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

Matrix Calculus

Numerator

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

Denominator

Table courtesy of Matt Gormley

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.

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

Table courtesy of Matt Gormley

Matrix Calculus: Denominator Layout

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

Table courtesy of Matt Gormley

Computing Gradients

weights from bias node in layer $\ell-1$

loss of a single data point $(x^{(i)}, y^{(i)})$

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

$$\in \mathbb{R}^{d^{(\ell)} \times (d^{(\ell-1)}+1)}$$

which is the same size as $W^{(\ell)}$

Computing Gradients: Intuition

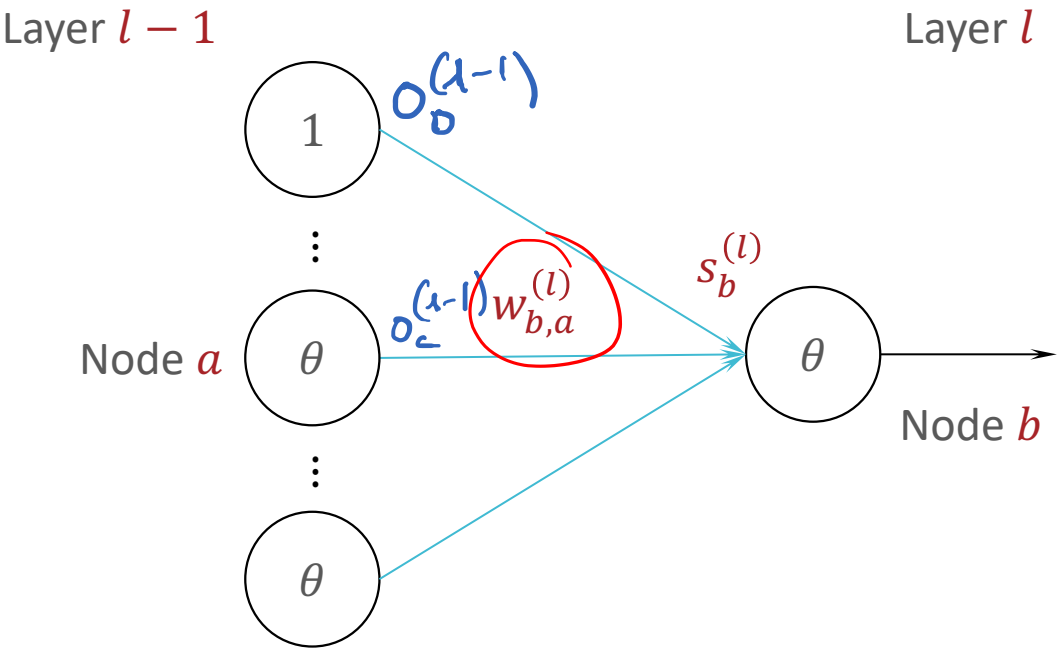
- A weight affects the prediction of the network (and therefore the error) through downstream signals/outputs
  - Use the chain rule!
- Any weight going into the same node will affect the prediction through the same downstream path
  - Compute derivatives starting from the last layer and move “backwards”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

Computing Partial Derivatives

Computing $\nabla_{W^{(l)}} \ell^{(i)} \left(W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)} \right)$ 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 $\nabla_{W^{(l)}} \ell^{(i)} \left(W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)} \right)$ 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: $\frac{\partial \ell^{(i)}}{\partial w_{b,a}^{(l)}} = \frac{\partial \ell^{(i)}}{\partial s_b^{(l)}} \left(\frac{\partial s_b^{(l)}}{\partial w_{b,a}^{(l)}} \right)$

