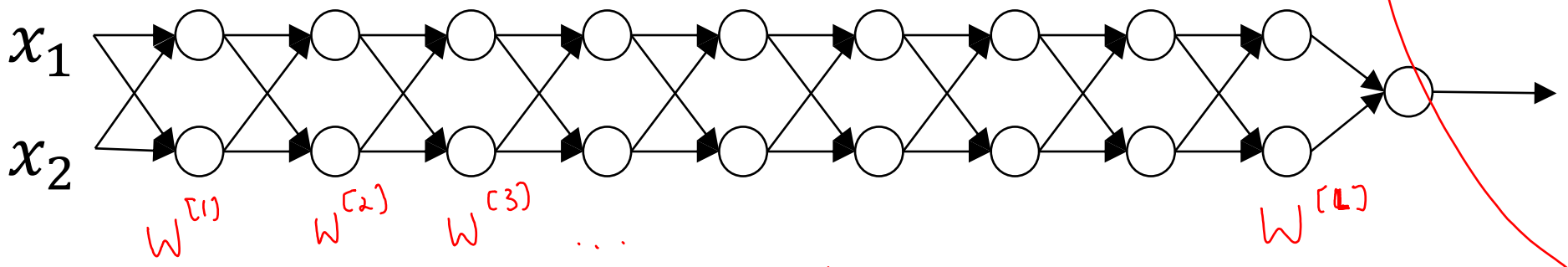
deeplearning.ai

Setting up your optimization problem

Vanishing/exploding gradients

Vanishing/exploding gradients



This argument was for Activations'

$W^{[l]}$ is small $\Rightarrow z$ is small $\Rightarrow a$ is small, the same argument can be for their gradient dw is small $\Rightarrow dz$ is small $\Rightarrow da$ is small

$g(z) = z$ (Linear Activation Func)

$b^{(l)} = 0$ (All bias $= 0$)

$\Rightarrow y = W^{[L]} \cdot W^{[L-1]} \cdot W^{[L-2]} \cdots W^{[3]} \cdot W^{[2]} \cdot W^{[1]} \cdot X$

$W^{[1]} \cdot X = z^{[1]}$ (because $b^{[1]} = 0$)

Also $g(z) = z = a$

$\Rightarrow a^{[1]} = z^{[1]}$

$\Rightarrow W^{[1]} X = a^{[1]}$

$\Rightarrow W^{[2]} a^{[1]} = z^{[2]}$, but $g(z^{[2]}) = a^{[2]} \Rightarrow W^{[2]} \cdot a^{[1]} = a^{[2]}$

$\Rightarrow W^{[L]} \cdot a^{[L-1]} = a^{[L]} = \hat{y}$

Conversely If $W^{[l]} = \begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}$

then we have $(0.5)^{30}$

$\Rightarrow$ Vanishing gradient

dim for each w

$n^{[l]}, n^{[l-1]}$

$= 2 \times 2$

Say we get values for each w as

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

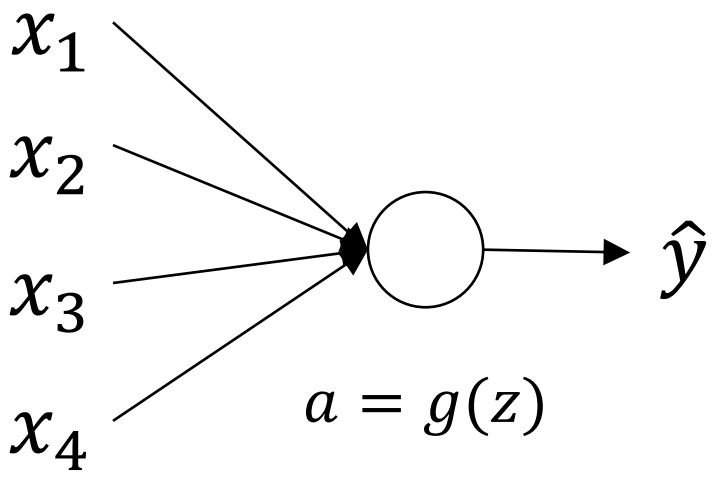
$\Rightarrow \hat{y} = W^{[L]} \cdot \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}^{L-1} \cdot X \Rightarrow W^{[L]} \cdot \begin{bmatrix} 1.5^{L-1} & 0 \\ 0 & 1.5^{L-1} \end{bmatrix} \cdot X$, $\Rightarrow$ If "L" is a large number, say 30

$\Rightarrow 1.5^{(30)}$ is large $\Rightarrow$ exploding gradient

$^{L-1}$ → to the power, ie, matrix A, multiplied (L-1) times by itself

Single neuron example

(weight Initialization for deep neural networks)
↳ partial solution to Vanishing/exploding gradients is careful initialization



$a = g(z)$

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

If n is large, ie, # features is large ⇒ you want all w_i to be small

- one way to do this

★ $\text{Var}(w_i) = \frac{1}{n}$, can also make $\text{Var}(w_i) = 2/n$ [same thing, prop to $1/n$]

If → $w^{[l]} = \text{np.random.randn}(\text{shape}(w)) * \boxed{\text{np.sqrt}\left(\frac{2}{n^{[l-1]}}\right)}$ ⇒ then $\text{Var}(w^{[l]}) = 2/n$

↳ # features feeding into layer "l"

Note
If you are using ReLU Activation then setting $\text{Var}(w_i) = 2/n$ is better

May be good to make Var(w) a hyper param to change based on Activation func

other variants

If using a tanh Activation use

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

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

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

This doesn't solve vanishing/exploding gradient, but ↓ it significantly
⇒ So as long as we normalize i/ps Initialize w's like this, we can solve this problem

Andrew Ng