deeplearning.ai

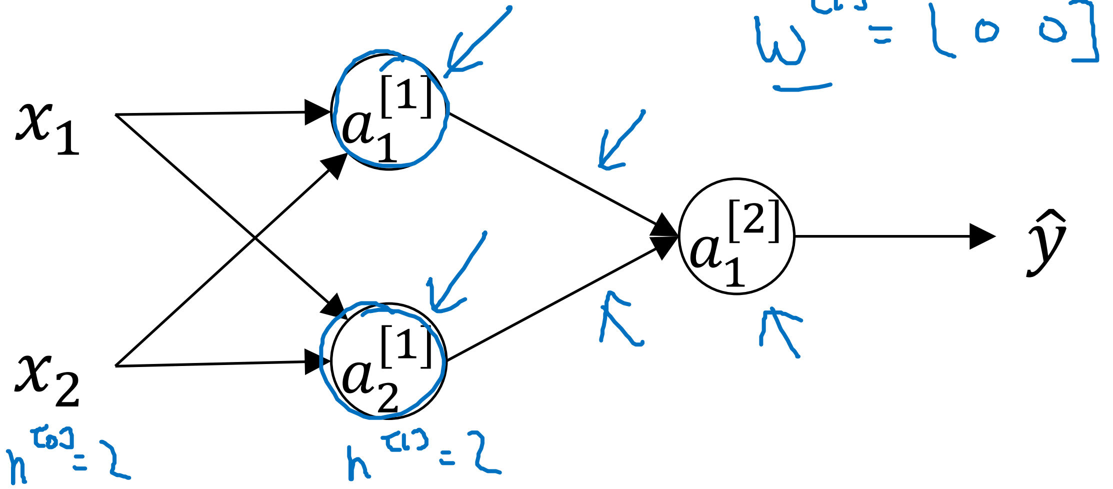
One hidden layer Neural Network

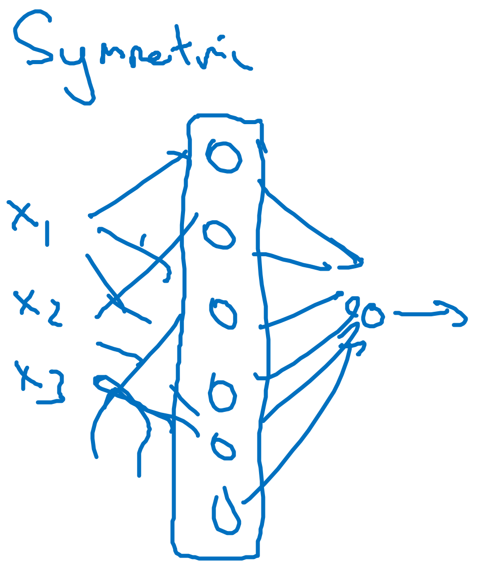
Random Initialization

When you train your network, it's very important to initialize the weights randomly, for logistic regression it was okay to initialize the weights to zero. But for a neural network of initialize the arrays of parameters to all zero, and apply gradient descent it won't work.

What happens if you initialize weights to zero?

No matter how long you train your neural network, both hidden units are still computing exactly the same function, and so in this case there's really no point to having more than one hidden unit, because they're all computing the same thing.





$$W^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

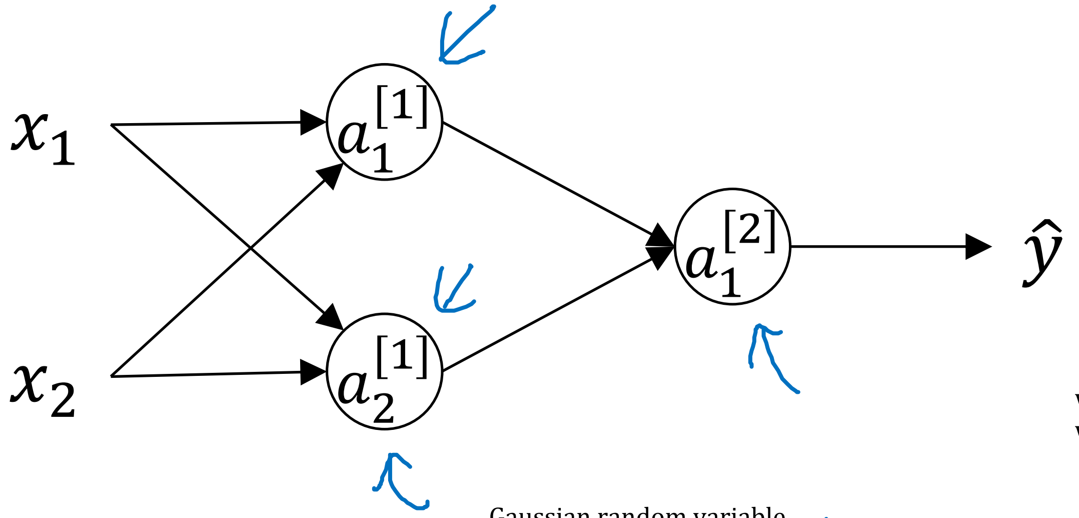
$$a_1^{[1]} = a_2^{[1]} \qquad dz_1^{[1]} = dz_2^{[1]}$$

$$W^{[1]} = \begin{bmatrix} \cdots\cdots \\ \cdots\cdots \end{bmatrix}$$

$$dw = \begin{bmatrix} u & v \\ u & v \end{bmatrix} \qquad W^{[1]} = W^{[1]} - \alpha dW$$

Andrew Ng

Random initialization



when you're training a very very deep neural network, then you might want to pick a different constant as 0.01

Gaussian random variable

$\rightarrow$ $W^{[1]} =$ np.random.randn((2,2)) $* \frac{0.01}{100?}$

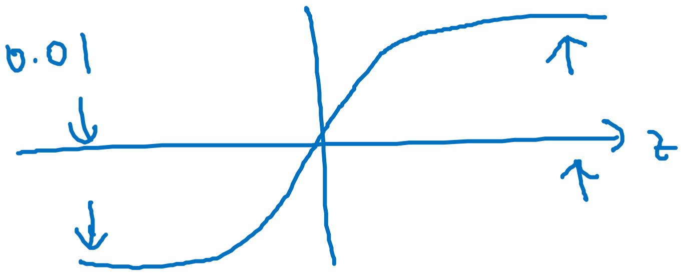
$b^{[1]} =$ np.zeros((2,1))

$W^{[2]} =$ np.random.randn((1,2)) $* 0.01$

$b^{[2]} = 0$

$\rightarrow$ $z^{[1]} = W^{[1]} x + b^{[1]}$

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



So that's it so this week videos you know how to setup a neural network of a hidden layer, initialize the parameters, make predictions using forward prop as well as compute derivatives and apply gradient descent using back prop.