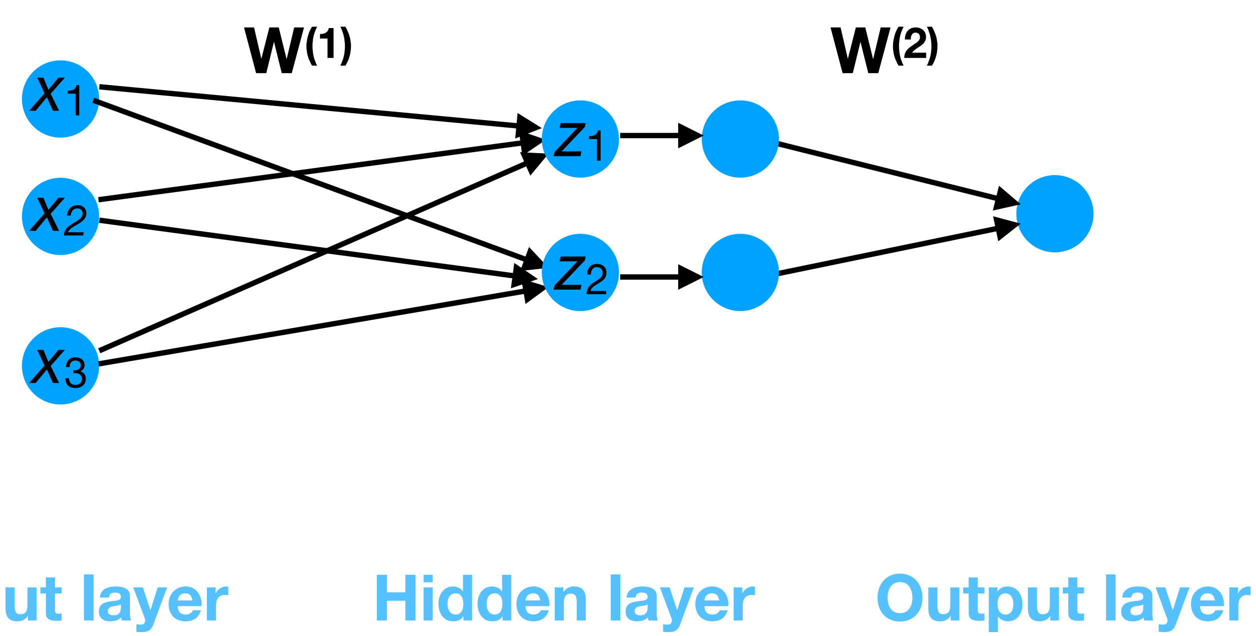
CS 4342: Class 19

Jacob Whitehill

Neural networks

Feed-forward NN

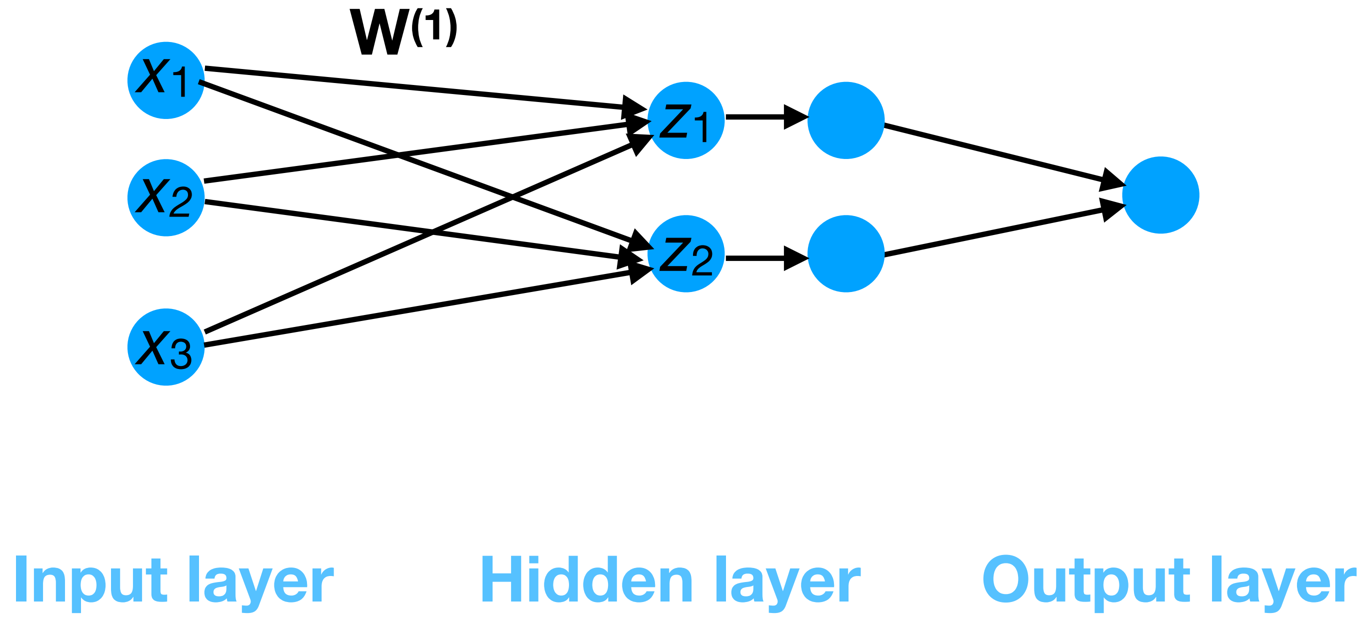
- Neural networks can have multiple neurons per layer.
- Between each adjacent pair of layers (input-hidden and hidden-output), there is a *matrix* of (synaptic) weights:



Feed-forward NN

- We can compute the pre-activation values **z** of the hidden layer as:

$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x}$$

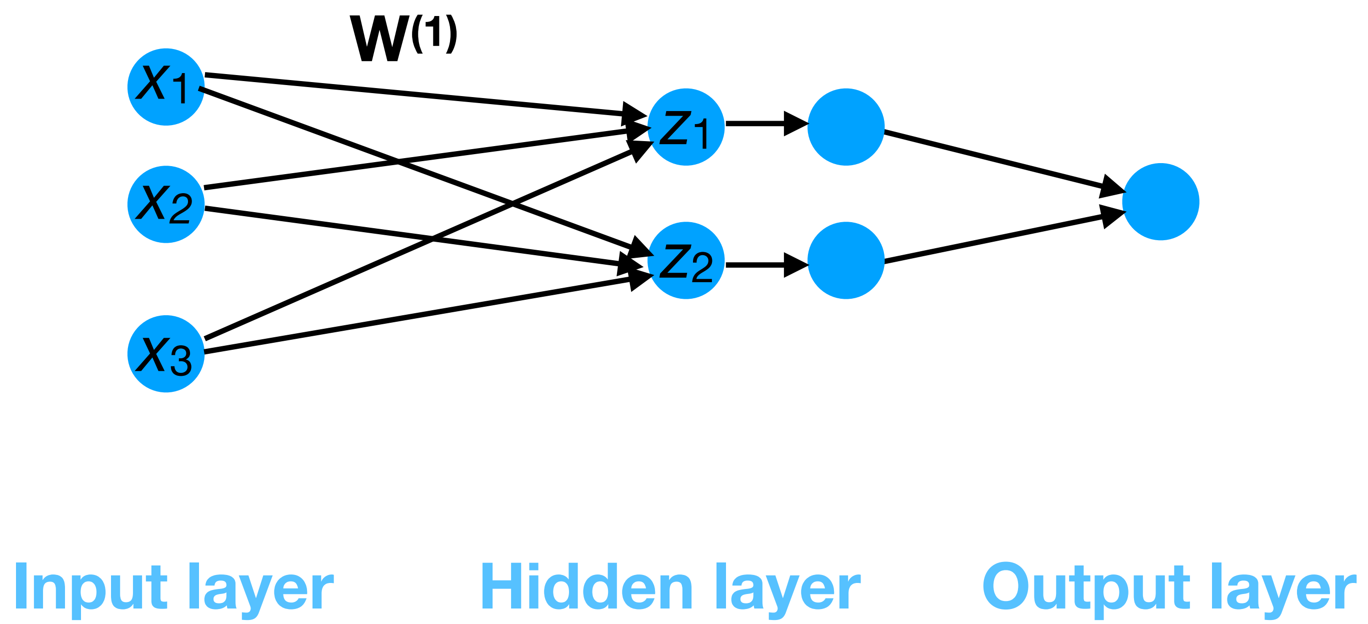


Feed-forward NN

- We can compute the pre-activation values **z** of the hidden layer as:

$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x}$$

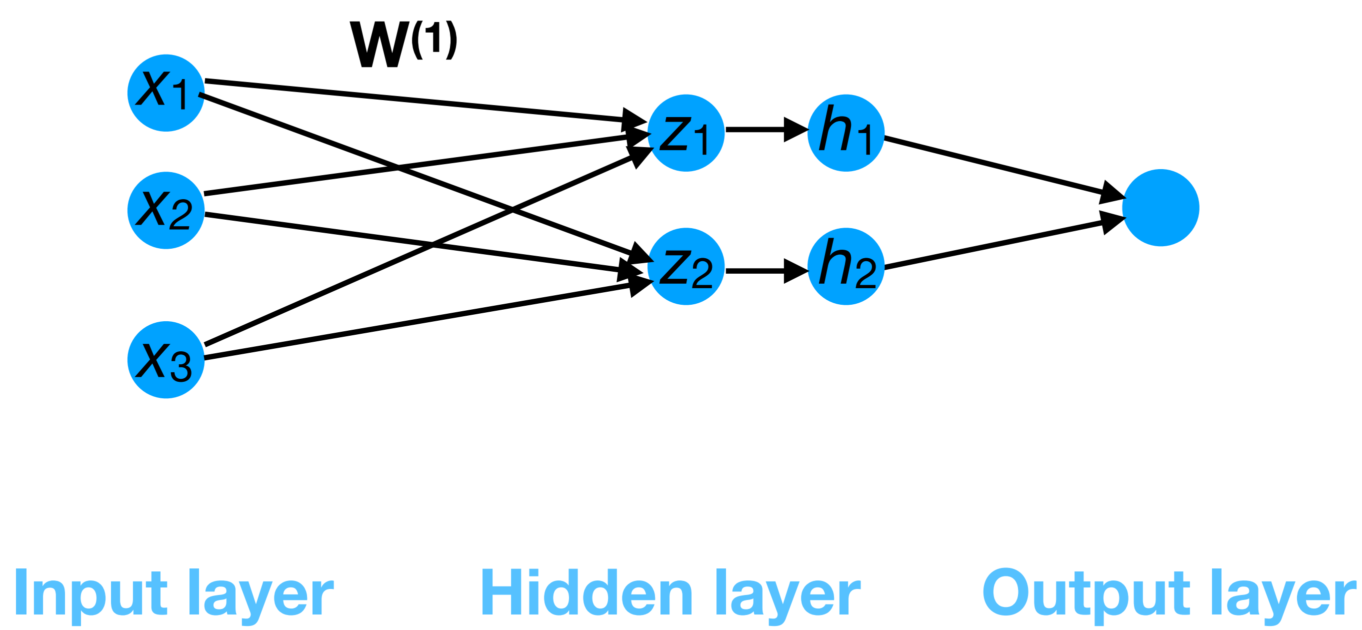
$\mathbf{W}^{(1)}$ **is 2 x 3.**



Feed-forward NN

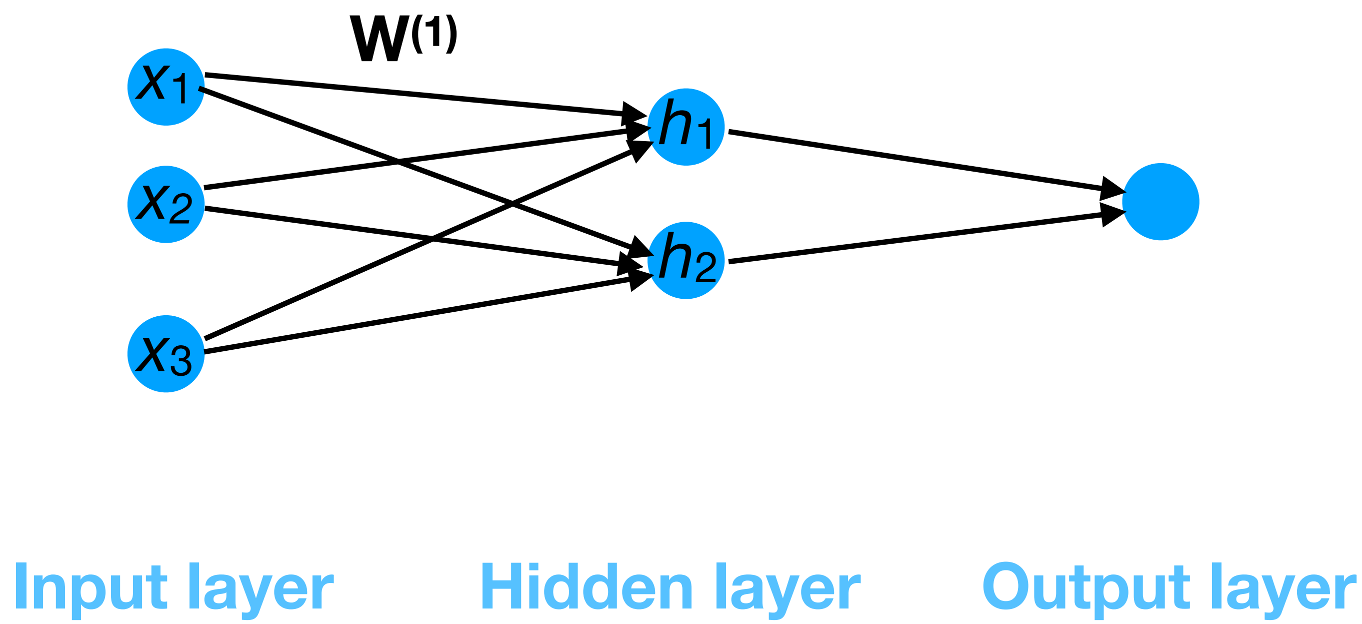
- We can then pass **z** to the activation function σ and compute the hidden neuron values **element-wise**, i.e.:

$$\mathbf{h}_j = \sigma(\mathbf{z}_j)$$



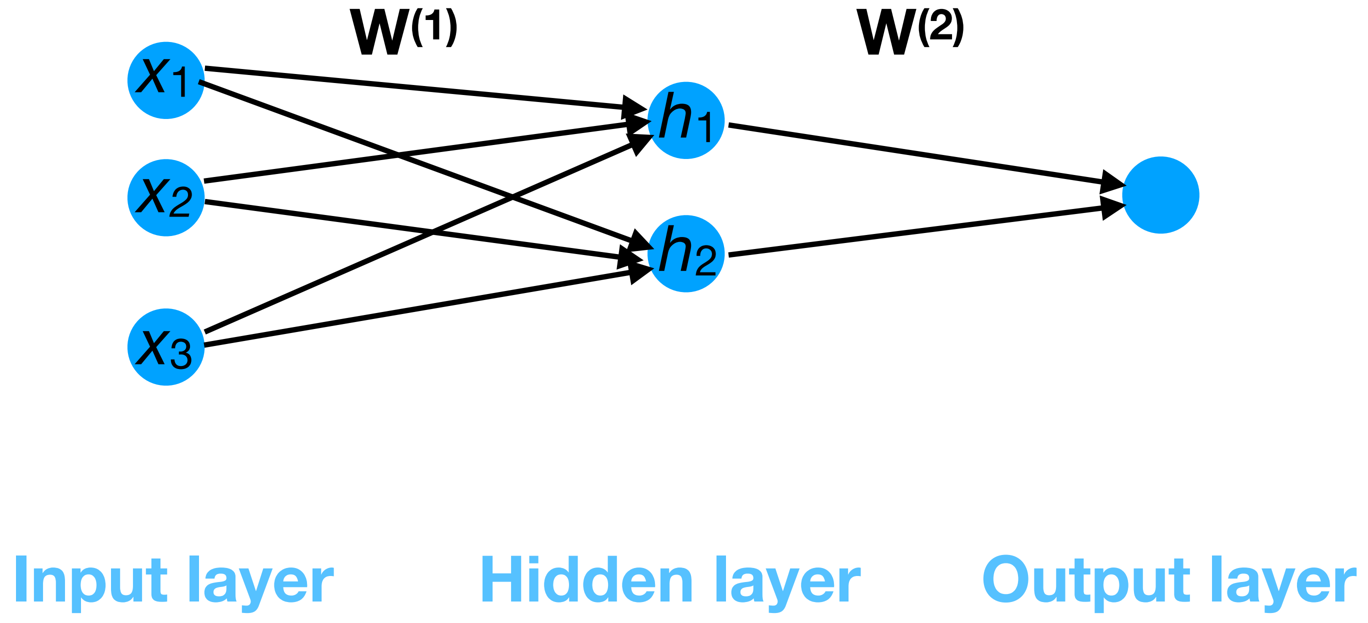
Feed-forward NN

- We typically do not shown the **z** layer explicitly; it is subsumed into the **h** layer to avoid clutter.



Feed-forward NN

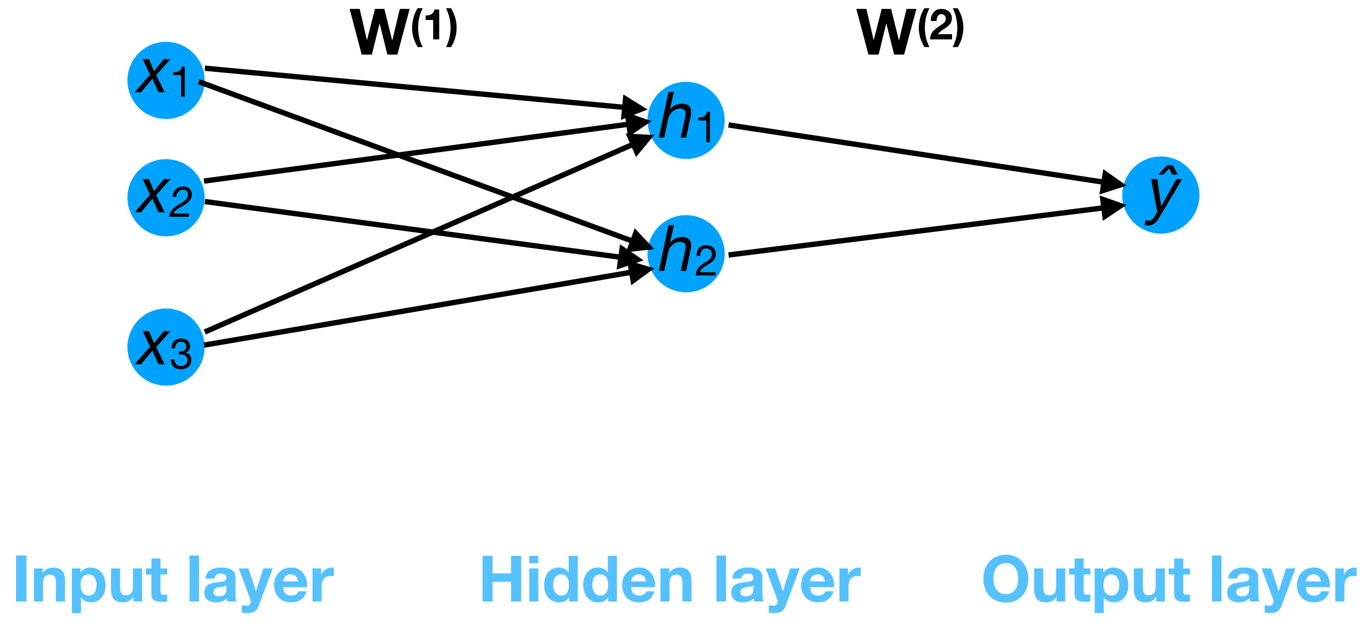
- Next, we pass **h** to the next layer…



