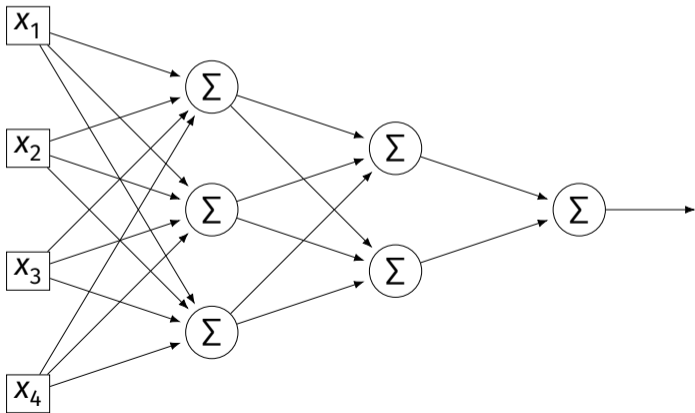dsc-capstone.org/enrollment

DSC 140B
Representation Learning

Lecture 14 | Part 1

Training Neural Networks

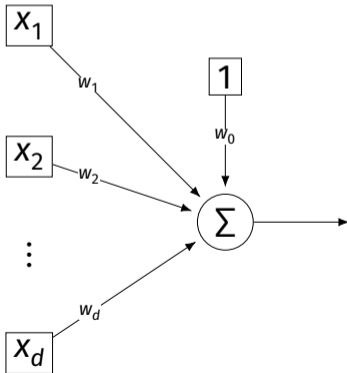# Training

▶ 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

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

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

2. 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:

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

$$(H-y)^2$$
$$2(H-y)$$

▶ 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 $\nabla_{\vec{w}} H$.

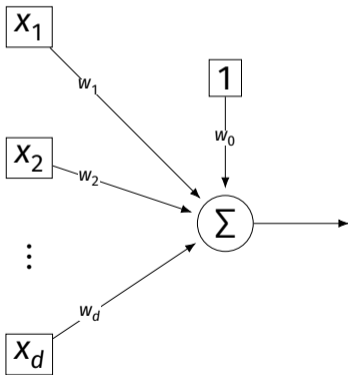► This will depend on the form of H.

Example: Linear Model

▶ Suppose *H* is a linear prediction function:

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

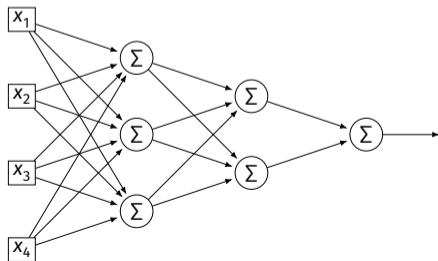▶ What is $\nabla_{\vec{w}} H$ with respect to $\vec{w}$?

Example: Linear Model

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

Example: Neural Networks

▶ Suppose *H* is a neural network (with nonlinear activations).

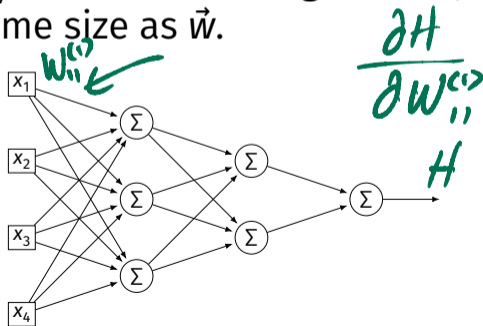▶ What is ∇*H*?
  ▶ It's more complicated...

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" $\vec{w}$.

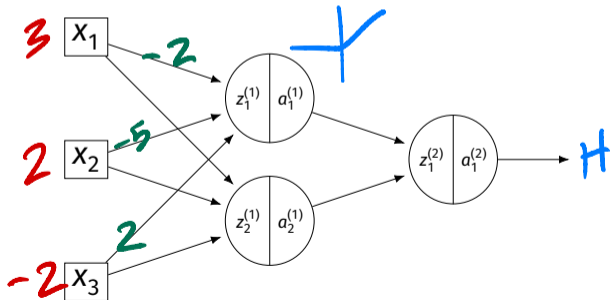# Gradient of Neural Network

▶ $\nabla_{\vec{w}} H$ 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 $\vec{w}$.

