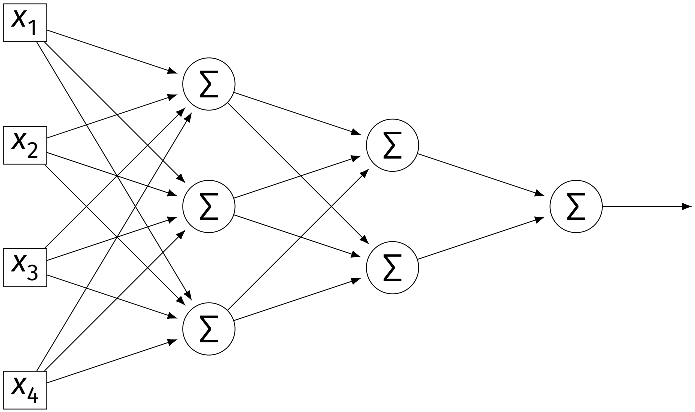
DSC 140B
Representation Learning

Lecture 12 | Part 1

Training Neural Networks

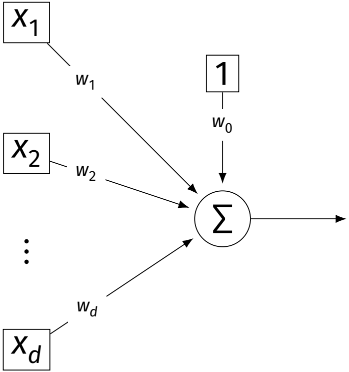
Training

- How do we learn the weights of a (deep) neural network?



Remember...

- How did we learn the weights in linear least squares regression?



Empirical Risk Minimization

0. Collect a training set, $\{(\vec{x}^{(i)}, y_i)\}$
1. Pick the form of the prediction function, H.
2. Pick a loss function.
3. Minimize the empirical risk w.r.t. that loss.

Remember: Linear Least Squares

1. Pick the form of the prediction function, H.
   - E.g., linear: $H(\vec{x}; \vec{w}) = w_0 + w_1 x_1 + \ldots + w_d x_d = \text{Aug}(\vec{x}) \cdot \vec{w}$

2. Pick a loss function.
   - E.g., the square loss.

3. Minimize the empirical risk w.r.t. that loss:

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} (H(\vec{x}^{(i)}) - y_i)^2 = \frac{1}{n} \sum_{i=1}^{n} (\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w} - y_i)^2$$

Minimizing Risk

- To minimize risk, we often use **vector calculus**.
  - Either set $\nabla_{\vec{w}} R(\vec{w}) = 0$ and solve...
  - Or use gradient descent: walk in opposite direction of $\nabla_{\vec{w}} R(\vec{w})$.
- Recall, $\nabla_{\vec{w}} R(\vec{w}) = (\partial R / \partial w_0, \partial R / \partial w_1, \ldots, \partial R / \partial w_d)^T$

In General

- Let ℓ be the loss function, let $H(\vec{x}; \vec{w})$ be the prediction function.

- The empirical risk:

$$R(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell(H(\vec{x}^{(i)}; \vec{w}), y_i)$$

- Using the chain rule:

$$\nabla_{\vec{w}} R(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \ell}{\partial H} \nabla_{\vec{w}} H(\vec{x}^{(i)}; \vec{w})$$

Gradient of H

- To minimize risk, we want to compute $\nabla_{\vec{w}} R$.
- To compute $\nabla_{\vec{w}} R$, we want to compute $\nabla_{\vec{w}} H$.
- This will depend on the form of H.

Example: Linear Model

- Suppose H is a linear prediction function:

$$H(\vec{x}; \vec{w}) = w_0 + w_1 x_1 + \ldots + w_d x_d$$

$$\vec{w} \cdot \text{Aug}(\vec{x})$$

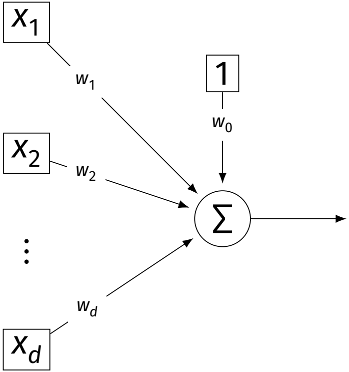
- What is $\nabla_{\vec{w}} H$ with respect to $\vec{w}$?

