

# 10-301/601: Introduction to Machine Learning

## Lecture 10 – Backpropagation

Henry Chai

2/19/24

# Front Matter

- Announcements
  - HW2 released 2/7, due **2/19** (today!) at 11:59 PM
  - HW3 released **2/19** (today!), due **2/28** (previously 2/26) at 11:59 PM
  - Lecture on 2/21 (Wednesday) and Recitation on 2/23 (Friday) have been swapped
- Recommended Readings
  - Mitchell, [Chapters 4.1 – 4.6](#)
  - Zhang, Lipton, Li & Smola, [Chapters 5.1 – 5.3](#)

# Recall: Stochastic Gradient Descent for Learning

- Input:  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N, \eta^{(0)}$
- Initialize all weights  $W_{(0)}^{(1)}, \dots, W_{(0)}^{(L)}$  to small, random numbers and set  $t = 0$
- While TERMINATION CRITERION is not satisfied
  - For  $i \in \text{shuffle}(\{1, \dots, N\})$ 
    - For  $l = 1, \dots, L$ 
      - Compute  $G^{(l)} = \nabla_{W^{(l)}} \ell^{(i)} \left( W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)} \right)$
      - Update  $W^{(l)}$ :  $W_{(t+1)}^{(l)} = W_{(t)}^{(l)} - \eta_0 G^{(l)}$
    - Increment  $t$ :  $t = t + 1$
- Output:  $W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)}$

# Matrix Calculus

		Numerator		
		scalar	vector	matrix
Denominator	Types of Derivatives			
	scalar	$\frac{\partial y}{\partial x}$	$\frac{\partial \mathbf{y}}{\partial x}$	$\frac{\partial \mathbf{Y}}{\partial x}$
	vector	$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}$
matrix	$\frac{\partial y}{\partial \mathbf{X}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$	

# Matrix Calculus: Denominator Layout

- Derivatives of a scalar always have the *same shape* as the entity that the derivative is being taken with respect to.

<i>Types of Derivatives</i>	<b>scalar</b>
<b>scalar</b>	$\frac{\partial y}{\partial x} = \left[ \frac{\partial y}{\partial x} \right]$
<b>vector</b>	$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_P} \end{bmatrix}$
<b>matrix</b>	$\frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial X_{11}} & \frac{\partial y}{\partial X_{12}} & \cdots & \frac{\partial y}{\partial X_{1Q}} \\ \frac{\partial y}{\partial X_{21}} & \frac{\partial y}{\partial X_{22}} & \cdots & \frac{\partial y}{\partial X_{2Q}} \\ \vdots & & & \vdots \\ \frac{\partial y}{\partial X_{P1}} & \frac{\partial y}{\partial X_{P2}} & \cdots & \frac{\partial y}{\partial X_{PQ}} \end{bmatrix}$

# Matrix Calculus: Denominator Layout

<i>Types of Derivatives</i>	scalar	vector
scalar	$\frac{\partial y}{\partial x} = \left[ \frac{\partial y}{\partial x} \right]$	$\frac{\partial \mathbf{y}}{\partial x} = \left[ \frac{\partial y_1}{\partial x} \quad \frac{\partial y_2}{\partial x} \quad \dots \quad \frac{\partial y_N}{\partial x} \right]$
vector	$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_P} \end{bmatrix}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \dots & \frac{\partial y_N}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \dots & \frac{\partial y_N}{\partial x_2} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial y_1}{\partial x_P} & \frac{\partial y_2}{\partial x_P} & \dots & \frac{\partial y_N}{\partial x_P} \end{bmatrix}$

