CMPE 258, Deep Learning

Backpropagation

Feb 22, 2018

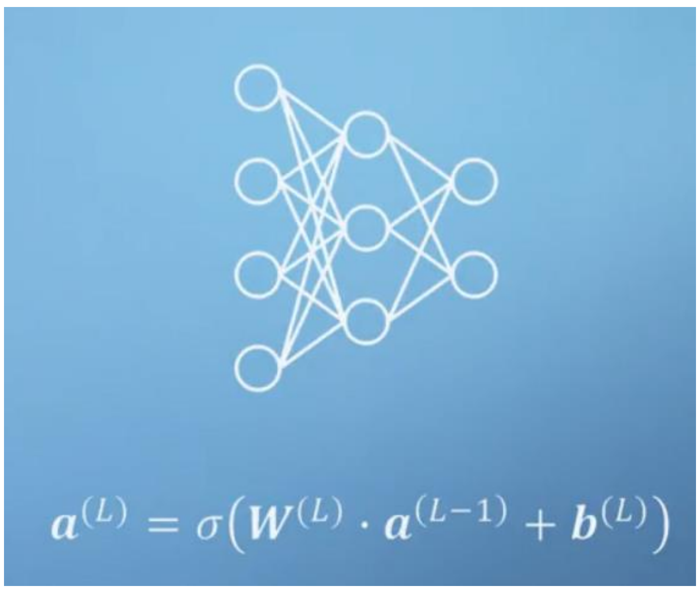
DMH 149A

Taehee Jeong

Ph.D., Data Scientist

SJSU SAN JOSÉ STATE UNIVERSITY

Neural Network



<Mathematics for Machine Learning>

- Input layer
- Hidden layer
- Output layer

Model optimizing procedure

- Initialize parameters
- Run the optimization loop
  - Forward propagation to compute the loss function
  - Backward propagation to compute the gradients with respect to the loss function
  - Using the gradients, update your parameter with the gradient descent update rule
- Return the learned parameters

<deep learning, Andrew Ng>

Optimization loop

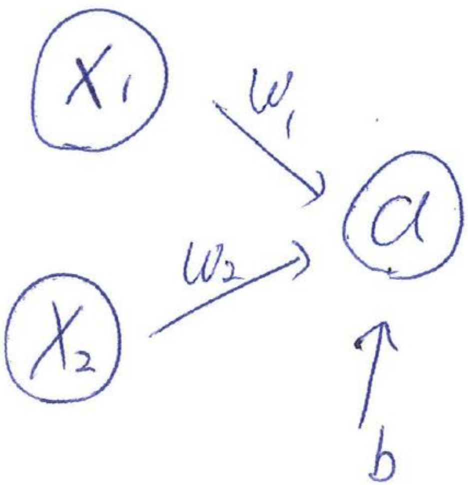
1. forward pass
2. cost computation
3. backward pass
4. parameter update

