Neural Networks
Part 2

Yin Li

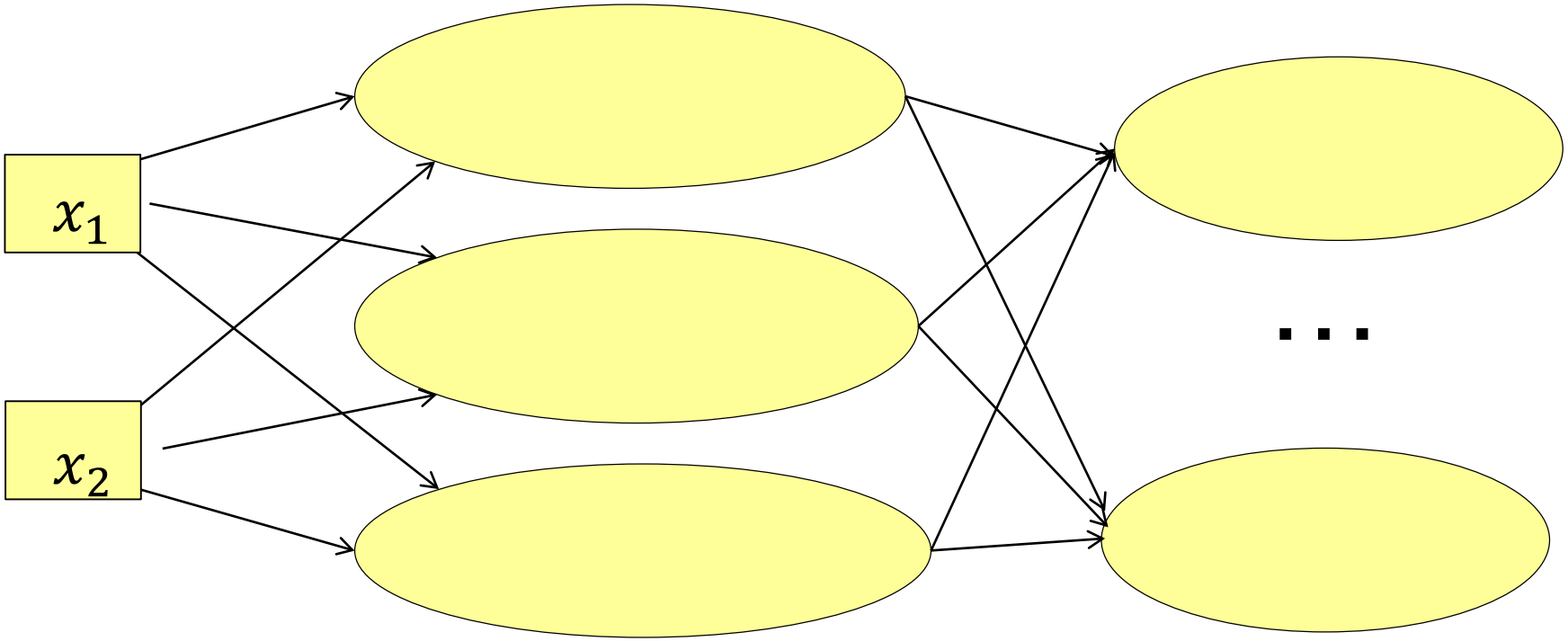
yin.li@wisc.edu

University of Wisconsin, Madison

[Some of the slides from Yingyu Liang, Jerry Zhu, Mohit Gupta]

Neural net for K-way classification

- Use K output units
- Training: encode a label y by an indicator vector
  - class1=(1,0,0,…,0), class2=(0,1,0,…,0) etc.
- Test: choose the class corresponding to the largest output unit



Mini-batch stochastic gradient descent

- Define a loss function $E_x = E(x, y)$
- Select a learning rate $\alpha > 0$
- Initialize the model parameters (edge weights) $w^{(0)}$
- For $t = 1, 2, \ldots$
  - Randomly sample a subset $\widehat{D}$ from D
  - Compute $\frac{\partial E_x}{\partial w}$ (per sample gradients w.r.t. w) for $x \in \widehat{D}$
  - Update the parameters

$$w^{(t)} = w^{(t-1)} - \alpha \frac{1}{|\widehat{D}|} \sum_{x \in \widehat{D}} \frac{\partial E_x}{\partial w}$$

