Backpropagation

國立中山大學
資訊工程系
張雲南

Adapted from NTU Prof.李宏毅's slides

Training process

- Suppose we want to find the parameter w to minimize $L(w)$
  - For example: $\mathrm{L(w)} = \frac{1}{N}\sum_{i=1}^{N}\{h(x_i; w, m) - y_i\}^2$
  - The most direct naïve approach is to find many w candidates.
  - What if we have multiple parameters to solve?

$$L(w_1, w_2, w_3, \cdots, w_{100})$$

$$w \rightarrow \boxed{L(w)} \rightarrow L$$

    - the number of candidates will be very huge. Ex: 10^{100}
  - the actual weights to train could be up to million.

- *What terms will influence the loss function?*

Gradient Descent Optimization Method

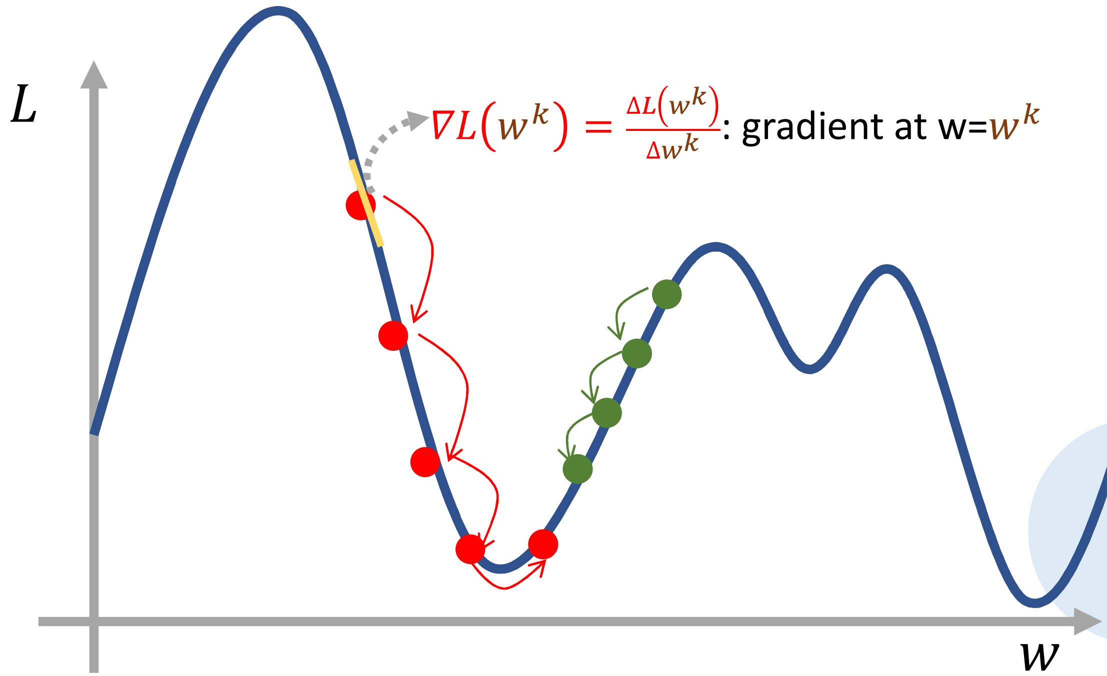
for k in range (max_iteration):

$w^{k+1} = w^k - \eta \nabla L(w^k)$ # update weight

if $\|w^{k+1} - w^k\| < \epsilon$ # stopping criteria

break

η: *adjustment (learning) rate*



- If the increase of W can reduce the loss, W will be increased.

Gradient Descent

Training data: (x_1, y_1)

