



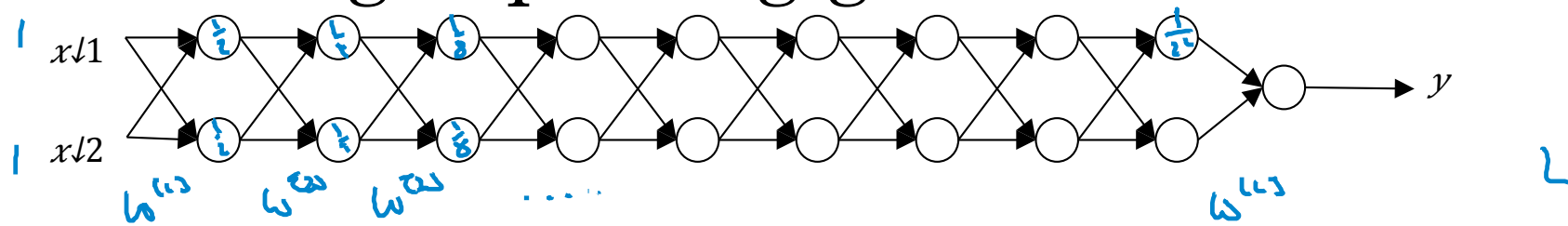
deeplearning.ai

Setting up your
optimization problem

Vanishing/exploding
gradients

Vanishing/exploding gradients

$L=150$



$g(z) = z$ $b^{(L)} = 0$

$\hat{y} = w^{(L)} \left(w^{(L-1)} \left(w^{(L-2)} \dots \left(w^{(1)} \left(w^{(1)} x \right) \right) \right) \right)$

15^L
 0.5^L

$w^{(1)} > 1$

$w^{(2)} < 1$ $\begin{bmatrix} 0.9 & \\ & 0.9 \end{bmatrix}$

$w^{(2)} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$

$z^{(1)} = w^{(1)} x$

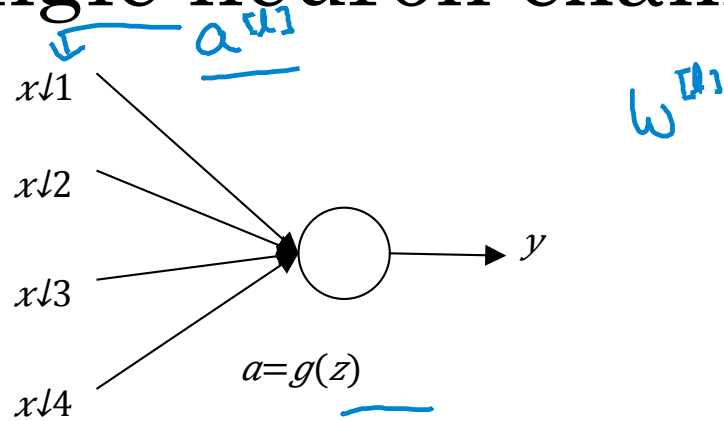
$a^{(1)} = g(z^{(1)}) = z^{(1)}$

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

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

$1.5^{L-1} x$
 $0.5^{L-1} x$

Single neuron example



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

large $n \rightarrow$ smaller w_i

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

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

ReLU $g^{[1]}(z) = \text{ReLU}(z)$

Other variants:

$$\text{tanh} \left(\frac{1}{n^{[1-1]}} \right)$$

Xavier initialization \uparrow

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

\uparrow