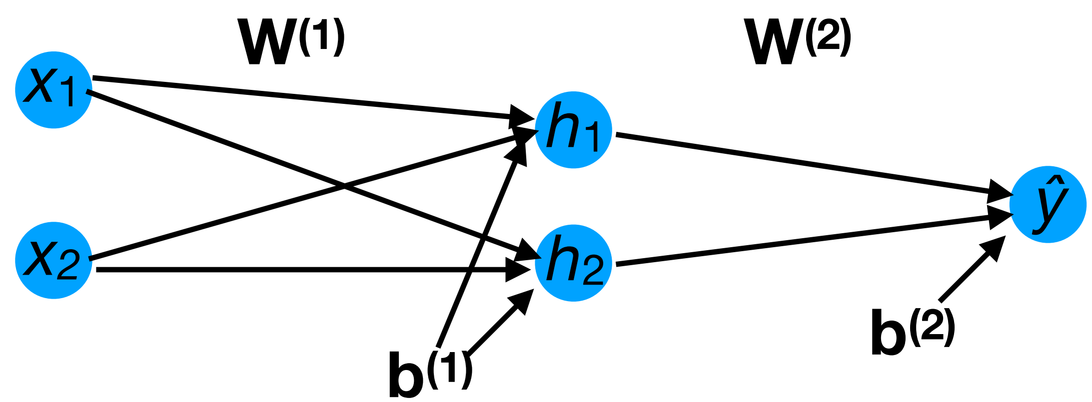
CS 4342: Class 20

Jacob Whitehill

Gradient descent: XOR problem

- Here is how we can conduct gradient descent for the XOR problem…



- Let's first define:

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}, \mathbf{W}^{(2)} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix}, \mathbf{b}^{(1)} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \mathbf{b}^{(2)} = \begin{bmatrix} b_3 \end{bmatrix}$$

- Then we can define *g* so that:

$$\begin{aligned} \hat{y} = g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) &= \mathbf{W}^{(2)}\sigma\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)} \\ &= \begin{bmatrix} w_5 \\ w_6 \end{bmatrix}^\mathsf{T} \sigma\left(\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}\right) + b_3 \\ &= w_5\sigma(w_1x_1 + w_2x_2 + b_1) + w_6\sigma(w_3x_1 + w_4x_2 + b_2) + b_3 \end{aligned}$$

Gradient descent: XOR problem

- From $\hat{y}$, we can compute the f_{MSE} cost as:

$$f_{\text{MSE}}(\hat{y}; w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3) = \frac{1}{2}(\hat{y} - y)^2$$

- We then calculate the derivative of f_{MSE} w.r.t. each parameter p using the chain rule as:

$$\frac{\partial f_{\text{MSE}}}{\partial p} = \frac{\partial f_{\text{MSE}}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial p}$$

where:

$$\frac{\partial f_{\text{MSE}}}{\partial \hat{y}} = (\hat{y} - y)$$

Gradient descent: XOR problem

- Now we just have to differentiate $\hat{y} = g(\mathbf{x})$ w.r.t each parameter *p*:, e.g.:

$$
\begin{aligned}
\frac{\partial \hat{y}}{\partial w_1} &= \frac{\partial}{\partial w_1}\left[w_5\sigma(w_1x_1 + w_2x_2 + b_1) + w_6\sigma(w_3x_1 + w_4x_2 + b_2) + b_3\right] \\
&= w_5\sigma'(w_1x_1 + w_2x_2 + b_1)x_1
\end{aligned}
$$

where:

$$
\sigma'(z) = \begin{cases} 0 & \text{if} \quad z \leq 0 \\ 1 & \text{if} \quad z > 0 \end{cases}
$$ **for ReLU**

Gradient descent: XOR problem

- Hence:

$$\frac{\partial \hat{y}}{\partial w_1} = \begin{cases} 0 & \text{if} \quad w_1 x_1 + w_2 x_2 + b_1 \leq 0 \\ w_5 x_1 & \text{if} \quad w_1 x_1 + w_2 x_2 + b_1 > 0 \end{cases}$$

since:

$$\sigma'(z) = \begin{cases} 0 & \text{if} \quad z \leq 0 \\ 1 & \text{if} \quad z > 0 \end{cases}$$ **for ReLU**

Gradient descent: XOR problem

- We need to derive all the other partial derivatives:

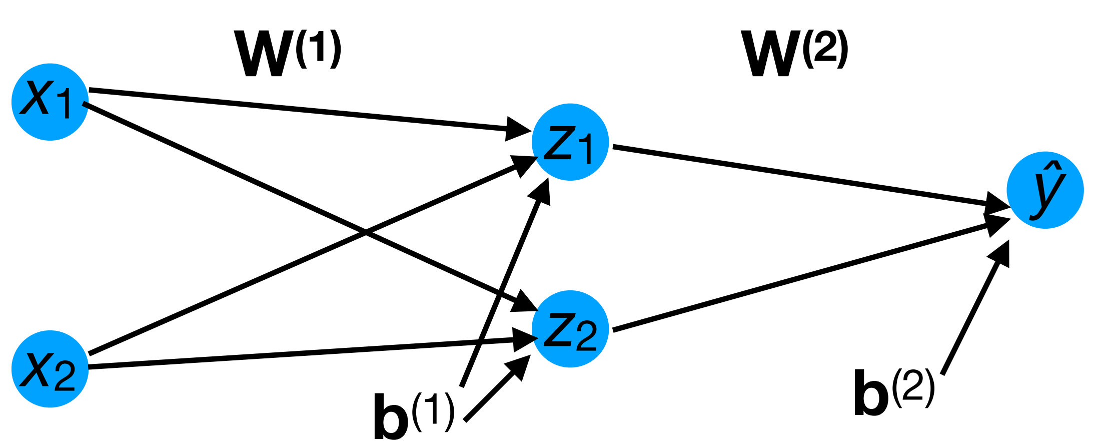
$$\frac{\partial \hat{y}}{\partial w_1}, \ldots \frac{\partial \hat{y}}{\partial w_6}, \frac{\partial \hat{y}}{\partial b_1}, \ldots \frac{\partial \hat{y}}{\partial w_3}$$

- Based on these, we can compute the gradient terms:

$$\frac{\partial f_{\mathrm{MSE}}}{\partial w_1} = \frac{\partial f_{\mathrm{MSE}}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_1} \qquad \frac{\partial f_{\mathrm{MSE}}}{\partial b_1} = \frac{\partial f_{\mathrm{MSE}}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b_1}$$

... ...

$$\frac{\partial f_{\mathrm{MSE}}}{\partial w_6} = \frac{\partial f_{\mathrm{MSE}}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_6} \qquad \frac{\partial f_{\mathrm{MSE}}}{\partial b_3} = \frac{\partial f_{\mathrm{MSE}}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b_3}$$



Gradient descent: XOR problem

- Given the gradients, we can conduct gradient descent:
  - For each epoch:
    - For each minibatch:
      - Compute all 9 partial derivatives (summed over all examples in the minibatch):
      - Update w_1: $w_1 \leftarrow w_1 - \epsilon \frac{\partial f_{\mathrm{MSE}}}{\partial w_1}$
        …
        Update w_6: $w_6 \leftarrow w_6 - \epsilon \frac{\partial f_{\mathrm{MSE}}}{\partial w_6}$

        Update b_1: $b_1 \leftarrow b_1 - \epsilon \frac{\partial f_{\mathrm{MSE}}}{\partial b_1}$
        …
        Update b_3: $b_3 \leftarrow b_3 - \epsilon \frac{\partial f_{\mathrm{MSE}}}{\partial b_3}$

Gradient descent: (any 3-layer NN)

