deeplearning.ai

One hidden layer Neural Network

Gradient descent for neural networks

Gradient descent for neural networks

Parameters: $W^{[1]}$, $b^{[1]}$, $W^{[2]}$, $b^{[2]}$ $\quad n_x = n^{[0]}, \; n^{[1]}, \; \underline{n^{[2]} = 1}$

$(n^{[1]}, n^{[0]})$ $\quad (n^{[1]}, 1)$ $\quad (n^{[2]}, n^{[1]})$ $\quad (n^{[2]}, 1)$

Cost function: $J(W^{[1]}, b^{[1]}, \underline{W^{[2]}}, \underline{b^{[2]}}) = \frac{1}{m} \sum_{i=1}^{n} \mathcal{L}(\hat{y}, y)$ $\quad \uparrow \; a^{[2]}$

Gradient descent:

→ Repeat {

→ Compute predicts $(\hat{y}^{(i)}, \; i = 1 \ldots, m)$

$\underline{dW^{[1]}} = \frac{\partial J}{\partial W^{[1]}}, \quad \underline{db^{[1]}} = \frac{\partial J}{\partial b^{[1]}}, \ldots$

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

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

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

}

Andrew Ng

Formulas for computing derivatives

Forward propagation:

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

$A^{[1]} = g^{[1]}(Z^{[1]})$

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

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

Back propagation:

$dZ^{[2]} = A^{[2]} - Y$

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

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

$(n^{[1]},)$

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

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

$dZ^{[1]} = \underbrace{W^{[2]T} dZ^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]'}(Z^{[1]})}_{(n^{[1]}, m)}$

element-wise product

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

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

$(n^{[1]}, 1)$ $(n^{[1]},)$ reshape

Andrew Ng