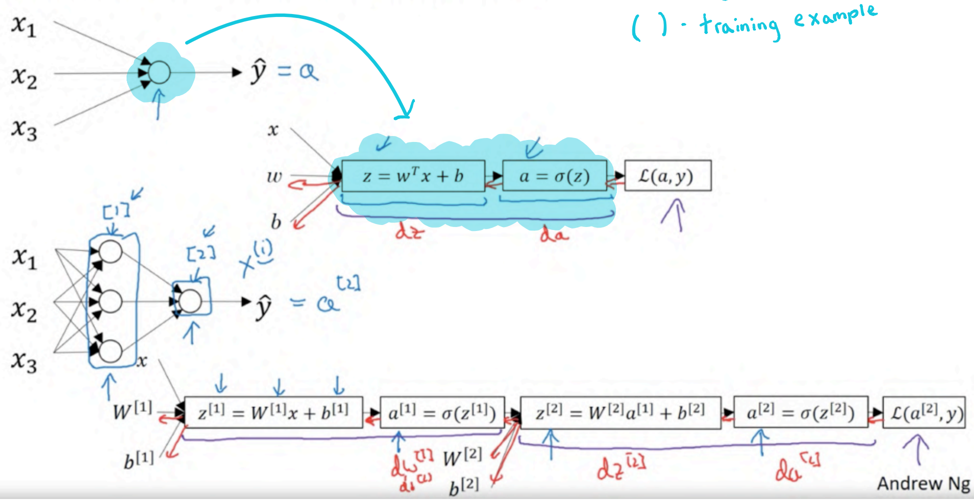
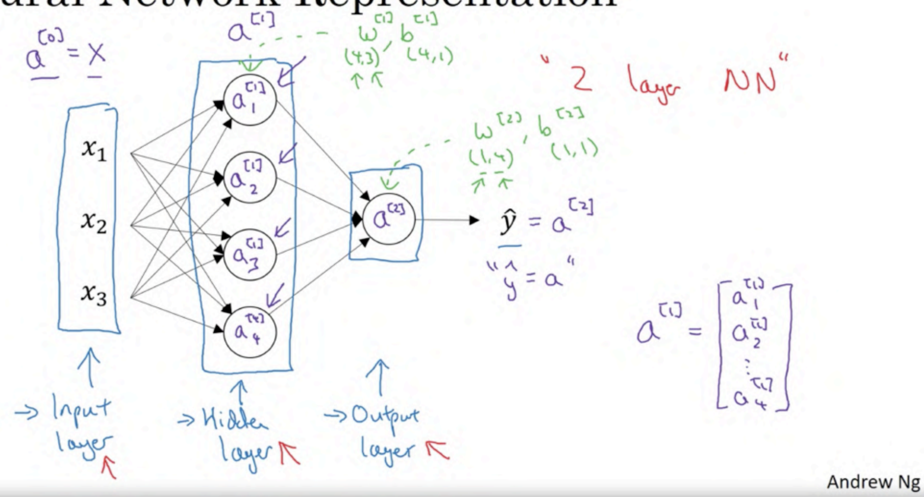
What is a Neural Network?