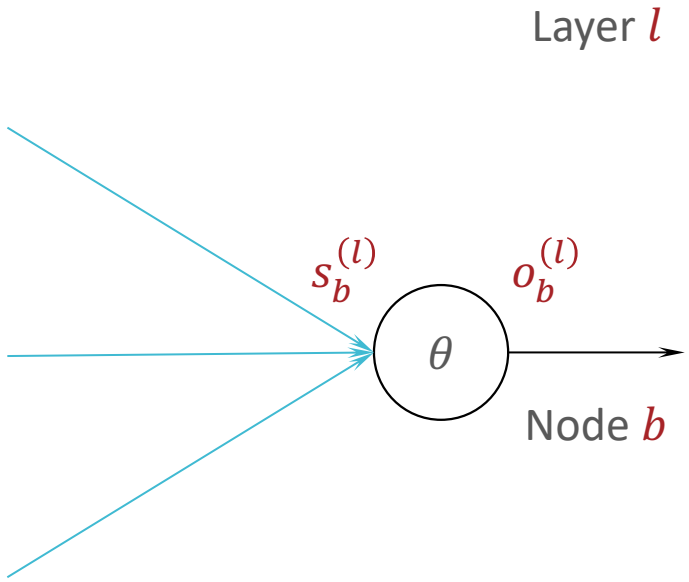
$s_b^{(l)} = \sum_{i=0}^{d^{(l-1)}} w_{b,i}^{(l)} o_i^{(l-1)} \Rightarrow \frac{\partial s_b^{(l)}}{\partial w_{b,a}^{(l)}} = o_a^{(l-1)}$

Compute outputs using forward propagation

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

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 $o_b^{(l)}$

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

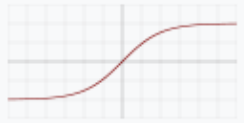
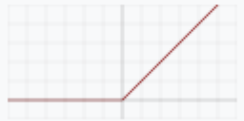
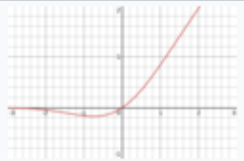
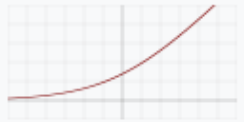
$o_b^{(l)} = \Theta(s_b^{(l)})$ e.g. $\Theta(\cdot) = \tanh(\cdot)$

$\Rightarrow \frac{\partial o_b^{(l)}}{\partial s_b^{(l)}} = 1 - \left(\tanh(s_b^{(l)})\right)^2$

