10-701: Introduction to Machine Learning Lecture 10 – Backpropagation

Nari Johnson, narij@andrew.cmu.edu

Slides adapted from Henry Chai

9/29/25

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	$\frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial X_{11}} & \frac{\partial y}{\partial X_{12}} & \cdots & \frac{\partial y}{\partial X_{1Q}} \\ \frac{\partial y}{\partial X_{21}} & \frac{\partial y}{\partial X_{22}} & \cdots & \frac{\partial y}{\partial X_{2Q}} \\ \vdots & & & \vdots \\ \frac{\partial y}{\partial X_{P1}} & \frac{\partial y}{\partial X_{P2}} & \cdots & \frac{\partial y}{\partial X_{PQ}} \end{bmatrix}$

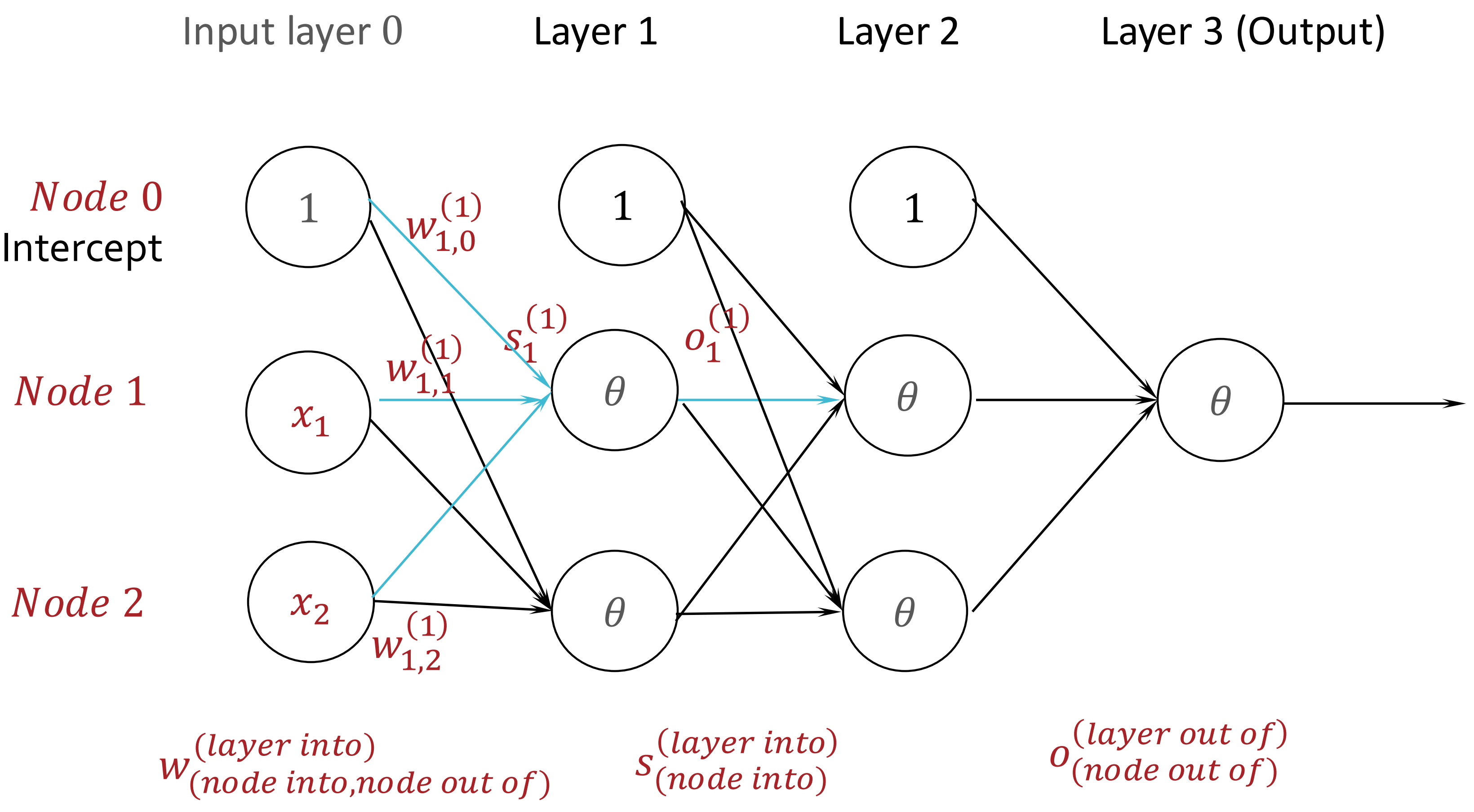
Table courtesy of Matt Gormley

Matrix Calculus: Denominator Layout

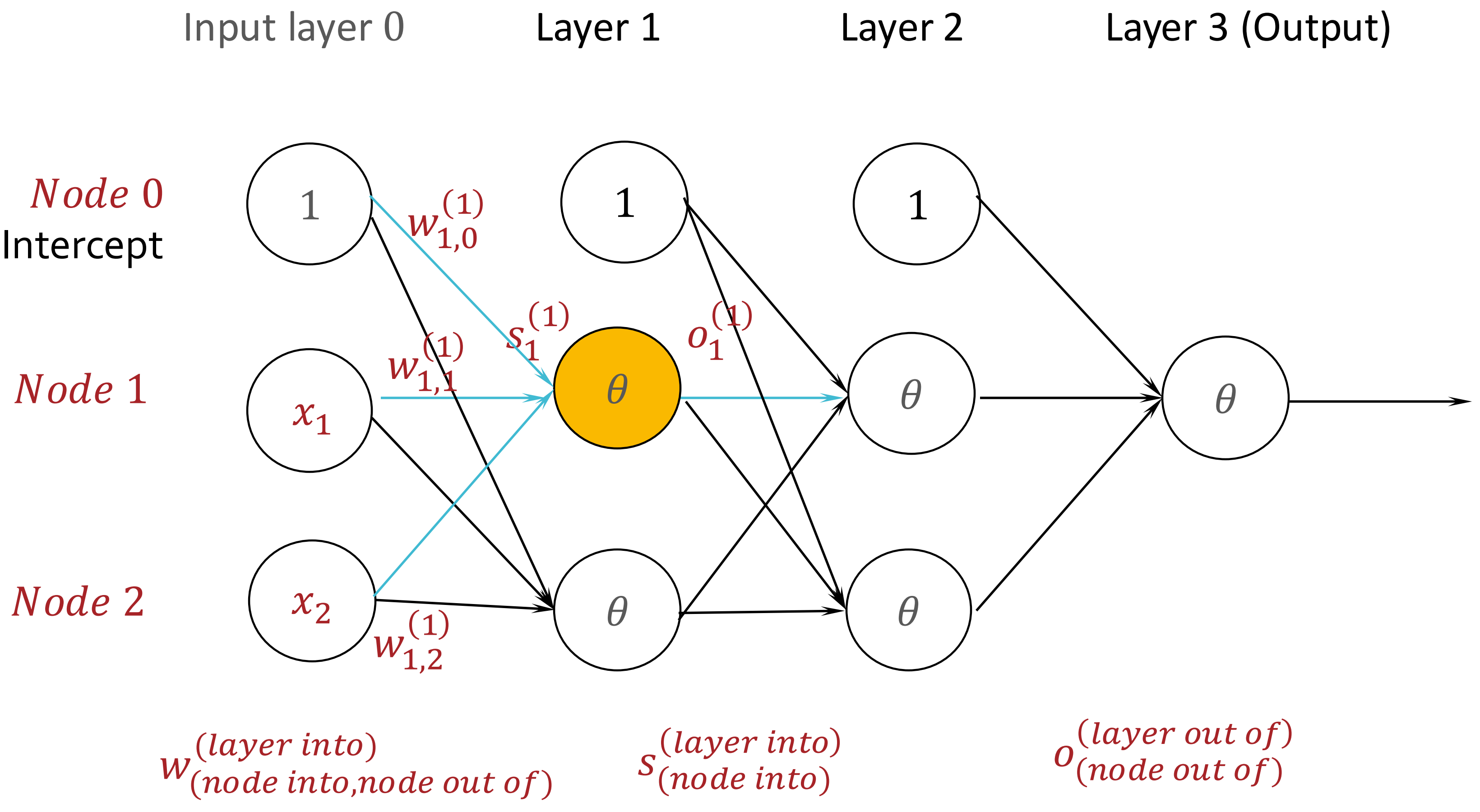
Types of Derivatives	scalar	vector
scalar	$\frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial x}\right]$	$\frac{\partial \mathbf{y}}{\partial x} = \left[\frac{\partial y_1}{\partial x} \quad \frac{\partial y_2}{\partial x} \quad \cdots \quad \frac{\partial y_N}{\partial x}\right]$
vector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