# Computing Gradients

$$\nabla_{W^{(l)}} \ell^{(i)} \left( W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)} \right) = \begin{bmatrix} \frac{\partial \ell^{(i)}}{\partial w_{1,0}^{(l)}} & \frac{\partial \ell^{(i)}}{\partial w_{1,1}^{(l)}} & \dots & \frac{\partial \ell^{(i)}}{\partial w_{1,d^{(l-1)}}^{(l)}} \\ \frac{\partial \ell^{(i)}}{\partial w_{2,0}^{(l)}} & \frac{\partial \ell^{(i)}}{\partial w_{2,1}^{(l)}} & \dots & \frac{\partial \ell^{(i)}}{\partial w_{2,d^{(l-1)}}^{(l)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \ell^{(i)}}{\partial w_{d^{(l)},0}^{(l)}} & \frac{\partial \ell^{(i)}}{\partial w_{d^{(l)},1}^{(l)}} & \dots & \frac{\partial \ell^{(i)}}{\partial w_{d^{(l)},d^{(l-1)}}^{(l)}} \end{bmatrix}$$

# Computing Gradients: Intuition

- A weight affects the prediction of the network (and therefore the error) through downstream signals/outputs
  - Use the chain rule!
- Any weight going into the same node will affect the prediction through the same downstream path
  - Compute derivatives starting from the last layer and move “backwards”
  - Derive a recursive definition for the relevant partial derivatives
  - Automatic differentiation: store intermediate values and reuse for efficiency (dynamic programming)

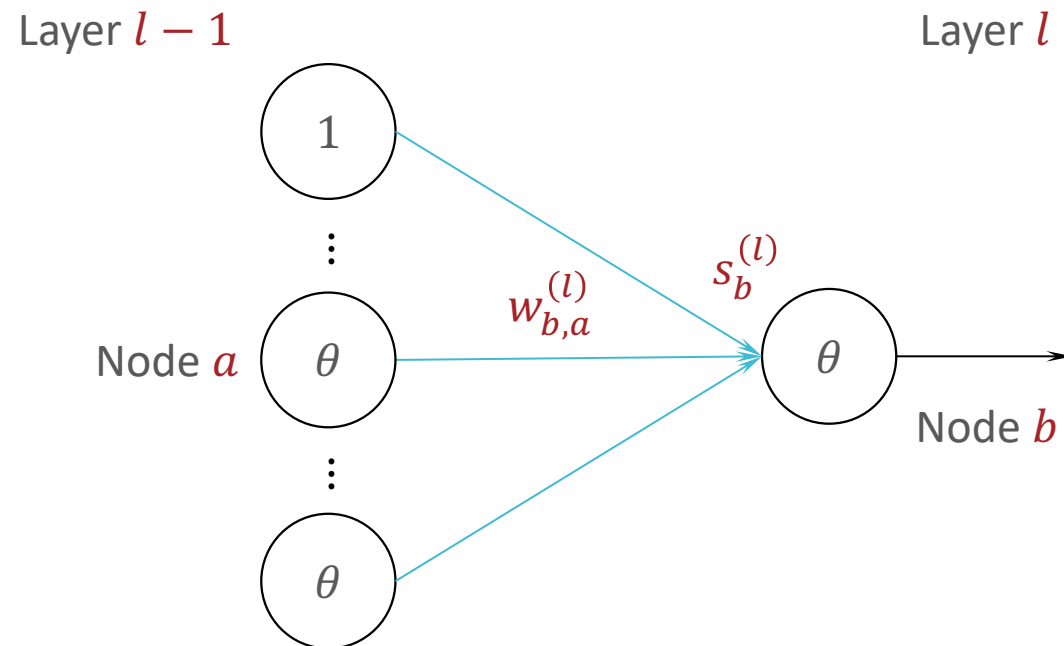


# Computing Partial Derivatives

Computing  $\nabla_{W^{(l)}} \ell^{(i)} \left( W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)} \right)$  reduces to computing

$$\frac{\partial \ell^{(i)}}{\partial w_{b,a}^{(l)}}$$

Insight:  $w_{b,a}^{(l)}$  only affects  $\ell^{(i)}$  via  $s_b^{(l)}$



# Computing Partial Derivatives

Computing  $\nabla_{W^{(l)}} \ell^{(i)} \left( W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)} \right)$  reduces to computing

