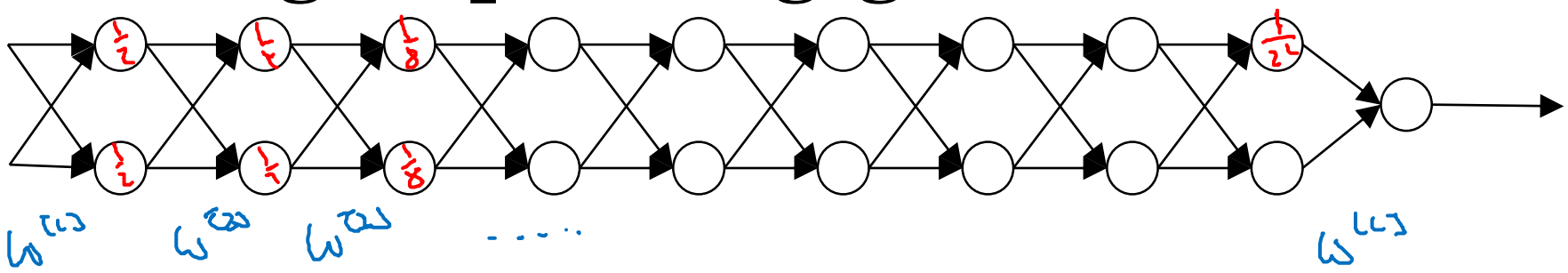
deeplearning.ai

Setting up your optimization problem

Vanishing/exploding gradients

Vanishing/exploding gradients



$L = 150$

$g(z) = z$ $\quad b^{[l]} = 0$

$\hat{y} = w^{[L]} w^{[L-1]} w^{[L-2]} \cdots w^{[3]} w^{[2]} w^{[1]} x$ $\quad a^{[3]}$

1.5^L

0.5^L

$w^{[l]} > I$

$w^{[l]} < I \quad \begin{bmatrix} 0.9 & \\ & 0.9 \end{bmatrix}$

$z^{[1]} = w^{[1]} x$

$a^{[1]} = g(z^{[1]}) = z^{[1]}$

$a^{[2]} = g(z^{[2]}) = g(w^{[2]} a^{[1]})$

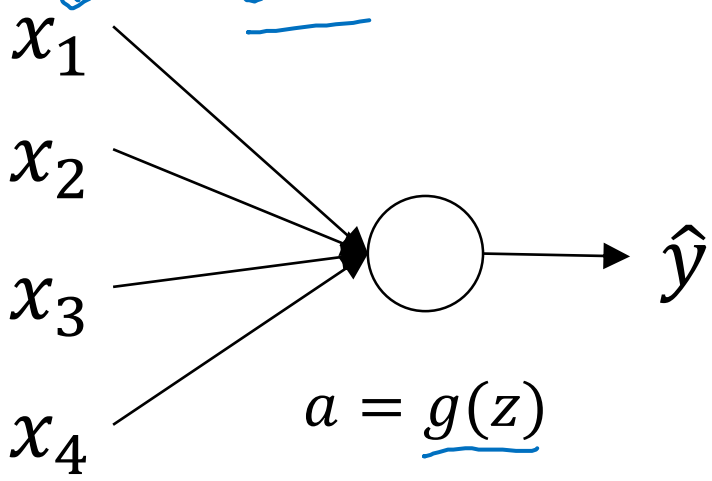
$w^{[l]} = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$ (1.5 → 0.5)

$\hat{y} = w^{[L]} \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}^{L-1} x$ (1.5 → 0.5)

$1.5^{L-1} x$

$0.5^{L-1} x$

Single neuron example



$w^{[l]}$

$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$

large n → smaller w_i

$Var(w_i) = \frac{1}{n} \quad \frac{2}{n}$

$w^{[l]} = np.random.randn(shape) * np.sqrt\left(\frac{2}{n^{[l-1]}}\right)$

ReLU $g^{[l]}(z) = ReLU(z)$

Other variants:

tanh

$\frac{1}{n^{[l-1]}}$

Xavier initialization

$\frac{2}{n^{[l-1]} + n^{[l]}}$