$$\nabla_{\vec{w}} H(\vec{x}) = \left(\frac{\partial H}{\partial w_0}, \frac{\partial H}{\partial w_1}, \ldots, \frac{\partial H}{\partial w_d} \right)$$

$$= (1, x_1, x_2, \ldots, x_d)$$

Example: Linear Model

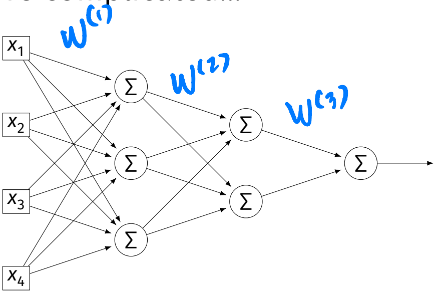
- Consider $\partial H / \partial w_1$:



$$\frac{\partial H}{\partial w_1} = x_1$$

Example: Neural Networks

- Suppose H is a neural network (with nonlinear activations).
- What is ∇H?
  - It's more complicated...



Parameter Vectors

- It is often useful to pack all of the network's weights into a **parameter vector**, $\vec{w}$.
- Order is arbitrary:

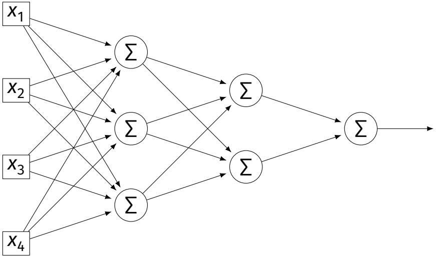
$$\vec{w} = (W_{11}^{(1)}, W_{12}^{(1)}, \ldots, b_1^{(1)}, b_2^{(1)}, W_{11}^{(2)}, W_{12}^{(2)}, \ldots, b_1^{(2)}, b_2^{(2)}, \ldots)^T$$

- The network is a function $H(\vec{x}; \vec{w})$.
- Goal of learning: find the "best" $\vec{w}$.

Gradient of Neural Network

- $\nabla_{\vec{w}} H$ is a vector-valued function.

- Plugging a data point, $\vec{x}$, and a parameter vector, $\vec{w}$, into $\nabla_{\vec{w}} H$ "evaluates the gradient", results in a vector, same size as $\vec{w}$.



Today

- **Backpropagation**: a strategy for computing ∇H when H is a neural network.

DSC 140B
Representation Learning

Lecture 12 | Part 2

The Chain Rule

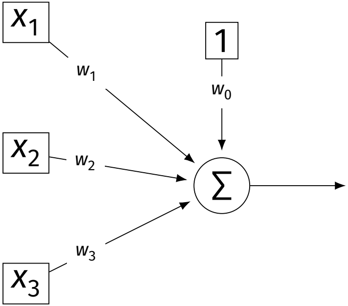
The Gradient

- The **gradient** $\nabla_{\vec{w}} H$ is the vector of partial derivatives of H with respect to each weight in $\vec{w}$:

$$\nabla_{\vec{w}} H = \left(\frac{\partial H}{\partial w_0}, \frac{\partial H}{\partial w_1}, \ldots, \frac{\partial H}{\partial w_d} \right)^T$$

- A partial derivative, $\partial H / \partial w_i$, measures the change in H due to a change in w_i.

Example: Linear Model



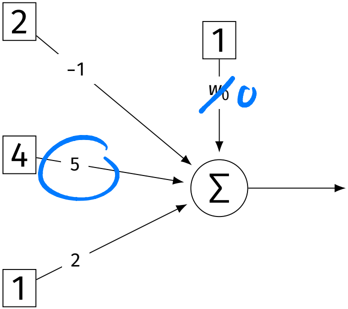
- Consider $\partial H / \partial w_1$:

$$\begin{aligned}\frac{\partial H}{\partial w_1} &= \frac{\partial}{\partial w_1}(w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d) \\ &= 0 + x_1 + 0 + \ldots + 0 \\ &= x_1\end{aligned}$$

Exercise

Suppose the input to H is $\vec{x} = (2, 4, 1)^T$.

How much does H change if we increase w_2 by 1?