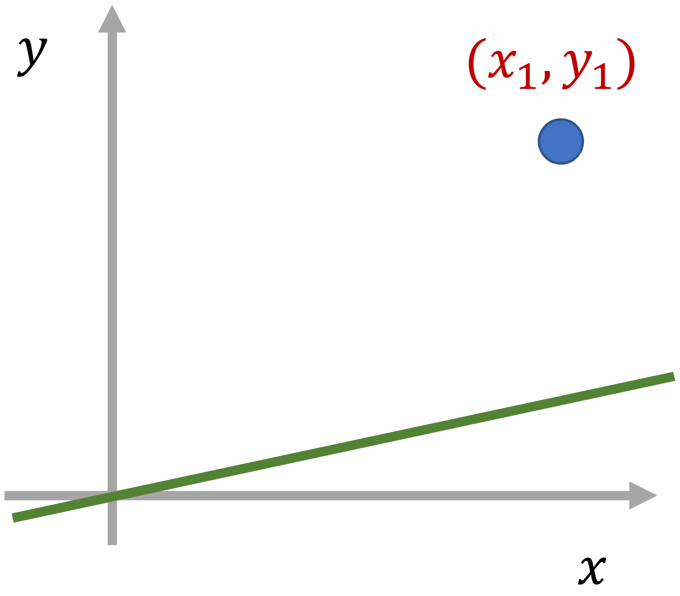
$h(x; w, m) = x \times w$

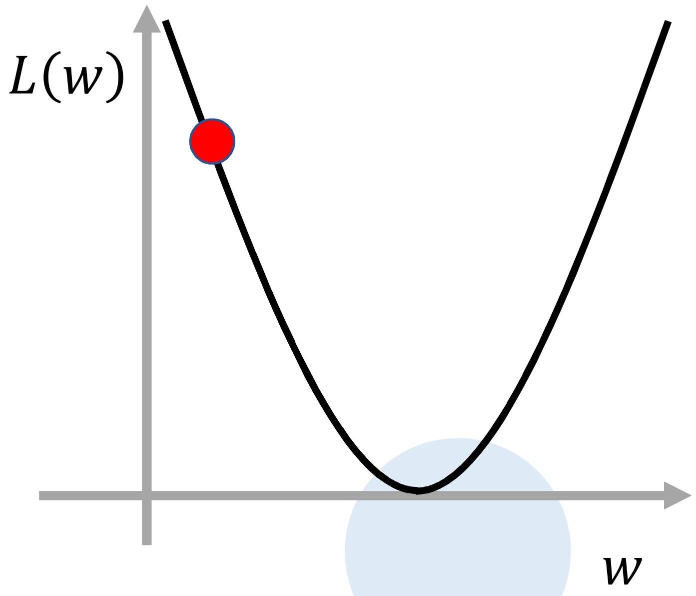
$$\boxed{L(w) = (x_1 \times w - y_1)^2}$$

Model

Epoch = 0

Loss





Gradient Descent

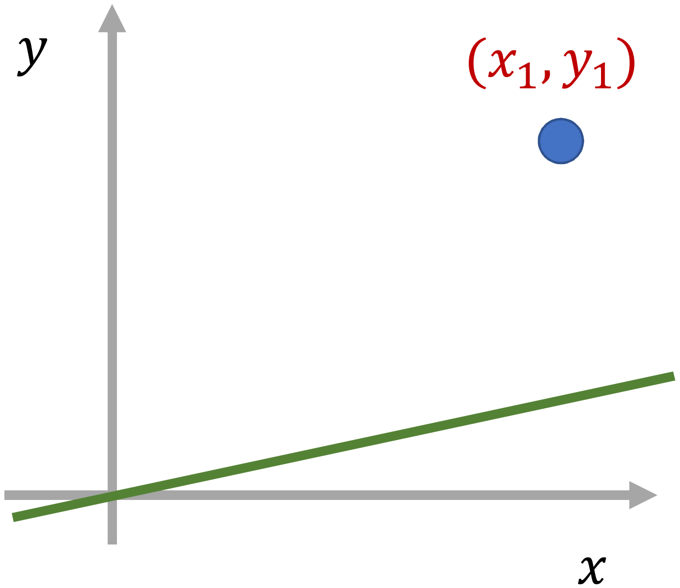
$$\mathrm{L(w)} = (x_1 \times w - y_1)^2$$

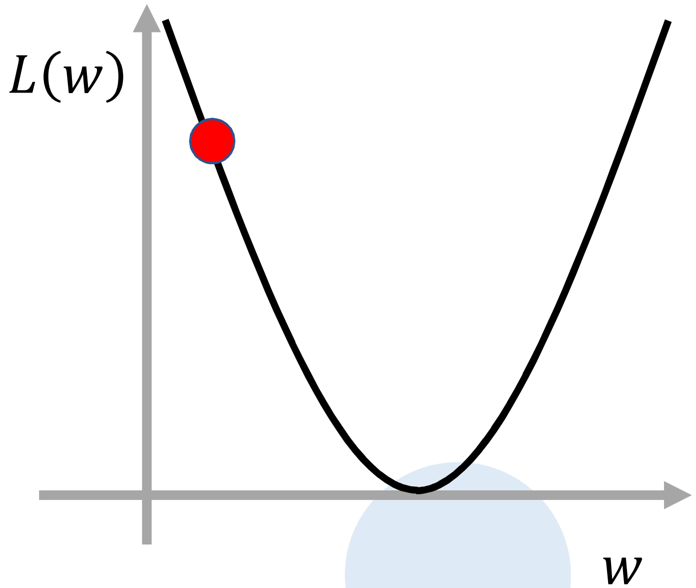
Find the best w to minimize L(w)

Model

Epoch = 0

Loss





$$y = h(x; w, m) = x \times w$$

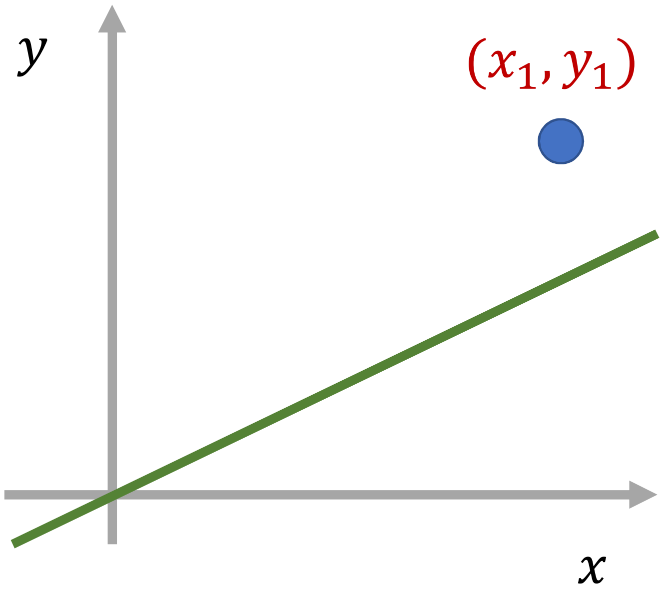
Only one training data: (x_1, y_1)