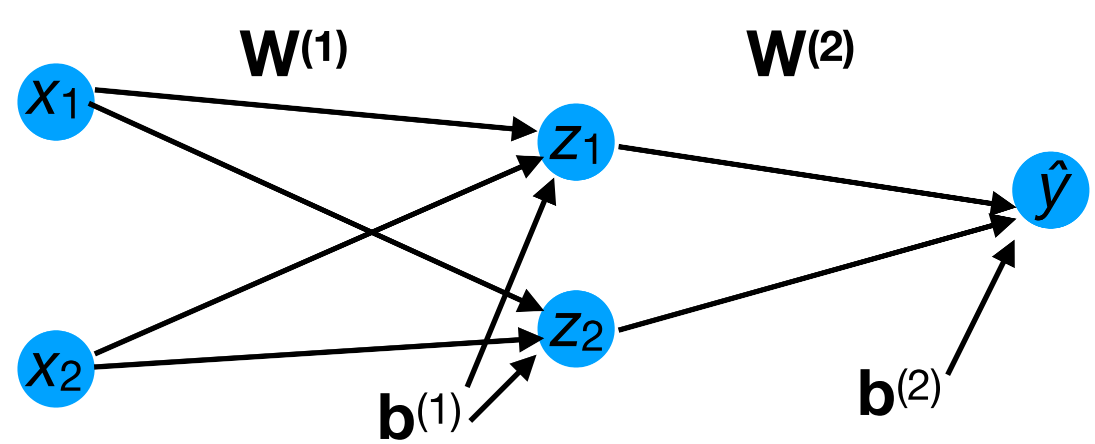
- It is more compact and efficient to represent and compute these gradients more abstractly:

$$\nabla_{\mathbf{W}^{(1)}} f_{\text{MSE}}$$
$$\nabla_{\mathbf{W}^{(2)}} f_{\text{MSE}}$$
$$\nabla_{\mathbf{b}^{(1)}} f_{\text{MSE}}$$
$$\nabla_{\mathbf{b}^{(2)}} f_{\text{MSE}}$$

where

$$\nabla_{\mathbf{W}^{(1)}} f_{\text{MSE}} = \begin{bmatrix} \frac{\partial f_{\text{MSE}}}{\partial w_1} & \frac{\partial f_{\text{MSE}}}{\partial w_2} \\ \frac{\partial f_{\text{MSE}}}{\partial w_3} & \frac{\partial f_{\text{MSE}}}{\partial w_4} \end{bmatrix}$$

etc.



Gradient descent (any 3-layer NN)

- Given the gradients, we can conduct gradient descent:
  - For each epoch:
    - For each minibatch:
      - Compute all gradient terms (summed over all examples in the minibatch):
      - Update **W**$^{(1)}$: $\mathbf{W}^{(1)} \leftarrow \mathbf{W}^{(1)} - \epsilon \nabla_{\mathbf{W}^{(1)}} f_{\mathrm{MSE}}$

        Update **W**$^{(2)}$: $\mathbf{W}^{(2)} \leftarrow \mathbf{W}^{(2)} - \epsilon \nabla_{\mathbf{W}^{(2)}} f_{\mathrm{MSE}}$

        Update **b**$^{(1)}$: $\mathbf{b}^{(1)} \leftarrow \mathbf{b}^{(1)} - \epsilon \nabla_{\mathbf{b}^{(1)}} f_{\mathrm{MSE}}$

        Update **b**$^{(2)}$: $\mathbf{b}^{(2)} \leftarrow \mathbf{b}^{(2)} - \epsilon \nabla_{\mathbf{b}^{(2)}} f_{\mathrm{MSE}}$

Deriving the gradient terms

- In the XOR problem, we derived each element of each gradient term by hand.
- For large networks with many layers, this would be prohibitively tedious.
- Fortunately, we can largely *automate* the process of computing each gradient term $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, …, $\mathbf{W}^{(d)}$ (for d layers).
- This procedure is enabled by the **chain rule of multivariate calculus**…

Jacobian matrices & the chain rule of multivariate calculus

Jacobian matrices

- Consider a function *f* that takes a vector as input *and produces a vector as output*, e.g.:

$$f\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right) = \left[\begin{array}{c} 2x_1 + x_2 \\ \pi x_2 \\ -x_1/x_2 \end{array}\right]$$

- What is the set of *f*'s partial derivatives?

Jacobian matrices

- For any function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$, we can define the **Jacobian matrix** of all partial derivatives:

Columns are the *inputs* to *f*.

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_1}{\partial \mathbf{x}_m} \\ & \ddots & \\ \frac{\partial f_n}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_n}{\partial \mathbf{x}_m} \end{bmatrix}$$

Row are the *outputs* of *f*.

f: $R^m \rightarrow R$:
Jacobian versus gradient

- Note, for a real-valued function $f : \mathbb{R}^m \rightarrow \mathbb{R}$, the Jacobian matrix is the transpose of the gradient.

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial \mathbf{x}_1} \\ \vdots \\ \frac{\partial f}{\partial \mathbf{x}_m} \end{bmatrix}$$

Gradient

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial \mathbf{x}_1} & \dots & \frac{\partial f}{\partial \mathbf{x}_m} \end{bmatrix}$$

Jacobian

Chain rule of multivariate calculus

- Suppose $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $g : \mathbb{R}^k \rightarrow \mathbb{R}^m$.
- Then $\frac{\partial (f \circ g)}{\partial \mathbf{x}} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial \mathbf{x}}$

Jacobian of *f*.

Chain rule of multivariate calculus

- Suppose $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $g : \mathbb{R}^k \rightarrow \mathbb{R}^m$.
- Then $\frac{\partial (f \circ g)}{\partial \mathbf{x}} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial \mathbf{x}}$

Jacobian of g.

Chain rule of multivariate calculus

- Suppose $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $g : \mathbb{R}^k \rightarrow \mathbb{R}^m$.
- Then $\frac{\partial (f \circ g)}{\partial \mathbf{x}} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial \mathbf{x}}$
- Note that the order matters!, i.e.:

$$\frac{\partial (f \circ g)}{\partial \mathbf{x}} \neq \frac{\partial g}{\partial \mathbf{x}} \frac{\partial f}{\partial g}$$

Chain rule of multivariate calculus: example

- Suppose:

$$f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \begin{bmatrix} x_1 - 2x_2 + x_3/4 \end{bmatrix} \quad g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 + 2x_2 \\ x_2 \\ -x_1 + 3 \end{bmatrix}$$

Chain rule of multivariate calculus: example

- Suppose:

$$f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \begin{bmatrix} x_1 - 2x_2 + x_3/4 \end{bmatrix} \quad g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 + 2x_2 \\ x_2 \\ -x_1 + 3 \end{bmatrix}$$

- What is $\frac{\partial (f \circ g)}{\partial \mathbf{x}}$? Two alternative methods:

1. Substitute *g* into *f* and differentiate.

2. Apply chain rule.

Chain rule of multivariate calculus: example

$$f\left(\left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array}\right]\right) = \left[\begin{array}{c} x_1 - 2x_2 + x_3/4 \end{array}\right] \quad g\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right) = \left[\begin{array}{c} x_1 + 2x_2 \\ x_2 \\ -x_1 + 3 \end{array}\right]$$

- Substitution:

$$f \circ g\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right) \quad = \quad f\left(g\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right)\right)$$

Exercise

- What is the Jacobian of the following function?

$$f(\mathbf{w}) = f(w_1, w_2) = \begin{bmatrix} w_1/w_2 \\ -w_2 \\ \log w_1 \end{bmatrix}$$

Jacobian matrices

- Note that we sometimes examine the *same* function from different perspectives, e.g.:

$$f(\mathbf{x}; \mathbf{w}) = f(x, y; a, b) = 2ax + b/y$$

Jacobian matrices

- Note that we sometimes examine the *same* function from different perspectives, e.g.:

$$f(\mathbf{x}; \mathbf{w}) = f(x, y; a, b) = 2ax + b/y$$

  - We can consider *x*, *y* to be the variables and *a*, *b* to be constants.

$$\frac{\partial f}{\partial \mathbf{x}} = \left[\begin{array}{cc} 2a & -b/y^2 \end{array} \right]$$

Jacobian matrices

- Note that we sometimes examine the *same* function from different perspectives, e.g.:

$$f(\mathbf{x}; \mathbf{w}) = f(x, y; a, b) = 2ax + b/y$$

- We can consider *a*, *b* to be the variables and *x*, *y* to be constants.

$$\frac{\partial f}{\partial \mathbf{w}} = \left[\begin{array}{cc} 2x & 1/y \end{array} \right]$$

Vectorizing a matrix

- Sometimes we need to differentiate a *vector*-valued function f w.r.t. a *matrix* of its parameters, e.g.:

$$f(\mathbf{x}; \mathbf{W}) = \left[\begin{array}{cc} w_1 & w_2 \\ w_3 & w_4 \end{array} \right] \left[\begin{array}{c} x_1 \\ x_2 \end{array} \right] = \left[\begin{array}{c} w_1 x_1 + w_2 x_2 \\ w_3 x_1 + w_4 x_2 \end{array} \right]$$

Vectorizing a matrix

- Sometimes we need to differentiate a *vector*-valued function f w.r.t. a *matrix* of its parameters, e.g.:

$$f(\mathbf{x}; \mathbf{W}) = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_1 x_1 + w_2 x_2 \\ w_3 x_1 + w_4 x_2 \end{bmatrix}$$