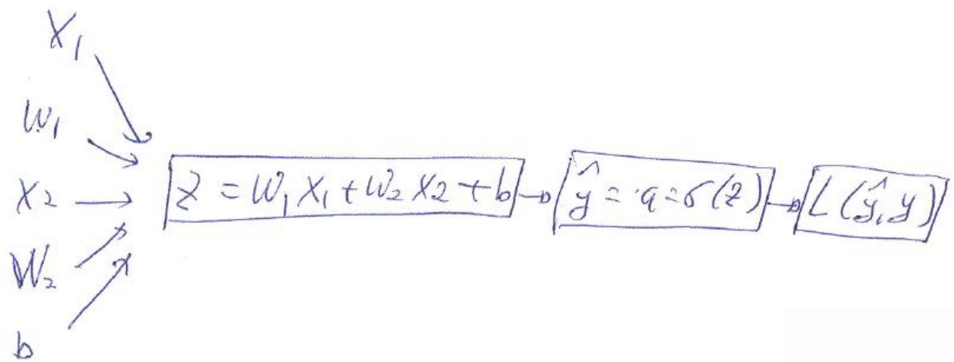
<deep learning, Andrew Ng>

Cache forward pass variables

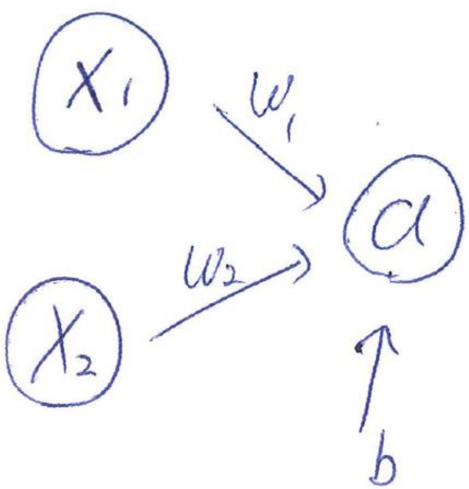
To compute the backward pass it is very helpful to have some of the variables that were used in the forward pass. In practice you want to structure your code so that you cache these variables, and so that they are available during backpropagation.

Logistic regression





Logistic regression



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

$$\hat{y} = a = \sigma(z) = \frac{1}{1+\exp(-z)}$$

$$L(\hat{y}, y) = -[y \log(a) + (1-y)\log(1-a)]$$

Derivative of da/dz

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

$$\frac{\partial a}{\partial z} = \frac{d}{dz}\left[(1+e^{-z})^{-1}\right]$$

$$= e^{-z}(1+e^{-z})^{-2}$$

$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}}$$

$$= \frac{1}{1+e^{-z}} \cdot \left[\frac{1+e^{-z}-1}{1+e^{-z}}\right]$$

$$= \frac{1}{1+e^{-z}} \cdot \left[1 - \frac{1}{1+e^{-z}}\right]$$

$$= a\,[1-a]$$

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

$$\hat{y} = a = \sigma(z) = \frac{1}{1+\exp(-z)}$$

$$L(\hat{y}, y) = -[y\log(a) + (1-y)\log(1-a)]$$

Derivative of dL/dz

$$\frac{\partial L}{\partial a} = \text{"}da\text{"} = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$\text{"}da\text{"} = \frac{\partial L(a,y)}{\partial a}$$

$$\frac{\partial L}{\partial z} = \text{"}dz\text{"} = \frac{\partial L(a,y)}{\partial z}$$

$$dz = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}$$

$$= \left[-\frac{y}{a} + \frac{1-y}{1-a}\right]\left[a(1-a)\right]$$

$$= a - y$$

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

$$\hat{y} = a = \sigma(z) = \frac{1}{1+\exp(-z)}$$

$$L(\hat{y}, y) = -\left[y\log(a) + (1-y)\log(1-a)\right]$$

Derivative of dL/dW

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

$$\hat{y} = a = \sigma(z) = \frac{1}{1+\exp(-z)}$$

$$L(\hat{y}, y) = -[y \log(a) + (1-y)\log(1-a)]$$

$$\frac{\partial L}{\partial w_1} = \text{"}dw_1\text{"}$$

$$= \underbrace{\frac{\partial L}{\partial a} \frac{\partial a}{\partial z}} \frac{\partial z}{\partial w_1}$$

$$= dz \cdot \frac{\partial z}{\partial w_1}$$

$$= (a-y) \cdot x_1$$

$$dw_2 = \frac{\partial L}{\partial w_2}$$

$$= (a-y) \cdot x_2$$

$$db = \frac{\partial L}{\partial b}$$

$$= (a-y) \cdot 1$$

$$= a - y$$

Gradient Descent on m observations

$$J(w,b) = \frac{1}{m} \sum L(a^{(i)}, y)$$

$$a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$$

$$\frac{\partial}{\partial w_1} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\frac{\partial}{\partial w_1} L(a^{(i)}, y)}_{dw_1^{(i)}}$$

Gradient Descent on m observations

Loop to calculate derivatives

For $i=1$ to m:

$z^{(i)} = w^T x^{(i)} + b$

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

$J \mathrel{+}= -[y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log(1-a^{(i)})]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 \mathrel{+}= dz^{(i)} \cdot x_1^{(i)}$

$dw_2 \mathrel{+}= dz^{(i)} \cdot x_2^{(i)}$

$db \mathrel{+}= dz^{(i)}$

After the loop

$J = J/m$

$dw_1 = dw_1/m$

$dw_2 = dw_2/m$

$db = db/m$

Gradient Descent

After the loop

$J = J/m$

$dw_1 = dw_1/m$

$dw_2 = dw_2/m$

$db = db/m$

For iteration in range(1000):

$w_1 := w_1 - \alpha dw_1$

$w_2 := w_2 - \alpha dw_2$

$b := b - \alpha db$

Gradient Descent using matrix form

For iter in range(1000):

$Z = np.dot(X, W) + b$: (m,1)

$A = \sigma(Z)$: (m,1)

$dZ = A - Y$: (m,1)

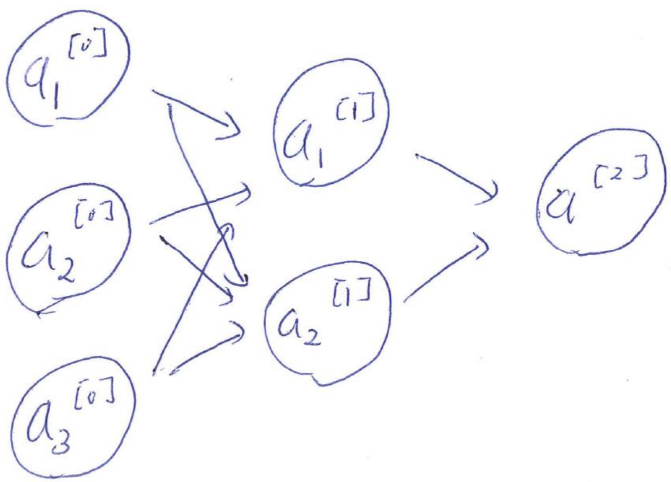
$dW = \frac{1}{m} np.dot(X.T, dZ)$: (n,1)

$db = \frac{1}{m} np.sum(dZ)$

$w := w - \alpha dw$

$b := b - \alpha db$

Neural Network: 1 hidden layer



Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$

Cost function: $J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]})$

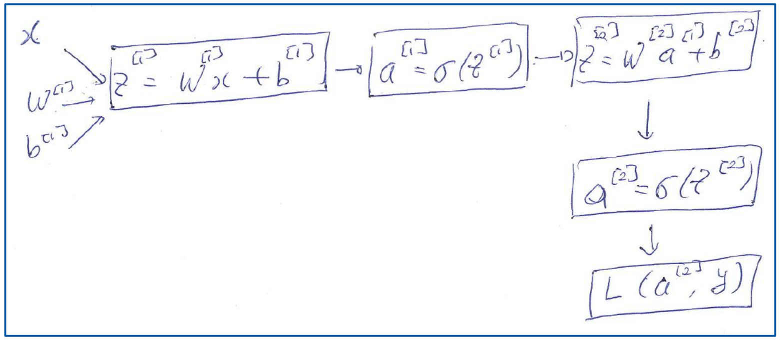
$$= \frac{1}{m} \sum_{i=1}^{m} L(\underset{\underset{a^{[2]}}{\|}}{\hat{y}}, y)$$

SJSU SAN JOSÉ STATE UNIVERSITY

Neural Network: 1 hidden layer

$$\left.\begin{matrix} x \\ W^{[1]} \\ b^{[1]} \end{matrix}\right\} \to \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \to \boxed{a^{[1]} = \sigma(z^{[1]})} \to \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}} \to \boxed{a^{[2]} = \sigma(z^{[2]})} \to \boxed{L(a^{[2]}, y)}$$