$= 1 - (o_b^{(l)})^2 \leq 1$

compute these from the forward propagation

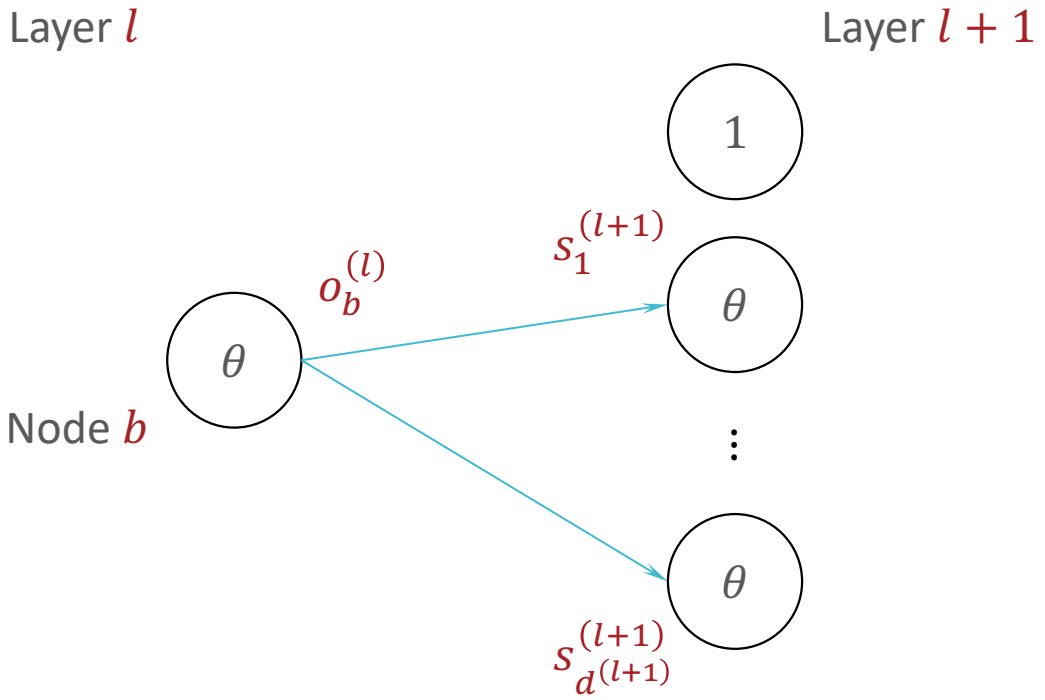
Recall: Other Activation Functions

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

Source: https://en.wikipedia.org/wiki/Activation_function

Computing Partial Derivatives

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



Computing Partial Derivatives

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

$$\delta_j^{(l)}$$

chain rule: $$\frac{\partial \ell^{(i)}}{\partial o_b^{(l)}} = \sum_{j=1}^{d^{(l+1)}} \frac{\partial \ell^{(i)}}{\partial s_j^{(l+1)}} \left(\frac{\partial s_j^{(l+1)}}{\partial o_b^{(l)}} \right)$$

$$s_j^{(l+1)} = \sum_{k=0}^{d^{(l)}} w_{j,k}^{(l+1)} o_k^{(l)} \Rightarrow \frac{\partial s_j^{(l+1)}}{\partial o_b^{(l)}} = w_{j,b}^{(l+1)}$$

$$\frac{\partial \ell^{(i)}}{\partial o_b^{(l)}} = \sum_{j=1}^{d^{(l+1)}} \delta_j^{(l+1)} w_{j,b}^{(l+1)}$$

Computing Partial Derivatives

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

$$= \left(\sum_{c=1}^{d^{(l+1)}} \delta_c^{(l+1)} \left(w_{c,b}^{(l+1)} \right) \right) \left(1 - \left(o_b^{(l)} \right)^2 \right)$$

$$\boldsymbol{\delta}^{(l)} := \nabla_{\boldsymbol{s}^{(l)}} \ell^{(i)} \left(W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)} \right)$$

Computing Partial Derivatives

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

$$= \left(\sum_{c=1}^{d^{(l+1)}} \delta_c^{(l+1)} \left(w_{c,b}^{(l+1)} \right) \right) \left(1 - \left(o_b^{(l)} \right)^2 \right)$$

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

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

Sanity check:

$e(o^{(L)}, y^{(i)}) = \ell^{(i)}(w_{(t)}^{(1)} \cdots w_{(t)}^{(L)})$

↳ prediction $\hat{y}(w_{(t)}^{(1)}, \ldots, w_{(t)}^{(L)})$

if $\Theta(\cdot) = \tanh(\cdot)$

sanity check: $W^{(l+1)} \in \mathbb{R}^{d^{(l+1)} \times (d^{(l)}+1)}$

$W^{(l+1)^T} \in \mathbb{R}^{(d^{(l)}+1) \times d^{(l+1)}}$

$W^{(l+1)^T} \delta^{(l+1)} \in \mathbb{R}^{d^{(l)}+1}$ is the same size as $o^{(l)}$

Computing Gradients

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

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

the same size as $W^{(l)}$

Sanity check:

$$\delta^{(l)} \in \mathbb{R}^{d^{(l)}}$$

$$o^{(l-1)} \in \mathbb{R}^{(d^{(l-1)}+1)}$$

$$\Rightarrow \delta^{(l)} o^{(l-1)T} \in \mathbb{R}^{d^{(l)} \times (d^{(l-1)}+1)}$$

Computing Partial Derivatives

- Can recursively compute $\boldsymbol{\delta}^{(l)}$ using $\boldsymbol{\delta}^{(l+1)}$; need to compute the base case: $\boldsymbol{\delta}^{(L)}$
- Assume the output layer is a single node and the error function is the squared error: $\boldsymbol{\delta}^{(L)} = \delta_1^{(L)}$, $\boldsymbol{o}^{(L)} = o_1^{(L)}$ and $\ell^{(i)}\left(W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)}\right) = \left(o_1^{(L)} - y^{(i)}\right)^2$

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

$$= 2\left(o_1^{(L)} - y^{(i)}\right)\frac{\partial o_1^{(L)}}{\partial s_1^{(L)}} = 2\left(o_1^{(L)} - y^{(i)}\right)(1)$$