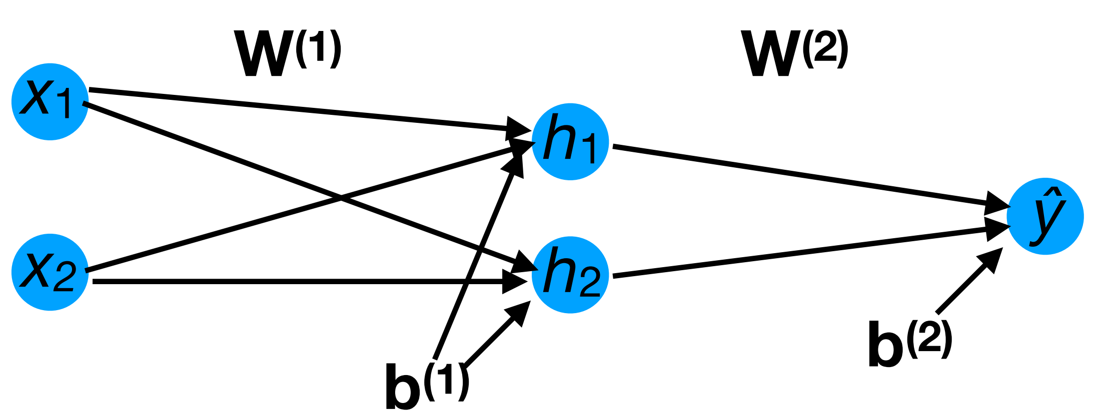
- We define:

$$\frac{\partial f}{\partial \text{vec}[\mathbf{W}]}(\mathbf{W}) = \begin{bmatrix} \frac{\partial f_1}{\partial w_1} & \frac{\partial f_1}{\partial w_2} & \frac{\partial f_1}{\partial w_3} & \frac{\partial f_1}{\partial w_4} \\ \frac{\partial f_2}{\partial w_1} & \frac{\partial f_2}{\partial w_2} & \frac{\partial f_2}{\partial w_3} & \frac{\partial f_2}{\partial w_4} \end{bmatrix}$$

$$= \begin{bmatrix} x_1 & x_2 & 0 & 0 \\ 0 & 0 & x_1 & x_2 \end{bmatrix}$$

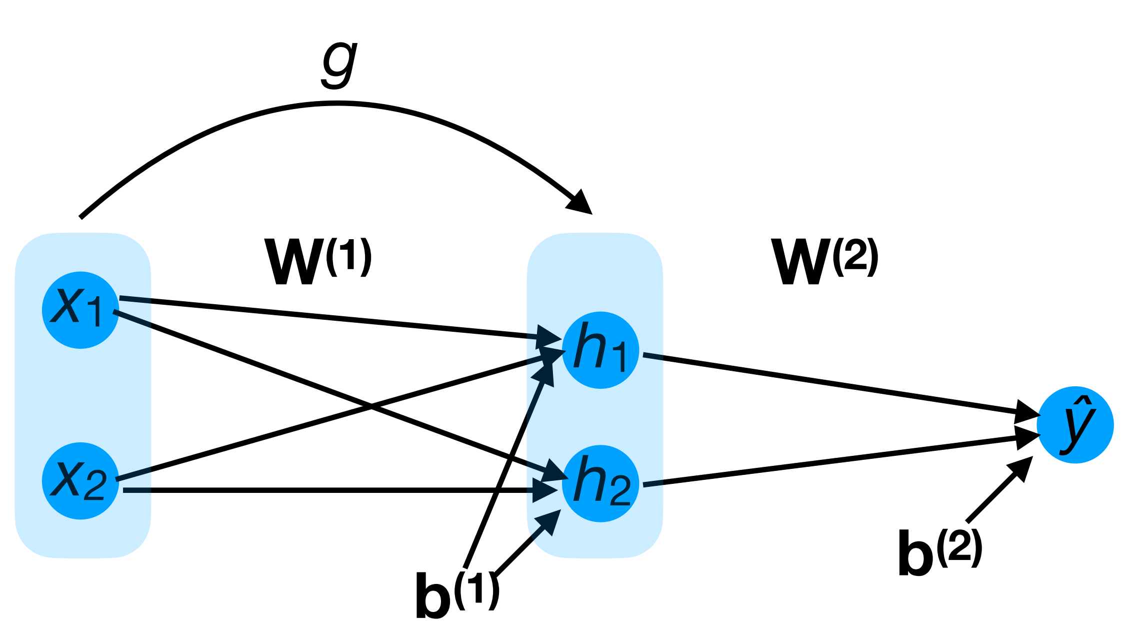
Neural networks as function compositions

- We can think of a NN as a **composition of functions**.
- Each layer is a function of the previous layer:



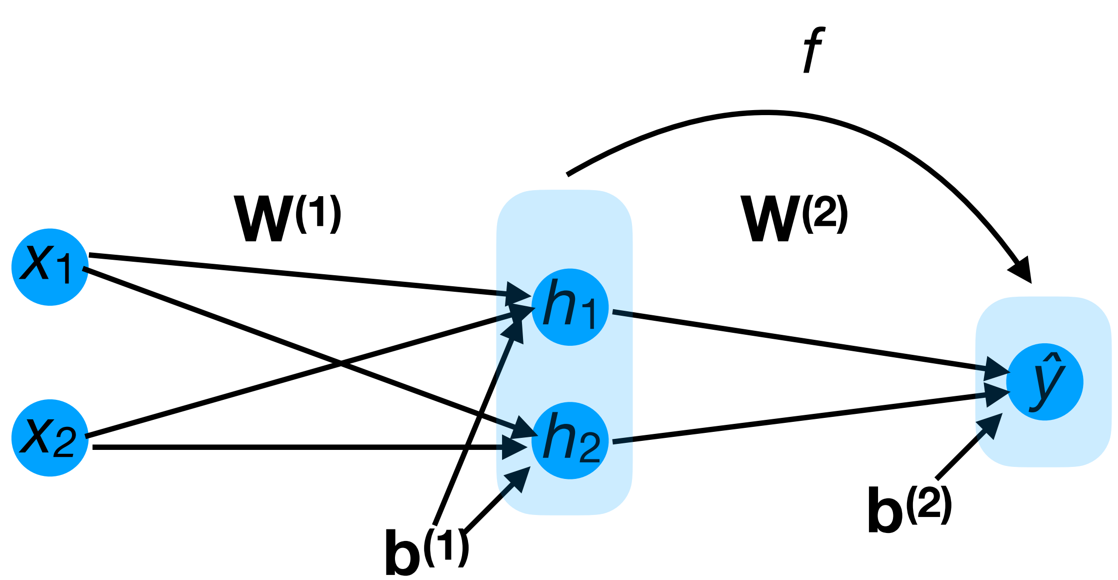
Neural networks as function compositions

- We can think of a NN as a **composition of functions**.
- Each layer is a function of the previous layer:
  - The hidden layer **h** is a function *g* of **x**.



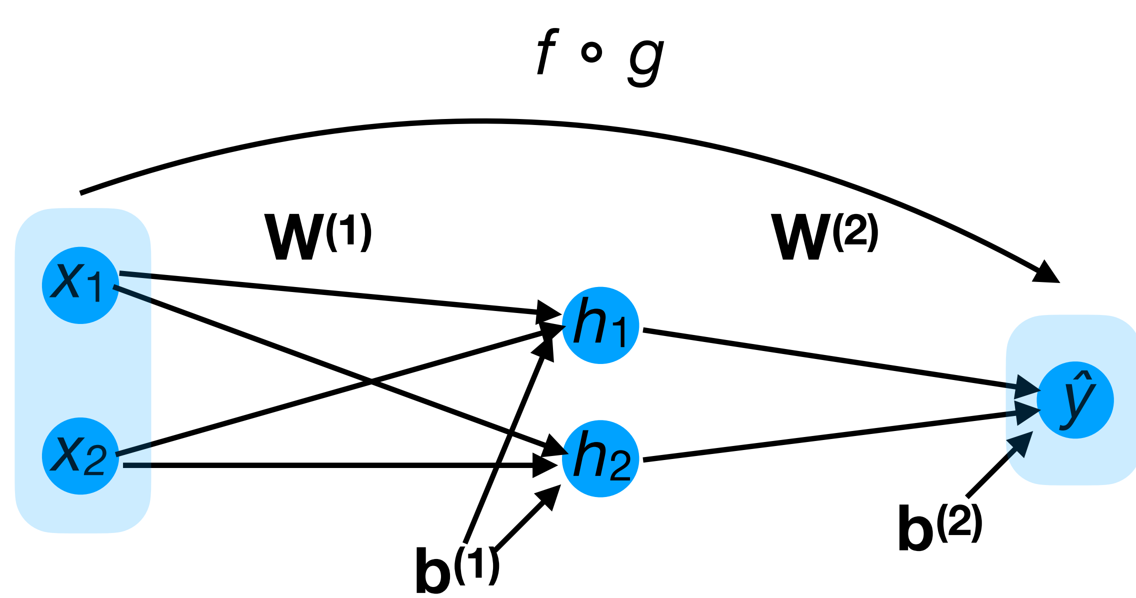
Neural networks as function compositions

- We can think of a NN as a **composition of functions**.
- Each layer is a function of the previous layer:
  - The hidden layer **h** is a function *g* of **x**.
  - The final layer $\hat{y}$ is a function *f* of **h**.