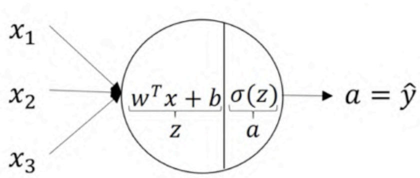
[ ] - layer
( ) - training example



Neural Network Representation

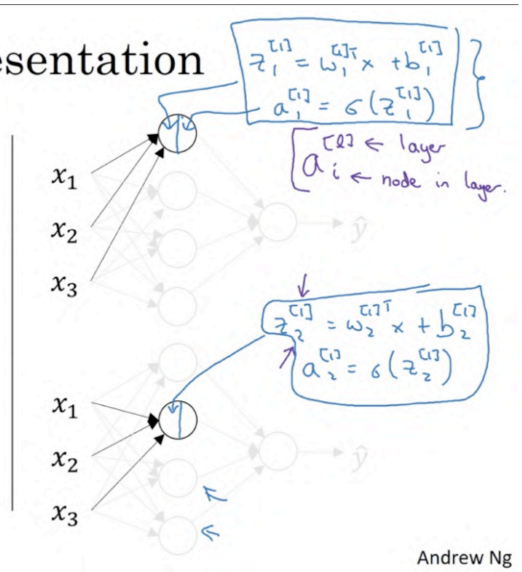


Neural Network Representation

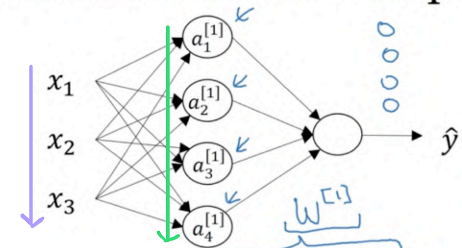


$z = w^T x + b$

$a = \sigma(z)$



Neural Network Representation



$(w_1^{[1]})^T x$ $\quad a^{[1]}$

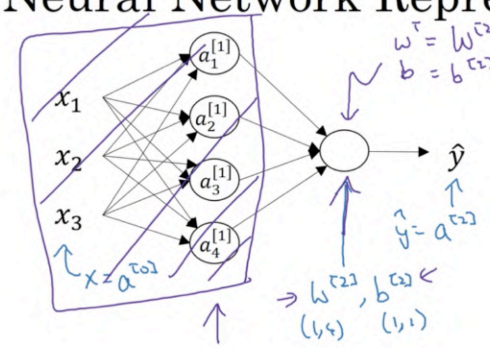
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

$$z^{[1]} = \underbrace{\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix}}_{4 \times 3}^{W^{[1]}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \underbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}_{b^{[1]}\ (4,1)} = \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \\ w_4^{[1]T} x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} \ (4,1)$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]}) \quad (4,1)$$

Andrew Ng

Neural Network Representation learning



$z = w^T x + b$

$\hat{y} = a = \sigma(z)$

Given input x:

$$\underset{(4,1)}{z^{[1]}} = \underset{(4,3)}{W^{[1]}} \underset{(3,1)}{\cancel{x}\, a^{[0]}} + \underset{(4,1)}{b^{[1]}}$$

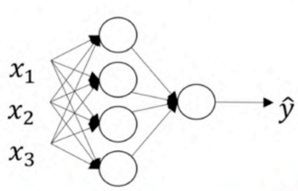
$$\underset{(4,1)}{a^{[1]}} = \sigma(\underset{(4,1)}{z^{[1]}})$$

$$\underset{(1,1)}{z^{[2]}} = \underset{(1,4)}{W^{[2]}} \underset{(4,1)}{a^{[1]}} + \underset{(1,1)}{b^{[2]}}$$

$$\underset{(1,1)}{a^{[2]}} = \sigma(\underset{(1,1)}{z^{[2]}})$$

Andrew Ng

Vectorizing across multiple examples



$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

$x \longrightarrow a^{[2]} = \hat{y}$

$x^{(1)} \longrightarrow a^{[2](1)} = \hat{y}^{(1)}$

$x^{(2)} \longrightarrow a^{[2](2)} = \hat{y}^{(2)}$

$\vdots$

$x^{(m)} \longrightarrow a^{[2](m)} = \hat{y}^{(m)}$

$a^{[2](i)}$ ← example i, ← layer 2

for $i = 1$ to m,

$\quad z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$

$\quad a^{[1](i)} = \sigma(z^{[1](i)})$

$\quad z^{[2](i)} = W^{[2]} a^{[1](i)} + b^{[2]}$

$\quad a^{[2](i)} = \sigma(z^{[2](i)})$

Andrew Ng

Vectorizing across multiple examples

```
for i = 1 to m:
```

$\boxed{z^{[1](i)}} = W^{[1]}x^{(i)} + b^{[1]}$

$\boxed{a^{[1](i)}} = \sigma(z^{[1](i)})$

$\boxed{z^{[2](i)}} = W^{[2]}a^{[1](i)} + b^{[2]}$

$\boxed{a^{[2](i)}} = \sigma(z^{[2](i)})$

$$\boxed{\begin{aligned} Z^{[1]} &= W^{[1]}X + b^{[1]} \\ A^{[1]} &= \sigma(Z^{[1]}) \\ Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\ A^{[2]} &= \sigma(Z^{[2]}) \end{aligned}}$$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad (n_x, m)$$

