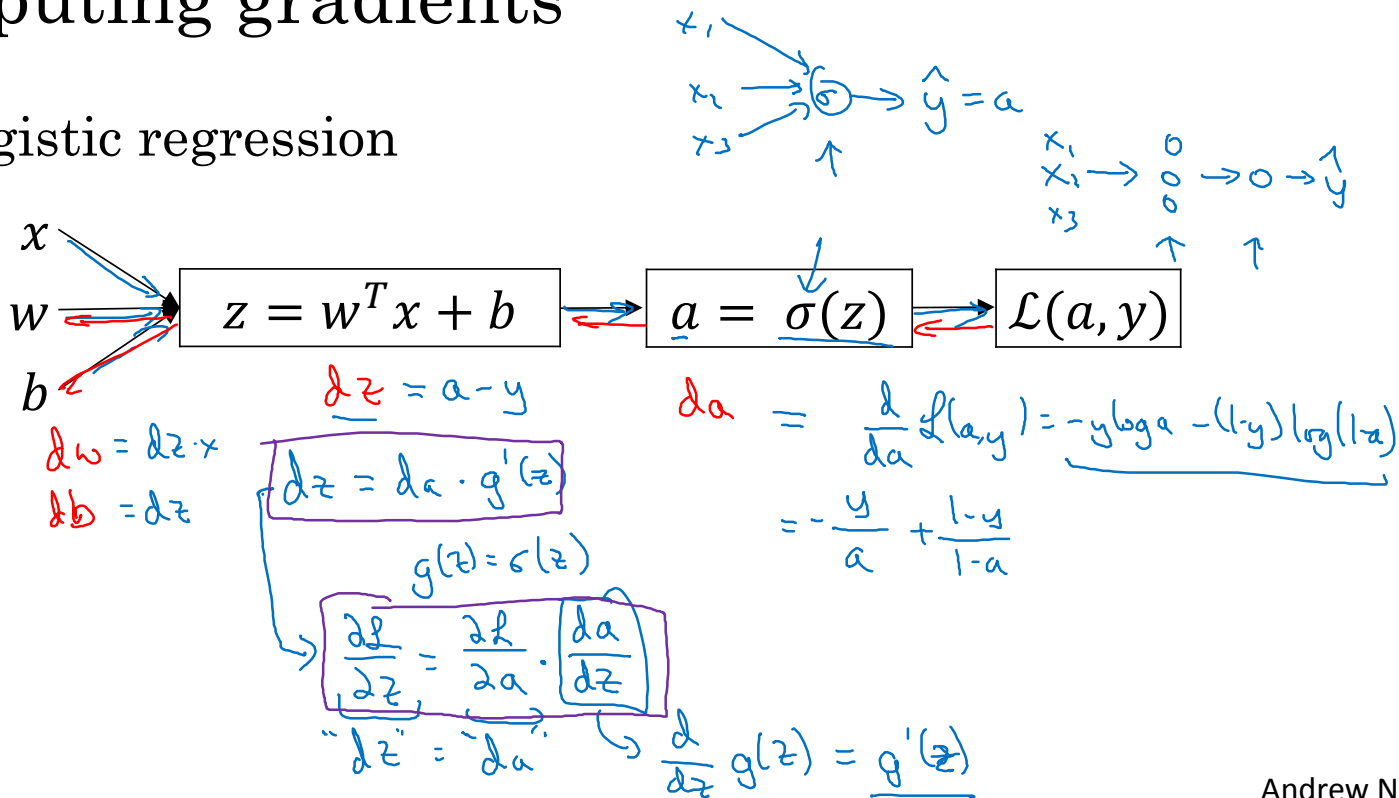deeplearning.ai

One hidden layer
Neural Network
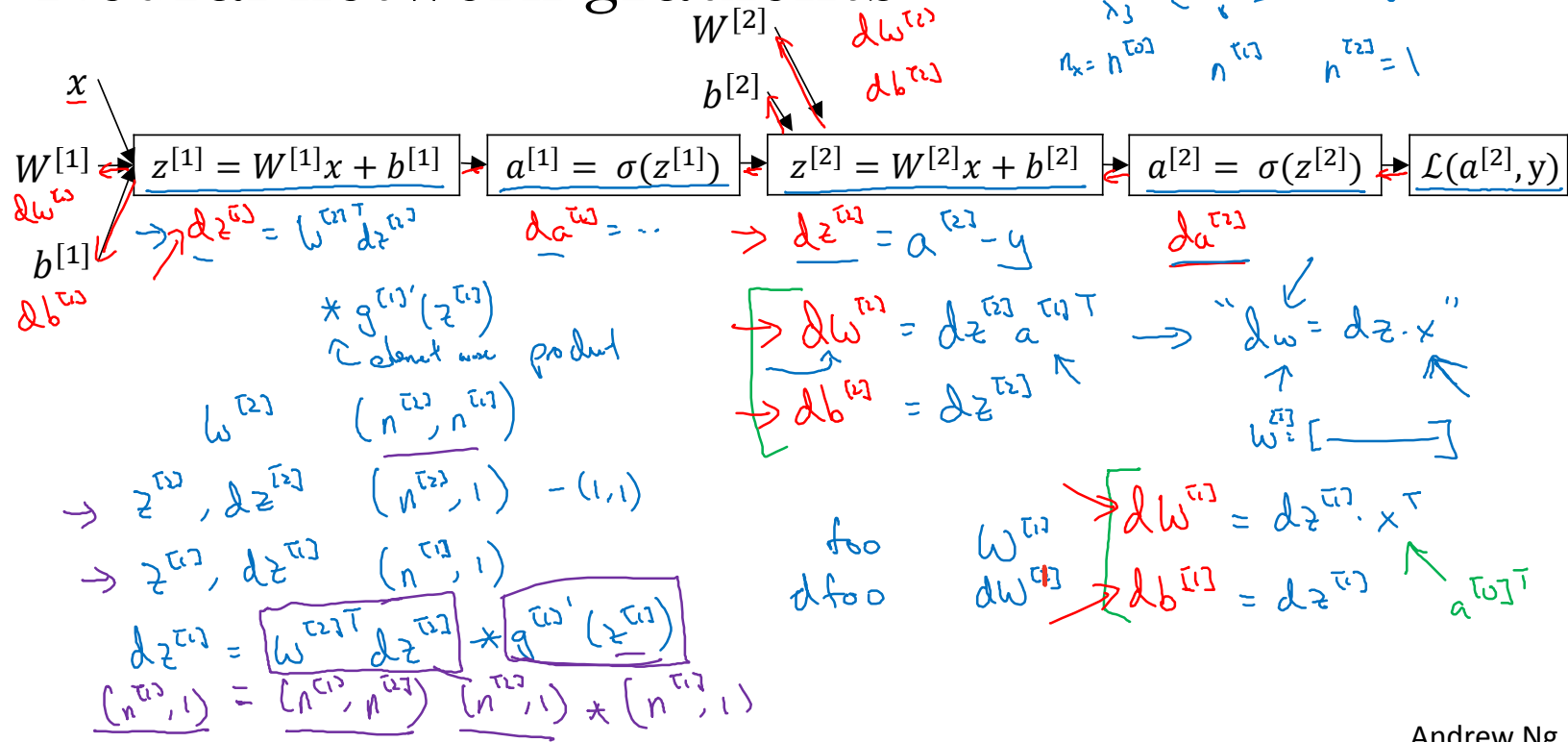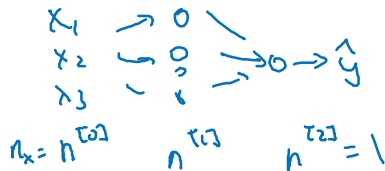
Backpropagation
intuition (Optional)

Computing gradients

Logistic regression

$$z = w^T x + b$$

$$a = \sigma(z)$$

$$\mathcal{L}(a, y)$$

$$dz = a - y$$

$$dw = dz \cdot x$$

$$db = dz$$

$$dz = da \cdot g'(z)$$

$$g(z) = \sigma(z)$$

$$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial a} \cdot \frac{da}{dz}$$

$$\text{"} dz \text{"} = \text{"} da \text{"} \cdot \frac{d}{dz} g(z) = g'(z)$$

$$da = \frac{d}{da} \mathcal{L}(a,y) = -y \log a - (1-y) \log(1-a)$$

$$= -\frac{y}{a} + \frac{1-y}{1-a}$$

Andrew Ng

Neural network gradients

$x_1 \to 0$
$x_2 \to 0 \to 0 \to \hat{y}$
$x_3 \to ?$

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

$W^{[2]}$ $\quad dW^{[2]}$

$b^{[2]}$ $\quad db^{[2]}$

x

$W^{[1]}$
$dW^{[1]}$

$b^{[1]}$
$db^{[1]}$

$$\boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \to \boxed{a^{[1]} = \sigma(z^{[1]})} \to \boxed{z^{[2]} = W^{[2]}x + b^{[2]}} \to \boxed{a^{[2]} = \sigma(z^{[2]})} \to \boxed{\mathcal{L}(a^{[2]}, y)}$$

$\to dz^{[1]} = W^{[1]T} dz^{[1]}$

$da^{[1]} = \cdots$ $\quad \to dz^{[2]} = a^{[2]} - y$ $\quad da^{[2]}$

$* g^{[1]'}(z^{[1]})$
$\uparrow$ element wise product

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

$\to z^{[2]}, dz^{[2]} \quad (n^{[2]}, 1) - (1,1)$

$\to z^{[1]}, dz^{[1]} \quad (n^{[1]}, 1)$

$dz^{[1]} = \boxed{W^{[2]T} \, dz^{[2]}} * \boxed{g^{[1]'}(z^{[1]})}$

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

