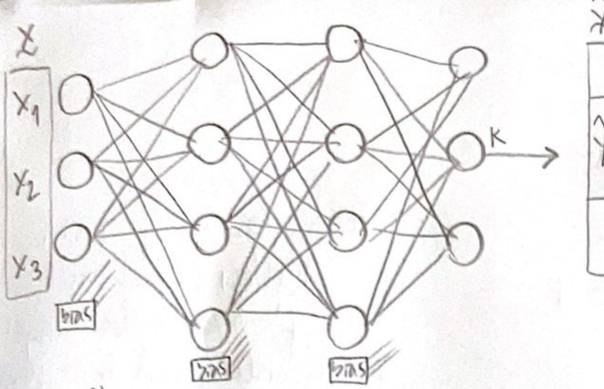


$\hat{y}_k = a_k^{(4)}$

$\underset{\sim}{a}^{(1)}$ INPUT — layer 1

$\underset{\sim}{a}^{(2)}$ layer 2, $\underset{\sim}{a}^{(3)}$ layer 3: HIDDEN

$\underset{\sim}{a}^{(4)}$ OUTPUT — layer 4

X: input

a: output after activation function

z: output <u>before</u> activation function

f: activation function, sigmoid

$\underset{\sim}{W}^{(l)}$: weight matrix from layer l to layer $l+1$ of size $\underbrace{S_{l+1}}_{\text{units in layer } l+1} \times (\underbrace{S_l}_{\text{units in layer } l} + \underbrace{1}_{\text{bias}})$

FORWARD PASS

$\underset{\sim}{a}^{(1)} = \underset{\sim}{x}$

$\hookrightarrow$ append bias: $\underset{\sim}{a}^{(1)} \leftarrow [a_0^{(1)} = 1, \underset{\sim}{a}^{(1)T}]^T = [1 \; x_1 \; x_2 \; x_3]^T$ $\quad \underbrace{(3+1)}_{4} \times 1$

$\underset{\sim}{z}^{(2)} = \underset{\sim}{W}^{(1)} \cdot \underset{\sim}{a}^{(1)} \quad (4\times4)\times(4\times1) = 4\times1$

$\underset{\sim}{a}^{(2)} = f(\underset{\sim}{z}^{(2)})$

$\hookrightarrow$ append bias: $\underset{\sim}{a}^{(2)} \leftarrow [a_0^{(2)} = 1, \underset{\sim}{a}^{(2)T}]^T$ $\quad \underbrace{(4+1)}_{5} \times 1$

$\underset{\sim}{z}^{(3)} = \underset{\sim}{W}^{(2)} \cdot \underset{\sim}{a}^{(2)} \quad (4\times5)\times(5\times1) = 4\times1$

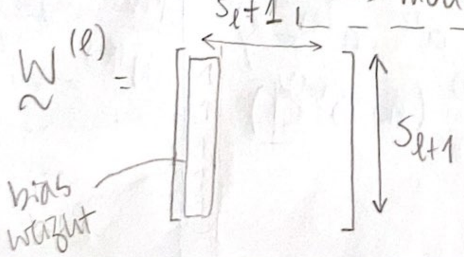
$\underset{\sim}{a}^{(3)} = f(\underset{\sim}{z}^{(3)})$

$\hookrightarrow$ append bias: $\underset{\sim}{a}^{(3)} \leftarrow [a_0^{(3)} = 1, \underset{\sim}{a}^{(3)T}]^T$ $\quad \underbrace{(4+1)}_{5} \times 1$

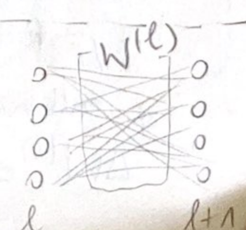
$\underset{\sim}{z}^{(4)} = \underset{\sim}{W}^{(3)} \cdot a^{(3)} \quad (3\times5)\times(5\times1) = 3\times1$

$\underset{\sim}{a}^{(4)} = f(z^{(4)}) = \underbrace{h_W}_{\hookrightarrow \text{ model / neural network.}}(\underset{\sim}{X}) : \; 3\times1$

Note: $\underset{\sim}{W}^{(l)} =$



$\underset{\sim}{W}^{(l)}$ is the weight matrix of connections that go from units in l to units in $l+1$