$$\frac{\partial \ell^{(i)}}{\partial w_{b,a}^{(l)}}$$

Insight:  $w_{b,a}^{(l)}$  *only* affects  $\ell^{(i)}$  via  $s_b^{(l)}$

$$\text{Chain rule: } \frac{\partial \ell^{(i)}}{\partial w_{b,a}^{(l)}} = \frac{\partial \ell^{(i)}}{\partial s_b^{(l)}} \left( \frac{\partial s_b^{(l)}}{\partial w_{b,a}^{(l)}} \right)$$

$$s_b^{(l)} = \sum_{a=0}^{d^{(l-1)}} w_{b,a}^{(l)} o_a^{(l-1)} \rightarrow \frac{\partial s_b^{(l)}}{\partial w_{b,a}^{(l)}} = o_a^{(l-1)}$$

Compute outputs  $o^{(l)} \forall l \in \{0, \dots, L\}$  by forward propagation

# Computing Partial Derivatives

Computing  $\nabla_{W^{(l)}} \ell^{(i)} \left( W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)} \right)$  reduces to computing

$$\frac{\partial \ell^{(i)}}{\partial w_{b,a}^{(l)}}$$

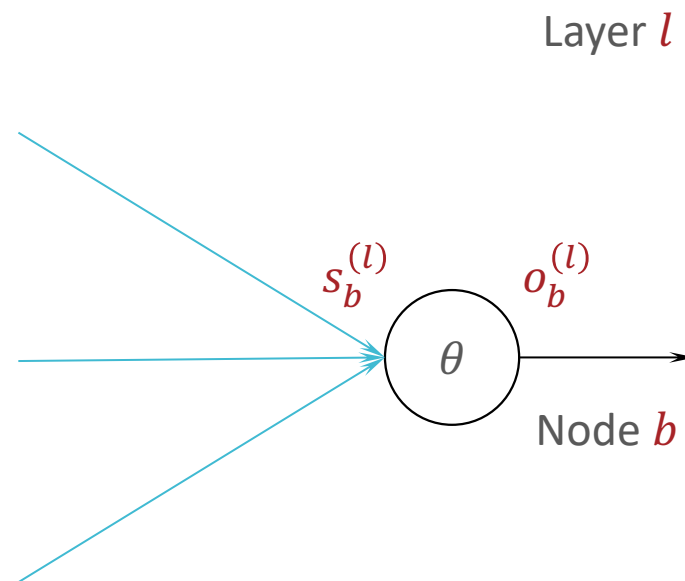
Insight:  $w_{b,a}^{(l)}$  *only* affects  $\ell^{(i)}$  via  $s_b^{(l)}$

Chain rule: 
$$\frac{\partial \ell^{(i)}}{\partial w_{b,a}^{(l)}} = \frac{\partial \ell^{(i)}}{\partial s_b^{(l)}} \left( \frac{\partial s_b^{(l)}}{\partial w_{b,a}^{(l)}} \right)$$

$$\delta_b^{(l)} := \frac{\partial \ell^{(i)}}{\partial s_b^{(l)}}$$

# Computing Partial Derivatives

Insight:  $s_b^{(l)}$  only affects  $\ell^{(i)}$  via  $o_b^{(l)}$



# Computing Partial Derivatives

Insight:  $s_b^{(l)}$  only affects  $\ell^{(i)}$  via  $o_b^{(l)}$

$$\text{Chain rule: } \delta_b^{(l)} = \frac{\partial \ell^{(i)}}{\partial o_b^{(l)}} \left( \frac{\partial o_b^{(l)}}{\partial s_b^{(l)}} \right)$$

$$\begin{aligned} o_b^{(l)} = \theta \left( s_b^{(l)} \right) &\rightarrow \frac{\partial o_b^{(l)}}{\partial s_b^{(l)}} = \frac{\partial \theta \left( s_b^{(l)} \right)}{\partial s_b^{(l)}} \\ &= 1 - \left( \tanh \left( s_b^{(l)} \right) \right)^2 \end{aligned}$$