Neural networks as function compositions

- We can think of a NN as a **composition of functions**.
- Each layer is a function of the previous layer:
  - The hidden layer $\mathbf{h}$ is a function g of $\mathbf{x}$.
  - The final layer $\hat{y}$ is a function f of $\mathbf{h}$.
- Hence, $\hat{y} = f(g(\mathbf{x}))$.



Neural networks as function compositions

- For a function *f* that depends *indirectly* on **x**:

  $$f(g(h(\ldots(\mathbf{x})\ldots))$$

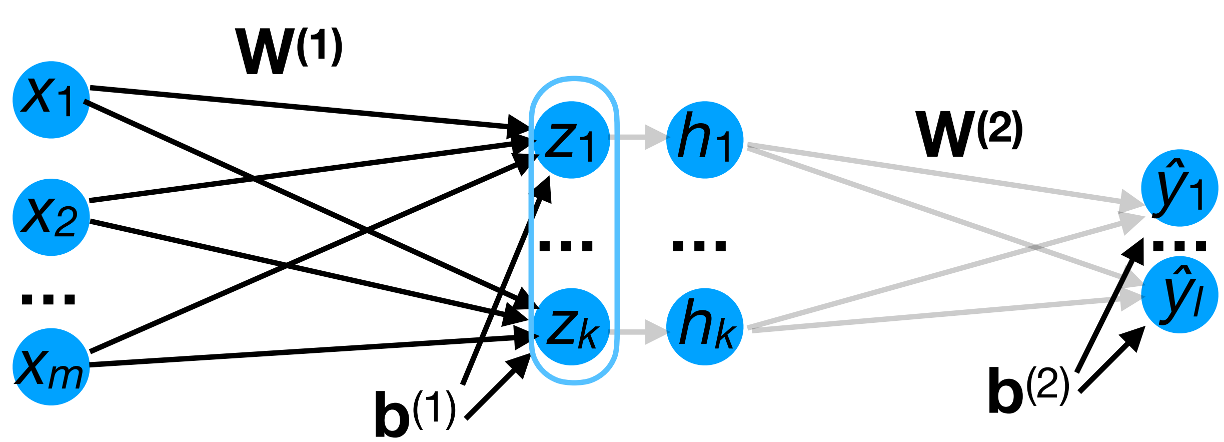
  we can compute its derivative w.r.t. **x** using the chain rule of multi-variate calculus as the product of Jacobians:

  $$\frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\ldots\frac{\partial \ldots}{\partial \mathbf{x}}$$

Forwards and backwards propagation

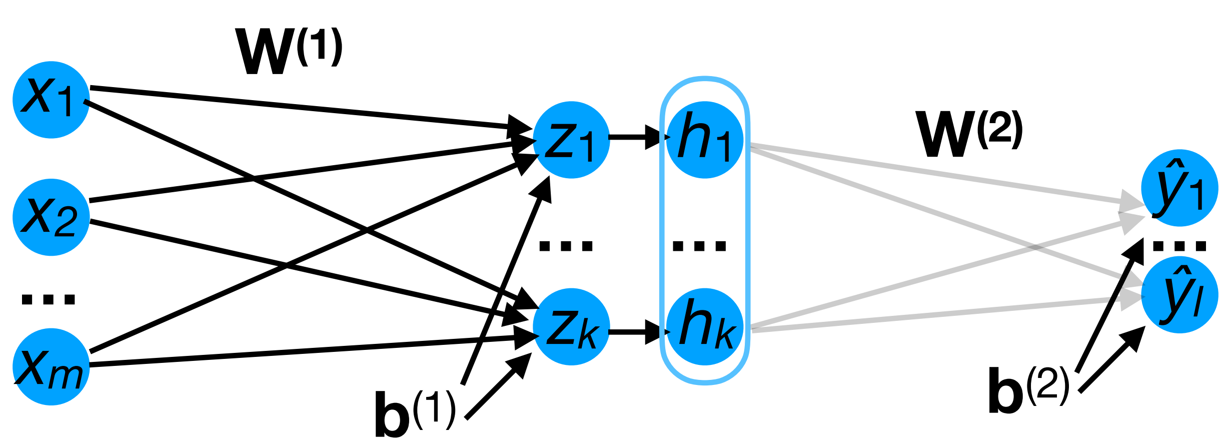
Computing the gradients

- Consider the 3-layer NN below:
  - From **x**, **W**$^{(1)}$, and **b**$^{(1)}$, we can compute **z**.



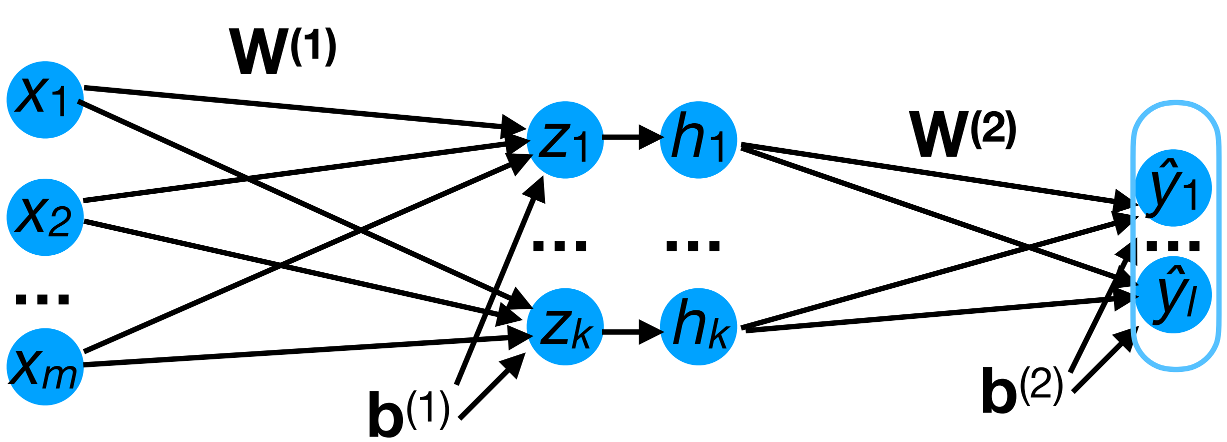
Computing the gradients

- Consider the 3-layer NN below:
  - From $\mathbf{x}$, $\mathbf{W}^{(1)}$, and $\mathbf{b}^{(1)}$, we can compute $\mathbf{z}$.
  - From $\mathbf{z}$ and σ, we can compute $\mathbf{h} = \sigma(\mathbf{z})$.



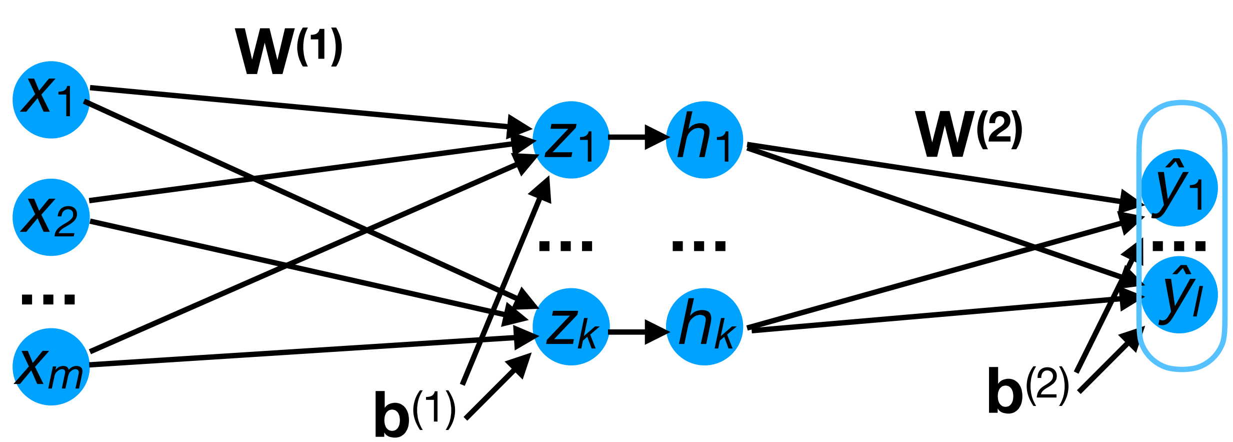
Computing the gradients

- Consider the 3-layer NN below:
  - From $\mathbf{x}$, $\mathbf{W}^{(1)}$, and $\mathbf{b}^{(1)}$, we can compute $\mathbf{z}$.
  - From $\mathbf{z}$ and σ, we can compute $\mathbf{h} = \sigma(\mathbf{z})$.
  - From $\mathbf{h}$, $\mathbf{W}^{(2)}$, and $\mathbf{b}^{(2)}$, we can compute $\hat{\mathbf{y}}$.