training examples

hidden units

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

$$A^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

hidden units

Andrew Ng

Justification for vectorized implementation

$z^{[1](1)} = W^{[1]}x^{(1)} + \cancel{b^{[1]}} \to 0, \quad z^{[1](2)} = W^{[1]}x^{(2)} + \cancel{b^{[1]}} \to 0, \quad z^{[1](3)} = W^{[1]}x^{(3)} + \cancel{b^{[1]}} \to 0$

$W^{[1]} = \begin{bmatrix} \text{---} \\ \text{---} \\ \text{---} \end{bmatrix} \qquad W^{[1]}x^{(1)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \qquad W^{[1]}x^{(2)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \qquad W^{[1]}x^{(3)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$

$$W^{[1]} \underbrace{\begin{bmatrix} | & | & | & \\ x^{(1)} & x^{(2)} & x^{(3)} & \cdots \\ | & | & | & \end{bmatrix}}_{X} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} | & | & | & \\ z^{[1](1)} & z^{[1](2)} & z^{[1](3)} & \cdots \\ | & | & | & \end{bmatrix} = Z^{[1]}$$

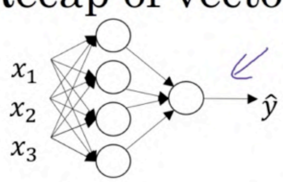
$+b^{[1]} \quad +b^{[1]} \quad +b^{[1]}$

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

$W^{[1]}x^{(i)} = z^{[1](i)}$

Andrew Ng

Recap of vectorizing across multiple examples



$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$$A^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

```
for i = 1 to m
```

$\boxed{z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}}$

$a^{[1](i)} = \sigma(z^{[1](i)})$

$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$

$a^{[2](i)} = \sigma(z^{[2](i)})$

$A^{[0]} \qquad X = a^{[0]} \qquad x^{(i)} = a^{[0](i)}$

$Z^{[1]} = W^{[1]}X + b^{[1]} \leftarrow W^{[1]}A^{[0]} + b^{[1]}$

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

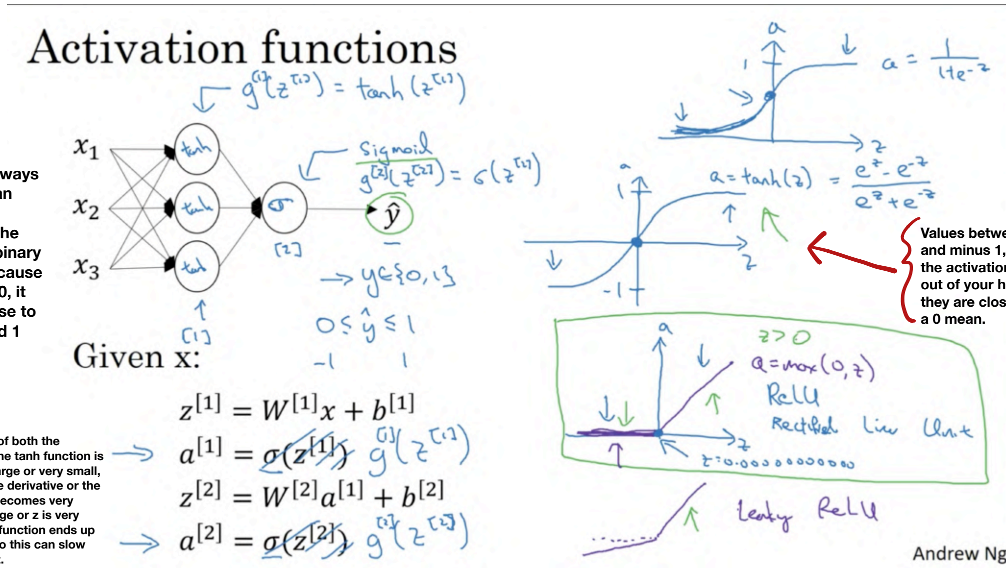
$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$

