deeplearning.ai

Multi-class classification

Trying a softmax classifier

Understanding softmax

(4,1)

$$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \qquad t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix}$$

C = 4 $\qquad g^{[L]}(\cdot)$

"soft max"

"hard max"

$$a^{[L]} = g^{[L]}(z^{[L]}) = \begin{bmatrix} e^5/(e^5+e^2+e^{-1}+e^3) \\ e^2/(e^5+e^2+e^{-1}+e^3) \\ e^{-1}/(e^5+e^2+e^{-1}+e^3) \\ e^3/(e^5+e^2+e^{-1}+e^3) \end{bmatrix} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Softmax regression generalizes logistic regression to C classes.

If $\underline{C=2}$, softmax reduces to logistic regression. $a^{[L]} = \begin{bmatrix} 0.842 \\ \cancel{0.158} \end{bmatrix}$

Andrew Ng

Loss function

$$\underset{(4,1)}{y^{(i)}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ — cat} \qquad y_2 = 1 \qquad y_1 = y_3 = y_4 = 0$$

$$a^{[L](i)} = \underset{(4,1)}{\hat{y}^{(i)}} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix} \qquad C = 4$$

$$\underbrace{\mathcal{L}(\hat{y}, y)}_{\text{small}} = \underbrace{-\sum_{j=1}^{4} y_j \log \hat{y}_j}$$

$$J(w^{[1]}, b^{[1]}, \dots) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

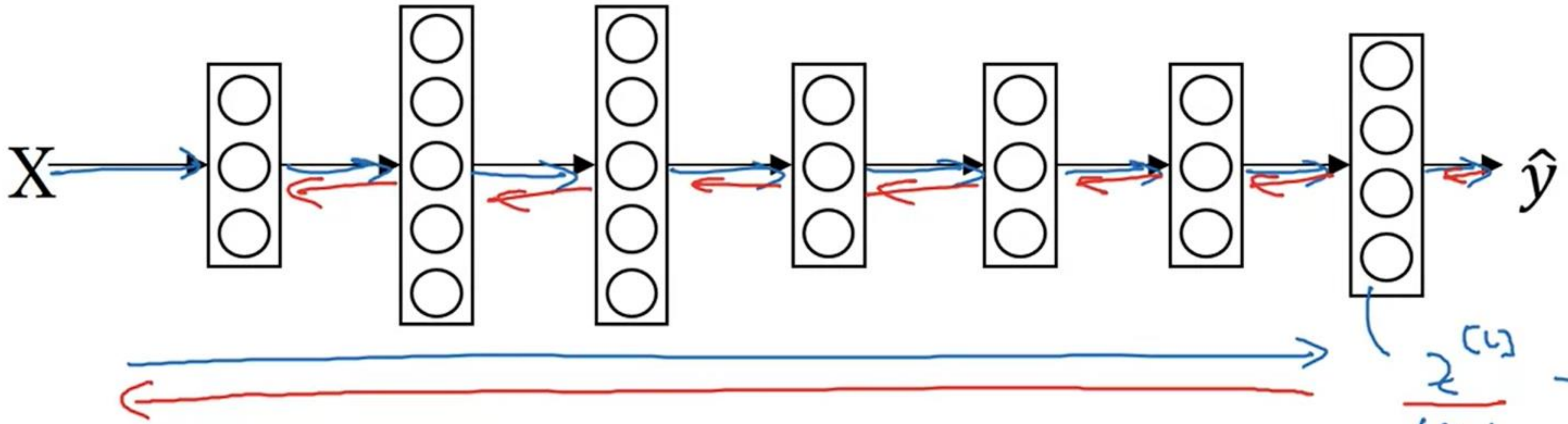
$$-y_2 \log \hat{y}_2 = \underline{-\log \hat{y}_2}. \qquad \text{Make } \hat{y}_2 \text{ big.}$$

$$Y = [y^{(1)}\ y^{(2)} \dots, y^{(m)}] = \underset{(4,m)}{\begin{bmatrix} 0 & 0 & 1 & \\ 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \\ 0 & 0 & 0 & \end{bmatrix}}$$

$$\hat{Y} = [\hat{y}^{(1)}, \dots, y^{(m)}] = \underset{(4,m)}{\begin{bmatrix} 0.3 & \\ 0.2 & \dots \\ 0.1 & \\ 0.4 & \end{bmatrix}}$$

Andrew Ng

Gradient descent with softmax



$\underset{(4,1)}{z^{[L]}} \longrightarrow a^{[L]} = \hat{y} \to \mathcal{L}(\hat{y}, y)$

Backprop: $\underset{(4,1)}{dz^{[L]}} = \hat{y} - y$

$\frac{\partial J}{\partial z^{[L]}}$