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 $\partial H/\partial W_{11}^{(1)}(\vec{x}, \vec{w})$? $= 0$
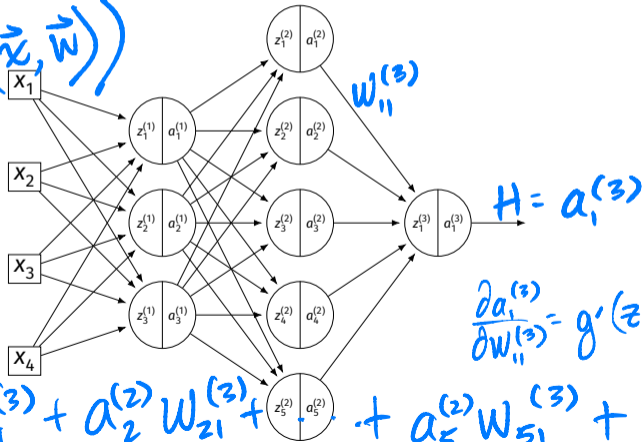
Example

$$\frac{\partial a_1^{(3)}}{\partial W_{11}^{(3)}} = g'(z_1^{(3)}) a_1^{(2)}$$

▶ Consider $\partial H / \partial W_{11}^{(3)}$:

$$a_1^{(3)} = g\left(z_1^{(3)}(\vec{x}, \vec{w})\right)$$

$$W_{11}^{(3)}$$

$$H = a_1^{(3)}$$

$$a_1^{(2)}$$

$$\frac{\partial a_1^{(3)}}{\partial W_{11}^{(3)}} = g'(z_1^{(3)}) \frac{\partial z_1^{(3)}}{\partial W_{11}^{(3)}}$$

$$z_1^{(3)} = a_1^{(2)} W_{11}^{(3)} + a_2^{(2)} W_{21}^{(3)} + \cdots + a_5^{(2)} W_{51}^{(3)} + b_1^{(3)}$$

$$\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$$

Example

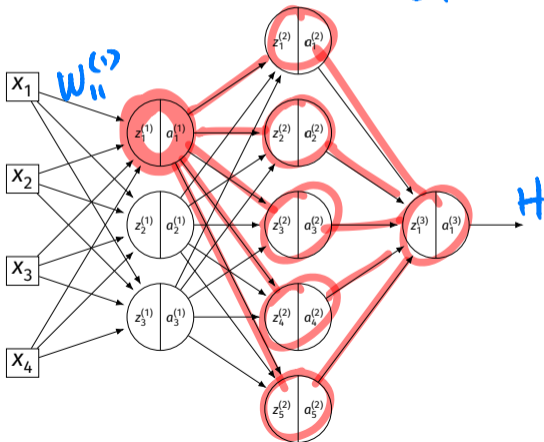▶ Consider $\partial H / \partial W_{11}^{(2)}$:

$$\frac{\partial H}{\partial W_{11}^{(2)}} = \frac{\partial H}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial W_{11}^{(2)}}$$

$$= \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}}$$

$$\times \frac{\partial z_1^{(2)}}{\partial W_{11}^{(2)}}$$

$W_{11}^{(2)}$

4
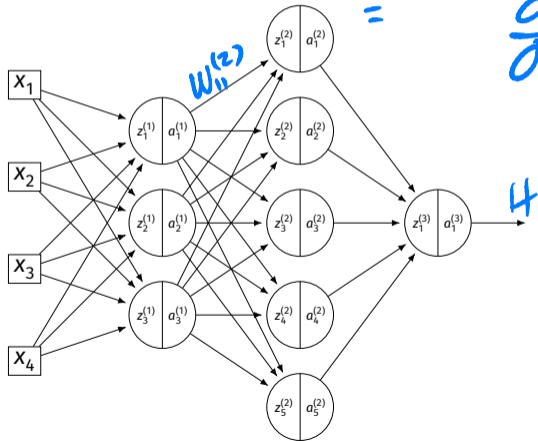
Example

▶ Consider $\partial H / \partial W_{11}^{(1)}$:

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 14 | Part 2

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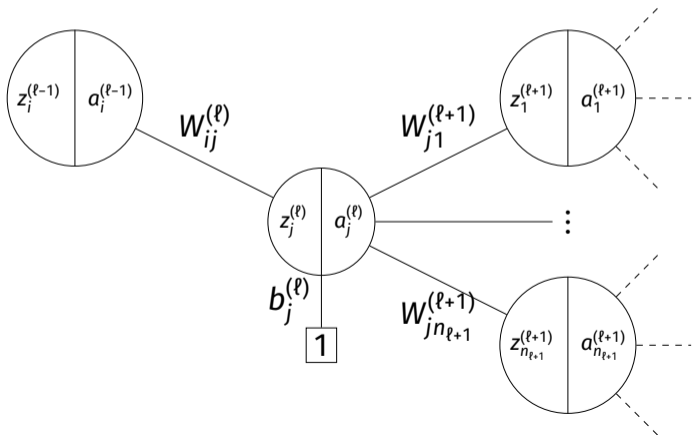
$\blacktriangleright \ \dfrac{\partial H}{\partial W_{ij}^{(\ell)}}$?