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

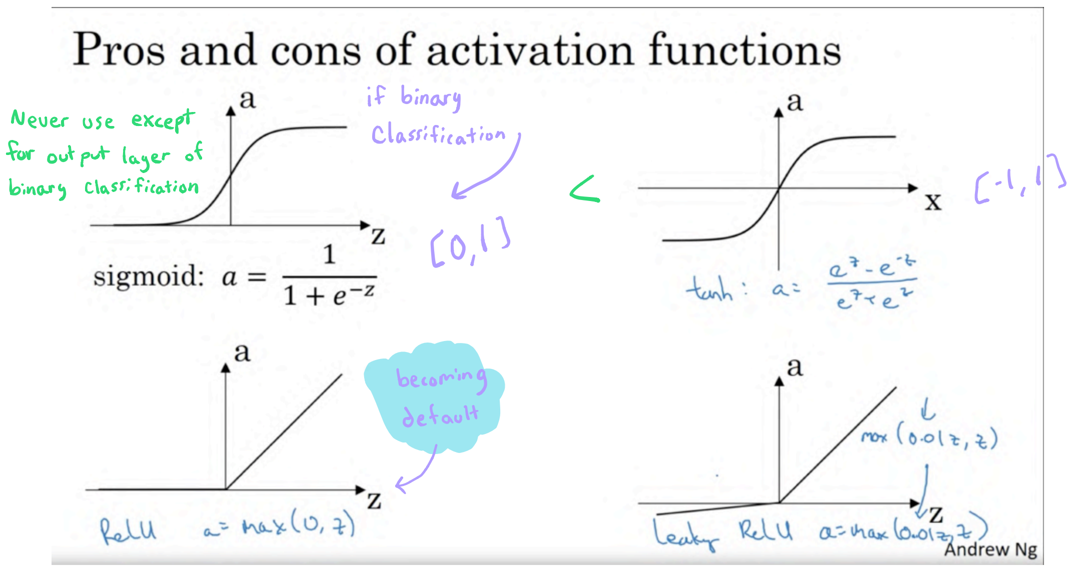
Andrew Ng

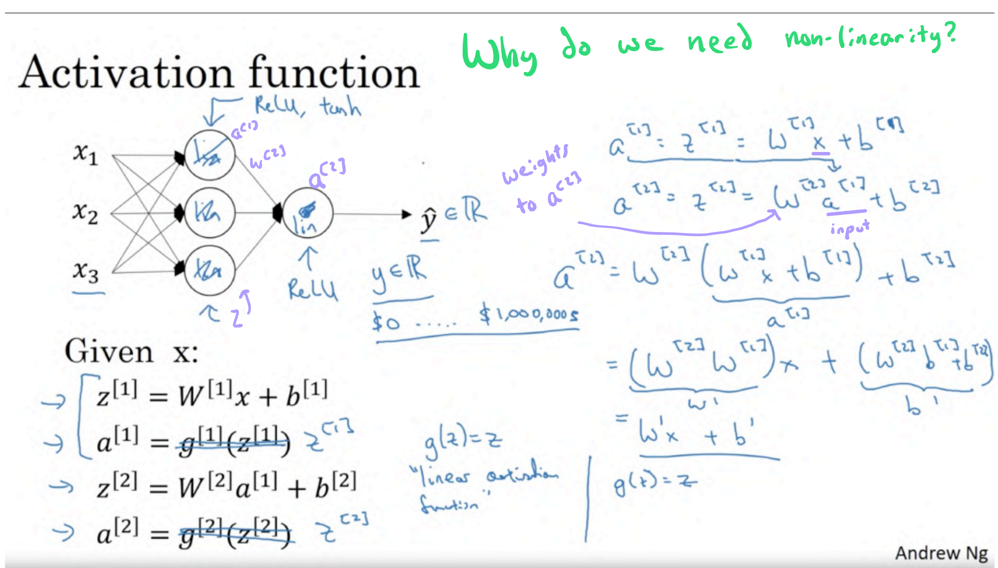
Tanh is almost always strictly better than sigmoid. Only exception is for the output layer for binary classification because if y is either 1 or 0, it makes more sense to be between 0 and 1 than 1 and -1.