Derivative of dL/dW$^{[2]}$



$$L(\hat{y}, y) = -[y \log(a^{[2]}) + (1-y)\log(1-a^{[2]})]$$

$$da^{[2]} = -\frac{y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}}$$

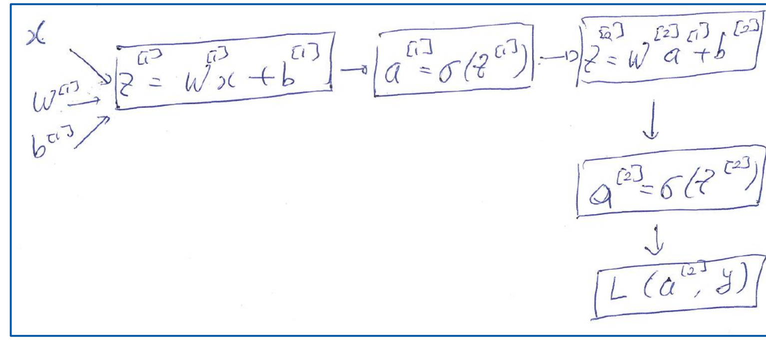
$$dz^{[2]} = a^{[2]} - y$$

$$dw^{[2]} = (a^{[2]} - y)\, a^{[1]} = dz^{[2]} \cdot a^{[1]}$$

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

SJSU SAN JOSÉ STATE UNIVERSITY

Derivative of dL/dZ$^{[1]}$



$$da^{[1]} = \frac{\partial L}{\partial a^{[1]}} = \underbrace{\frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}}} \frac{\partial z^{[2]}}{\partial a^{[1]}}$$

$$= dz^{[2]} \cdot w^{[2]}$$

$$= (a^{[2]} - y) \cdot w^{[2]}$$

$$dz^{[1]} = \frac{\partial L}{\partial z^{[1]}} = \underbrace{\frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}}} \frac{\partial a^{[1]}}{\partial z^{[1]}}$$

$$= da^{[1]} \cdot a^{[1]}(1 - a^{[1]})$$

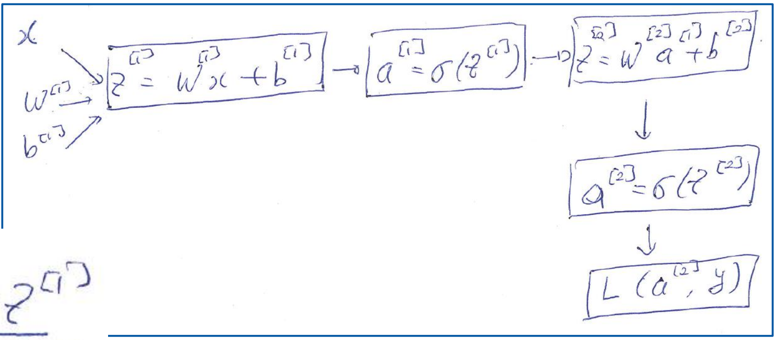
$$= (a^{[2]} - y) \cdot w^{[2]} \cdot a^{[1]} \cdot (1 - a^{[1]})$$

Derivative of dL/dW[1]

$$
\begin{array}{l}
x, \; W^{[1]}, \; b^{[1]} \rightarrow \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \rightarrow \boxed{a^{[1]} = \sigma(z^{[1]})} \rightarrow \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}} \\
\downarrow \\
\boxed{a^{[2]} = \sigma(z^{[2]})} \\
\downarrow \\
\boxed{L(a^{[2]}, y)}
\end{array}
$$

$$
\begin{aligned}
dW^{[1]} = \frac{\partial L}{\partial W^{[1]}} &= \underbrace{\frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial z^{[1]}}} \frac{\partial z^{[1]}}{\partial W^{[1]}} \\
&= dz^{[1]} \cdot \frac{\partial z^{[1]}}{\partial W^{[1]}} \\
&= (a^{[2]} - y) \cdot W^{[2]} \cdot a^{[1]} \cdot (1 - a^{[1]}) \cdot x \\
&= dz^{[1]} \cdot x \\
&= dz^{[1]} \cdot a^{[0]}
\end{aligned}
$$