$\blacktriangleright \ \dfrac{\partial H}{\partial b_j^{(\ell)}}$?

Claim #1

$$z_j^{(\ell)} = W_{1j}^{(\ell)} a_1^{(\ell-1)} + W_{2j}^{(\ell)} a_2^{(\ell-1)} + \cdots + W_{n_{\ell-1}j}^{(\ell)} a_{n_{\ell-1}}^{(\ell-1)} + b_j^{(\ell)}$$

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

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

$$a_i^{(\ell-1)}$$
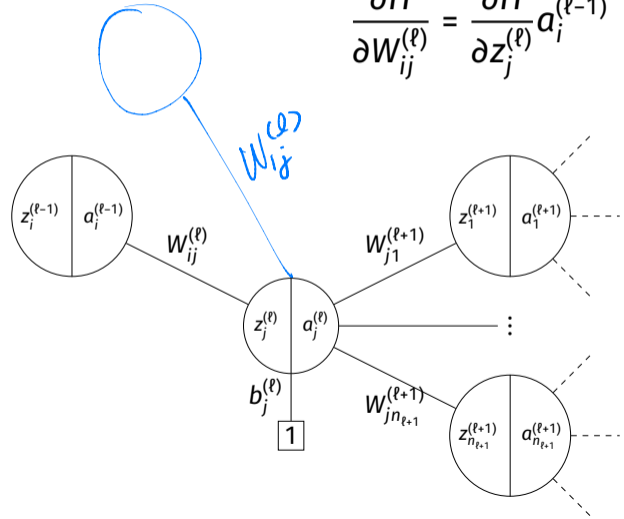
Claim #2

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

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

$$a_j^{(\ell)} = g\left(z_j^{(\ell)}\right)$$

$$\frac{\partial a_j^{(\ell)}}{\partial z_j^{(\ell)}} = \frac{\partial}{\partial z_j^{(\ell)}} g\left(z_j^{(\ell)}\right)$$

$$= g'\left(z_j^{(\ell)}\right)$$
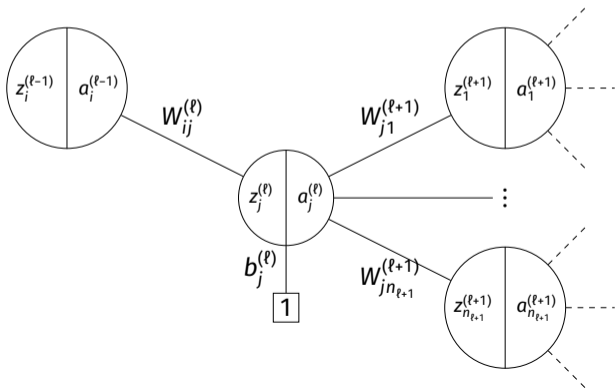
Claim #3

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

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

$$W_{jk}^{(\ell+1)}$$
$$=$$

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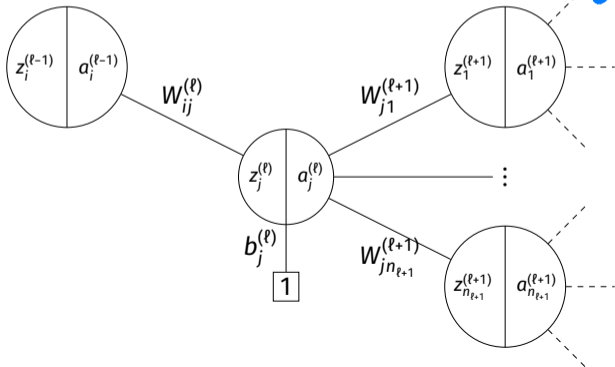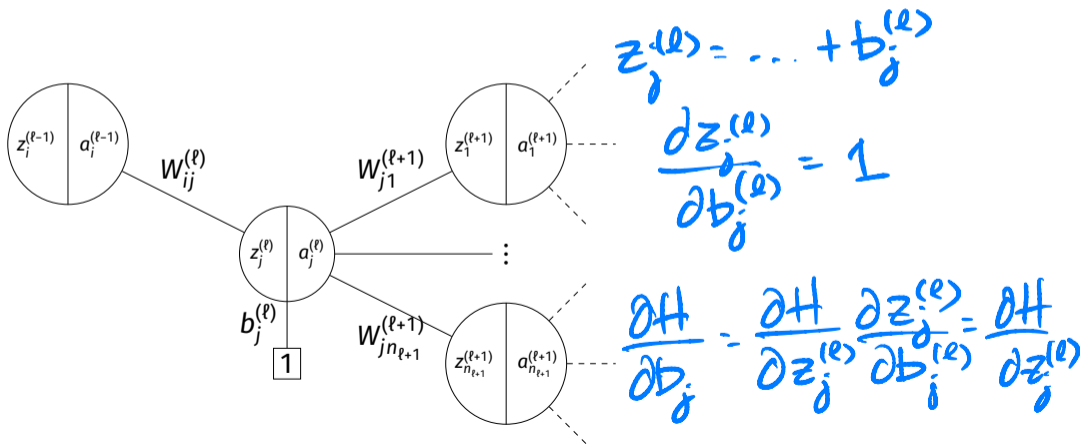
What is $\partial H / \partial b_j^{(\ell)}$?