when  $\theta(\cdot) = \tanh(\cdot)$

## Aside: Vanishing Gradients


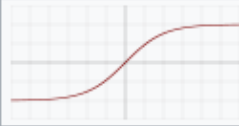
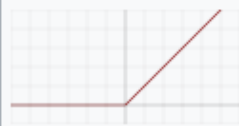
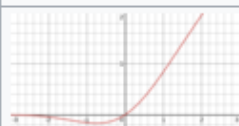
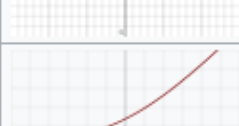



Insight:  $s_b^{(l)}$  only affects  $\ell^{(i)}$  via  $o_b^{(l)}$

$$\text{Chain rule: } \delta_b^{(l)} = \frac{\partial \ell^{(i)}}{\partial o_b^{(l)}} \left( \frac{\partial o_b^{(l)}}{\partial s_b^{(l)}} \right)$$

$$\begin{aligned} o_b^{(l)} = \theta \left( s_b^{(l)} \right) &\rightarrow \frac{\partial o_b^{(l)}}{\partial s_b^{(l)}} = \frac{\partial \theta \left( s_b^{(l)} \right)}{\partial s_b^{(l)}} \\ &= 1 - \left( \tanh \left( s_b^{(l)} \right) \right)^2 \leq 1 \end{aligned}$$