Table courtesy of Matt Gormley

Recall our neural network notation…

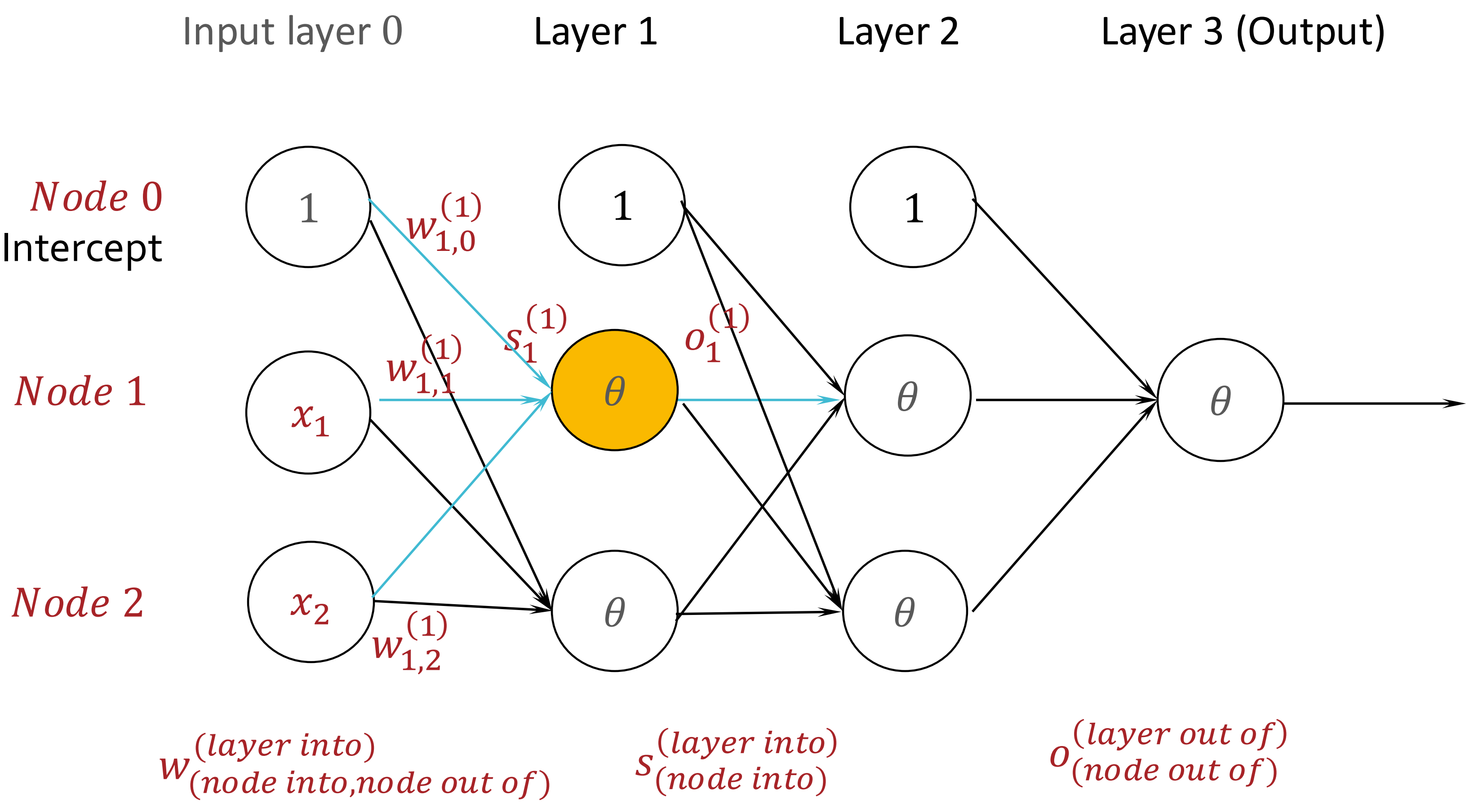


Recall our neural network notation…



Write equations for the signal $s_1^{(1)}$ and output $o_1^{(1)}$ for the highlighted node:

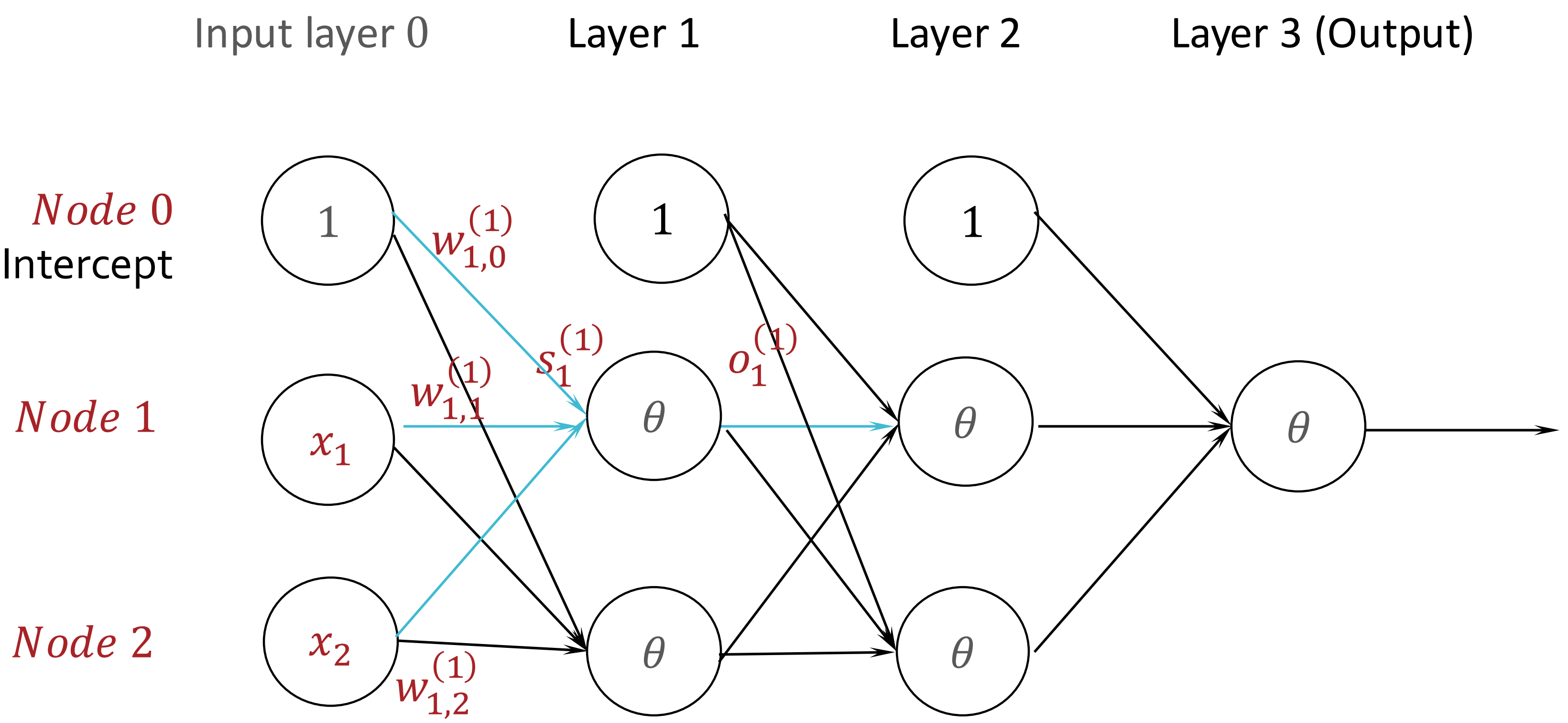
Recall our neural network notation…



Write equations for the signal $s_1^{(1)}$ and output $o_1^{(1)}$ for the highlighted node:

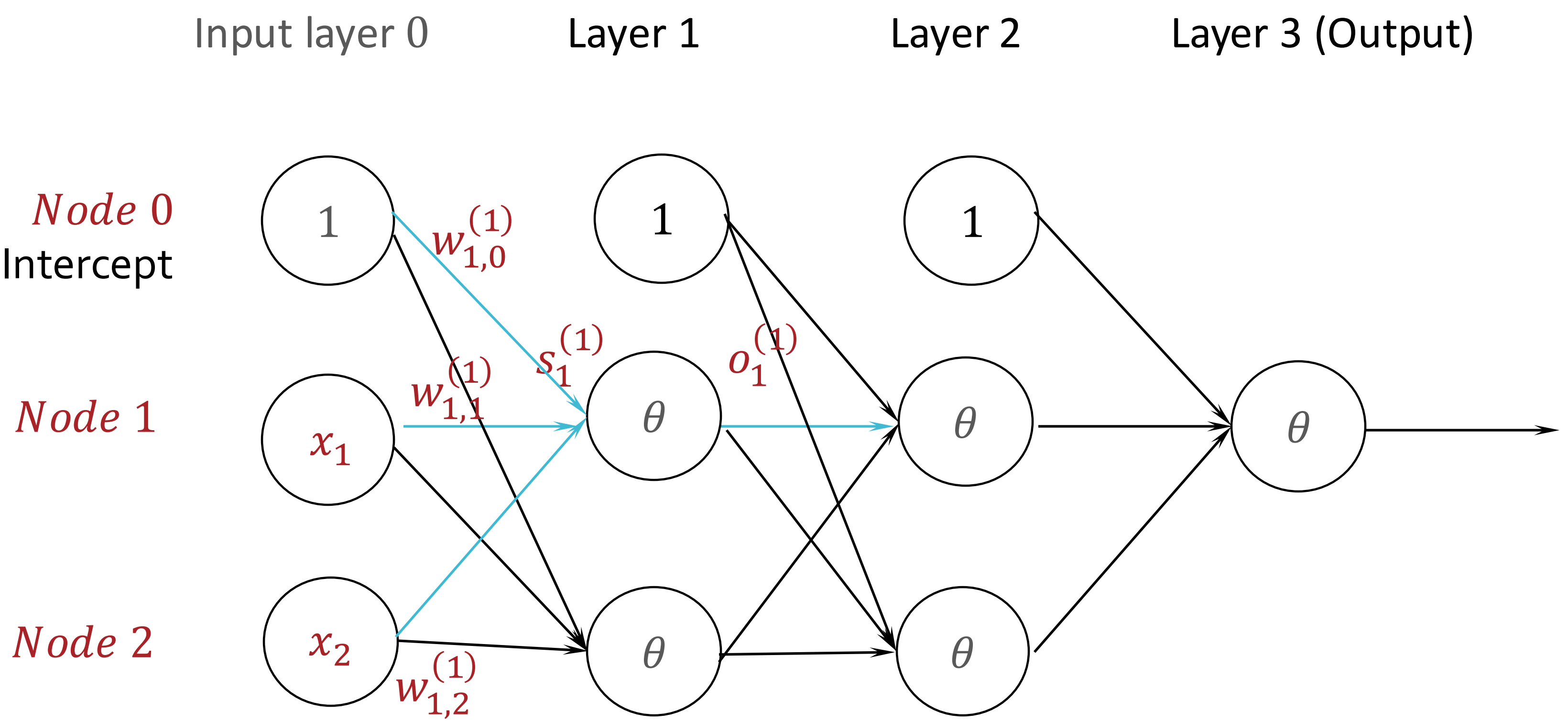
$$s_1^{(1)} = w_{1,0}^{(1)} + w_{1,1}^{(1)} x_1 + w_{1,2}^{(1)} x_2 \qquad o_1^{(1)} = \theta(s_1^{(1)})$$

Recall our neural network notation...



We can also write our weights, signals, and outputs using matrices:

Recall our neural network notation…

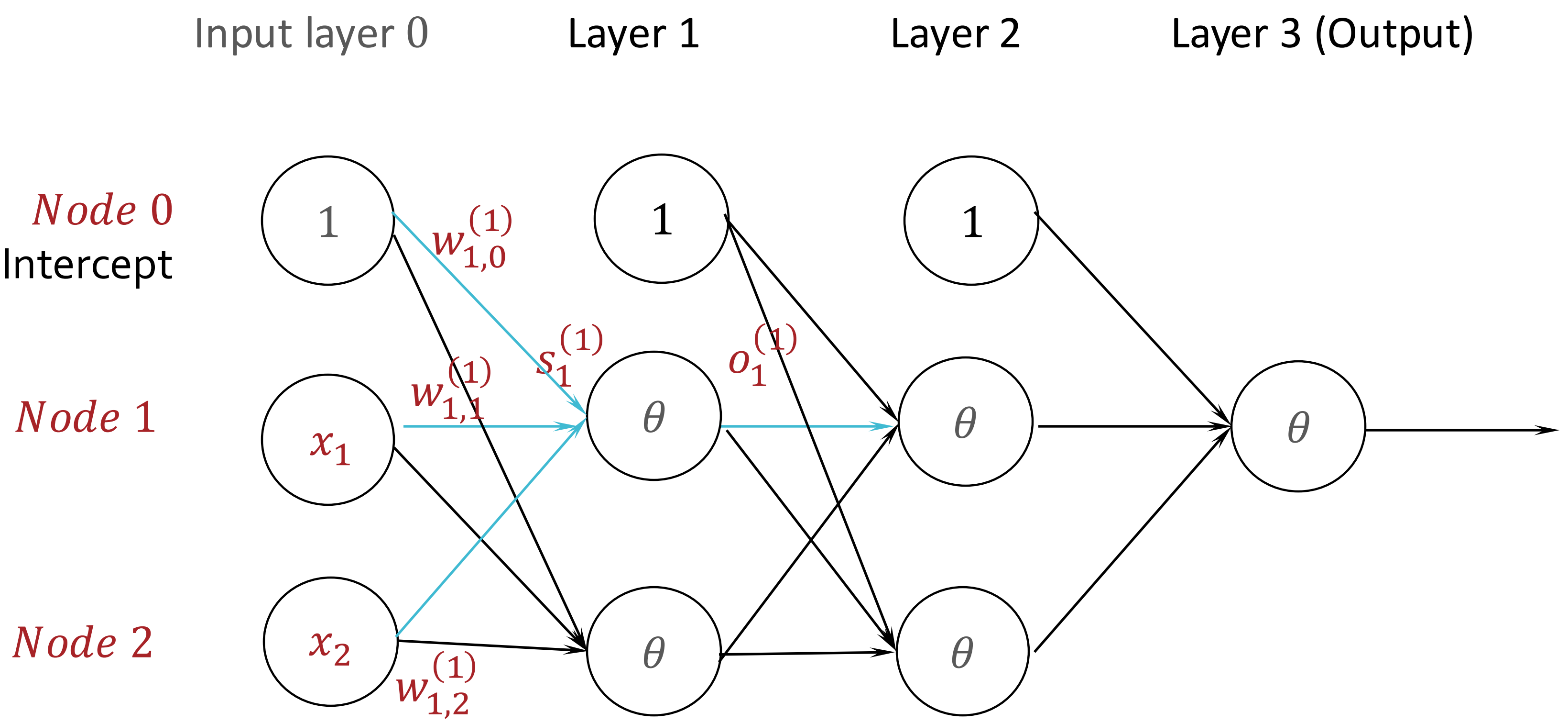


out of

$$W^{(1)} = \begin{bmatrix} w_{1,0}^{(1)} & w_{1,1}^{(1)} & w_{1,2}^{(1)} \\ w_{2,0}^{(1)} & w_{2,1}^{(1)} & w_{2,2}^{(1)} \end{bmatrix} \quad \text{into} \qquad o^{(1)} = \begin{bmatrix} 1 \\ o_1^{(1)} \\ o_2^{(1)} \end{bmatrix}$$

Can you write $s^{(2)}$ as a matrix product of $W^{(2)}$ and $o^{(1)}$?
What shape is each matrix?

Recall our neural network notation…



Can you write $s^{(2)}$ as a matrix product of $W^{(2)}$ and $o^{(1)}$?
What shape is each matrix?

$$s^{(2)} = W^{(2)} o^{(1)}$$

$$s^{(2)}: [2, 1], W^{(2)}: [2, 3], o^{(1)}: [3, 1]$$

Recall: the chain rule

The chain rule tells us how to differentiate composite functions.

It is important for neural networks because the output of our neural network is a composite function of the network weights.

A simple example:

$$a = 2x^3 - 4x$$
$$b = \sin(a)$$

$$\frac{\partial b}{\partial x} =$$

Recall: the chain rule

The chain rule tells us how to differentiate composite functions.

It is important for neural networks because the output of our neural network is a composite function of the network weights.

A simple example:

$$a = 2x^3 - 4x$$
$$b = \sin(a)$$

$$\frac{\partial b}{\partial x} = \frac{\partial b}{\partial a}\frac{\partial a}{\partial x}$$

$$= \cos(a)\,(6x^2 - 4)$$

Applying the chain rule to our neural network example

Recall from our toy neural network that:

$$s_1^{(1)} = w_{1,0}^{(1)} + w_{1,1}^{(1)} x_1 + w_{1,2}^{(1)} x_2 \qquad o_1^{(1)} = \theta(s_1^{(1)})$$

Use the chain rule to calculate the following partial derivative.
Assume that the activation function $\theta(-) = \tanh(-)$

$$\frac{\partial o_1^{(1)}}{\partial w_{1,1}^{(1)}} =$$

Hint: $\frac{\partial}{\partial z} \tanh(z) = 1 - (\tanh(z))^2$

A: $= \left(1 - (o_1^{(1)})^2\right)(x_1)$

B: $= \left(1 - o_1^{(1)}\right)\left(w_{1,1}^{(1)}\right)$

C: $= \left(1 + o_1^{(1)}\right)\left(w_{1,1}^{(1)} x_1\right)$

D: $= o_1^{(1)} w_{1,1}^{(1)}$

Applying the chain rule to our neural network example

Recall from our toy neural network that:

$$s_1^{(1)} = w_{1,0}^{(1)} + w_{1,1}^{(1)} x_1 + w_{1,2}^{(1)} x_2 \qquad o_1^{(1)} = \theta(s_1^{(1)})$$

Use the chain rule to calculate the following partial derivative.
Assume that the activation function $\theta(-) = \tanh(-)$