Derivative of dL/db[1]



$$db^{[1]} = \frac{\partial L}{\partial b^{[1]}} = \underbrace{\frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial z^{[1]}}} \frac{\partial z^{[1]}}{\partial b^{[1]}}$$

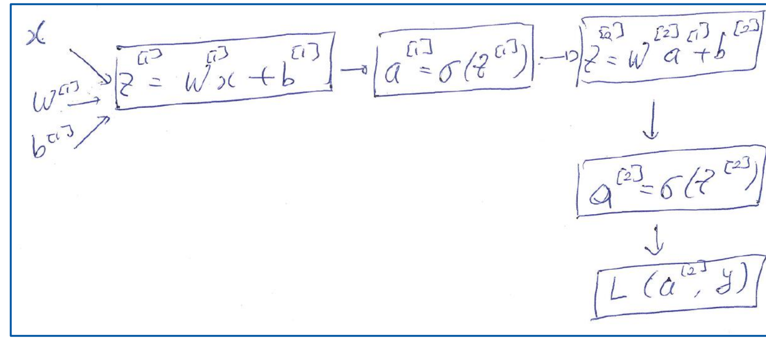
$$= dz^{[1]} \cdot \frac{\partial z^{[1]}}{\partial b^{[1]}}$$

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

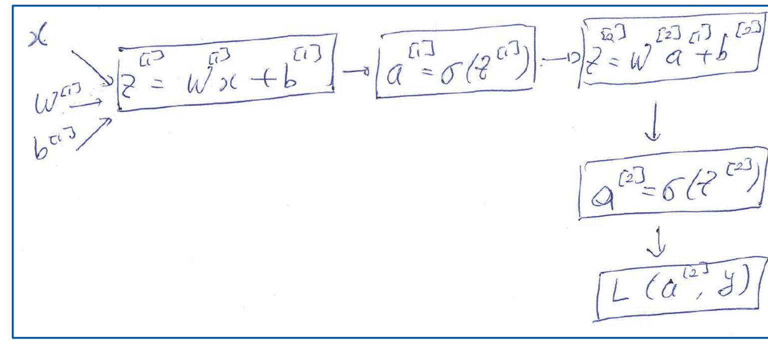
SJSU SAN JOSÉ STATE UNIVERSITY

Forward propagation

$$
\begin{aligned}
z^{[1]} &= W^{[1]} \cdot x + b^{[1]} \\
&= W^{[1]} \cdot A^{[0]} + b^{[1]} \\
A^{[1]} &= g^{[1]}(z^{[1]}) \\
&= \sigma(z^{[1]}) \\
z^{[2]} &= W^{[2]} \cdot A^{[1]} + b^{[2]} \\
A^{[2]} &= g^{[2]}(z^{[2]}) \\
&= \sigma(z^{[2]}) \\
&= \hat{Y}
\end{aligned}
$$



Backward propagation



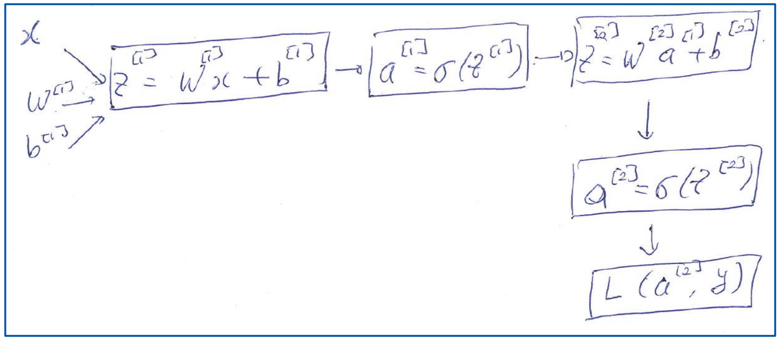
$$dA^{[2]} = -\frac{Y}{A^{[2]}} + \frac{1-Y}{1-A^{[2]}} \rightarrow \text{skip}$$