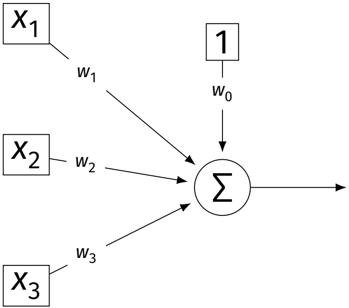
4

$-2 + 20 + 2 = 20$

Exercise

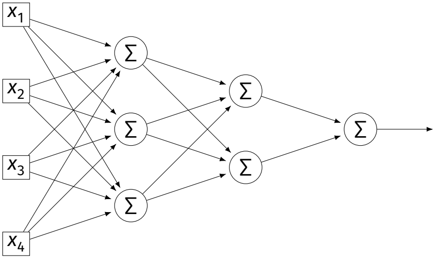
Suppose the input to H is $\vec{x} = (x_1, x_2, x_3)^T$.

What is $\partial H / \partial w_2$? $= x_2$



Neural Networks

- When H is a neural network, $\nabla_{\vec{w}} H$ is more complicated.



A “Simple” Strategy

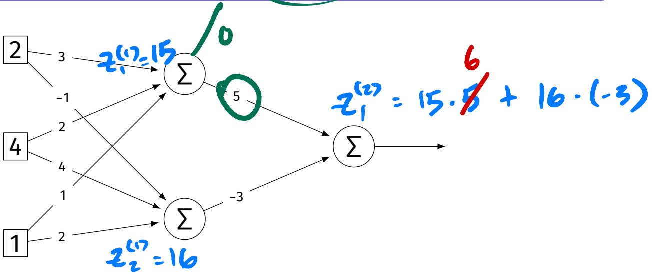
- However, there is a **simple** strategy for computing $\nabla_{\vec{w}} H$ when H is a neural network.
- We will derive it via examples.

Live Q&A

Exercise

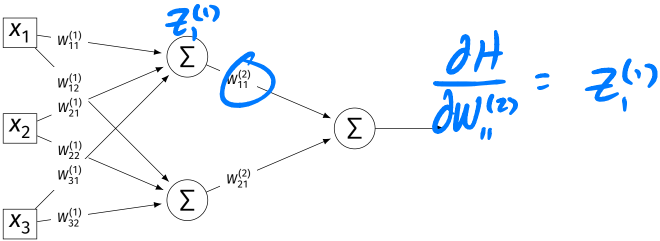
Now suppose H is the neural network shown below and $\vec{x} = (2, 4, 1)^T$. How much does H change if we increase $W_{11}^{(2)}$ by 1?

15



Exercise

Suppose H is the neural network shown below and $\vec{x} = (x_1, x_2, x_3)^T$. What is $\partial H / \partial W_{11}^{(2)}$?

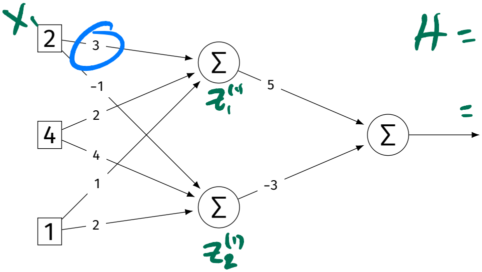


Live QA

Exercise

Suppose H is the neural network shown below and $\vec{x} = (2, 4, 1)^T$. How much does H change if we increase $W_{11}^{(1)}$ by 1?

10

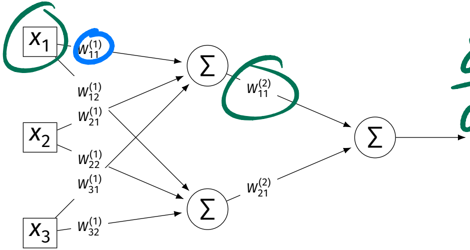


$H = 5 z_1^{(1)} - 3 z_2^{(1)}$

$= 5[\not{3}\,4 \cdot 2 + 2 \cdot 4 + 1 \cdot 1] - 3 z_2^{(1)}$

Exercise

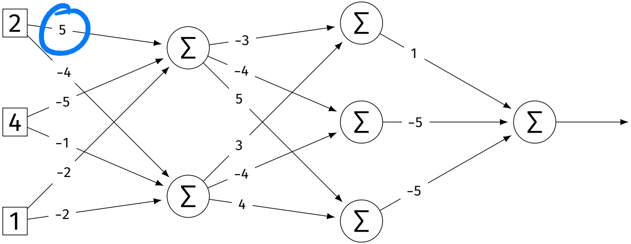
Suppose H is the neural network shown below and $\vec{x} = (x_1, x_2, x_3)^T$. What is $\partial H / \partial W_{11}^{(1)}$?



