deeplearning.ai

Setting up your optimization problem

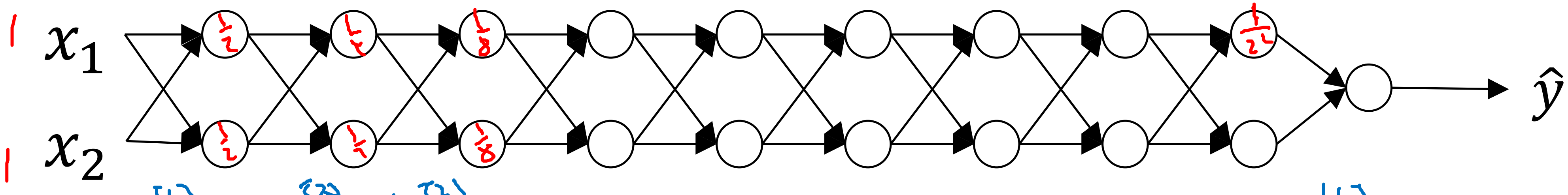
Vanishing/exploding gradients

Your derivatives or your slopes can sometimes get either very very big, or very very small, maybe exponentially small, and this makes training difficult.

a similar argument can be used to show that the derivatives or the gradients the computer is going to send will also increase exponentially or decrease exponentially as a function of the number of layers.

Vanishing/exploding gradients

$L = 150$



This makes training difficult,especially if your gradients are exponentially smaller than L. Then gradient descent will take tiny little steps. It will take a long time for gradient descent to learn anything.

L

$W^{[1]} \quad W^{[2]} \quad W^{[3]} \quad \cdots \quad W^{[L]}$

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

$\hat{y} = W^{[L]} W^{[L-1]} W^{[L-2]} \cdots \underbrace{W^{[3]} W^{[2]} W^{[1]} x}_{a^{[3]}}$

1.5^L

0.5^L

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

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

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

$W^{[l]} > I$ Identity matrices

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

Increasing/Decreasing exponentially.

$W^{[l]} = \begin{bmatrix} \cancel{1.5} \; 0.5 & 0 \\ 0 & \cancel{1.5} \; 0.5 \end{bmatrix}$

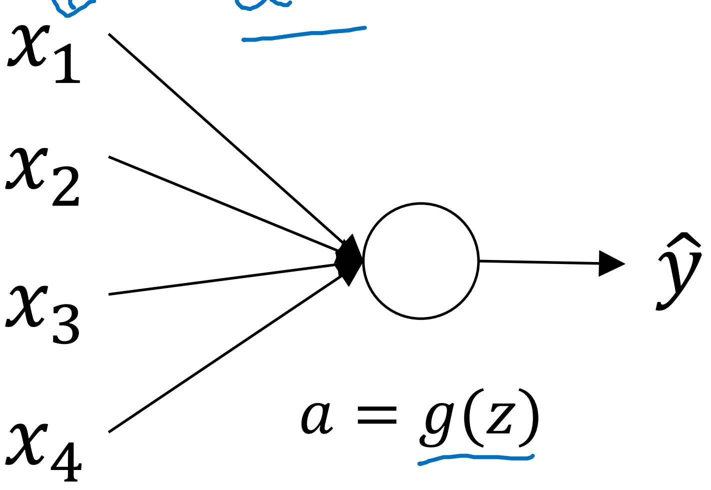
$\hat{y} = W^{[L]} \begin{bmatrix} \cancel{1.5} \; 0.5 & 0 \\ 0 & \cancel{1.5} \; 0.5 \end{bmatrix}^{L-1} x$

$1.5^{L-1} x$

$0.5^{L-1} x$

Andrew Ng

Single neuron example



$a^{[l]}$

$w^{[l]}$

$a = g(z)$

$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \cancel{+b}$

Large n → Smaller w_i

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

n #of features

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

ReLU

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

Other variants:

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

Xavier initialization

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