$$

take img run these numbers & compare what class is going to be

α = hyper parameter not trained by model

α = Learning rate

APPLY ACTIVATION FUNCTION



$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \le 0 \end{cases}$$

$A^{(1)}$ = to ReLU of (x) applied to every value of in z_1

Once you apply tanh or sigmoid function to all of the nodes in a layer its no longer linear once you move on to the second layer thats now adding a second layer of complexity & non linear complexity rather than just a linear model

$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$

$10 \times m$, 10×10, $10 \times m$, $10 \times 1 \to 10 \times m$

2nd weights parameter in 2nd layer times A' which is activated first layer + b^2 another constant bias term

unactivated 2nd layer values

$$A^{[2]} = softmax(Z^{[2]})$$

each of the output after the soft max activation are ...

e^{z} stroke node is always just part of the whole that is the sum ...

output layer

1.3
5.1
2.2
0.7
1.1

soft max activation function

$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Probabilities

0.02
0.90
0.05
0.01
0.02

each of 10 nodes correspond to each of the 10 digits that could be recognized we want each of them to have a probability

takes each of the nodes in the layer that you feed into it e to the node / sum of e all of the nodes

BACKWARDS PROPEGATION

going opposite way and starting with prediction and going to find out how much our prediction deviated by the actual LABEL basically we error and then we're going to see and then we're going to see how much each of the previous weights and biases contributed to that error and then we adjust those things accordingly

$$dZ^{[2]} = A^{[2]} - Y$$

$10 \times m$, $10 \times m$, $10 \times m$

error of the second layer how much the output layer is off by

get predictions and subtract actual labels from it

0.1
0.2
0.03
0.90

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$

10×10, $10 \times m$, $m \times 10$

$$db^{[2]} = \frac{1}{m} \sum dZ^{[2]}$$

10×1, 10×1

$$dZ^{[1]} = W^{[2]T} dZ^{[2]} .* g'(Z^{[1]})$$

$10 \times m$, 10×1

derivative of the activation function because we have to undo the activation function to get the proper error for the first layer

derivative of lost function with respect to weights in layer 2

avg of absolute error how much the output was off by

how much the hidden layer was off by

Propagation in reverse

weights for the second term transposed times $dZ^{[2]}$

taking the error from the second layer and applying the weights to it in reverse to get to the errors of the first layer

After we do all this math finding our weights and biases with forward and backward propegation and figure out each weight term and bias contributed to the error we update our parameters accordingly

$$W^{[1]} = W^{[1]} - \alpha dW^{[1]}$$

$$b^{[1]} = b^{[1]} - \alpha db^{[1]}$$

$$W^{[2]} = W^{[2]} - \alpha dW$$

$$b^{[2]} = b^{[2]} - \alpha db^{[2]}$$

Update Params

then we do it again its a cycle of continued improvement so that the prediction is so ever closer the actual thing

VIZ-PRO WHITE BOARD ERASER