Computing the gradients

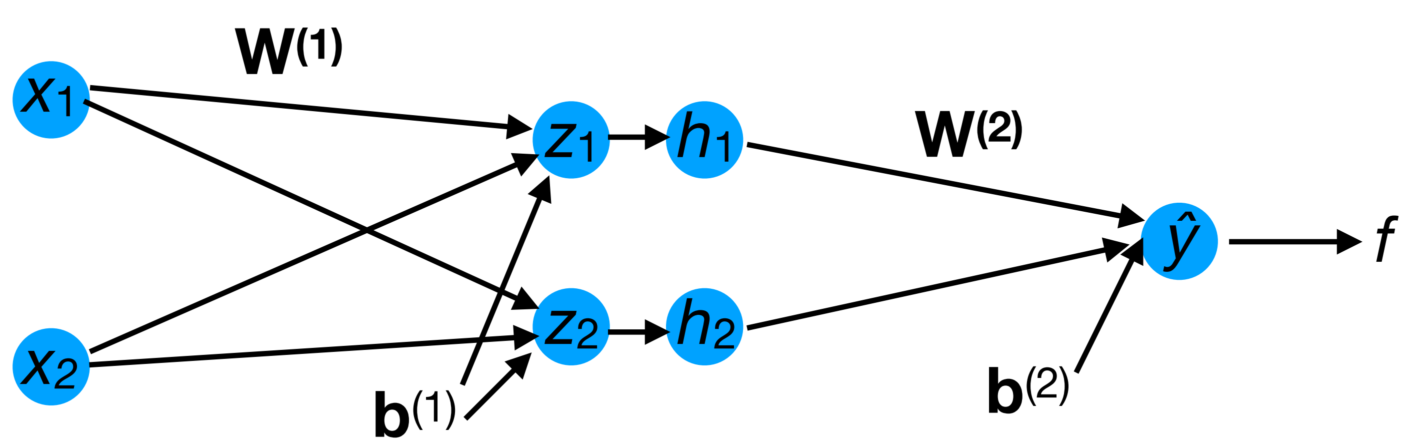
- This process is known as **forward propagation**.
  - It produces all the intermediary (**h**, **z**) and final (**ŷ**) network outputs.



Computing the gradients

- Now, let's look at how to compute each gradient term:

$$
\begin{aligned}
\frac{\partial f}{\partial \mathbf{W}^{(2)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{W}^{(2)}} \\
\frac{\partial f}{\partial \mathbf{b}^{(2)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{b}^{(2)}} \\
\frac{\partial f}{\partial \mathbf{W}^{(1)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{W}^{(1)}} \\
\frac{\partial f}{\partial \mathbf{b}^{(1)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{b}^{(1)}}
\end{aligned}
$$

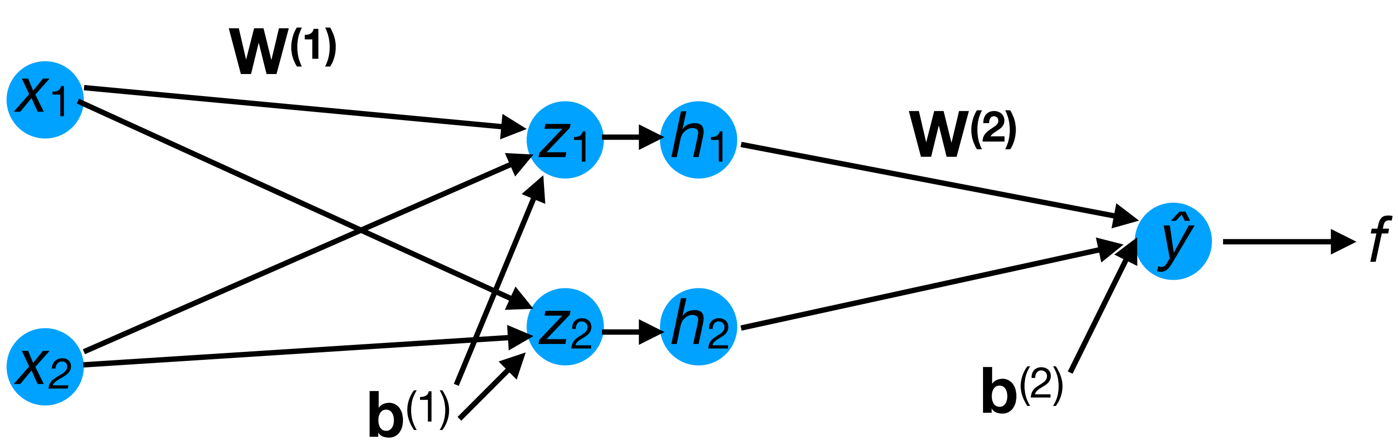


Computing the gradients

- Now, let's look at how to compute each gradient term:

$$
\begin{aligned}
\frac{\partial f}{\partial \mathbf{W}^{(2)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{W}^{(2)}} \\
\frac{\partial f}{\partial \mathbf{b}^{(2)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{b}^{(2)}} \\
\frac{\partial f}{\partial \mathbf{W}^{(1)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{W}^{(1)}} \\
\frac{\partial f}{\partial \mathbf{b}^{(1)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{b}^{(1)}}
\end{aligned}
$$

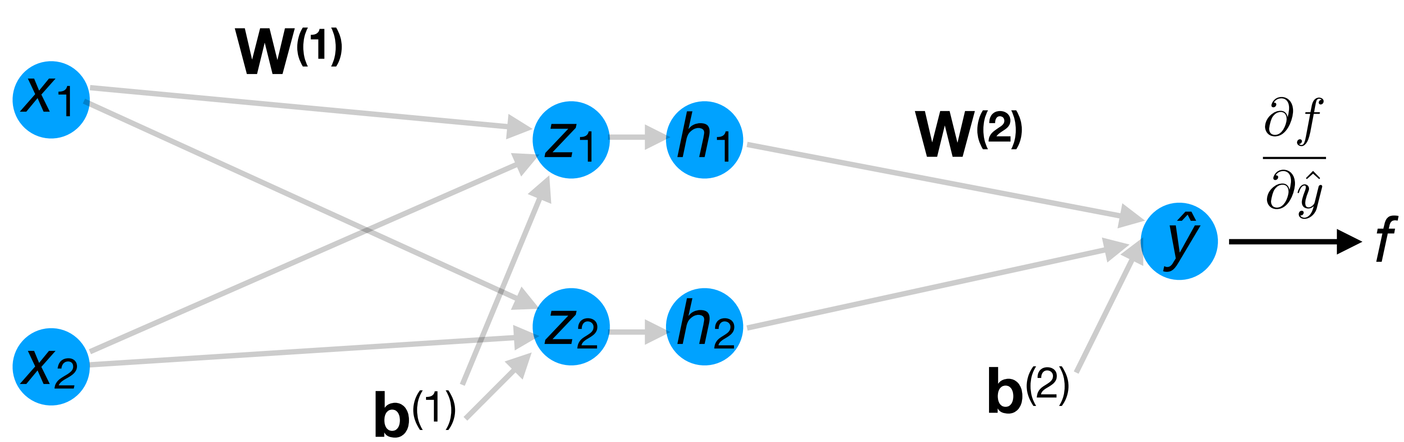
Redundant computation



Computing the gradients

- Here's how we can compute all these *efficiently*:

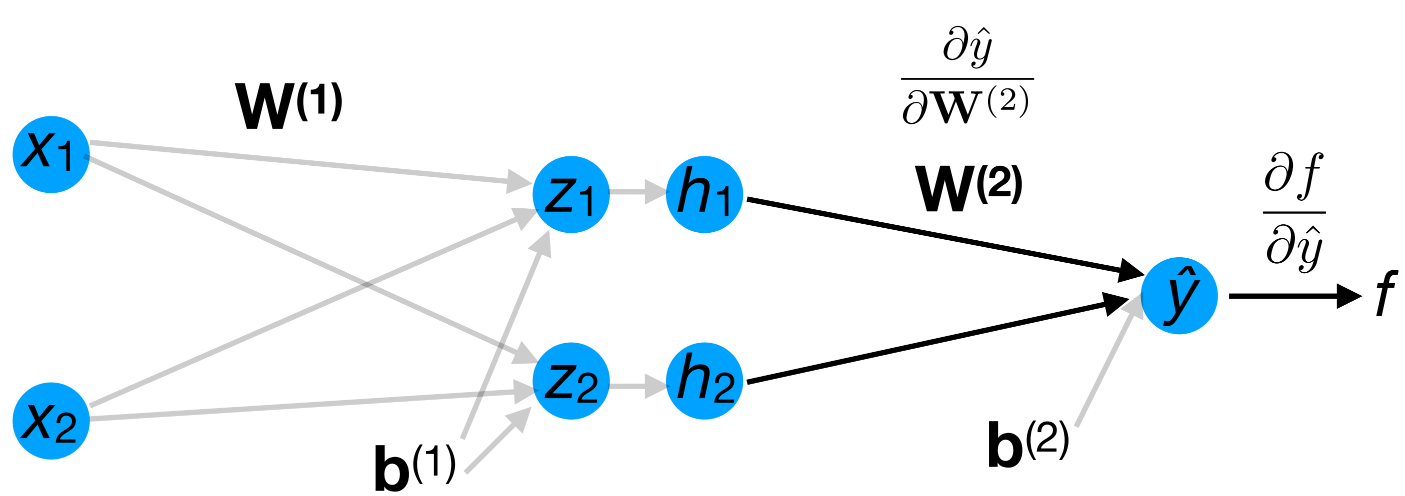
$$\frac{\partial f}{\partial \mathbf{W}^{(2)}} = \frac{\partial f}{\partial \hat{y}}$$



Computing the gradients

- Here's how we can compute all these *efficiently*:

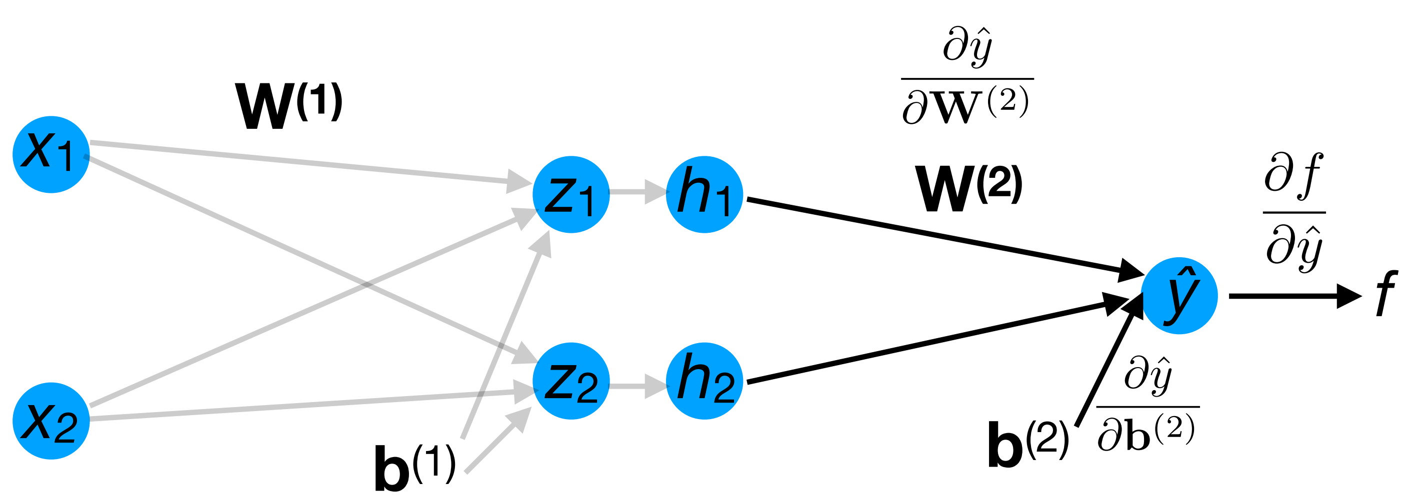
$$\frac{\partial f}{\partial \mathbf{W}^{(2)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{W}^{(2)}}$$



Computing the gradients

- Here's how we can compute all these *efficiently*:

$$\frac{\partial f}{\partial \mathbf{W}^{(2)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{W}^{(2)}}$$
$$\frac{\partial f}{\partial \mathbf{b}^{(2)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{b}^{(2)}}$$



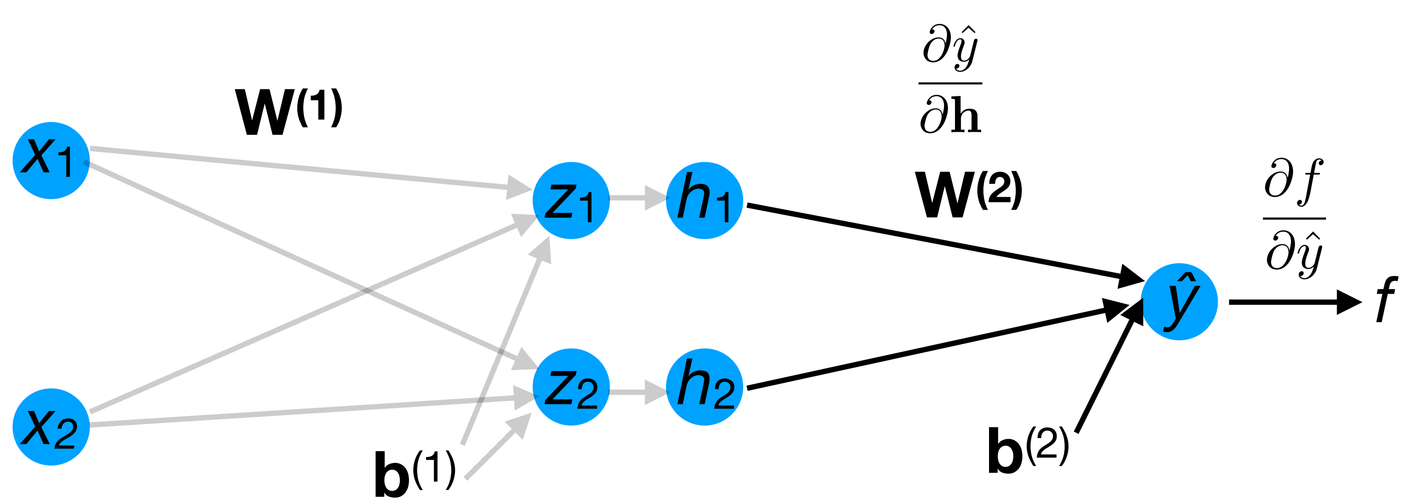
Computing the gradients

- Here's how we can compute all these *efficiently*:

$$\frac{\partial f}{\partial \mathbf{W}^{(2)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{W}^{(2)}}$$

$$\frac{\partial f}{\partial \mathbf{b}^{(2)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{b}^{(2)}}$$

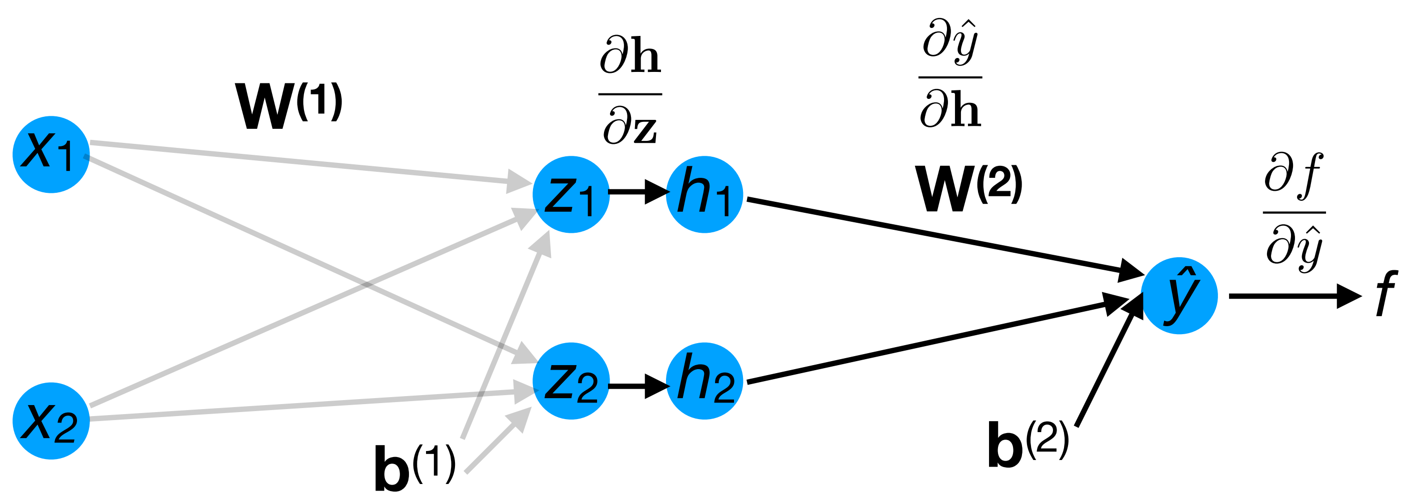
$$\frac{\partial f}{\partial \mathbf{W}^{(1)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{h}}$$