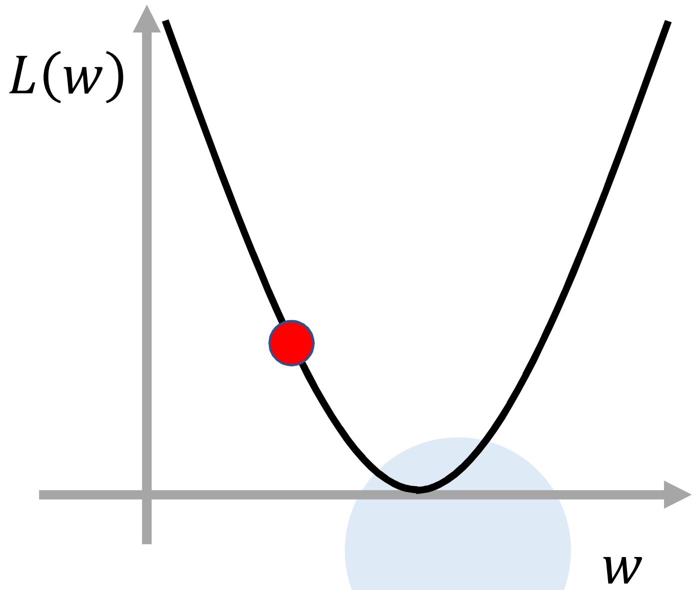
Gradient Descent

Model

Epoch = 10

Loss



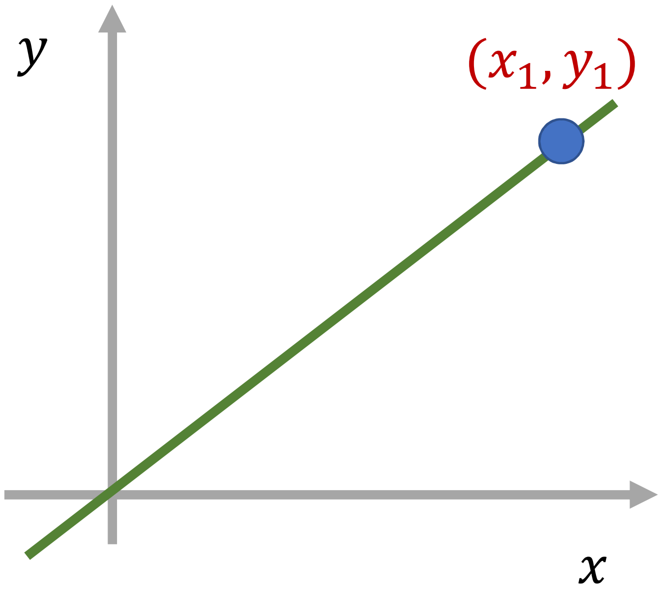


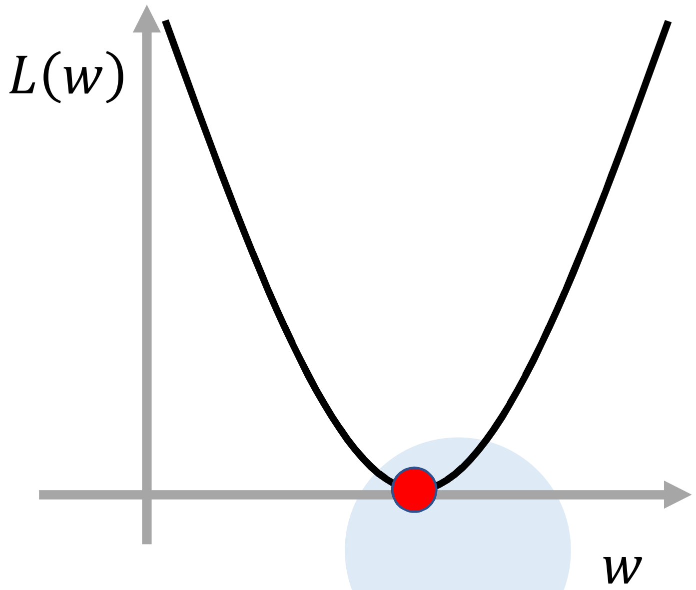
Gradient Descent

Model

Epoch = 100

Loss





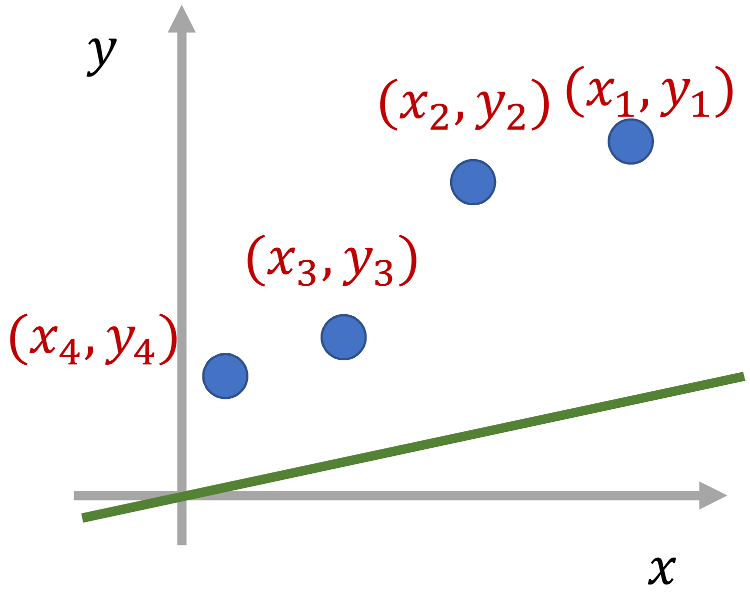
Gradient Descent

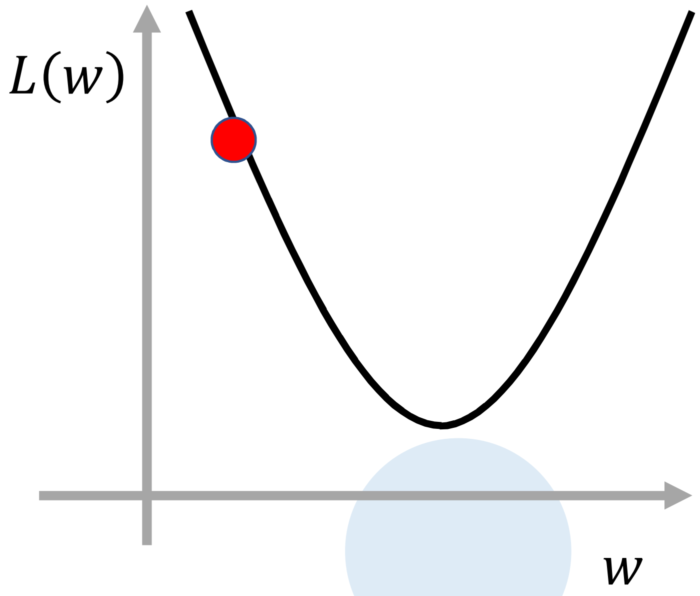
$$L(w) = \sum_{i=1}^{4}(x_i \times w - y_i)^2$$

Model

Epoch = 0

Loss





Gradient Descent

Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

Starting Parameters $\theta^0 \longrightarrow \theta^1 \longrightarrow \theta^2 \longrightarrow$ ……

$$\nabla L(\theta) = \begin{bmatrix} \partial L(\theta)/\partial w_1 \\ \partial L(\theta)/\partial w_2 \\ \vdots \\ \partial L(\theta)/\partial b_1 \\ \partial L(\theta)/\partial b_2 \\ \vdots \end{bmatrix}$$

$Compute\ \nabla L(\theta^0)$ $\qquad \theta^1 = \theta^0 - \eta \nabla L(\theta^0)$

$Compute\ \nabla L(\theta^1)$ $\qquad \theta^2 = \theta^1 - \eta \nabla L(\theta^1)$

Millions of parameters ……

To compute the gradients efficiently, we use ***backpropagation***.

Chain Rule

Case 1

$$y = g(x) \qquad z = h(y)$$

$$\Delta x \rightarrow \Delta y \rightarrow \Delta z \qquad\qquad \frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