One of the downsides of both the sigmoid function and the tanh function is that if z is either very large or very small, then the gradient or the derivative or the slope of this function becomes very small. So if z is very large or z is very small, the slope of the function ends up being close to 0. And so this can slow down gradient descent.



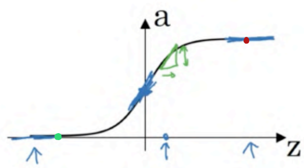
Values between plus 1 and minus 1, the mean of the activations that come out of your head, and they are closer to having a 0 mean.





Sigmoid activation function

$g'(z) = s(1-s)$



$$g(z) = \frac{1}{1+e^{-z}}$$

$a = g(z) = \frac{1}{1+e^{-z}}$

$g'(z) = \boxed{\frac{d}{dz} g(z)}$ = slope of $g(x)$ at z

$= \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right)$

$= g(z)\,(1 - g(z))$

$= \boxed{a\,(1-a)}$

$g'(z) = a(1-a)$

- $z = 10$. $g(z) \approx 1$

  $\frac{d}{dz} g(z) \approx 1\,(1-1) \approx 0$

- $z = -10$ $g(z) \approx 0$

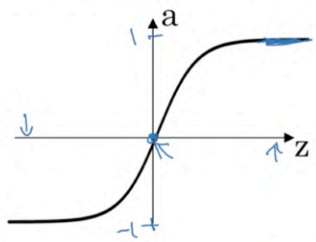
  $\frac{d}{dz} g(z) \approx 0 \cdot (1-0) \approx 0$

$z = 0$ $g(z) = \frac{1}{2}$

$\frac{d}{dz} g(z) = \frac{1}{2}\left(1 - \frac{1}{2}\right) = \frac{1}{4}$

Andrew Ng

Tanh activation function



$$g(z) = \tanh(z)$$

$$= \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$g'(z) = \frac{d}{dz} g(z)$ = slope of $g(z)$ at z

$= 1 - (\tanh(z))^2$

$a = g(z)$, $g'(z) = 1 - a^2$

$z = 10$ $\tanh(z) \approx 1$

$g'(z) \approx 0$

$z = -10$ $\tanh(z) \approx -1$

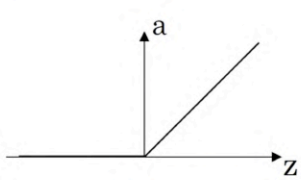
$g'(z) \approx 0$

$z = 0$ $\tanh(z) = 0$

$g'(z) = 1$

Andrew Ng

ReLU and Leaky ReLU

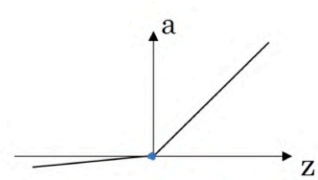


ReLU

$g(z) = \max(0, z)$

$$g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \\ \cancel{\text{undef}} & \cancel{\text{if } z = 0} \end{cases}$$

$z = 0.00000\ldots0$



Leaky ReLU

$g(z) = \max(0.01z, z)$

$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Andrew Ng

Gradient descent for neural networks

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

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 $(\hat{y}^{(i)}, \; i = 1 \dots m)$

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

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

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

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:

$dZ^{[2]} = A^{[2]} - Y$ ← $\quad Y = [y^{(1)} \; y^{(2)} \dots y^{(m)}]$

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

Sum horizontally

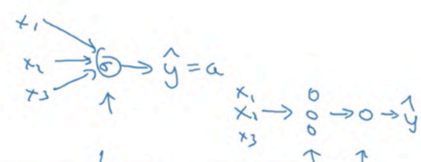
$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis}=1, \text{keepdims}=\text{True})$ $\quad (n^{[2]}, 1)$ ← $\quad (n,) \leftarrow$ No $\quad \rightarrow (n,1)$