$$z_j^{(\ell)} = \ldots + b_j^{(\ell)}$$

$$\frac{\partial z_j^{(\ell)}}{\partial b_j^{(\ell)}} = 1$$

$$\frac{\partial H}{\partial b_j} = \frac{\partial H}{\partial z_j^{(\ell)}} \frac{\partial z_j^{(\ell)}}{\partial b_j^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}}$$

Diagram labels:
- $z_i^{(\ell-1)} \;\; a_i^{(\ell-1)}$
- $W_{ij}^{(\ell)}$
- $z_j^{(\ell)} \;\; a_j^{(\ell)}$
- $b_j^{(\ell)}$ — 1
- $W_{j1}^{(\ell+1)}$
- $z_1^{(\ell+1)} \;\; a_1^{(\ell+1)}$
- $W_{jn_{\ell+1}}^{(\ell+1)}$
- $z_{n_{\ell+1}}^{(\ell+1)} \;\; a_{n_{\ell+1}}^{(\ell+1)}$

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 ℓ – 1.
  ▶ Use to compute derivatives in layer ℓ – 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 $\frac{\partial H}{\partial W_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} a_i^{(\ell-1)}$
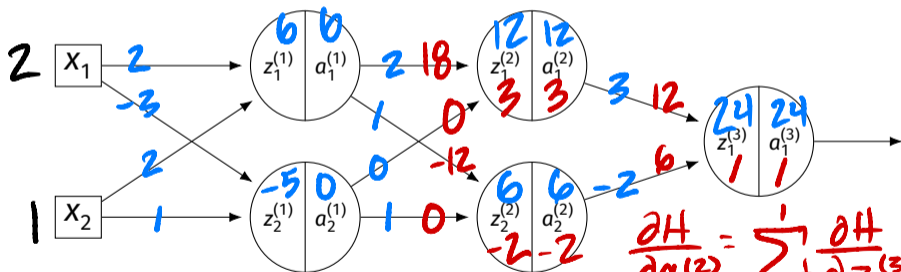   - Compute $\frac{\partial H}{\partial b_j^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}}$

Example

$H = a_1^{(3)}$

Compute the entries of the gradient given:

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

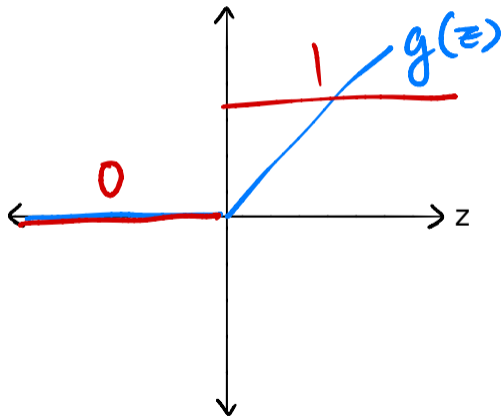$$\frac{\partial H}{\partial a_j^{(\ell)}} = \sum_{k=1}^{n_{\ell+1}} \frac{\partial H}{\partial z_k^{(\ell+1)}} W_{jk}^{(\ell+1)} \qquad \frac{\partial H}{\partial z_j^{(\ell)}} = \frac{\partial H}{\partial a_j^{(\ell)}} g'(z_j^\ell) \qquad \frac{\partial H}{\partial W_{ij}^{(\ell)}} = \frac{\partial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

$$\frac{\partial H}{\partial a_1^{(2)}} = \sum_{k=1}^{1} \frac{\partial H}{\partial z_k^{(3)}} W_{1k}^{(3)}$$
$$= 1 \times W_{11}^{(3)}$$
$$= 1 \times 3 = 3$$

Aside: Derivative of ReLU

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

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

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.