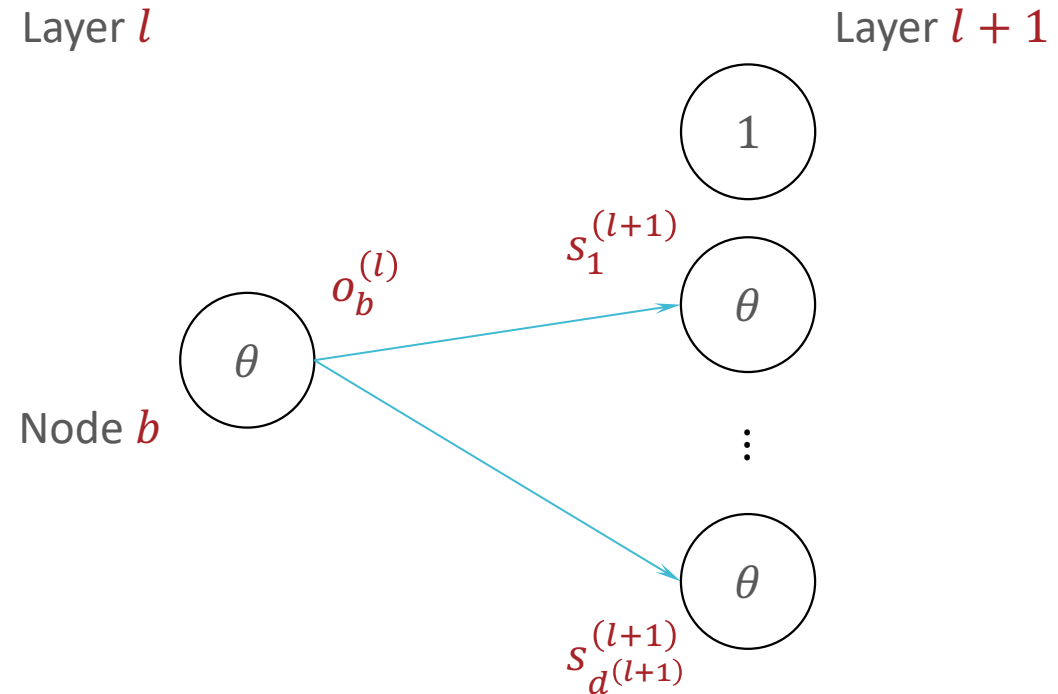
when  $\theta(\cdot) = \tanh(\cdot)$

# Recall: Other Activation Functions

Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1 + e^{-x}}$
Hyperbolic tangent (tanh)		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Rectified linear unit (ReLU) <sup>[7]</sup>		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x \mathbf{1}_{x>0}$
Gaussian Error Linear Unit (GELU) <sup>[4]</sup>		$\frac{1}{2}x \left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$ $= x\Phi(x)$
Softplus <sup>[8]</sup>		$\ln(1 + e^x)$
Exponential linear unit (ELU) <sup>[9]</sup>		$\begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ with parameter $\alpha$
Leaky rectified linear unit (Leaky ReLU) <sup>[11]</sup>		$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$
Parametric rectified linear unit (PReLU) <sup>[12]</sup>		$\begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameter $\alpha$

# Computing Partial Derivatives

Insight:  $o_b^{(l)}$  affects  $\ell^{(i)}$  via  $s_1^{(l+1)}, \dots, s_d^{(l+1)}$





# Computing Partial Derivatives

Insight:  $o_b^{(l)}$  affects  $\ell^{(i)}$  via  $s_1^{(l+1)}, \dots, s_{d^{(l+1)}}^{(l+1)}$

$$\text{Chain rule: } \frac{\partial \ell^{(i)}}{\partial o_b^{(l)}} = \sum_{c=1}^{d^{(l+1)}} \frac{\partial \ell^{(i)}}{\partial s_c^{(l+1)}} \left( \frac{\partial s_c^{(l+1)}}{\partial o_b^{(l)}} \right)$$

$$s_c^{(l+1)} = \sum_{b=0}^{d^{(l)}} w_{c,b}^{(l+1)} o_b^{(l)} \rightarrow \frac{\partial s_c^{(l+1)}}{\partial o_b^{(l)}} = w_{c,b}^{(l+1)}$$

$$= \sum_{c=1}^{d^{(l+1)}} \delta_c^{(l+1)} \left( w_{c,b}^{(l+1)} \right)$$

# Computing Partial Derivatives

$$\begin{aligned}\delta_b^{(l)} &= \frac{\partial \ell^{(i)}}{\partial o_b^{(l)}} \left( \frac{\partial o_b^{(l)}}{\partial s_b^{(l)}} \right) \\ &= \left( \sum_{c=1}^{d^{(l+1)}} \delta_c^{(l+1)} \left( w_{c,b}^{(l+1)} \right) \right) \left( 1 - \left( o_b^{(l)} \right)^2 \right) \\ \boldsymbol{\delta}^{(l)} &:= \nabla_{\mathbf{s}^{(l)}} \ell^{(i)} \left( W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)} \right)\end{aligned}$$

# Computing Partial Derivatives

$$\begin{aligned}\delta_b^{(l)} &= \frac{\partial \ell^{(i)}}{\partial o_b^{(l)}} \left( \frac{\partial o_b^{(l)}}{\partial s_b^{(l)}} \right) \\ &= \left( \sum_{c=1}^{d^{(l+1)}} \delta_c^{(l+1)} \left( w_{c,b}^{(l+1)} \right) \right) \left( 1 - \left( o_b^{(l)} \right)^2 \right)\end{aligned}$$

$$\boldsymbol{\delta}^{(l)} = W^{(l+1)T} \boldsymbol{\delta}^{(l+1)} \odot \left( 1 - \mathbf{o}^{(l)} \odot \mathbf{o}^{(l)} \right)$$

where  $\odot$  is the element-wise product operation

Sanity check:

$$\boldsymbol{\delta}^{(l+1)} \in \mathbb{R}^{d^{(l+1)} \times 1} \text{ so}$$

$$W^{(l+1)T} \boldsymbol{\delta}^{(l+1)} \in \mathbb{R}^{(d^{(l)}+1) \times 1}, \text{ the same size as } \mathbf{o}^{(l)}!$$