Computing the gradients

- Here's how we can compute all these *efficiently*:

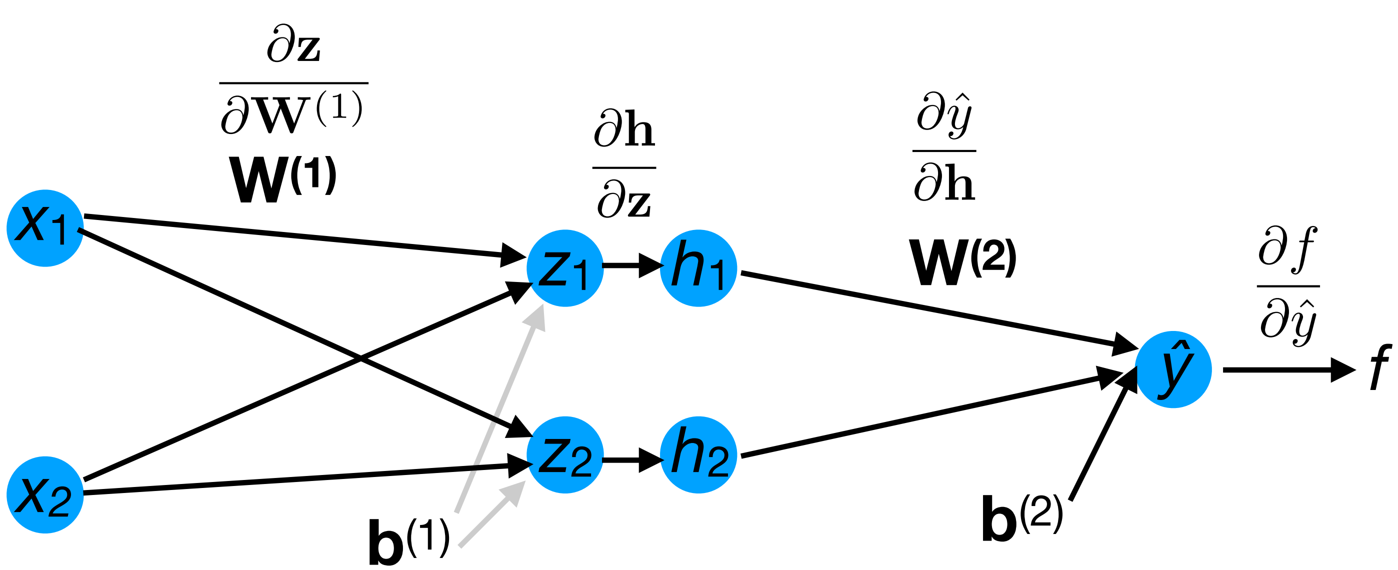
$$\frac{\partial f}{\partial \mathbf{W}^{(2)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{W}^{(2)}}$$
$$\frac{\partial f}{\partial \mathbf{b}^{(2)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{b}^{(2)}}$$
$$\frac{\partial f}{\partial \mathbf{W}^{(1)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}}$$



Computing the gradients

- Here's how we can compute all these *efficiently*:

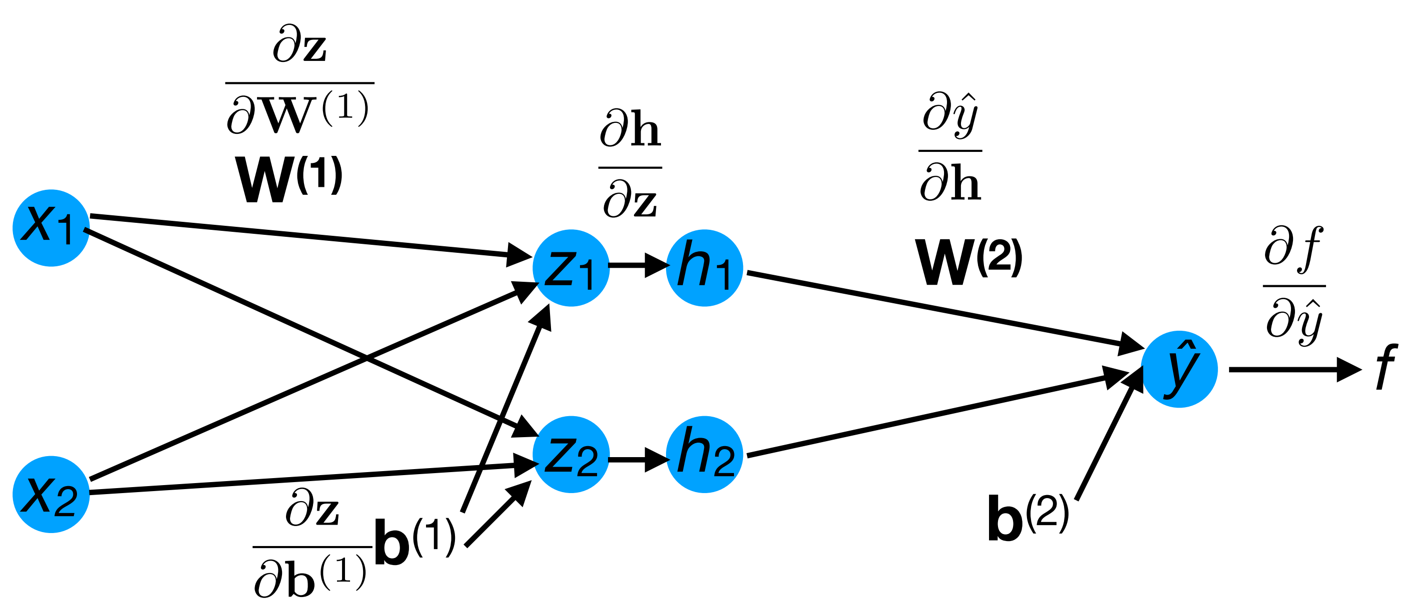
$$\frac{\partial f}{\partial \mathbf{W}^{(2)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{W}^{(2)}}$$
$$\frac{\partial f}{\partial \mathbf{b}^{(2)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{b}^{(2)}}$$
$$\frac{\partial f}{\partial \mathbf{W}^{(1)}} = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{W}^{(1)}}$$