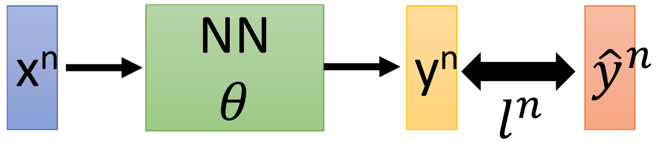
Case 2

$$x = g(s) \qquad y = h(s) \qquad z = k(x, y)$$

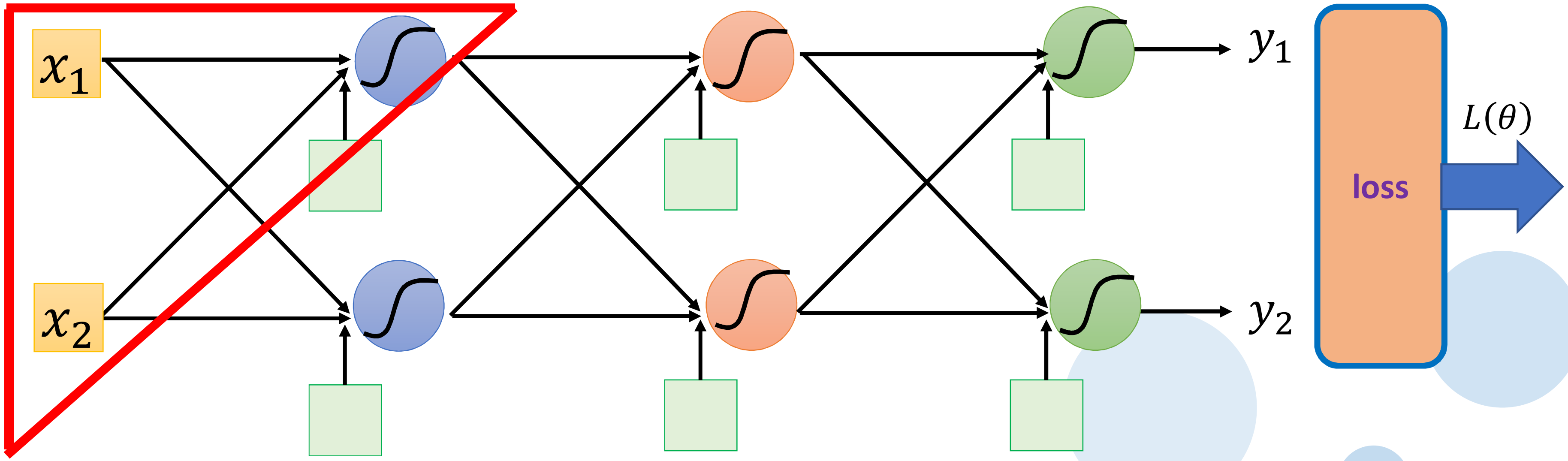
$$\begin{tikzcd}
 & \Delta x \arrow[dr] & \\
\Delta s \arrow[ur] \arrow[dr] & & \Delta z \\
 & \Delta y \arrow[ur] &
\end{tikzcd}$$

$$\frac{dz}{ds} = \frac{\partial z}{\partial x}\frac{dx}{ds} + \frac{\partial z}{\partial y}\frac{dy}{ds}$$

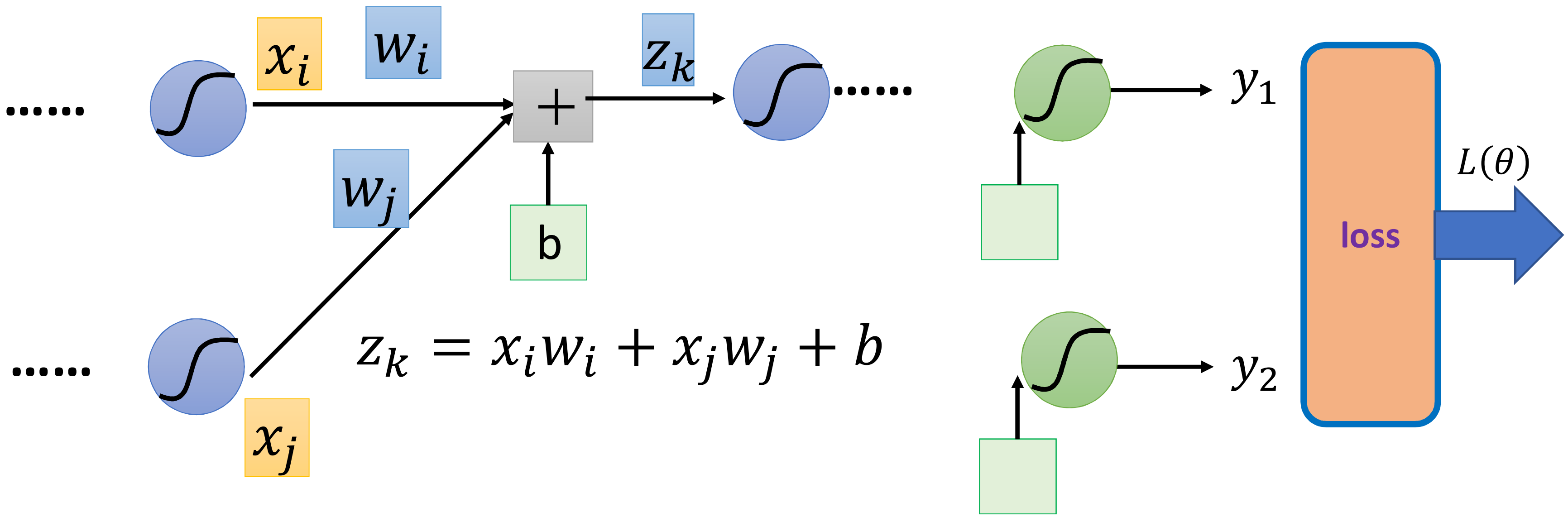
Backpropagation



$$L(\theta) = \sum_{n=1}^{N} l^n(\theta) \quad \Rightarrow \quad \frac{\partial L(\theta)}{\partial w} = \sum_{n=1}^{N} \frac{\partial l^n(\theta)}{\partial w}$$



Backpropagation



$$\frac{\partial l}{\partial w_i} =? \quad \frac{\partial l}{\partial z_k}\frac{\partial z_k}{\partial w_i}$$

(Chain rule)