# Computing Gradients

$$\frac{\partial \ell^{(i)}}{\partial w_{b,a}^{(l)}} = \delta_b^{(l)} \left( \frac{\partial s_b^{(l)}}{\partial w_{b,a}^{(l)}} \right) = \delta_b^{(l)} \left( o_a^{(l-1)} \right)$$

$$\nabla_{W^{(l)}} \ell^{(i)} \left( W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)} \right) = \boldsymbol{\delta}^{(l)} \mathbf{o}^{(l-1)T}$$

Sanity check:

$$\boldsymbol{\delta}^{(l)} \in \mathbb{R}^{d^{(l)} \times 1} \text{ so}$$
$$\boldsymbol{\delta}^{(l)} \mathbf{o}^{(l-1)T} \in \mathbb{R}^{d^{(l)} \times (d^{(l-1)} + 1)}, \text{ the same size as } W^{(l)}!$$

# Computing Partial Derivatives

- Can recursively compute  $\delta^{(l)}$  using  $\delta^{(l+1)}$ ; need to compute the base case:  $\delta^{(L)}$
- Assume the output layer is a single node and the error function is the squared error:  $\delta^{(L)} = \delta_1^{(L)}$ ,  $\mathbf{o}^{(L)} = o_1^{(L)}$

$$\text{and } \ell^{(i)} \left( W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)} \right) = \left( o_1^{(L)} - y^{(i)} \right)^2$$

$$\delta_1^{(L)} = \frac{\partial \ell^{(i)}}{\partial s_1^{(L)}} = \frac{\partial}{\partial s_1^{(L)}} \left( o_1^{(L)} - y^{(i)} \right)^2$$

$$= 2 \left( o_1^{(L)} - y^{(i)} \right) \frac{\partial o_1^{(L)}}{\partial s_1^{(L)}} = 2 \left( o_1^{(L)} - y^{(i)} \right)$$