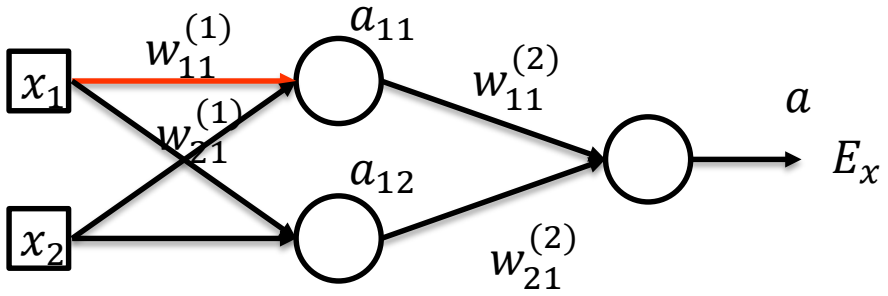
- Repeat until E converges

The key challenge is to compute $\frac{\partial E_x}{\partial w}$!

Back-propagation

- In theory

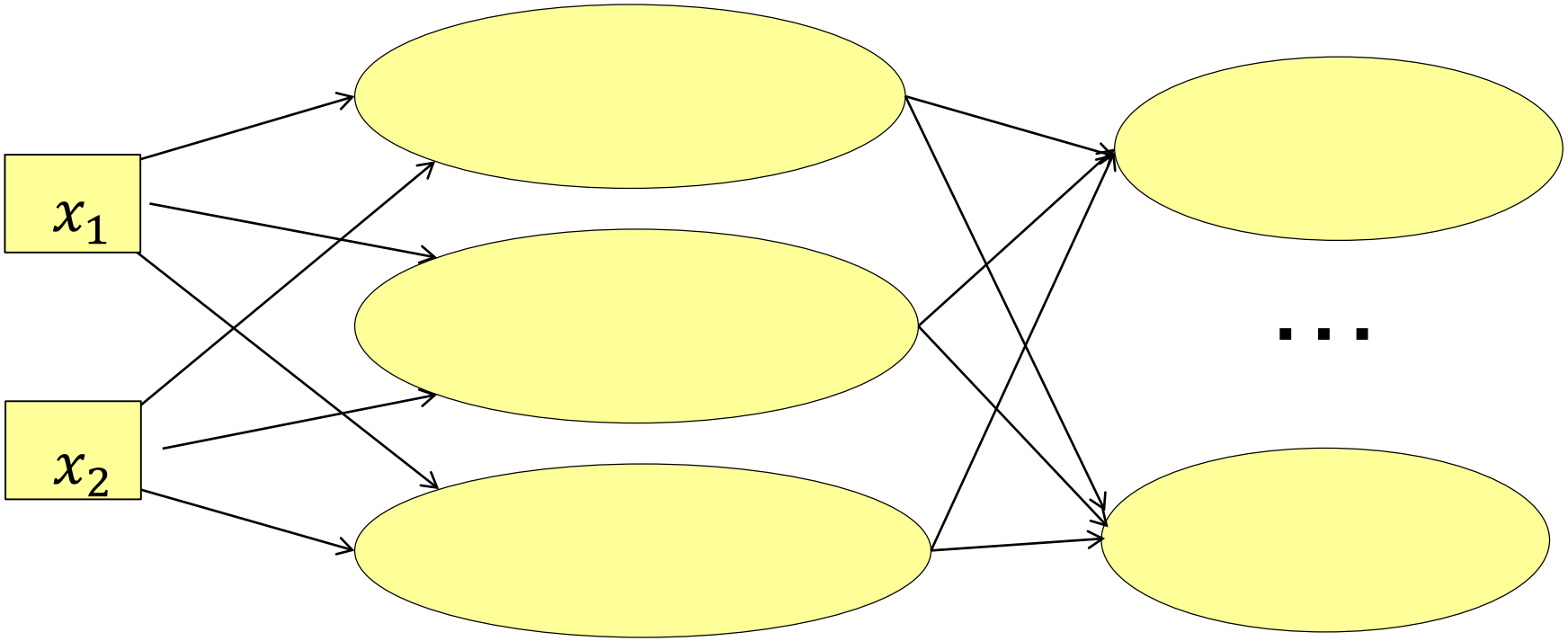
$$\frac{\partial E_x}{\partial w_{11}} = (a - y) w_{11}^{(2)} a_{11}(1 - a_{11}) x_1$$