Feed-forward NN

- …and compute the product:

$$\hat{\mathbf{y}} = \mathbf{W}^{(2)}\mathbf{h}$$

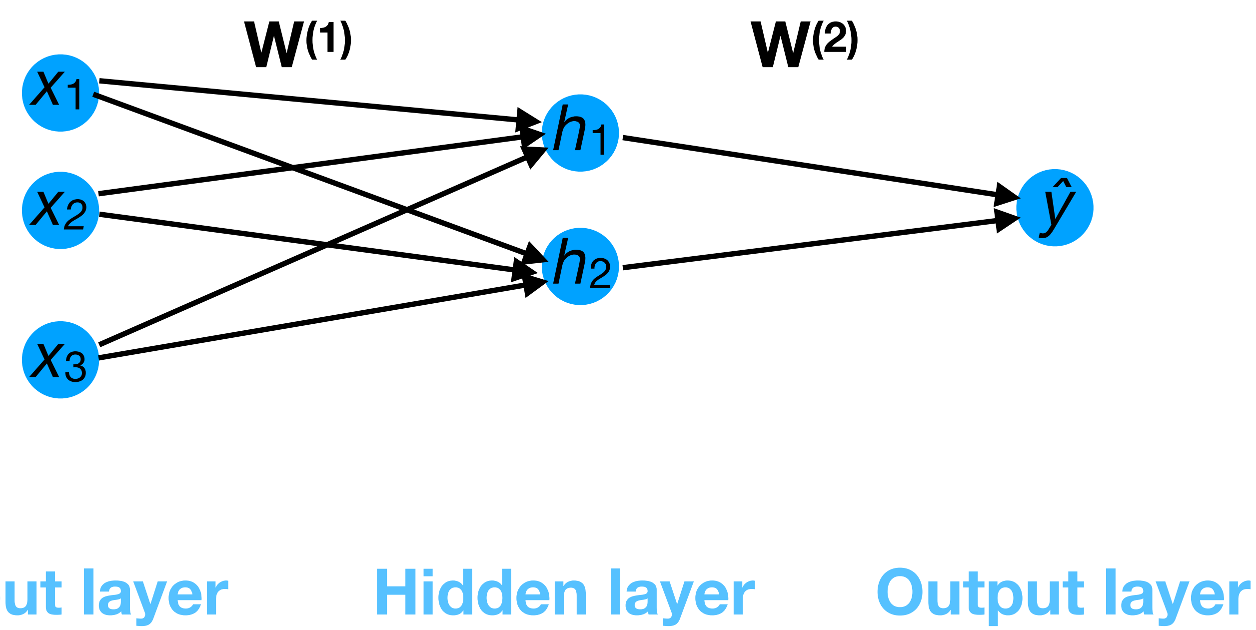


Feed-forward NN

- …and compute the product:

$$\hat{\mathbf{y}} = \mathbf{W}^{(2)}\mathbf{h}$$

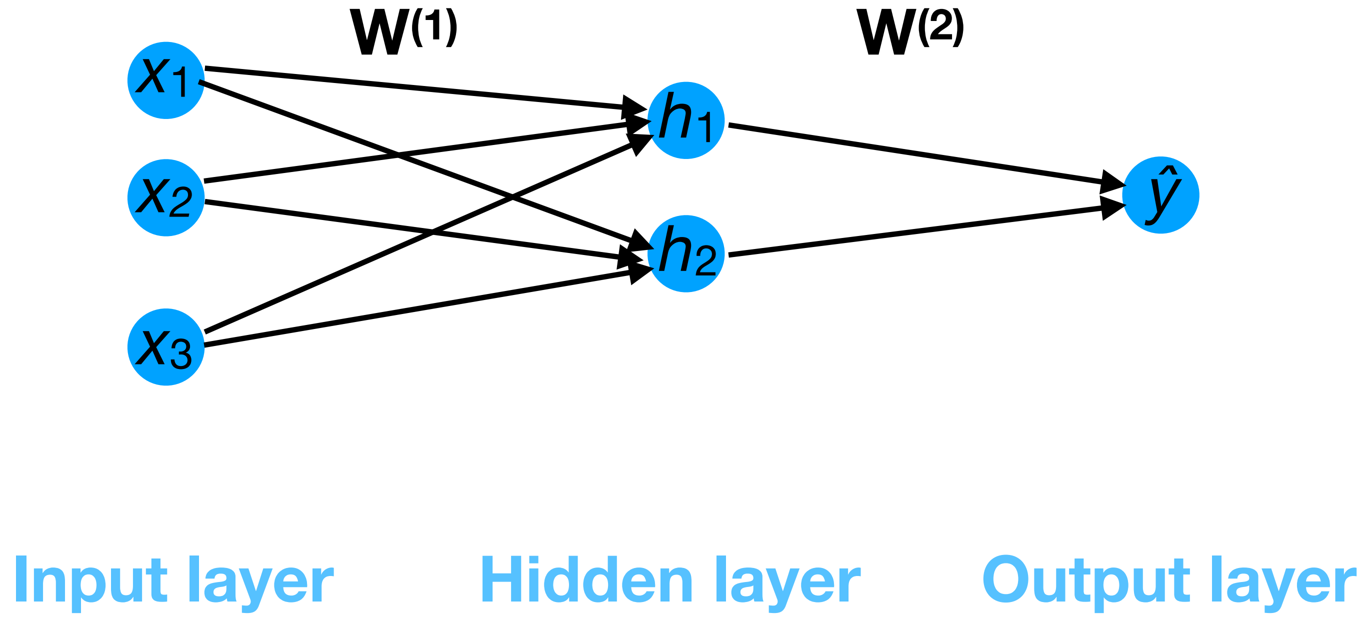
$\mathbf{W}^{(2)}$ is 1 x 2.



Feed-forward NN

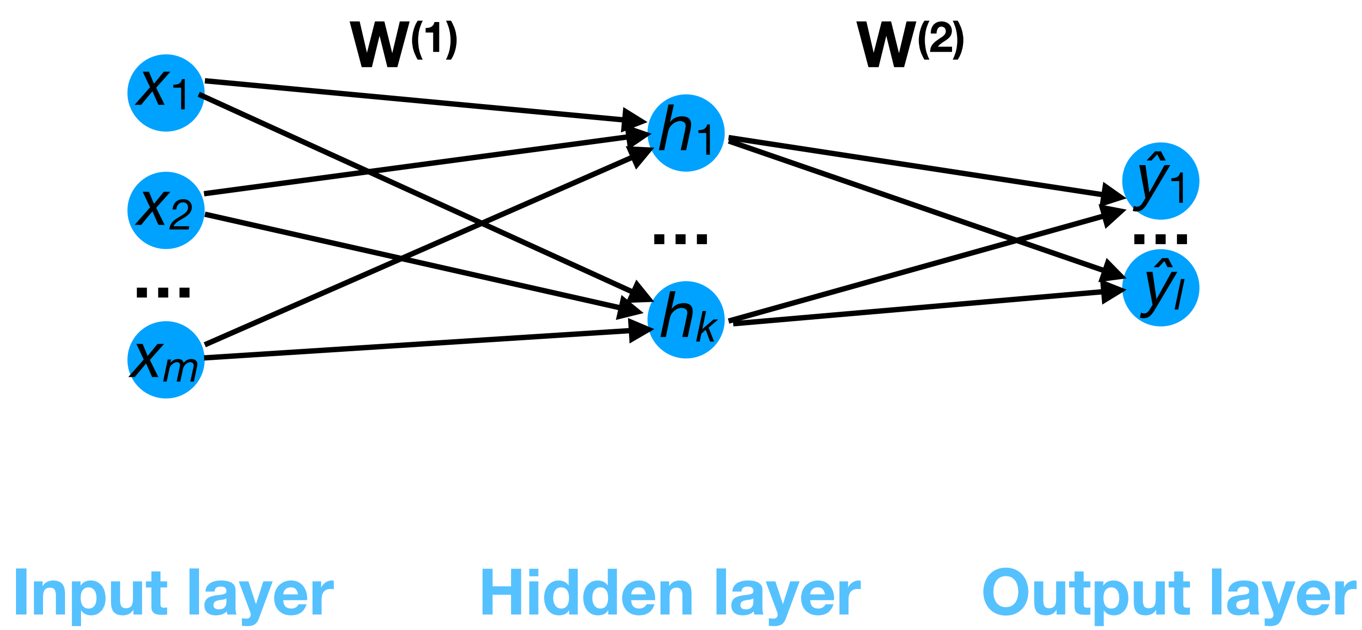
- Note that the final layer could also have an activation function (if we wanted one), e.g.:

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}^{(2)}\mathbf{h}\right)$$



Multiple output neurons

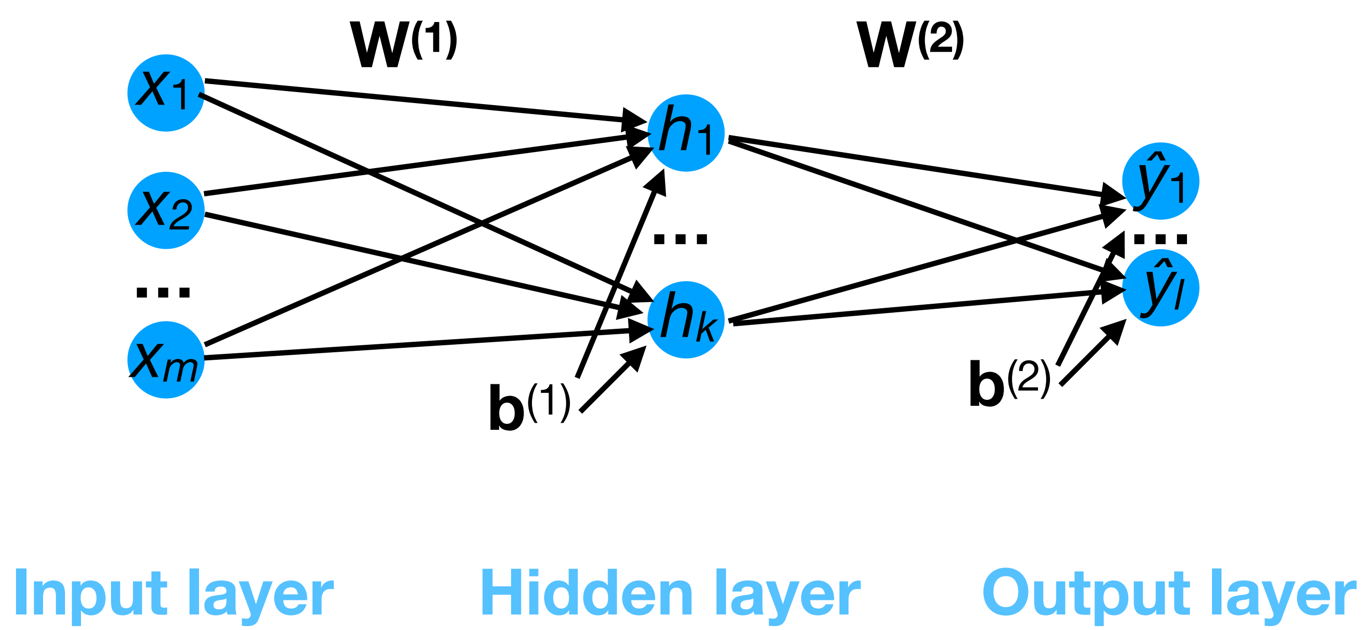
- We can also have a NN with multiple output neurons, e.g.:



Bias terms

- We typically include a **bias term** for every neuron, so that the layers' values are computed as:

$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \qquad \textbf{and} \qquad \hat{\mathbf{y}} = \mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}$$



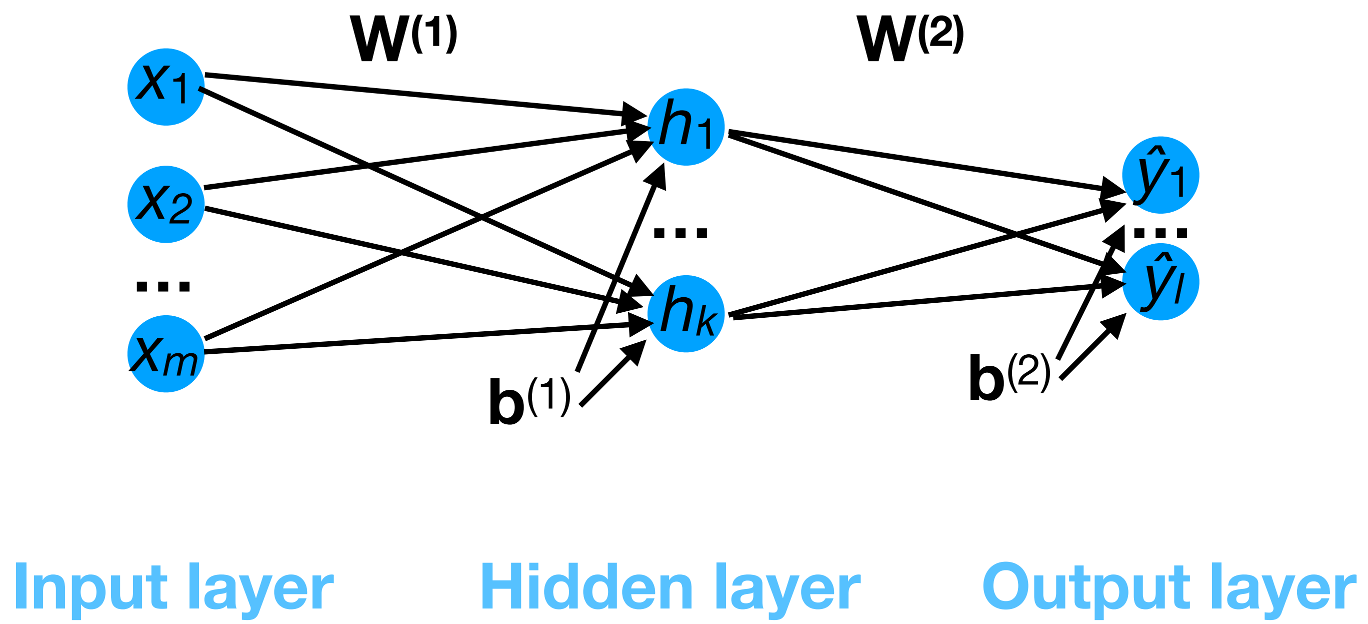
Exercise: bias terms

- What will $\hat{\mathbf{y}}$ be for $\mathbf{x} = [1\ 0\ {-2}]^T$,

$$\hat{y} = g(\mathbf{x}) = \mathbf{W}^{(2)}\sigma\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}$$

$$\mathbf{W}^{(1)} = \left[\begin{array}{ccc} 1 & 0 & 1 \\ -1 & 2 & 3 \end{array}\right] \quad \mathbf{b}^{(1)} = \left[\begin{array}{c} -1 \\ 2 \end{array}\right]$$

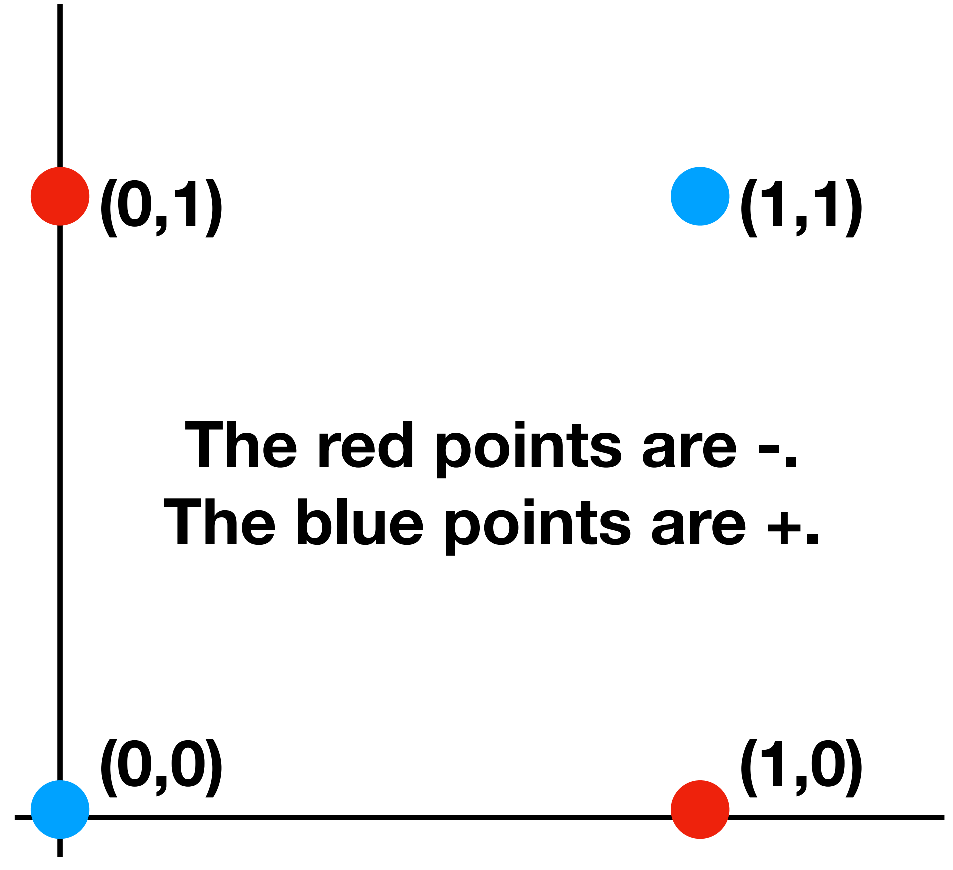
$$\mathbf{W}^{(2)} = \left[\begin{array}{cc} 1 & -1 \\ -1 & 1 \end{array}\right] \quad \mathbf{b}^{(2)} = \left[\begin{array}{c} 0 \\ 1 \end{array}\right]$$



Neural networks for non-linear classification

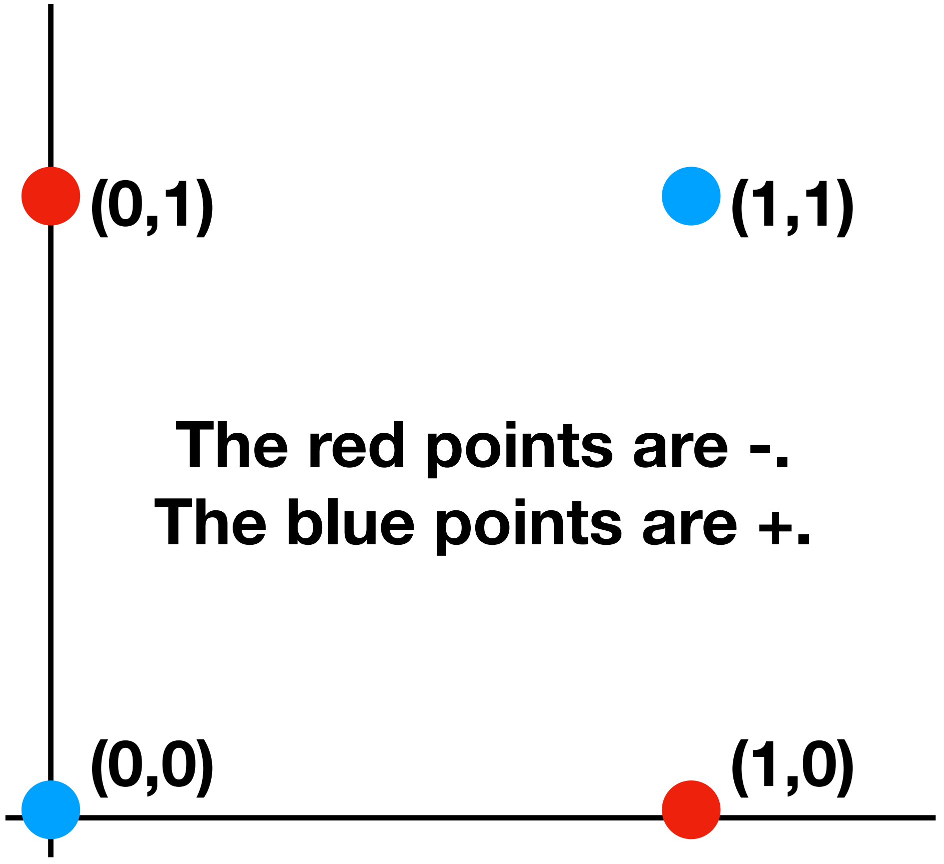
XOR problem

- Recall that no linear decision boundary (e.g., linear SVM, linear regression) can solve the XOR problem.