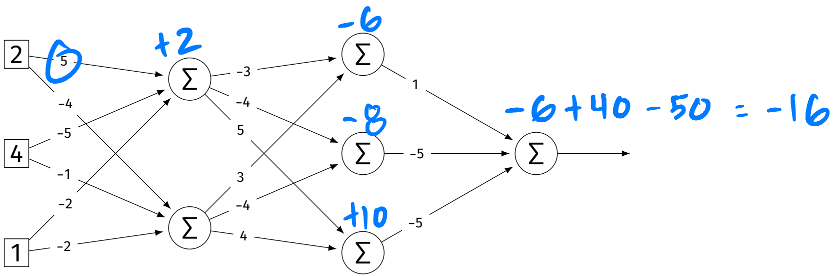
$$\frac{\partial H}{\partial W_{11}^{(1)}} = x_1 W_{11}^{(2)}$$

Exercise

Suppose H is the neural network shown below and $\vec{x} = (2, 4, 1)^T$. How much does H change if we increase $W_{11}^{(1)}$ by 1?



Solution



$\Delta H = -16$

$= 1 \cdot \Delta z_1^{(2)} + (-5)\Delta z_2^{(2)} + (-5)\Delta z_3^{(2)}$

$\Delta z_1^{(2)} = -6$

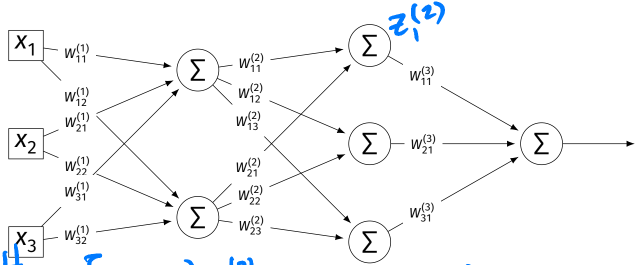
$\Delta z_2^{(2)} = -8$

$\Delta z_3^{(2)} = 10$

$\Delta z_1^{(1)} = 2$

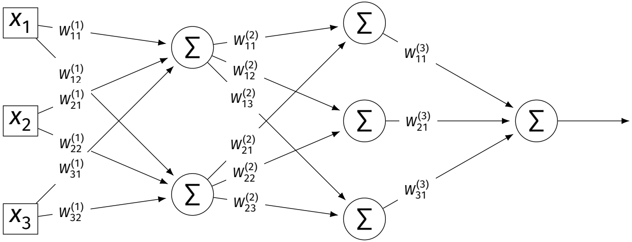
Exercise

Suppose H is the neural network shown below and $\vec{x} = (x_1, x_2, x_3)^T$. What is $\partial H / \partial W_{11}^{(1)}$?



$$\frac{\partial H}{\partial W_{11}^{(1)}} = \left[W_{11}^{(3)} \frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}} + W_{21}^{(3)} \frac{\partial z_2^{(2)}}{\partial W_{11}^{(1)}} + W_{31}^{(3)} \frac{\partial z_3^{(2)}}{\partial W_{11}^{(1)}} \right]$$

Solution



x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{12}^{(1)}$
$w_{21}^{(1)}$
$w_{22}^{(1)}$
$w_{31}^{(1)}$
$w_{32}^{(1)}$
Σ
Σ
$w_{11}^{(2)}$
$w_{12}^{(2)}$
$w_{13}^{(2)}$
$w_{21}^{(2)}$
$w_{22}^{(2)}$
$w_{23}^{(2)}$
Σ
Σ
Σ
$w_{11}^{(3)}$
$w_{21}^{(3)}$
$w_{31}^{(3)}$
Σ

Chain Rule

- We are rediscovering the **chain rule**.
- Example: if $H(x) = f_1(f_2(f_3(x)))$, then

$$\frac{\partial H}{\partial x} = \frac{\partial f_1}{\partial f_2} \cdot \frac{\partial f_2}{\partial f_3} \cdot \frac{\partial f_3}{\partial x}$$

Activations?