$$\frac{\partial o_1^{(1)}}{\partial w_{1,1}^{(1)}} =$$

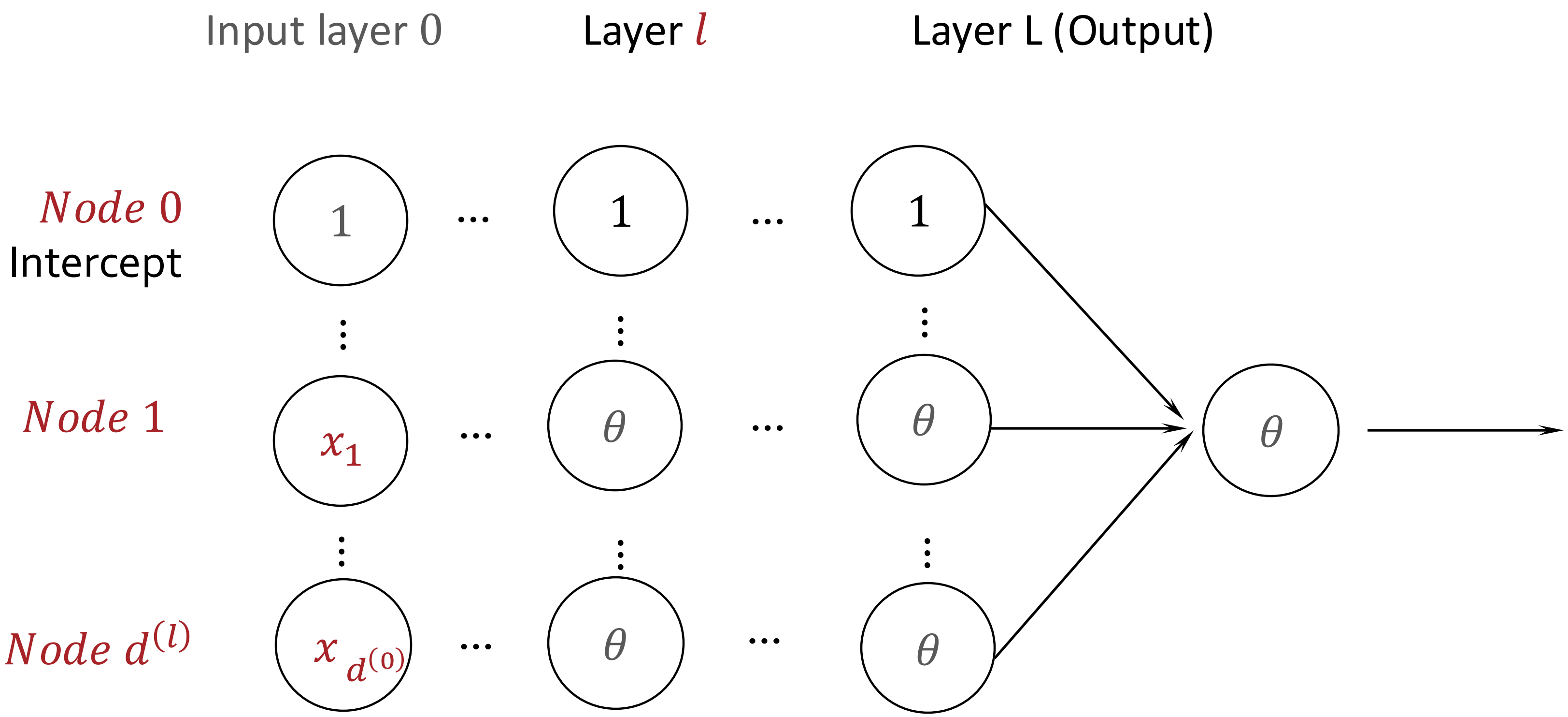
Hint: $\frac{\partial}{\partial z}\tanh(z) = 1 - (\tanh(z))^2$

$$= \frac{\partial o_1^{(1)}}{\partial s_1^{(1)}} \frac{\partial s_1^{(1)}}{\partial w_{1,1}^{(1)}}$$

$$= \left(1 - (\tanh(s_1^{(1)}))^2\right)(x_1)$$

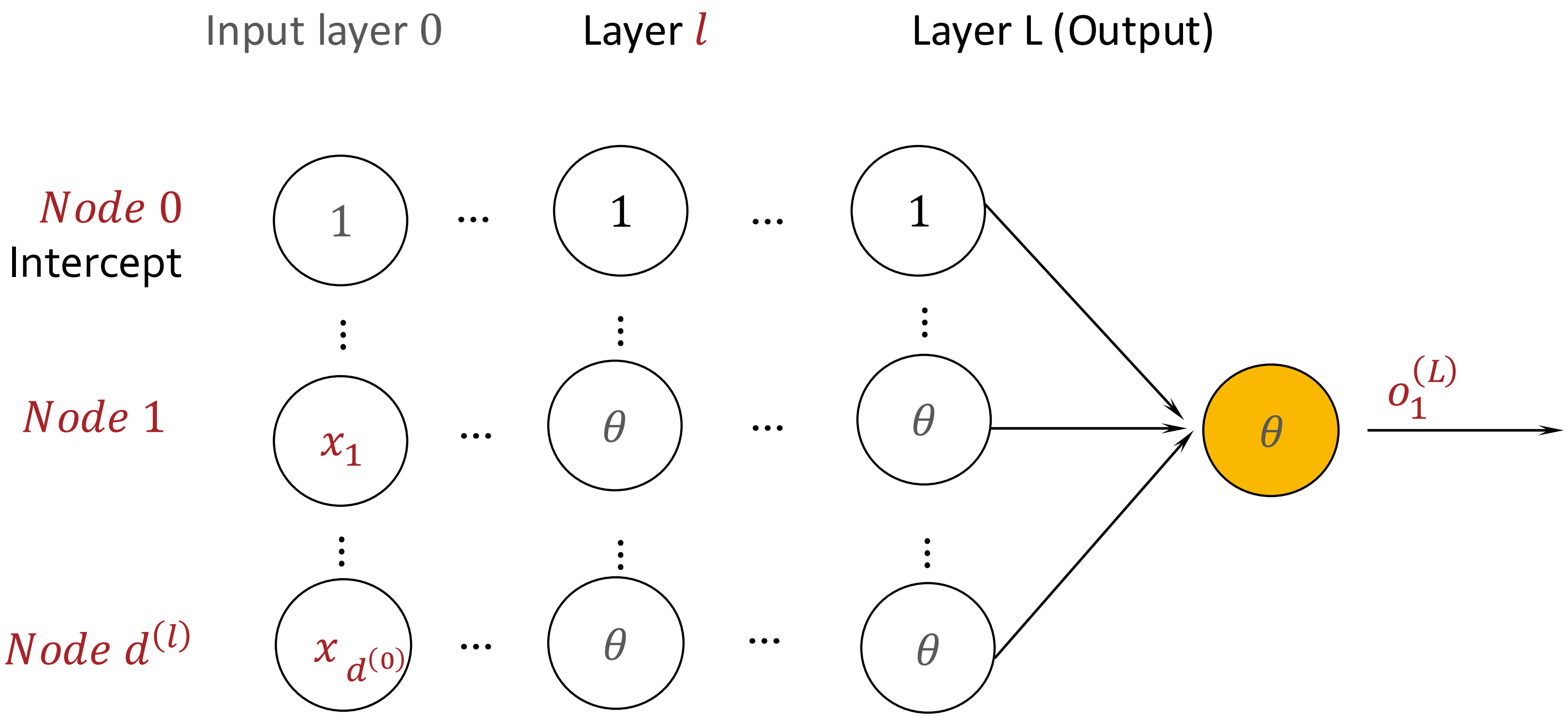
$$= \left(1 - (o_1^{(1)})^2\right)(x_1)$$

Zooming out...



A neural network has L layers.
Each layer has $d^{(l)}$ nodes, and its own weight matrix $W^{(l)}$
The matrix $W^{(l)}$ connects layer $(l-1)$ to layer l