$\partial z_k / \partial w_i = x_i$

$\partial l / \partial z_k =$

Forward pass

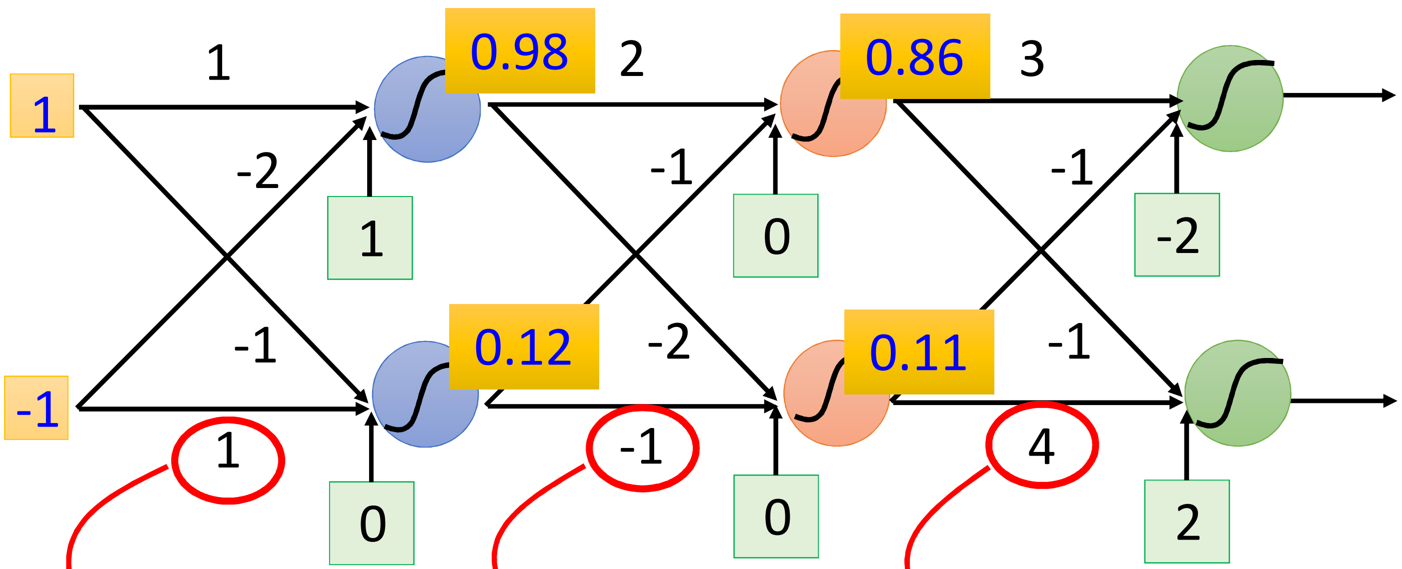
Compute $\partial z / \partial w$ for all parameters

Backward pass

Compute $\partial l / \partial z$ for all activation function inputs z

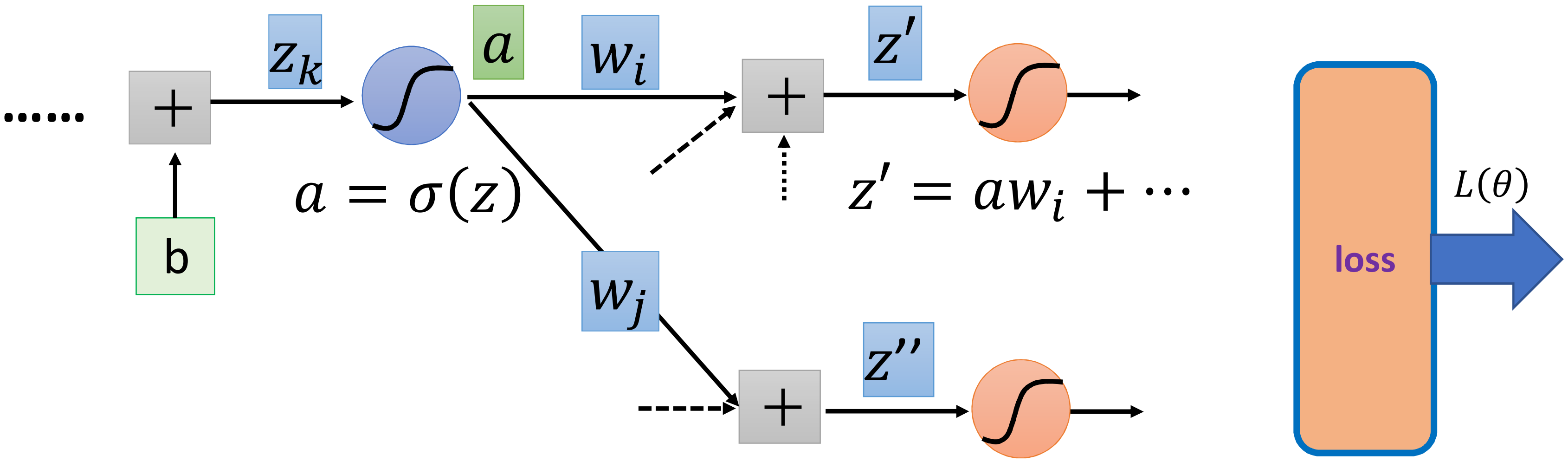
Backpropagation – Forward pass

Compute $\partial z/\partial w$ for all parameters



$$\frac{\partial z}{\partial w} = -1 \qquad \frac{\partial z}{\partial w} = 0.12 \qquad \frac{\partial z}{\partial w} = 0.11$$