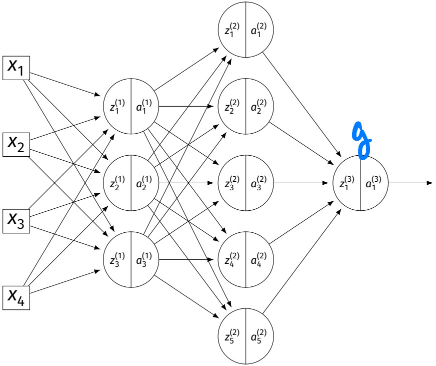
- So far, we have only considered linear activations.
- What happens if we have nonlinear activations?

$g(x)$

Example

$$H(\vec{x}) = a_1^{(3)} = g(z_1^{(3)})$$

So $$\frac{\partial H}{\partial W_{11}^{(1)}} = \frac{\partial}{\partial W_{11}^{(1)}} g(z_1^{(3)}) = g'(z_1^{(3)}) \frac{\partial}{\partial W_{11}^{(1)}} z_1^{(3)}$$



- Consider $\partial H / \partial W_{11}^{(1)}$.
- Let g be the activation function.

A Better Way

- Computing the gradient is straightforward...
- But can involve a lot of repeated work.
- **Backpropagation** is an algorithm for efficiently computing the gradient of a neural network.

DSC 140B

Representation Learning

Lecture 12 | Part 3

Backpropagation

Gradient of a Network

- We want to compute the gradient $\nabla_{\vec{w}} H$.
  - That is, $\partial H / \partial W_{ij}^{(\ell)}$ and $\partial H / \partial b_i^{(\ell)}$ for all valid i, j, ℓ.

- A network is a composition of functions.

- We'll make good use of the **chain rule**.

Recall: The Chain Rule

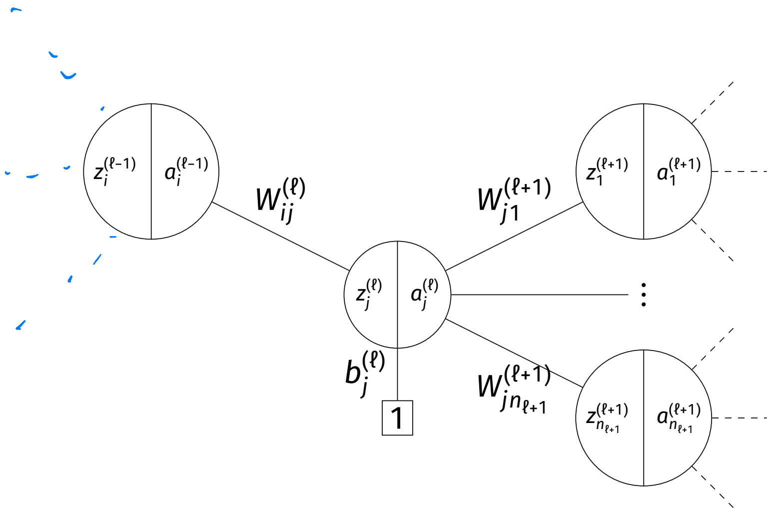
$$\frac{d}{dx} f(g(x)) = \frac{df}{dg}\frac{dg}{dx}$$

$$= f'(g(x))\, g'(x)$$

Some Notation

- We'll consider an arbitrary node in layer ℓ of a neural network.
- Let g be the activation function.
- n_ℓ denotes the number of nodes in layer ℓ.

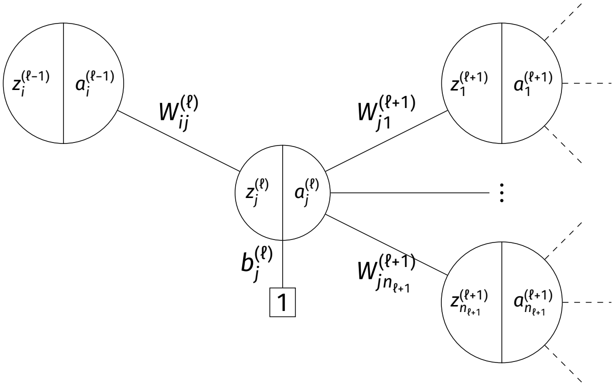
Arbitrary Node



- $\dfrac{\partial H}{\partial W_{ij}^{(\ell)}}$?
- $\dfrac{\partial H}{\partial b_j^{(\ell)}}$?