Building intuition for backprop



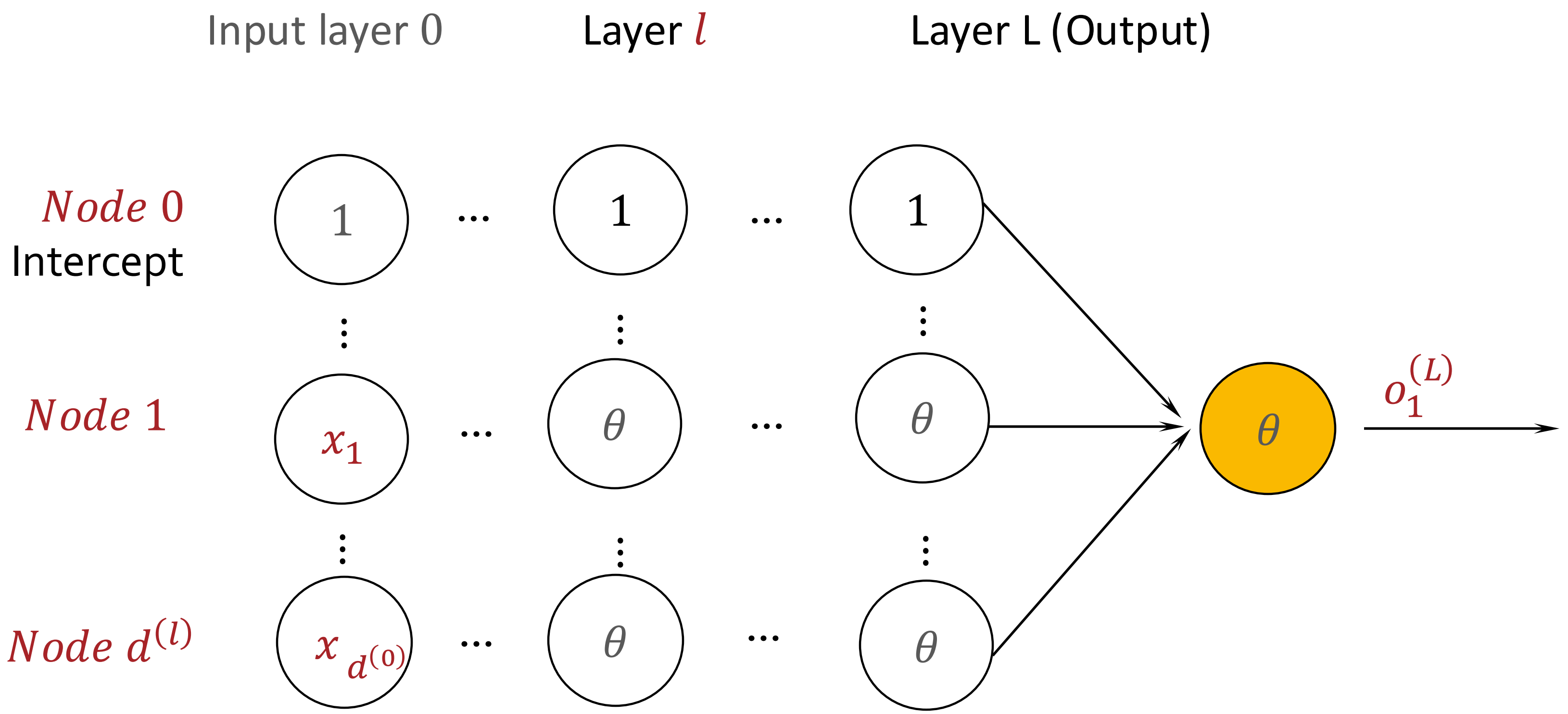
The final loss is a function of the *output of the entire network:*

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

But the final output is a composite function of *all of the weights that came before*…

$$o^{(L)} = \theta\left(s^{(L)}\right)$$
$$= \theta\left(W^{(L)}\left(\theta\left(s^{(L-1)}\right)\right)\right)$$
$$= \cdots$$

Building intuition for backprop



Idea: We can *apply the chain rule* to calculate the gradients at each layer.

$$\frac{\partial \ell^{(i)}}{\partial W^{(L)}} = \frac{\partial \ell^{(i)}}{\partial o^{(L)}} \left(\frac{\partial o^{(L)}}{\partial s^{(L)}}\right) \left(\frac{\partial s^{(L)}}{\partial W^{(L)}}\right)$$

$$\frac{\partial \ell^{(i)}}{\partial W^{(L-1)}} = \frac{\partial \ell^{(i)}}{\partial o^{(L)}} \left(\frac{\partial o^{(L)}}{\partial s^{(L)}}\right) \left(\frac{\partial s^{(L)}}{\partial o^{(L-1)}}\right) \left(\frac{\partial o^{(L-1)}}{\partial s^{(L-1)}}\right) \left(\frac{\partial s^{(L-1)}}{\partial W^{(L-1)}}\right)$$

Computing Gradients

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

Computing Gradients

Recall: $W^{(l)}: [\, d^{(l)}, d^{(l-1)} + 1]$

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

Computing Partial Derivatives

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

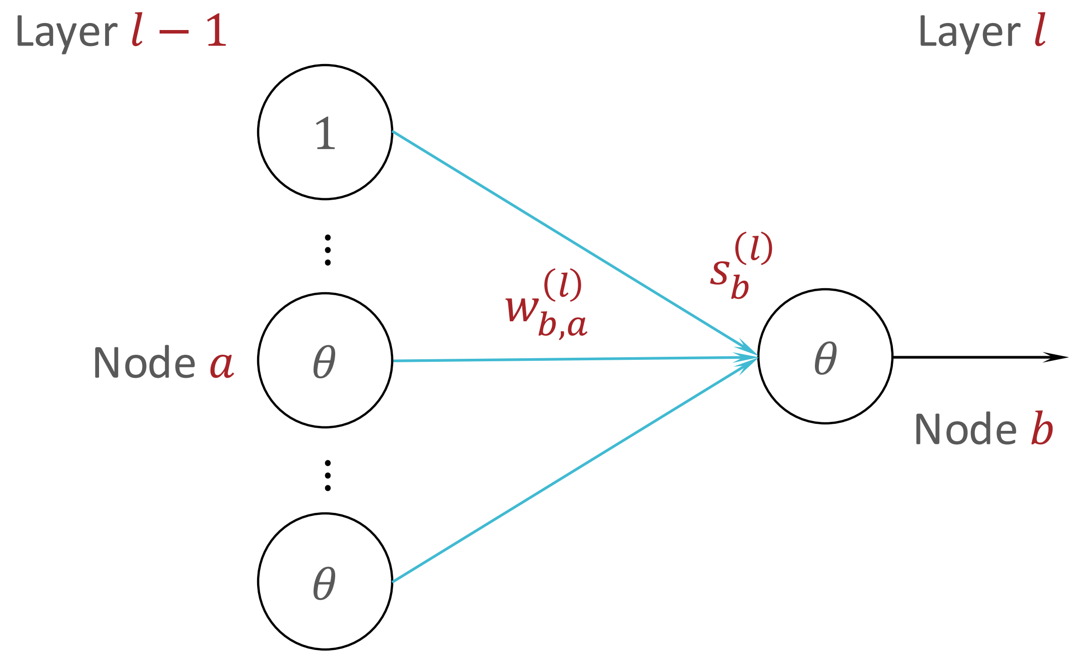
$$\frac{\partial \ell^{(i)}}{\partial w_{b,a}^{(l)}}$$

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

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

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

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

Computing Partial Derivatives

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

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

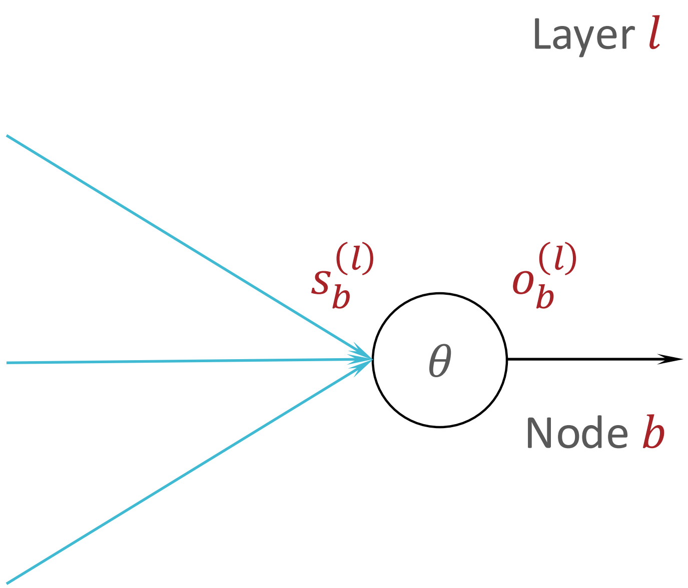
Insight: $w_{b,a}^{(l)}$ *only* affects $\ell^{(i)}$ via $s_b^{(l)}$