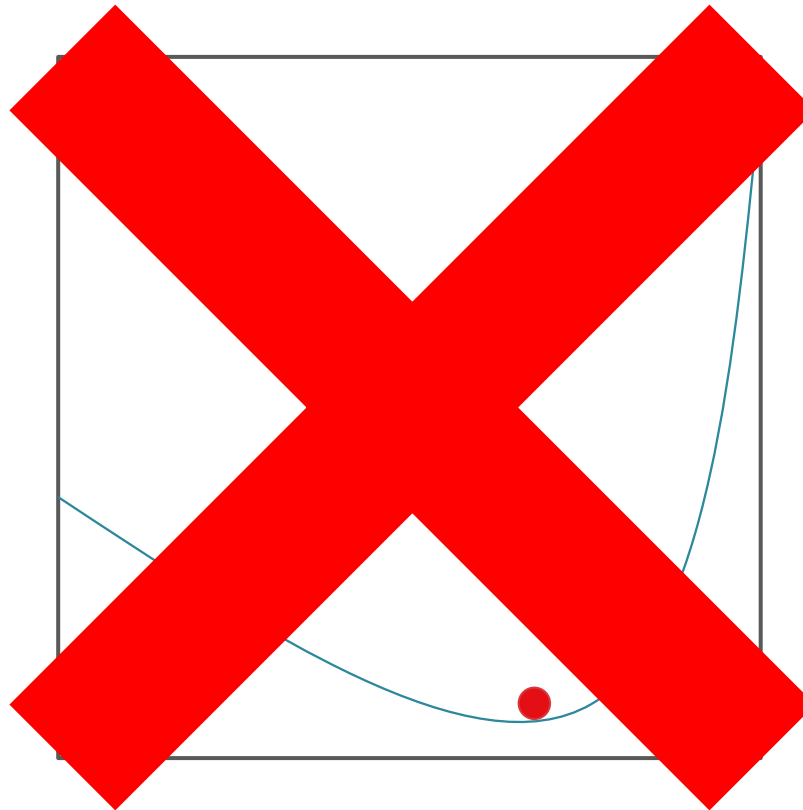
when  $\theta(z) = z$

# Back- propagation

- Input:  $W^{(1)}, \dots, W^{(L)}$  and  $(\mathbf{x}^{(i)}, y^{(i)})$
- Run forward propagation with  $\mathbf{x}^{(i)}$  to get  $\mathbf{o}^{(1)}, \dots, \mathbf{o}^{(L)}$
- (Optional) Compute  $\ell^{(i)} = (o^{(L)} - y^{(i)})^2$
- Initialize:  $\delta^{(L)} = 2(o_1^{(L)} - y^{(i)})$
- For  $l = L - 1, \dots, 1$ 
  - Compute  $\delta^{(l)} = W^{(l+1)T} \delta^{(l+1)} \odot (1 - \mathbf{o}^{(l)} \odot \mathbf{o}^{(l)})$
  - Compute  $G^{(l)} = \delta^{(l)} \mathbf{o}^{(l-1)T}$
- Output:  $G^{(1)}, \dots, G^{(L)}$ , the gradients of  $\ell^{(i)}$  w.r.t  $W^{(1)}, \dots, W^{(L)}$

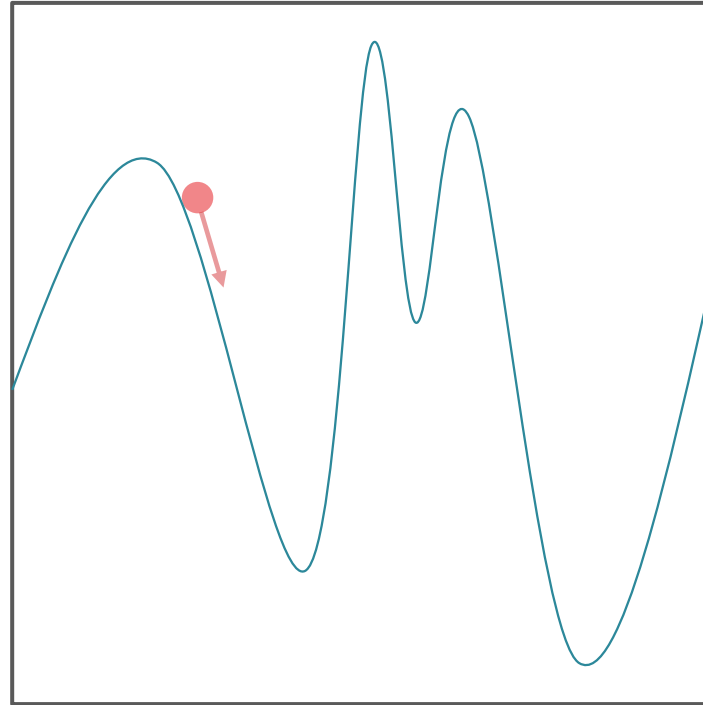
# Recall: Gradient Descent

- Iterative method for minimizing functions
- Requires the gradient to exist everywhere



# Non-convexity

- Gradient descent is not guaranteed to find a global minimum on non-convex surfaces





# Stochastic Gradient Descent for Learning

- Input:  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N, \eta^{(0)}$
- Initialize all weights  $W_{(0)}^{(1)}, \dots, W_{(0)}^{(L)}$  to small, random numbers and set  $t = 0$
- While TERMINATION CRITERION is not satisfied
  - For  $i \in \text{shuffle}(\{1, \dots, N\})$ 
    - For  $l = 1, \dots, L$ 
      - Compute  $G^{(l)} = \nabla_{W^{(l)}} \ell^{(i)} \left( W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)} \right)$
      - Update  $W^{(l)}$ :  $W_{(t+1)}^{(l)} = W_{(t)}^{(l)} - \eta_0 G^{(l)}$
    - Increment  $t$ :  $t = t + 1$
- Output:  $W_{(t)}^{(1)}, \dots, W_{(t)}^{(L)}$