$\to dW^{[2]} = dz^{[2]} a^{[1]T} \quad \to \text{``} dw = dz \cdot x \text{''}$

$\to db^{[2]} = dz^{[2]} \uparrow$
$\qquad\qquad\qquad\qquad \uparrow$
$\qquad\qquad\qquad W^{[2]}[\underline{\qquad}]$

$foo \qquad W^{[1]} \to dW^{[1]} = dz^{[1]} \cdot x^T$

$dfoo \qquad dW^{[1]} \to db^{[1]} = dz^{[1]} \qquad a^{[0]T}$

Andrew Ng

Summary of gradient descent

$dz^{[2]} = a^{[2]} - y$

$dW^{[2]} = dz^{[2]}a^{[1]T}$

$db^{[2]} = dz^{[2]}$

$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]'}(z^{[1]})$

$dW^{[1]} = dz^{[1]}x^T$

$db^{[1]} = dz^{[1]}$

Vectorized Implementation:

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

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

$$Z^{[1]} = \begin{bmatrix} z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](n)} \end{bmatrix}$$

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

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

Andrew Ng

Summary of gradient descent ☆

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} a^{[1]^T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]^T} dz^{[2]} * g^{[1]'}(z^{[1]})$$
$$(n^{[1]}, 1)$$

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

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

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

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

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

elementwise product

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

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

$$J(..) = \frac{1}{m} \sum_{i=1}^{n} \mathcal{L}(\hat{y}, y)$$

Andrew Ng