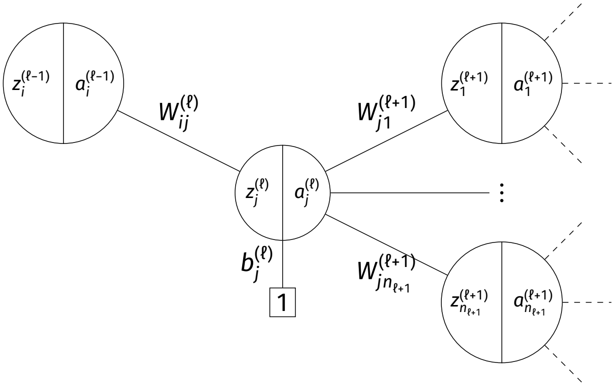
Claim #1

$$\frac{\partial H}{\partial W_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} \frac{\partial z_j^{(\ell)}}{\partial W_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} a_i^{(\ell-1)}$$



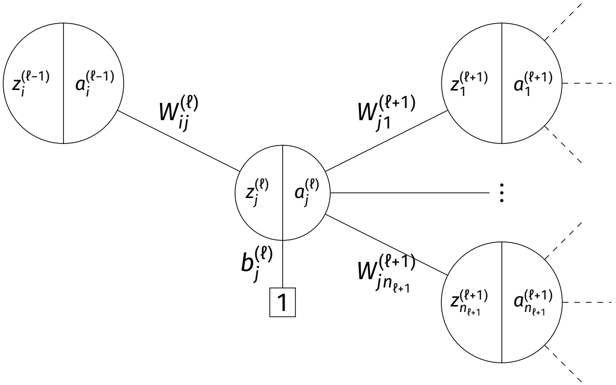
Claim #2

$$\frac{\partial H}{\partial z_j^{(\ell)}} = \frac{\partial H}{\partial a_j^{(\ell)}} \frac{\partial a_j^{(\ell)}}{\partial z_j^{(\ell)}} = \frac{\partial H}{\partial a_j^{(\ell)}} g'(z_j^{\ell})$$



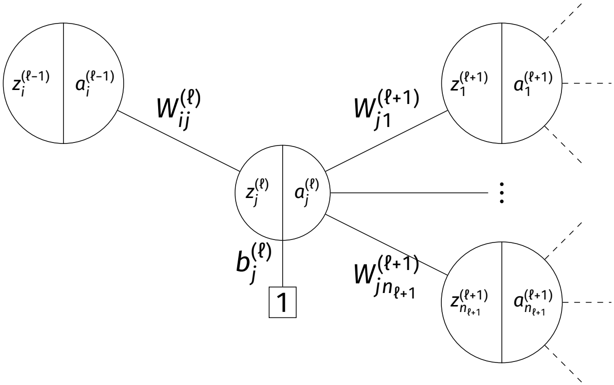
Claim #3

$$\frac{\partial H}{\partial a_j^{(\ell)}} = \sum_{k=1}^{n_{\ell+1}} \frac{\partial H}{\partial z_k^{(\ell+1)}} W_{jk}^{(\ell+1)}$$



Exercise

What is $\partial H / \partial b_j^{(\ell)}$?



General Formulas

- For any node in any neural network[1], we have the following recursive formulas:
  - $\frac{\partial H}{\partial a_j^{(\ell)}} = \sum_{k=1}^{n_{\ell+1}} \frac{\partial H}{\partial z_k^{(\ell+1)}} W_{jk}^{(\ell+1)}$
  - $\frac{\partial H}{\partial z_j^{(\ell)}} = \frac{\partial H}{\partial a_j^{(\ell)}} g'(z_j^{\ell})$
  - $\frac{\partial H}{\partial W_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} a_i^{(\ell-1)}$
  - $\frac{\partial H}{\partial b_j^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}}$

[1] Fully-connected, feedforward network

Main Idea

The derivatives in layer ℓ depend on derivatives in layer $\ell + 1$.

Backpropagation

- **Idea:** compute the derivatives in last layers, first.
- That is:
  - Compute derivatives in last layer, ℓ; store them.
  - Use to compute derivatives in layer $\ell - 1$.
  - Use to compute derivatives in layer $\ell - 2$.
  - ...