XOR problem

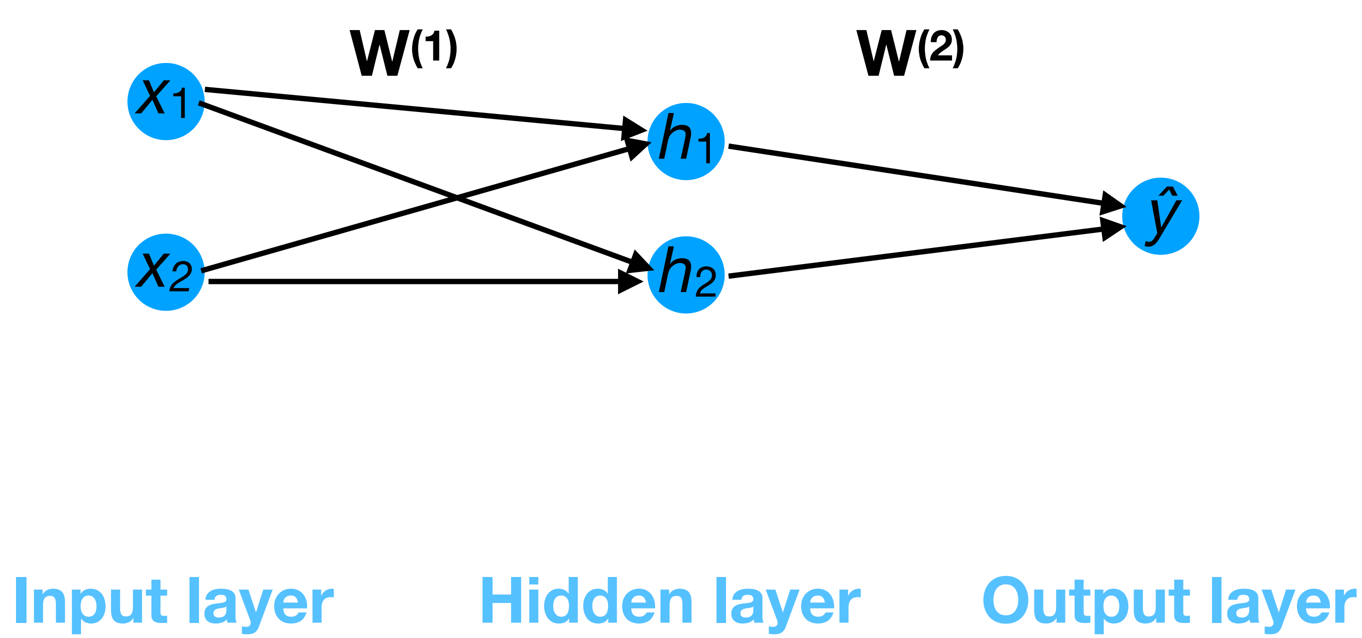
- Recall that no linear decision boundary (e.g., linear SVM, linear regression) can solve the XOR problem.



- Let's see how using a hidden layer can help us solve it…

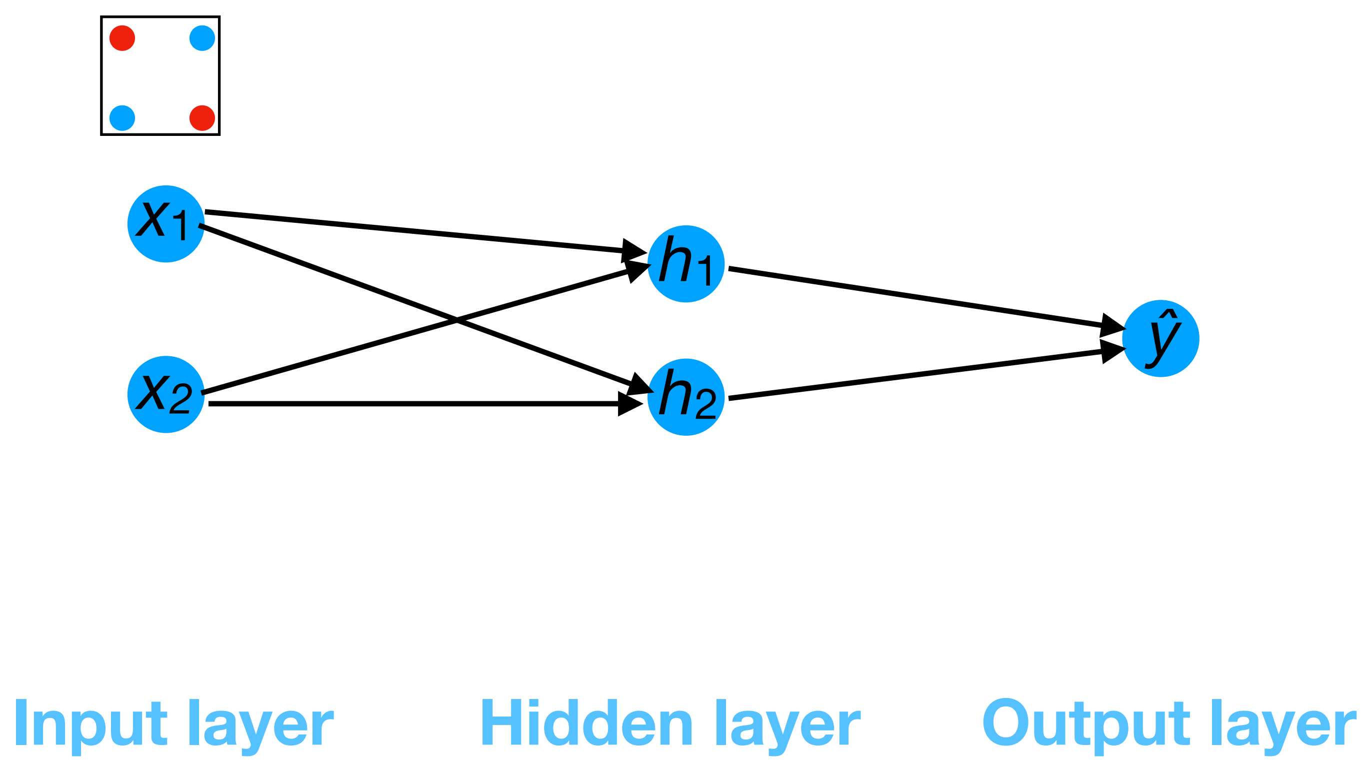
XOR problem

- We want to use a NN to define a function g such that:
  - $g(0,0) = 0$
    $g(0,1) = 1$
    $g(1,0) = 1$
    $g(1,1) = 0$



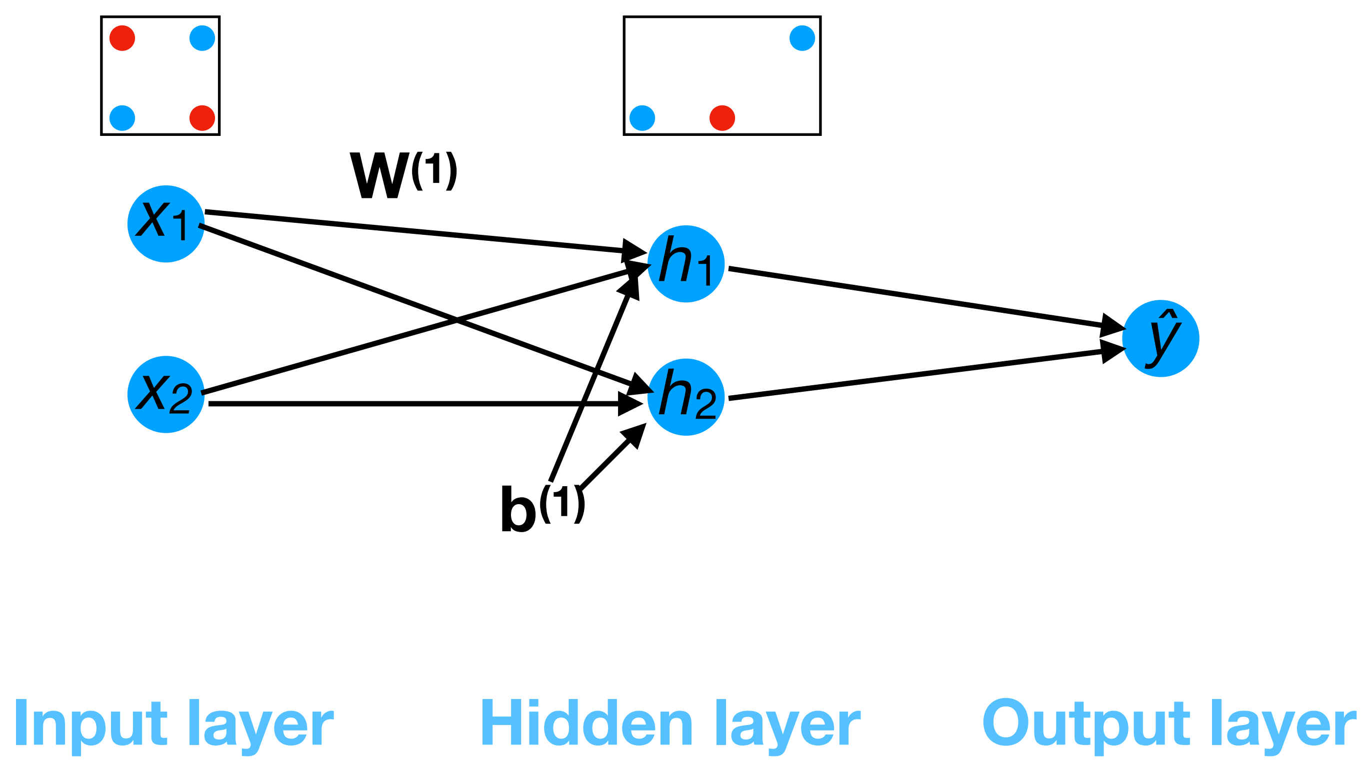
XOR problem

- Here's how a 3-layer NN with a non-linear (ReLU) activation function can solve it:



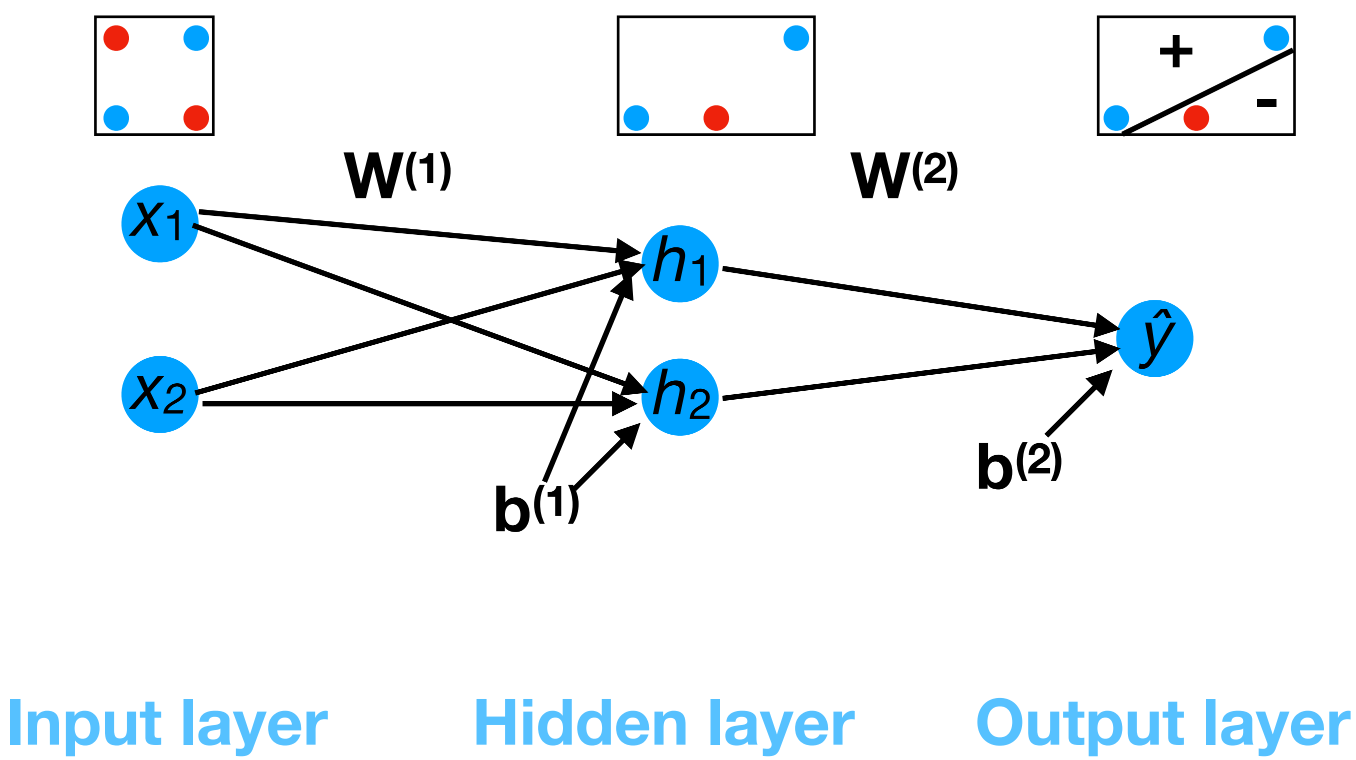
XOR problem

- Here's how a 3-layer NN with a non-linear (ReLU) activation function can solve it:
  - $\mathbf{W}^{(1)}$, $\mathbf{b}^{(1)}$ will "collapse" the two - data points onto one point in the "hidden" 2-D space.



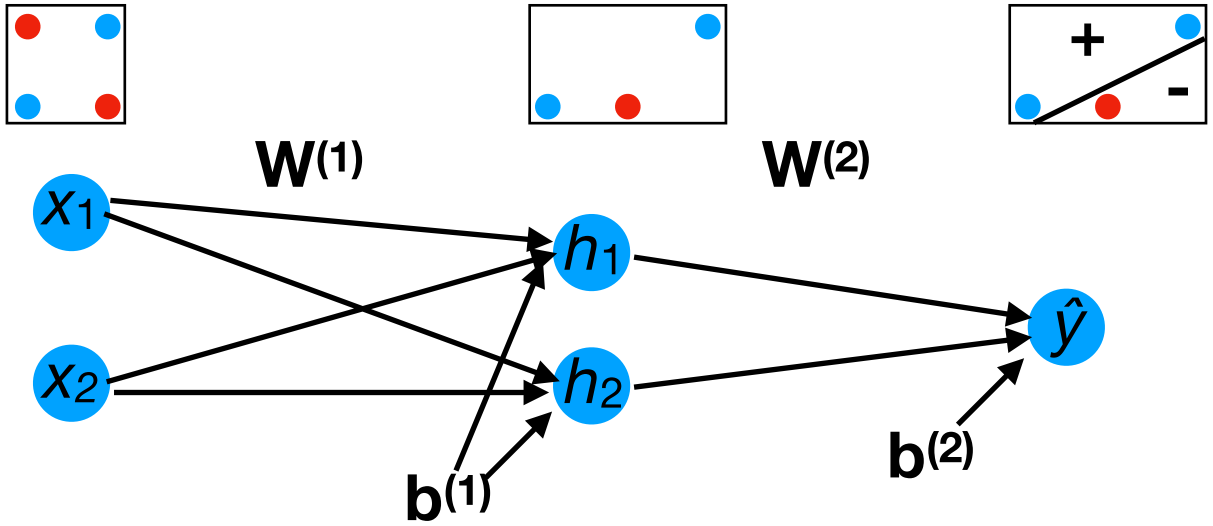
XOR problem

- Here's how a 3-layer NN with a non-linear (ReLU) activation function can solve it:
  - $\mathbf{W}^{(1)}$, $\mathbf{b}^{(1)}$ will "collapse" the two - data points onto one point in the "hidden" 2-D space.
  - Since the + and - data are now linearly separated, $\mathbf{W}^{(2)}$, $\mathbf{b}^{(2)}$ can easily distinguish the two classes.



XOR problem

- Here's how a 3-layer NN with a non-linear (ReLU) activation function can solve it:
  - **W**$^{(1)}$, **b**$^{(1)}$ will "collapse" the two - data points onto one point in the "hidden" 2-D space.
  - Since the + and - data are now linearly separated, **W**$^{(2)}$, **b**$^{(2)}$ can easily distinguish the two classes.



What values for W$^{(1)}$, b$^{(1)}$ will make the 4 data points linearly separable?
(Hint: set b = [-1 -1]^{T}; W contains only 1s and 2s.)

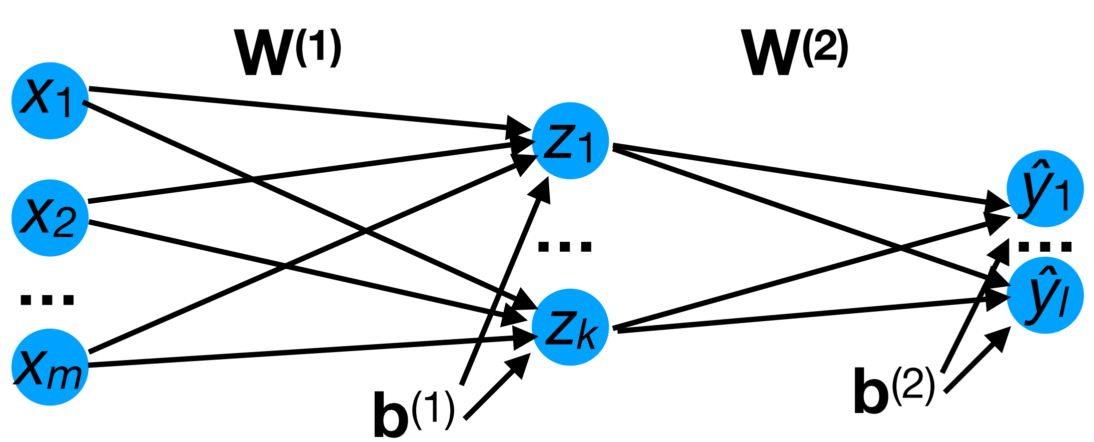
XOR problem

- Note that the ability of the 3-layer NN to solve the XOR problem relies crucially on the non-linear ReLU activation function.
  - Note that other non-linear functions would also work.
- Without non-linearity, a multi-layer NN is no more powerful than a 2-layer network!