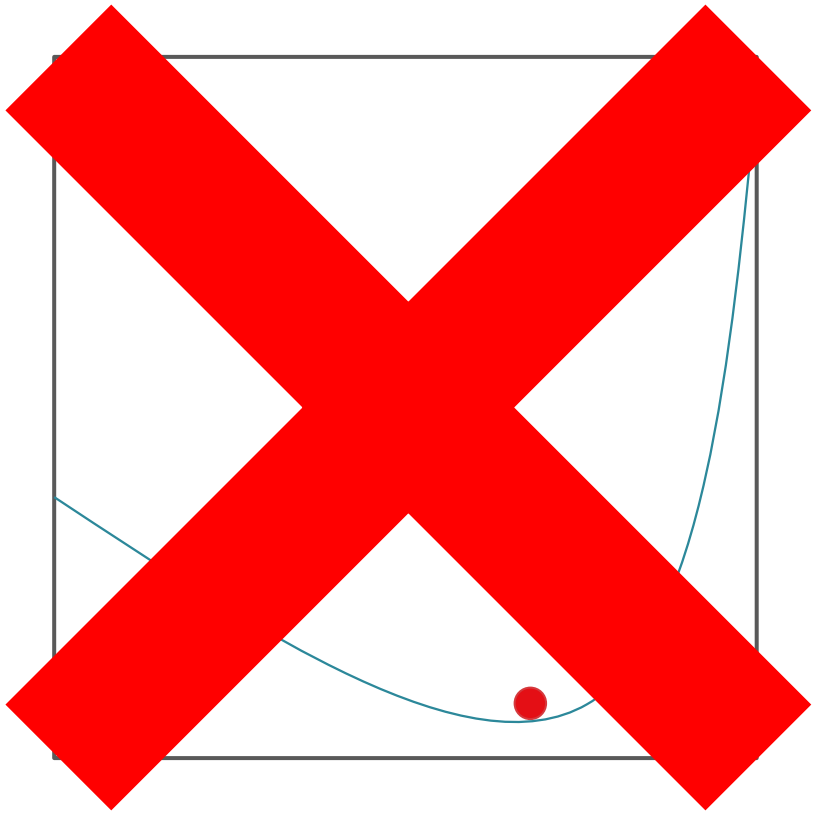
when $\theta(z) = z$

Back-propagation

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

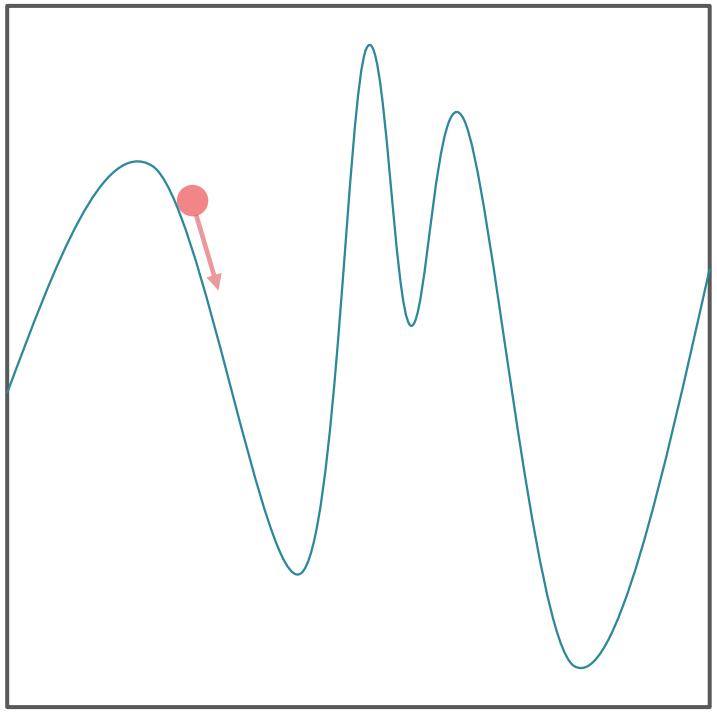
Recall: Gradient Descent

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