Computing the gradients

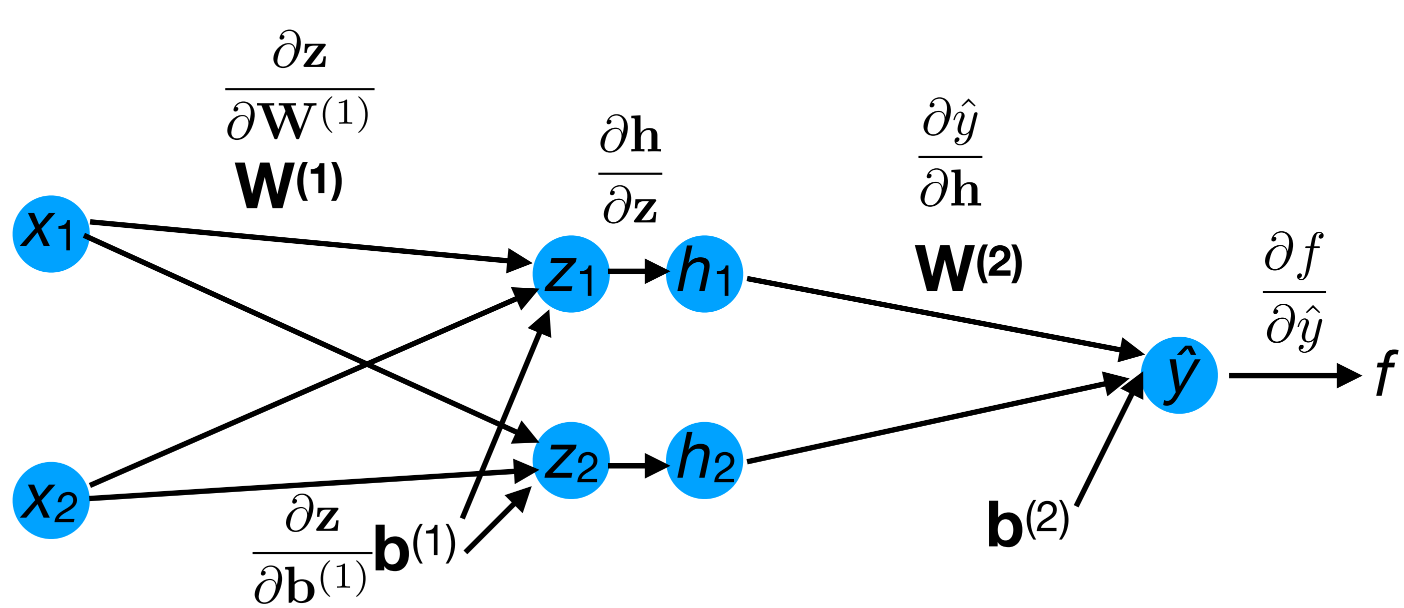
- Here's how we can compute all these *efficiently*:

$$\begin{aligned}
\frac{\partial f}{\partial \mathbf{W}^{(2)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{W}^{(2)}} \\
\frac{\partial f}{\partial \mathbf{b}^{(2)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{b}^{(2)}} \\
\frac{\partial f}{\partial \mathbf{W}^{(1)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{W}^{(1)}} \\
\frac{\partial f}{\partial \mathbf{b}^{(1)}} &= \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{b}^{(1)}}
\end{aligned}$$

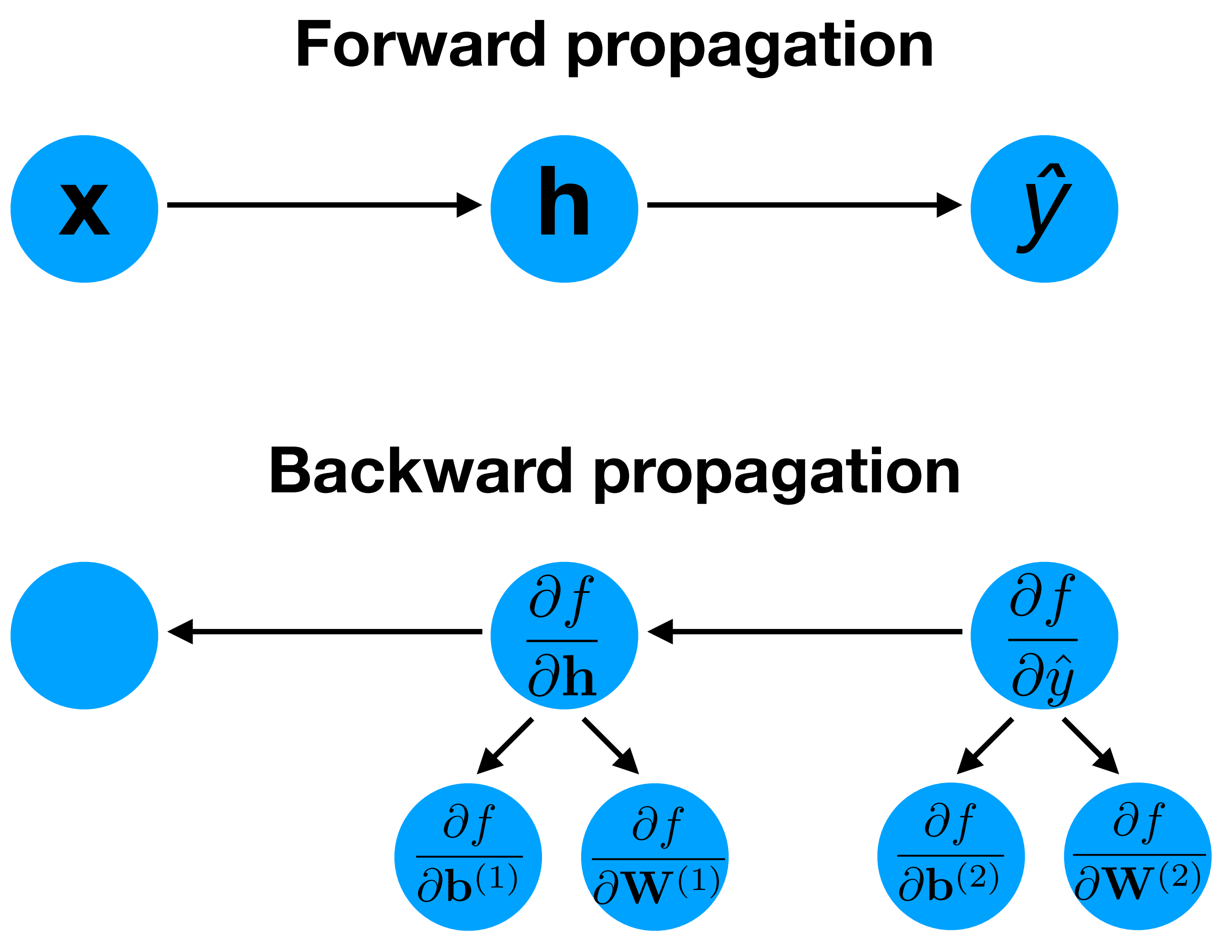


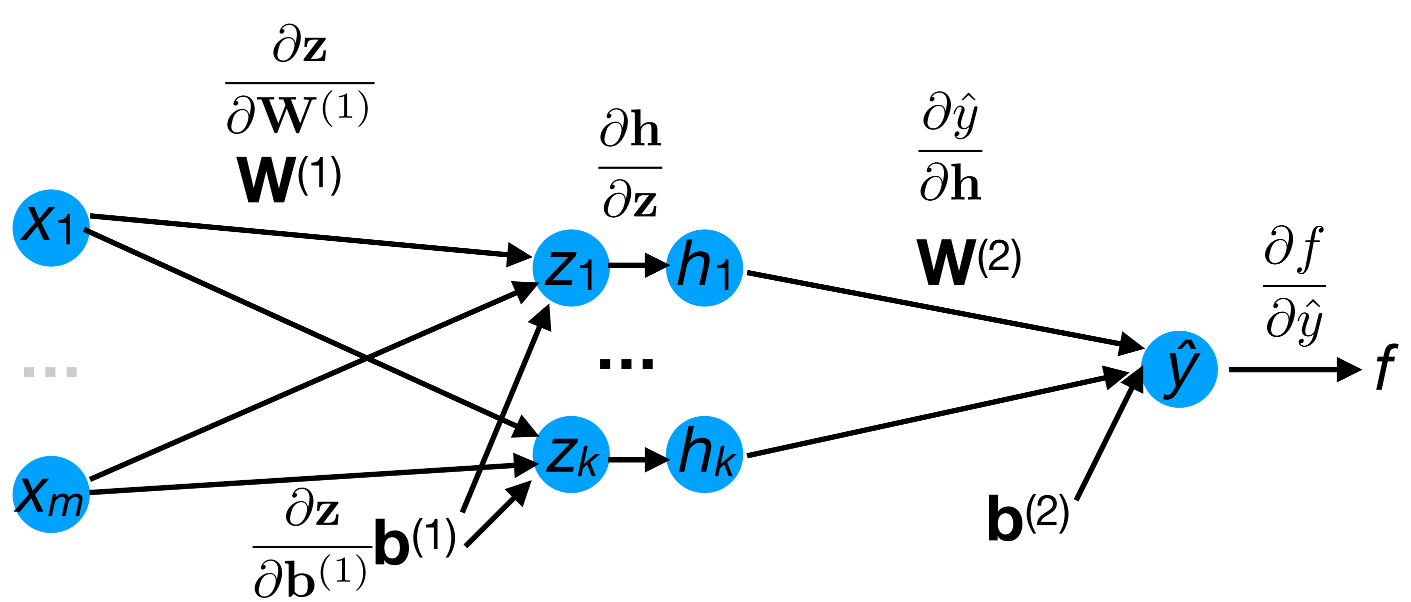
Computing the gradients

- This process is known as **backwards propagation ("backprop"):**
  - It produces the gradient terms of all the weight matrices and bias vectors.
  - It requires first conducting forward propagation.



Computing the gradients





Jacob Whitehill, WPI