Crucial role of non-linearity

- Suppose we define a 3-layer NN **without** non-linearity:

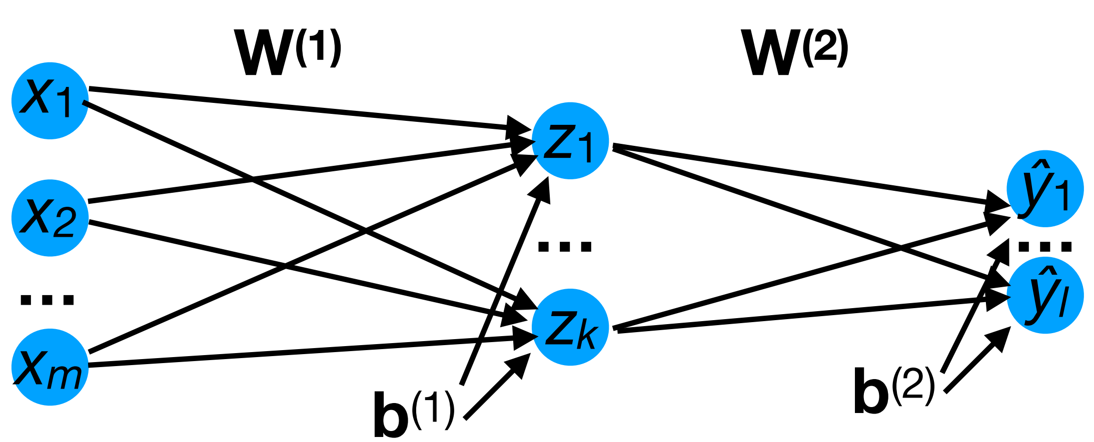
$$g(\mathbf{x}) \quad = \quad \mathbf{W}^{(2)} \left(\mathbf{W}^{(1)} \mathbf{x} + \mathbf{b}^{(1)} \right) + \mathbf{b}^{(2)}$$



Crucial role of non-linearity

- Then we can simplify *g* to be:

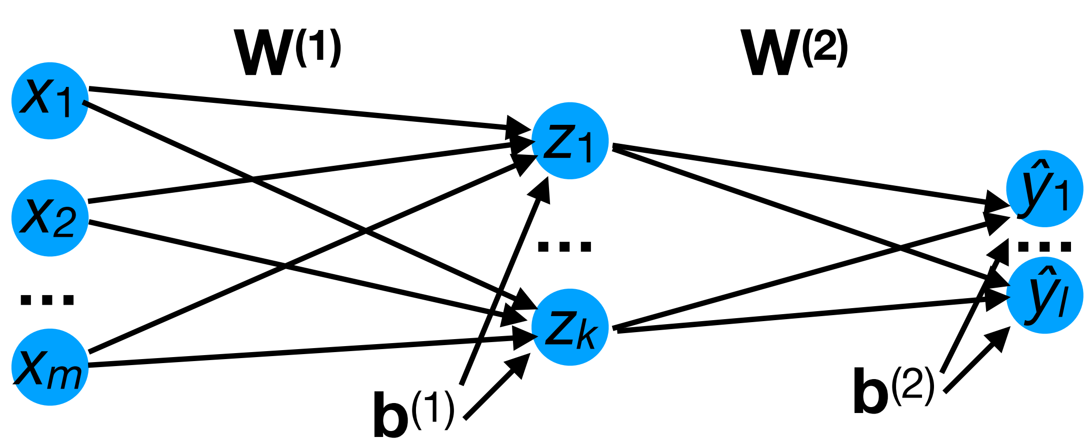
$$g(\mathbf{x}) \quad = \quad \mathbf{W}^{(2)}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}$$



Crucial role of non-linearity

- Then we can simplify g to be:

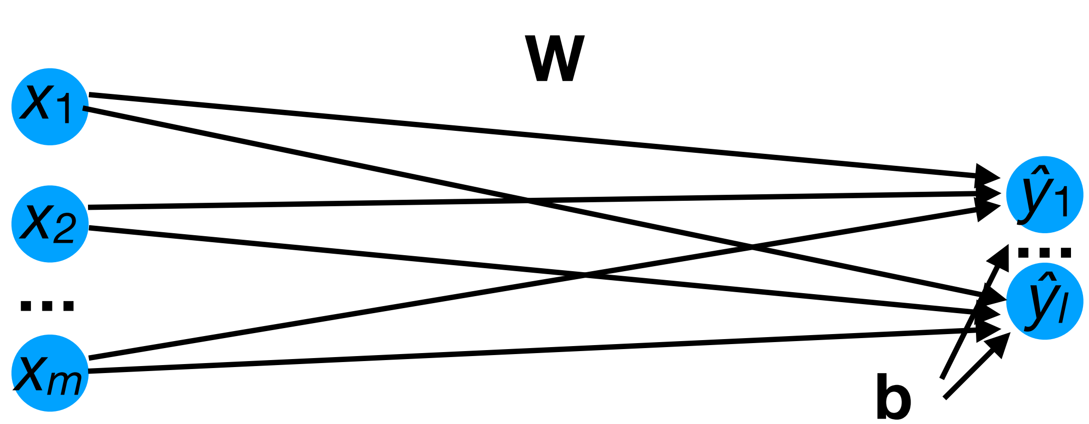
$$
\begin{aligned}
g(\mathbf{x}) &= \mathbf{W}^{(2)}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)} \\
&= \left(\mathbf{W}^{(2)}\mathbf{W}^{(1)}\right)\mathbf{x} + \mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}
\end{aligned}
$$



Crucial role of non-linearity

- Then we can simplify *g* to be:

$$\begin{aligned} g(\mathbf{x}) &= \mathbf{W}^{(2)}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)} \\ &= \left(\mathbf{W}^{(2)}\mathbf{W}^{(1)}\right)\mathbf{x} + \mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)} \\ &= \mathbf{W}\mathbf{x} + \mathbf{b} \end{aligned}$$



Training neural networks

Training neural networks

- While training neural networks by hand is (arguably) fun, it is completely impractical except for toy examples.

- How can we find good values for the weights and bias terms automatically?

Training neural networks

- While training neural networks by hand is (arguably) fun, it is completely impractical except for toy examples.
- How can we find good values for the weights and bias terms automatically?
    - Gradient descent.

Exercise

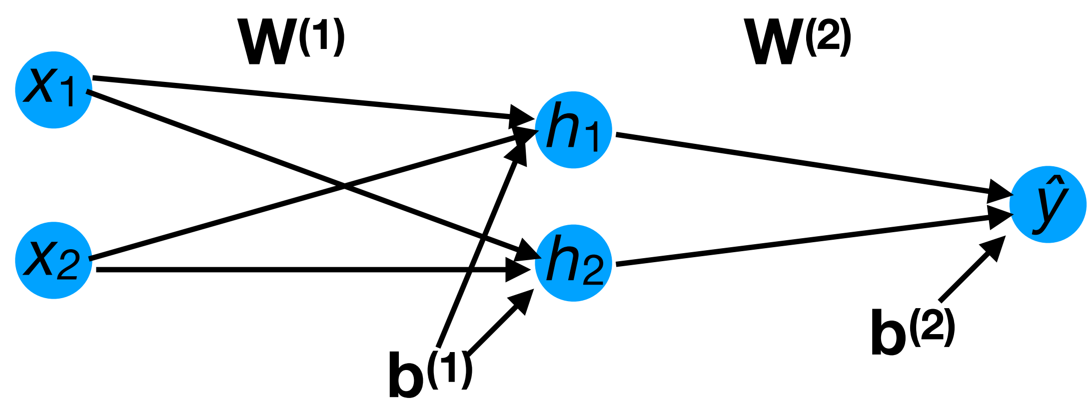
- Suppose σ is a differentiable function with derivative σ'.
- What is the derivative of the expression $v\sigma(uw + z)$ with respect to w (where u, v, w, z are all scalars)?

Exercise

- Suppose σ is a differentiable function with derivative σ'.
- What is the derivative of the expression $v\sigma(uw + z)$ with respect to w (where u, v, w, z are all scalars)?
  - $uv\sigma'(uw + z)$

Gradient descent: XOR problem

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



- Let's first define:

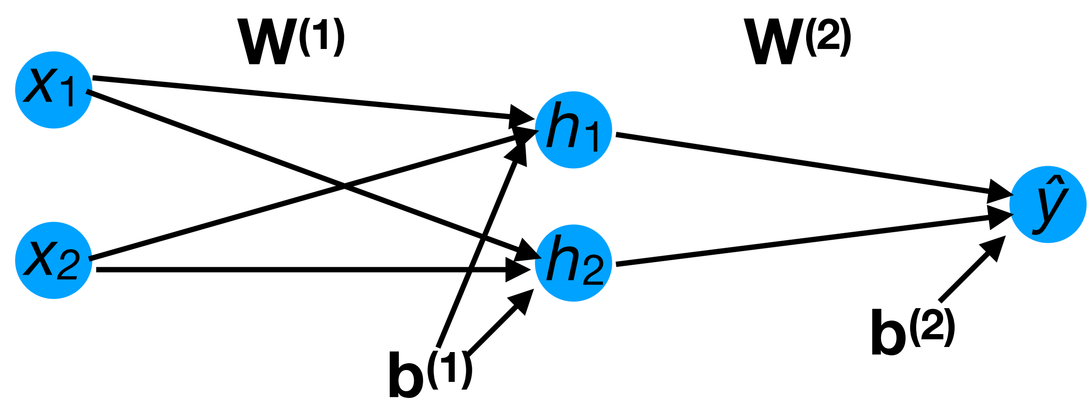
$$\mathbf{W}^{(1)} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}, \mathbf{W}^{(2)} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix}, \mathbf{b}^{(1)} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \mathbf{b}^{(2)} = \begin{bmatrix} b_3 \end{bmatrix}$$

- Then we can define *g* so that:

$$\hat{y} = g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) \quad = \quad \mathbf{W}^{(2)}\sigma\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}$$

Gradient descent: XOR problem

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



- Let's first define:

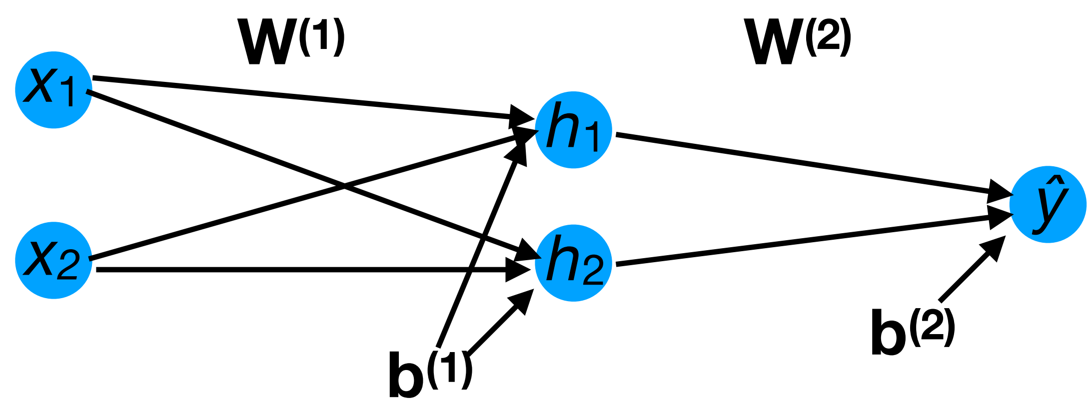
$$\mathbf{W}^{(1)} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}, \mathbf{W}^{(2)} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix}, \mathbf{b}^{(1)} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \mathbf{b}^{(2)} = \begin{bmatrix} b_3 \end{bmatrix}$$

- Then we can define *g* so that:

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

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:

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

Gradient descent: XOR problem

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

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

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

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

where:

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

Gradient descent: XOR problem

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

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

Gradient descent: XOR problem

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

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

This is the only term that depends on w_1. It has the form: $c*\sigma(z)$ where z is a function of w_1. Recall that:

$$\begin{aligned}\frac{\partial}{\partial w_1}\left[c\sigma(z)\right] &= c\frac{\partial \sigma}{\partial z}(z)\frac{\partial z}{\partial w_1} \\ &= c\sigma'(z)\frac{\partial z}{\partial w_1}\end{aligned}$$

Gradient descent: XOR problem

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

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

Gradient descent: XOR problem

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

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

Gradient descent: XOR problem

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

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

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:

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

Exercise

- Find:

$$\frac{\partial \hat{y}}{\partial b_2} \quad = \quad \frac{\partial}{\partial b_2}\left[w_5\sigma(w_1x_1 + w_2x_2 + b_1) + w_6\sigma(w_3x_1 + w_4x_2 + b_2) + b_3\right]$$

Gradient descent: XOR problem

- We need to derive all the other partial derivatives:

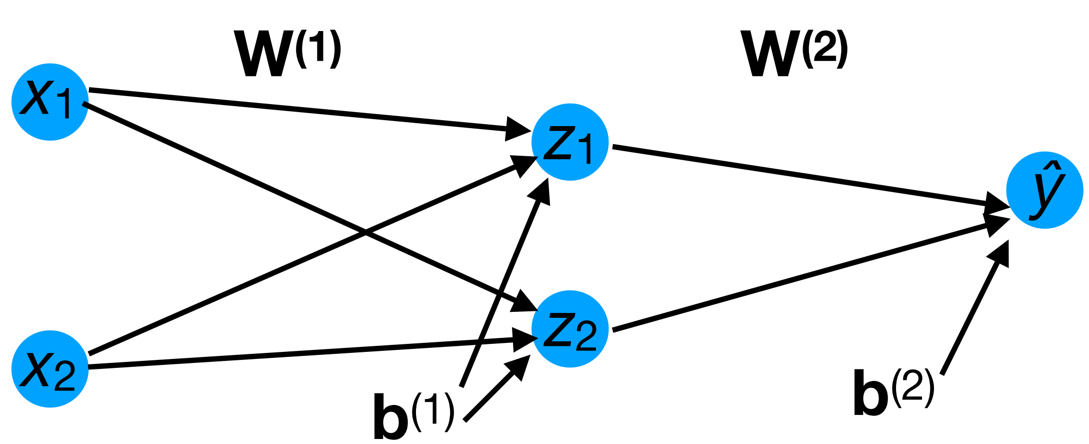
$$\frac{\partial \hat{y}}{\partial w_1}, \ldots \frac{\partial \hat{y}}{\partial w_6}, \frac{\partial \hat{y}}{\partial b_1}, \ldots \frac{\partial \hat{y}}{\partial w_3}$$

- 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: $w_1 \leftarrow w_1 - \epsilon \frac{\partial f_{\text{MSE}}}{\partial w_1}$
        …
        Update w_6: $w_6 \leftarrow w_6 - \epsilon \frac{\partial f_{\text{MSE}}}{\partial w_6}$

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

Gradient descent: (any 3-layer NN)

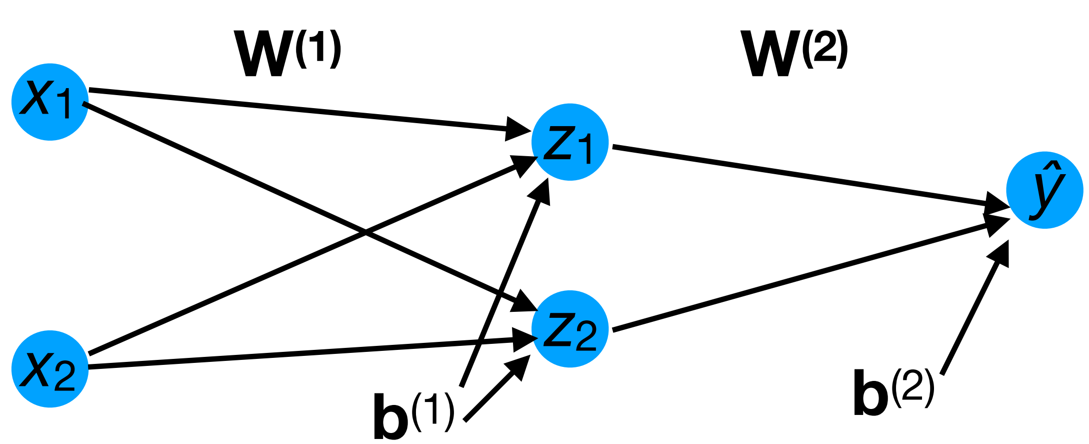
- It is more compact and efficient to represent and compute these gradients more abstractly:

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

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

- 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?

$$\begin{array}{ll} \frac{\partial f_1}{\partial x_1} = 2 & \frac{\partial f_1}{\partial x_2} = 1 \\ \frac{\partial f_2}{\partial x_1} = 0 & \frac{\partial f_2}{\partial x_2} = \pi \\ \frac{\partial f_3}{\partial x_1} = -1/x_2 & \frac{\partial f_3}{\partial x_2} = x_1/x_2^2 \end{array}$$

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?

$$\left[\begin{array}{ll} 2 & 1 \\ 0 & \pi \\ -1/x_2 & x_1/x_2^2 \end{array}\right]$$

Jacobian matrix

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*.

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

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 $\dfrac{\partial (f \circ g)}{\partial \mathbf{x}} = \dfrac{\partial f}{\partial g} \dfrac{\partial g}{\partial \mathbf{x}}$
- 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:

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

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:

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

Chain rule of multivariate calculus: example

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

- Substitution:

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

Chain rule of multivariate calculus: example

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

- Substitution:

$$\begin{aligned} f \circ g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) &= f\left(g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)\right) = x_1 + 2x_2 - 2(x_2) + (-x_1 + 3)/4 \\ &= \frac{3}{4}(x_1 + 1) \end{aligned}$$

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:

$$\begin{array}{rcl} f \circ g\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right) & = & f\left(g\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]\right)\right) = x_1 + 2x_2 - 2(x_2) + (-x_1 + 3)/4 \\ & = & \frac{3}{4}(x_1 + 1) \\ \implies \frac{\partial (f \circ g)}{\partial \mathbf{x}} & = & \left[\begin{array}{cc} \frac{3}{4} & 0 \end{array}\right] \end{array}$$

Chain rule of multivariate calculus: example

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

- Chain rule:

$$\frac{\partial f}{\partial g} = \begin{bmatrix} \boxed{?} \end{bmatrix}$$

1 x 3

Chain rule of multivariate calculus: example

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

- Chain rule:

$$\frac{\partial f}{\partial g} = \begin{bmatrix} 1 & -2 & \frac{1}{4} \end{bmatrix}$$ **1 x 3**

Chain rule of multivariate calculus: example

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

- Chain rule:

$$\frac{\partial f}{\partial g} = \begin{bmatrix} 1 & -2 & \frac{1}{4} \end{bmatrix} \quad \textbf{1 x 3}$$

$$\frac{\partial g}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \textbf{3 x 2}$$

Chain rule of multivariate calculus: example

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

- Chain rule:

$$\frac{\partial f}{\partial g} = \begin{bmatrix} 1 & -2 & \frac{1}{4} \end{bmatrix}$$ **1 x 3**

$$\frac{\partial g}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}$$ **3 x 2**

$$\implies \frac{\partial f \circ g}{\partial \mathbf{x}} =$$ **1 x 2**

Chain rule of multivariate calculus: example

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

- Chain rule:

$$\frac{\partial f}{\partial g} = \begin{bmatrix} 1 & -2 & \frac{1}{4} \end{bmatrix} \quad \textbf{1 x 3}$$

$$\frac{\partial g}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \textbf{3 x 2}$$

$$\implies \frac{\partial f \circ g}{\partial \mathbf{x}} = \begin{bmatrix} 1 & -2 & \frac{1}{4} \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{3}{4} & 0 \end{bmatrix}$$

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.

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\begin{array}{c} \frac{\partial f}{\partial \mathbf{x}_1} \\ \vdots \\ \frac{\partial f}{\partial \mathbf{x}_m} \end{array} \right]$$

Gradient

$$\frac{\partial f}{\partial \mathbf{x}} = \left[\begin{array}{ccc} \frac{\partial f}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f}{\partial \mathbf{x}_m} \end{array} \right]$$

Jacobian