BACKWARD PASS

$\delta_j^{(l)}$: delta
error of node/unit j in layer l

- For each output unit j in last layer $l=4$

$$\delta_j^{(4)} = \overbrace{(y_j - \hat{y}_j)}^{e^{(4)}} \cdot f'(z_j^{(3)})$$

in Udacity it's multiplied by $f'()$; Andrew Ng didn't use f' in the last layer, but I understand that depends in the activation function!

in Udacity, the es are the errors, δ is the delta

$\hookrightarrow$ assemble: $\underset{\sim}{\delta}^{(4)} = \underbrace{[\delta_1^{(4)}, \dots, \delta_j^{(4)}, \dots]}_{\text{num classes}}$

note: for the sigmoid: $f' = f \cdot (1-f)$;
if no activation: $f(x) = x$, $f' = 1$.

- Propagate errors to all units of all layers

extend because of bias, so that dims match, but then it's removed, I understand we could extend the bias at the end of the vector, too...

$$\underset{\sim}{\delta}^{(3)} = \overbrace{\underbrace{\underbrace{\left(\underset{\sim}{W}^{(3)}\right)^T}_{5\times 3} \cdot \underbrace{\underset{\sim}{\delta}^{(4)}}_{3\times 1}}_{5\times 1 \to 1\times 5}}^{e^{(3)}} .* \underbrace{\left[1, f'(\underset{\sim}{z}^{(3)})\right]^T}_{1\times 4}$$

$\to$ multiply 1 by 1

$= [\underbrace{\delta_0^{(3)}}_{\text{bias}}, \delta_1^{(3)}, \dots]^T$: each unit in layer 3 has an error.

$\hookrightarrow \delta^{(3)} \longleftarrow \delta^{(3)}[1:]$ remove bias component : 4×1

$$\underset{\sim}{\delta}^{(2)} = \overbrace{\underbrace{\left(\underbrace{\underset{\sim}{W}^{(2)}}_{5\times 4}\right)^T \cdot \underbrace{\delta^{(3)}}_{4\times 1}}_{5\times 1}}^{e^{(2)}} .* \underbrace{\left[1, f'(z^{(2)})^T\right]}_{4\times 1}$$

$\hookrightarrow \delta^{(2)} \longleftarrow \delta^{(2)}[1:]$ remove bias component : 4×1

Note: if we are doing complete batch gradient descend, we update weights after each epoch, but if stochastic, we can update weights for every example pass

initialized as $\underset{\sim}{\Delta W} = 0$

- Weight changes: For each sample:

$\hat{y} = \text{forward}(\underset{\sim}{x}) \longrightarrow a^{(1)}, a^{(2)}, a^{(3)}, a^{(4)} = \hat{y}$

$\text{backward}(\hat{y}) \longrightarrow \delta^{(4)}, \delta^{(3)}, \delta^{(2)}$ (no $\delta^{(1)}$!)

$$\underset{\sim}{\Delta W}^{(l)} = \underset{\sim}{\Delta W}^{(l)} + \underbrace{\delta^{(l+1)}}_{S_{l+1}\times 1} \cdot \underbrace{a^{(l)T}}_{1\times(S_l+1)}$$

Then, after each epoch

$$\underset{\sim}{W}^{(l)} = \underbrace{\underset{\sim}{W}^{(l)}}_{\text{old}} + \underbrace{\frac{\alpha}{m}\underset{\sim}{\Delta W}^{(l)}}_{\text{step}}$$

NEURAL NETWORKS	Notes on Udacity notation

In the videos an exercises from Udacity, the weight matrix is defined the other way around, compared to how Andrew Ng does it:

- Andrew Ng: $\underset{\sim}{W} : (S_{l+1} , S_l + 1)$: (new units, old units + 1 bias)
- Udacity: $\underset{\sim}{W}$: (old units, new units)

Given the order how we insert the sizes the layers / weight matrizes, the Udacity approach is maybe more intuitive.
Note that in that case, the multiplication order needs to be changed to match the sizes:

$$\underset{\sim}{Z} = \underset{\sim}{a} \cdot \underset{\sim}{W} \quad : \quad (\text{batch}, \text{old}) \times (\text{old}, \text{new}) = (\text{batch}, \text{new})$$

$$\underset{\sim}{Z} \leftarrow \underset{\sim}{Z} + \underset{\sim}{b} \quad : \text{ bias is added, not inserted in } \underset{\sim}{W}$$

$$\underset{\sim}{b} : (\text{batch}, \text{new})$$