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

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

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 $o_b^{(l)}$

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

Computing Partial Derivatives

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

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

$$o_b^{(l)} = \theta\left(s_b^{(l)}\right) \rightarrow \frac{\partial o_b^{(l)}}{\partial s_b^{(l)}} = \frac{\partial \theta\left(s_b^{(l)}\right)}{\partial s_b^{(l)}}$$

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

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

Computing Partial Derivatives

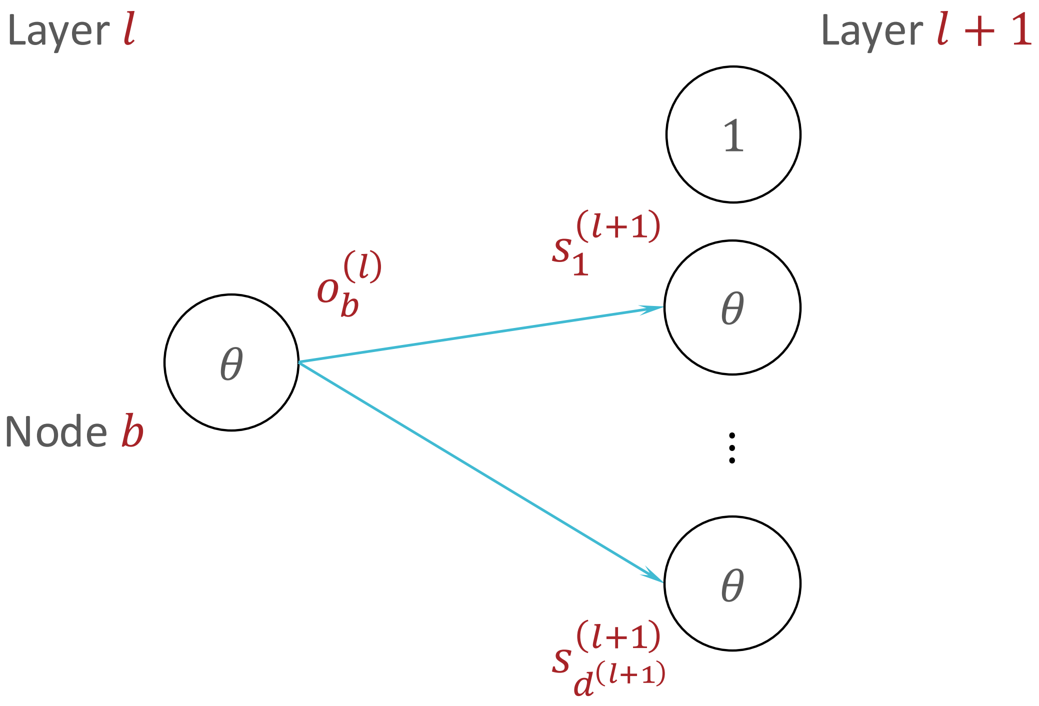
Recap:

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

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

Computing Partial Derivatives

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



Computing Partial Derivatives

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

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

Computing Partial Derivatives

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

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

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

Computing Partial Derivatives

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

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

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

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

Computing Partial Derivatives

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

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

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

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

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

Computing Partial Derivatives

Recap:

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

We just learned that we can write $\delta_b^{(l)}$ as a function of the terms $\delta_c^{(l+1)}$ at the next layer!

Computing Partial Derivatives

Pasting over what we calculated previously:

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

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

Computing Partial Derivatives

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

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

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

How would you write the vector $\boldsymbol{\delta}^{(l)}$ as a function of the matrices $\boldsymbol{\delta}^{(l+1)}, \boldsymbol{W}^{(l+1)}$ and $\boldsymbol{o}^{(l)}$?

Hint: $\boldsymbol{s}^{(l)}: [d^{(l)}, 1]$, $\boldsymbol{o}^{(l)}: [d^{(l)}+1, 1]$, $\boldsymbol{W}^{(l)}: [d^{(l)}, d^{(l-1)} + 1]$

Computing Partial Derivatives

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

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

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

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

Sanity check:

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

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

Computing Partial Derivatives

Putting it all together:

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

$$\frac{\partial \ell^{(i)}}{\partial \boldsymbol{W}^{(l)}} =$$

Computing Partial Derivatives

Putting it all together:

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

$$\frac{\partial \ell^{(i)}}{\partial \boldsymbol{W}^{(l)}} = \boldsymbol{\delta}^{(l)} {\boldsymbol{o}^{(l-1)}}^T$$

Sanity check:

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

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

Computing Partial Derivatives

- Can recursively compute $\boldsymbol{\delta}^{(l)}$ using $\boldsymbol{\delta}^{(l+1)}$; need to compute the base case: $\boldsymbol{\delta}^{(L)}$

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

- Assume the output layer is a single node and the error function is the squared error: $\boldsymbol{\delta}^{(L)} = \delta_1^{(L)}, \boldsymbol{o}^{(L)} = o_1^{(L)}$

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

Computing Partial Derivatives

- 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 $\theta(z) = z$, calculate $\delta_1^{(L)} = \frac{\partial \ell^{(i)}}{\partial s_1^{(L)}}$:

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

Computing Partial Derivatives

- Assume the output layer is a single node and the error function is the squared error: $\boldsymbol{\delta}^{(L)} = \delta_1^{(L)}, \boldsymbol{o}^{(L)} = o_1^{(L)}$

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

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

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

when $\theta(z) = z$

Computing Partial Derivatives

We can compute $\boldsymbol{\delta}^{(L)}$!!!

We can use these to compute $\boldsymbol{\delta}^{(L-1)}, \boldsymbol{\delta}^{(L-2)}$, all the way back to $\boldsymbol{\delta}^{(1)}$, using our equation from before….

