One hidden layer
Neural Network

Gradient descent for
neural networks

deeplearning.ai

Gradient descent for neural networks

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$
$\quad (n^{[1]}, n^{[0]}) \quad (n^{[1]}, 1) \quad (n^{[2]}, n^{[1]}) \quad (n^{[2]}, 1)$

$n_x = n^{[0]}, \ n^{[1]}, \ \underline{n^{[2]} = 1}$

Cost function: $J(W^{[1]}, b^{[1]}, \underline{W^{[2]}}, \underline{b^{[2]}}) = \dfrac{1}{m} \sum\limits_{i=1}^{n} \mathcal{L}(\hat{y}, y)$
$\qquad\qquad\qquad\qquad\qquad\quad \uparrow \qquad \uparrow \qquad\qquad\qquad\qquad\quad \hookleftarrow a^{[2]}$

Gradient Descent:

$\rightarrow$ Repeat {

$\quad \rightarrow$ Compute predicts $(\hat{y}^{(i)}, \ i=1,\dots,m)$

$\quad dW^{[1]} = \dfrac{\partial J}{\partial W^{[1]}}, \quad db^{[1]} = \dfrac{\partial J}{\partial b^{[1]}}, \quad \dots$

$\quad W^{[1]} := W^{[1]} - \alpha \, dW^{[1]}$

$\quad b^{[1]} := b^{[1]} - \alpha \, db^{[1]}$

$\quad W^{[2]} := \dots \qquad b^{[2]} := \dots$

}

Andrew Ng

Formulas for computing derivatives

Forward propagation:

$$z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = g^{[1]}(z^{[1]}) \leftarrow$$

$$z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(z^{[2]}) = \sigma(z^{[2]})$$

Back propagation:

$$dz^{[2]} = A^{[2]} - Y \leftarrow$$

$$dW^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dz^{[2]}, axis=1, \underline{keepdims=True})$$

$$dz^{[1]} = \underbrace{W^{[2]T} dz^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]'}(z^{[1]})}_{\uparrow \text{ element-wise product}} \quad (n^{[1]}, m)$$

$$dW^{[1]} = \frac{1}{m} dz^{[1]} X^T$$

$$db^{[1]} = \frac{1}{m} np.sum(dz^{[1]}, axis=1, \underline{keepdims=True})$$

$$\underline{(n^{[1]}, 1)} \qquad (n^{[1]},) \qquad \underset{\uparrow \ reshape}{}$$

$$Y = [y^{(1)} \ y^{(2)} \cdots , y^{(m)}]$$

$$(n^{[3]}_s) \leftarrow$$

$$(n^{[2]}, 1) \leftarrow$$

Andrew Ng