Backpropagation

Given an input $\vec{x}$ and a current parameter vector $\vec{w}$:

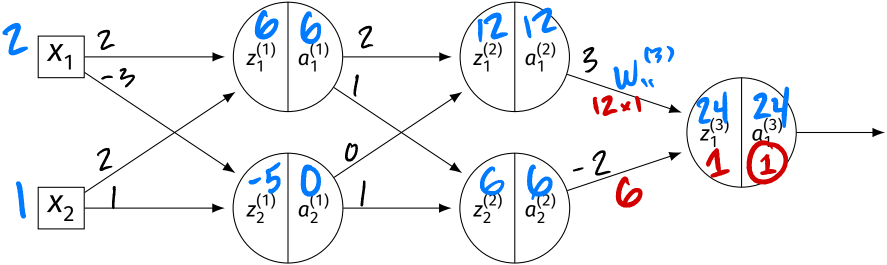
1. Evaluate the network to compute $z_i^{(\ell)}$ and $a_i^{(\ell)}$ for all nodes.
2. For each layer ℓ from last to first:
   - Compute $\frac{\partial H}{\partial a_j^{(\ell)}} = \sum_{k=1}^{n_{\ell+1}} \frac{\partial H}{\partial z_k^{(\ell+1)}} W_{jk}^{(\ell+1)}$
   - Compute $\frac{\partial H}{\partial z_j^{(\ell)}} = \frac{\partial H}{\partial a_j^{(\ell)}} g'(z_j^{\ell})$
   - Compute $\frac{\partial H}{\partial W_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} a_i^{(\ell-1)}$
   - Compute $\frac{\partial H}{\partial b_j^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}}$

This computes $\nabla_{\vec{w}} H$ and evaluates it at $\vec{x}$ and $\vec{w}$.

Example

Compute the entries of the gradient given:

$W^{(1)} = \begin{pmatrix} 2 & -3 \\ 2 & 1 \end{pmatrix}$ $W^{(2)} = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$ $W^{(3)} = \begin{pmatrix} 3 \\ -2 \end{pmatrix}$ $\vec{x} = (2, 1)^T$ $g(z) = \text{ReLU}$

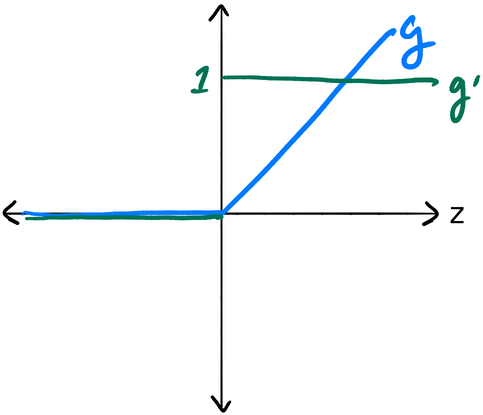


$$\frac{\partial H}{\partial a_j^{(\ell)}} = \sum_{k=1}^{n_{\ell+1}} \frac{\partial H}{\partial z_k^{(\ell+1)}} W_{jk}^{(\ell+1)} \qquad \frac{\partial H}{\partial z_j^{(\ell)}} = \frac{\partial H}{\partial a_j^{(\ell)}} g'(z_j^{\ell}) \qquad \frac{\partial H}{\partial W_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} a_i^{(\ell-1)}$$

Aside: Derivative of ReLU

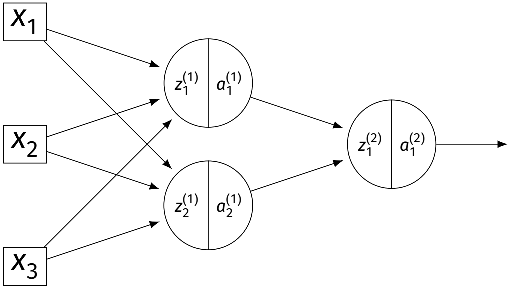
$$g(z) = \max\{0, z\}$$

$$g'(z) = \begin{cases} 0, & z < 0 \\ 1, & z > 0 \end{cases}$$



Exercise

Suppose $W_{11}^{(1)} = -2, W_{21}^{(1)} = -5, W_{31}^{(1)} = 2$ and $\vec{x} = (3, 2, -2)^T$ and all biases are 0. ReLU activations are used. What is $\partial H / \partial W_{11}^{(1)}(\vec{x}, \vec{w})$?



Summary: Backprop

- **Backprop** is an algorithm for efficiently computing the gradient of a neural network

- It is not an algorithm **you** need to carry out by hand: your NN library can do it for you.