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

And we know how to use the $\boldsymbol{\delta}^{(l)}$ to compute the gradient:

$$\frac{\partial \ell^{(i)}}{\partial \boldsymbol{W}^{(l)}} = \boldsymbol{\delta}^{(l)} \boldsymbol{o}^{(l-1)^T}$$

Back-propagation

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

Key take-aways

- 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)

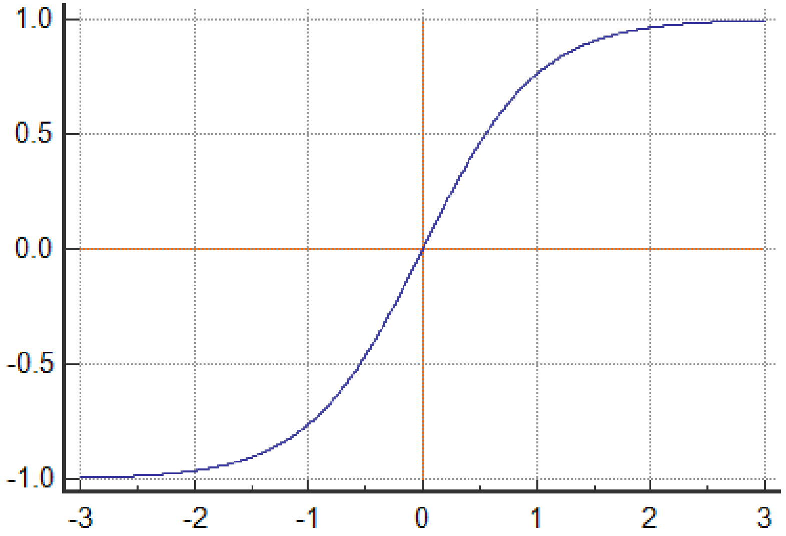
Aside: Vanishing Gradients

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

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

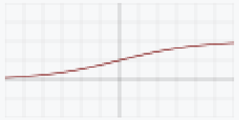
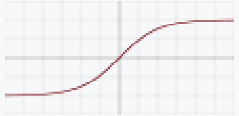
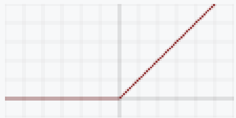
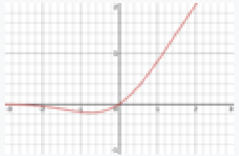
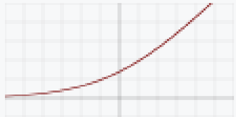
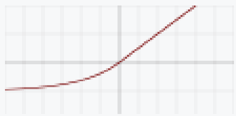
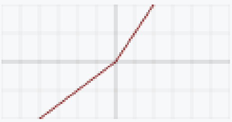
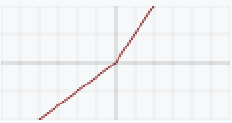
$$o_b^{(l)} = \theta\left(s_b^{(l)}\right) \rightarrow \frac{\partial o_b^{(l)}}{\partial s_b^{(l)}} = \frac{\partial \theta\left(s_b^{(l)}\right)}{\partial s_b^{(l)}}$$

$$= 1 - \left(\tanh\left(s_b^{(l)}\right)\right)^2 \leq 1$$



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

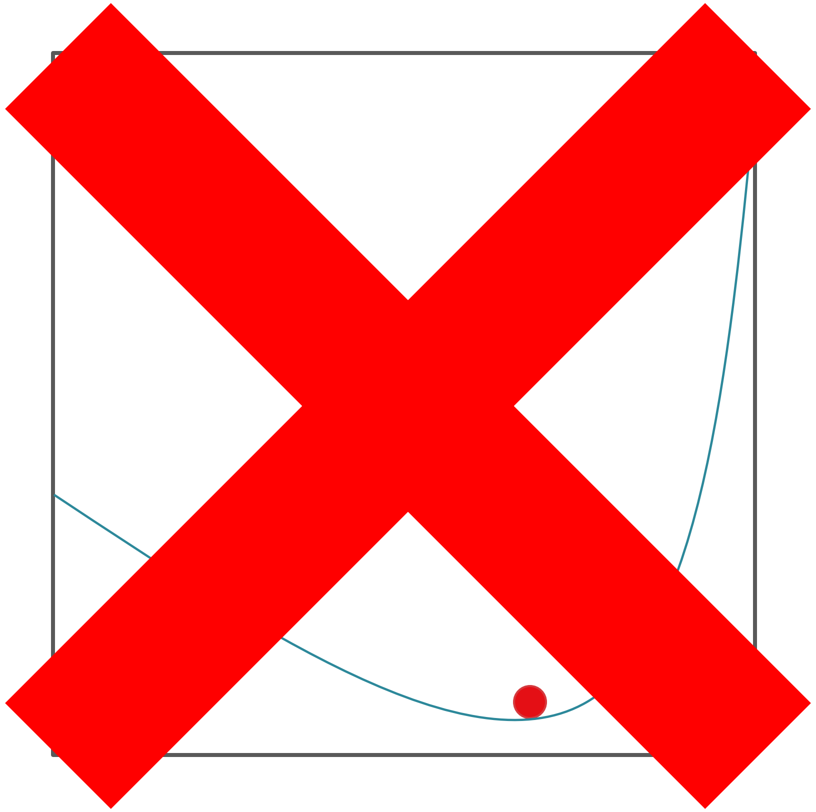
Recall: Other Activation Functions

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)[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]		$\frac{1}{2}x\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$ $= x\Phi(x)$
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

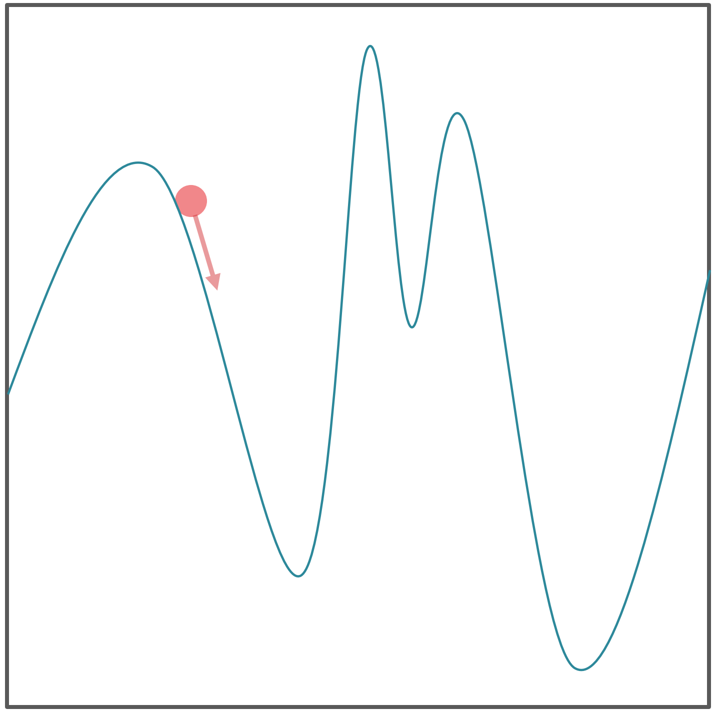
Recall: Gradient Descent

- 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: $W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)}$

Mini-batch Stochastic Gradient Descent for Learning

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}, \eta_{MB}^{(0)}, B$

1. Initialize all weights $W_{(0)}^{(1)}, \ldots, W_{(0)}^{(L)}$ to small, random numbers and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Randomly sample B data points from $\mathcal{D}$, $\left\{\left(\boldsymbol{x}^{(b)}, y^{(b)}\right)\right\}_{b=1}^{B}$
   b. Compute the gradient w.r.t. the sampled *batch*,
   $$G^{(l)} = \frac{1}{B} \sum_{b=1}^{B} \nabla_{W^{(l)}} \ell^{(b)} \left(W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)}\right) \ \forall\, l$$
   c. Update $W^{(l)}$: $W_{t+1}^{(l)} \leftarrow W_{t}^{(l)} - \eta_{MB}^{(0)} G^{(l)} \ \forall\, l$
   d. Increment t: $t \leftarrow t + 1$

- Output: $W_{t}^{(1)}, \ldots, W_{t}^{(L)}$