Backpropagation – Backward pass



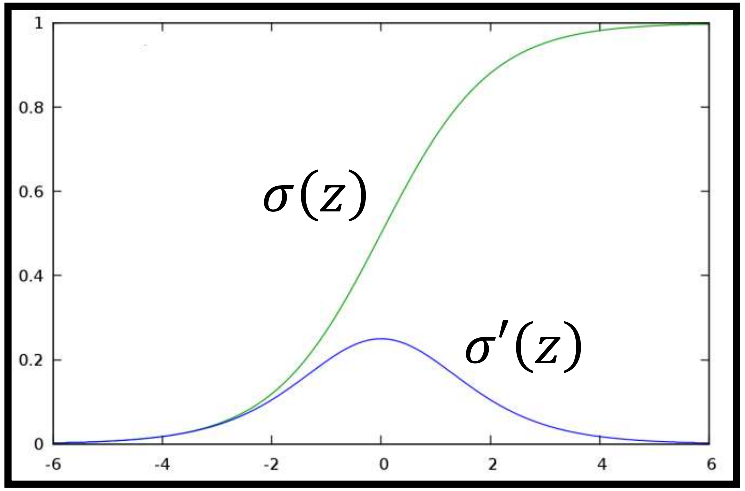
$$\frac{\partial l}{\partial z_k} = \underbrace{\frac{\partial a}{\partial z_k}}_{\sigma'(z_k)} \frac{\partial l}{\partial a}$$

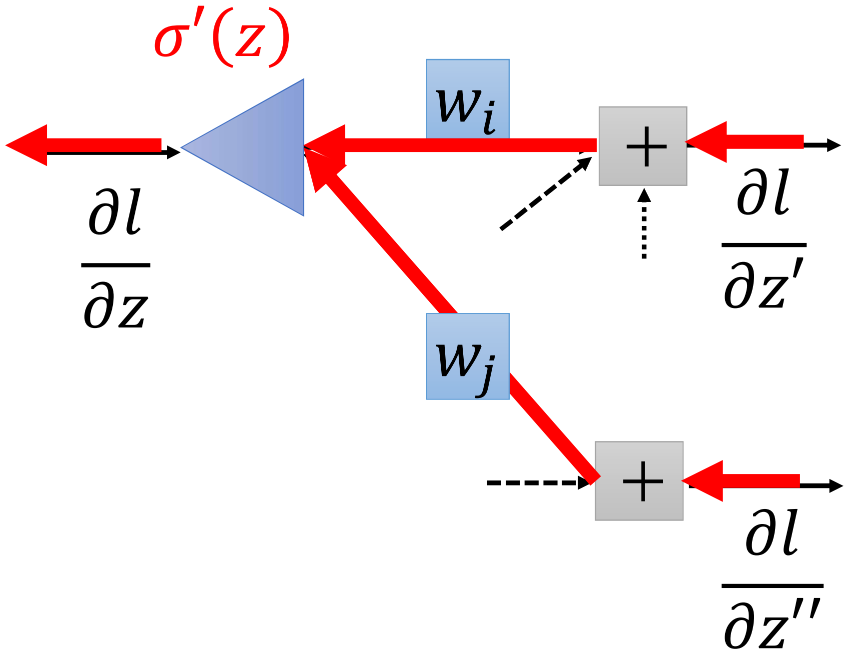
$$\frac{\partial l}{\partial a} = \underbrace{\frac{\partial z'}{\partial a}}_{w_i} \underbrace{\frac{\partial l}{\partial z'}}_{?} + \underbrace{\frac{\partial z''}{\partial a}}_{w_j} \underbrace{\frac{\partial l}{\partial z''}}_{?} = w_i \frac{\partial l}{\partial z'} + w_j \frac{\partial l}{\partial z''}$$

Backpropagation – Backward pass

$\sigma'(z)$ is a constant because z is already determined in the forward pass.

$$\frac{\partial l}{\partial z} = \sigma'(z)\left[w_i \frac{\partial l}{\partial z'} + w_j \frac{\partial l}{\partial z''}\right]$$