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

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

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

Backward propagation



$$dA^{[1]} = dZ^{[2]} \cdot W^{[2]}$$

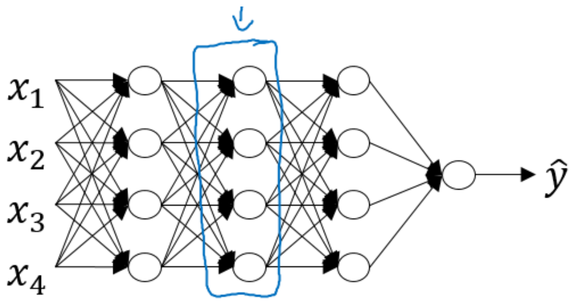
$$dZ^{[1]} = dA^{[1]} \cdot A^{[1]}(1 - A^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} \cdot A^{[0]}$$

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

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

Forward and backward functions



layer l: $W^{[l]}, b^{[l]}$

→ Forward: Input $a^{[l-1]}$, output $a^{[l]}$

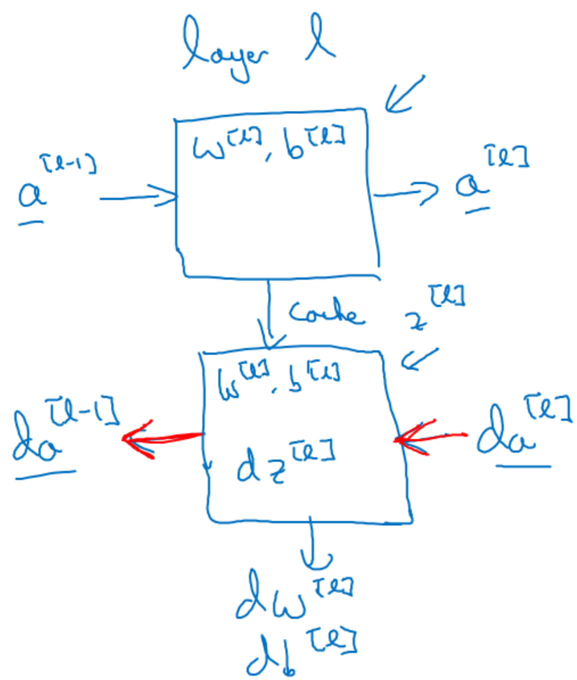
$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$ cache $z^{[l]}$

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

→ Backward: Input $da^{[l]}$ cache ($z^{[l]}$) output $da^{[l-1]}$

$dW^{[l]}$

$db^{[l]}$



<deep learning, Andrew Ng>

Forward and backward propagation

$$
\begin{aligned}
Z^{[1]} &= W^{[1]}X + b^{[1]} \\
A^{[1]} &= g^{[1]}(Z^{[1]}) \\
Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\
A^{[2]} &= g^{[2]}(Z^{[2]}) \\
&\vdots \\
A^{[L]} &= g^{[L]}\left(Z^{[L]}\right) = \hat{Y}
\end{aligned}
$$

$$
\begin{aligned}
dZ^{[L]} &= A^{[L]} - Y \\
dW^{[L]} &= \frac{1}{m} dZ^{[L]} A^{[L]^T} \\
db^{[L]} &= \frac{1}{m} np.\mathrm{sum}(\mathrm{d}Z^{[L]}, axis = 1, keepdims = True) \\
dZ^{[L-1]} &= dW^{[L]^T} dZ^{[L]} g'^{[L]}(Z^{[L-1]}) \\
&\vdots \\
dZ^{[1]} &= dW^{[L]^T} dZ^{[2]} g'^{[1]}(Z^{[1]}) \\
dW^{[1]} &= \frac{1}{m} dZ^{[1]} A^{[1]^T} \\
db^{[1]} &= \frac{1}{m} np.\mathrm{sum}(\mathrm{d}Z^{[1]}, axis = 1, keepdims = True)
\end{aligned}
$$

<deep learning, Andrew Ng>

Summary

- Derivatives for logistic regression
- Derivatives for Neural Network
- Forward and Backward propagation