$dZ^{[1]} = \underbrace{W^{[2]T} dZ^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]\prime}(Z^{[1]})}_{(n^{[1]}, m)}$ $\quad \uparrow$ element-wise product

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

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

Computing gradients



Logistic regression

$x, w, b \rightarrow \boxed{z = w^T x + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(a, y)}$

$dz = a - y$

$dw = dz \cdot x$

$db = dz$

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

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

$\boxed{dz = da \cdot g'(z)}$

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

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

"dz" = "da" $\quad \frac{d}{dz} g(z) = \underline{g'(z)}$

Neural network gradients

x_1, x_2, x_3 → o → $\hat{y}$

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

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

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

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

$dW^{[1]}$ $\quad db^{[1]}$

$dz^{[1]} = W^{[2]T} dz^{[2]}$ $\quad da^{[1]} = \dots$ $\quad dz^{[2]} = a^{[2]} - y$ $\quad da^{[2]}$

$* \, g^{[1]\prime}(z^{[1]})$

element wise product

$dW^{[2]} = dz^{[2]} a^{[1]T}$ $\quad$ "$dw = dz \cdot x$"

$db^{[2]} = dz^{[2]}$ $\quad W^{[2]} = [\text{———}]$

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

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

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

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

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

$dz^{[1]} = \boxed{W^{[2]T} dz^{[2]}} * \boxed{g^{[1]\prime}(z^{[1]})}$

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

Andrew Ng

Summary of gradient descent

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

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

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

$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$

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

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

Vectorized Implementation:

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

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

$Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \dots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$

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

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

Andrew Ng

Summary of gradient descent

$dz^{[2]} = a^{[2]} - y$	$dZ^{[2]} = A^{[2]} - Y$
$dW^{[2]} = dz^{[2]} a^{[1]^T}$	$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]^T}$
$db^{[2]} = dz^{[2]}$	$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$
$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$ $(n^{[1]}, 1)$	$dZ^{[1]} = \underbrace{W^{[2]T} dZ^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]\prime}(Z^{[1]})}_{(n^{[1]}, m)}$ $(n^{[1]}, m)$
$dW^{[1]} = dz^{[1]} x^T$	$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$
$db^{[1]} = dz^{[1]}$	$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$

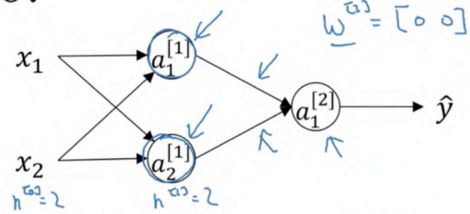
$J(\dots) = \frac{1}{m} \sum_{i=1}^{n} \mathcal{L}(\hat{y}, y)$

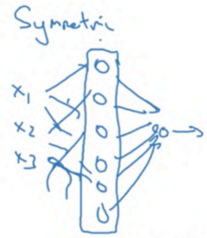
element-wise product

Andrew Ng

What happens if you initialize weights to zero?

$W^{[2]} = [0 \quad 0]$





If you initialize the weights to zero, then all of your hidden units are symmetric. And no matter how long you're upgrading the center, all continue to compute exactly the same function. So that's not helpful, because you want the different hidden units to compute different functions. The solution to this is to initialize your parameters randomly.

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

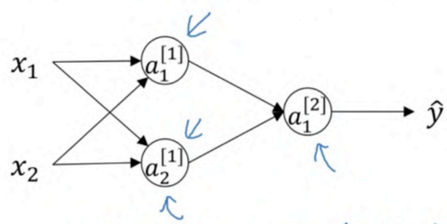
$$W^{[1]} = \begin{bmatrix} - - - - \\ - - - - \end{bmatrix}$$

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

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

Andrew Ng

Random initialization



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

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

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

$b^{[2]} = 0$

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

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

Andrew Ng