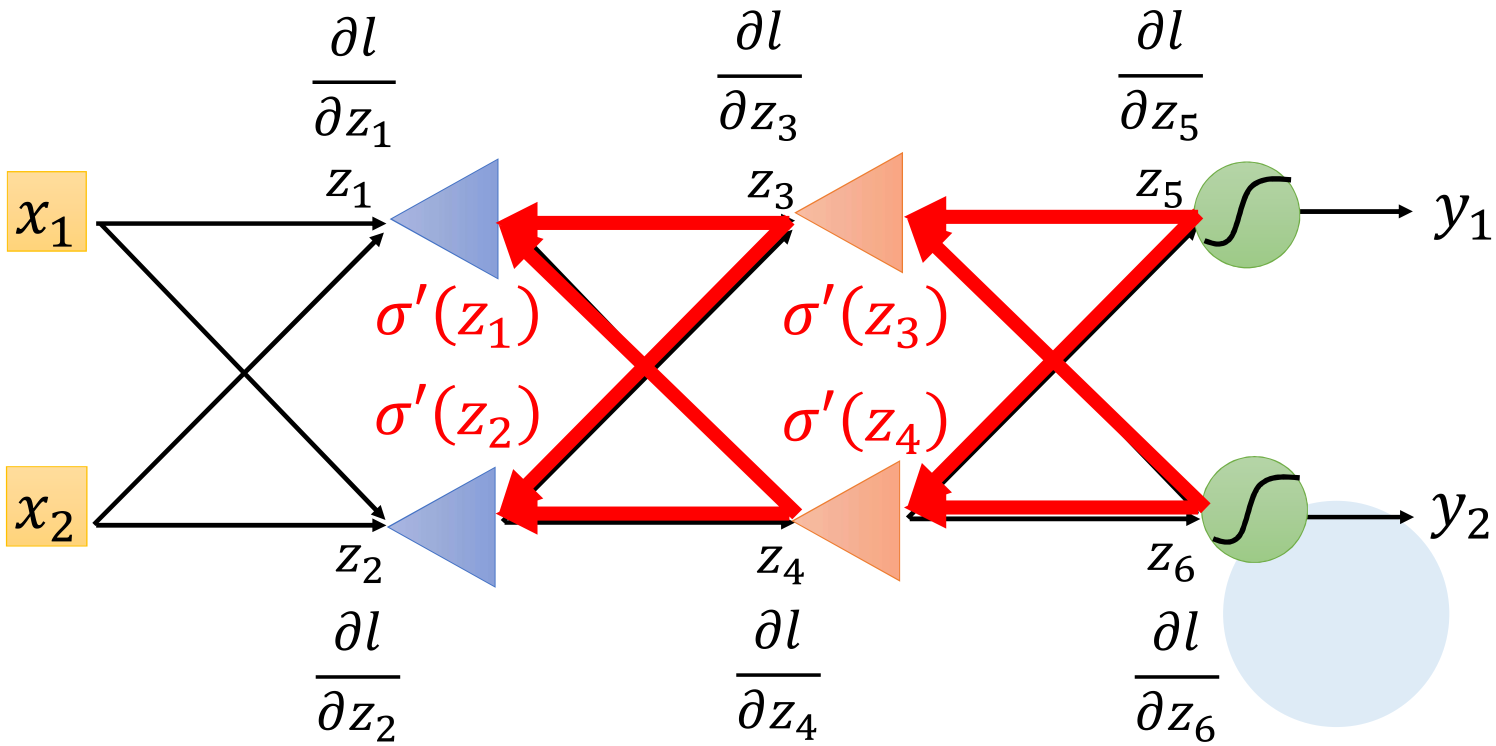




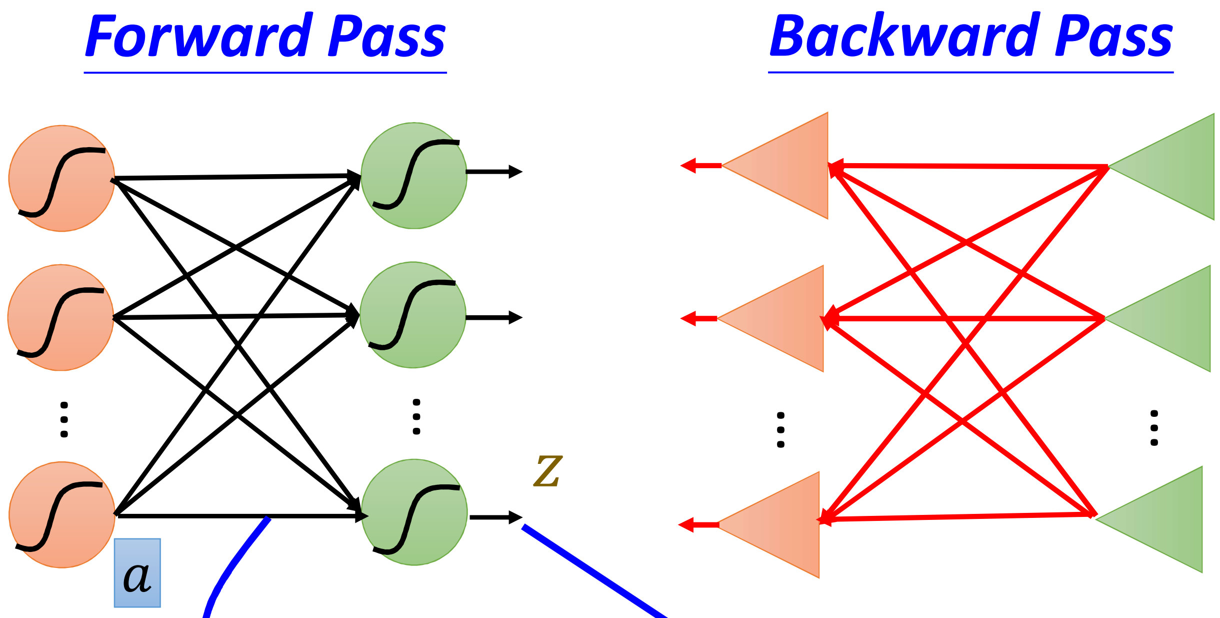
Backpropagation – Backward Pass

Compute $\partial l/\partial z$ for all activation function inputs z

Compute $\partial l/\partial z$ from the output layer



Backpropagation – Summary



$$\frac{\partial z}{\partial w} = a \times \frac{\partial l}{\partial z} = \frac{\partial l}{\partial w}$$