- In practice
  - Define the model and the loss function
  - Select the optimization method (e.g., stochastic gradient descent)
  - Back-propagation handled by automatic differentiation
  - How?

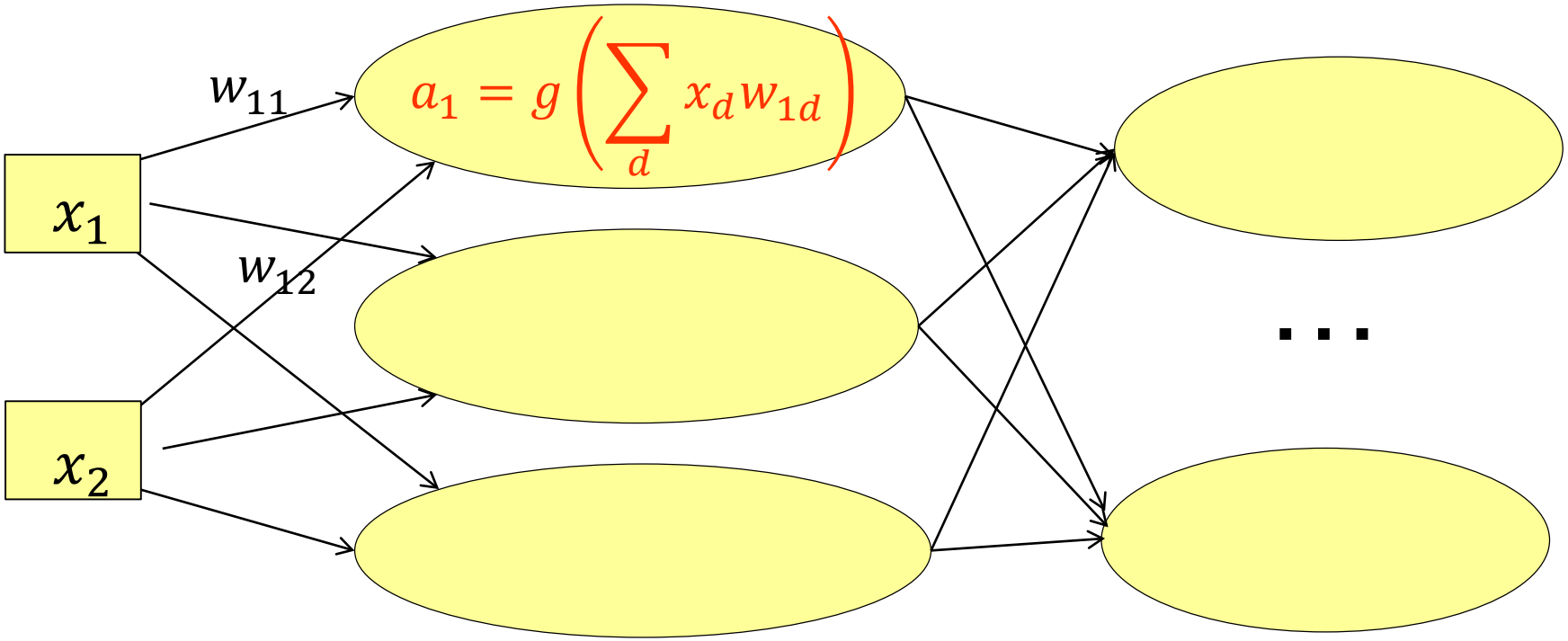
A single layer in neural network

- Let $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$



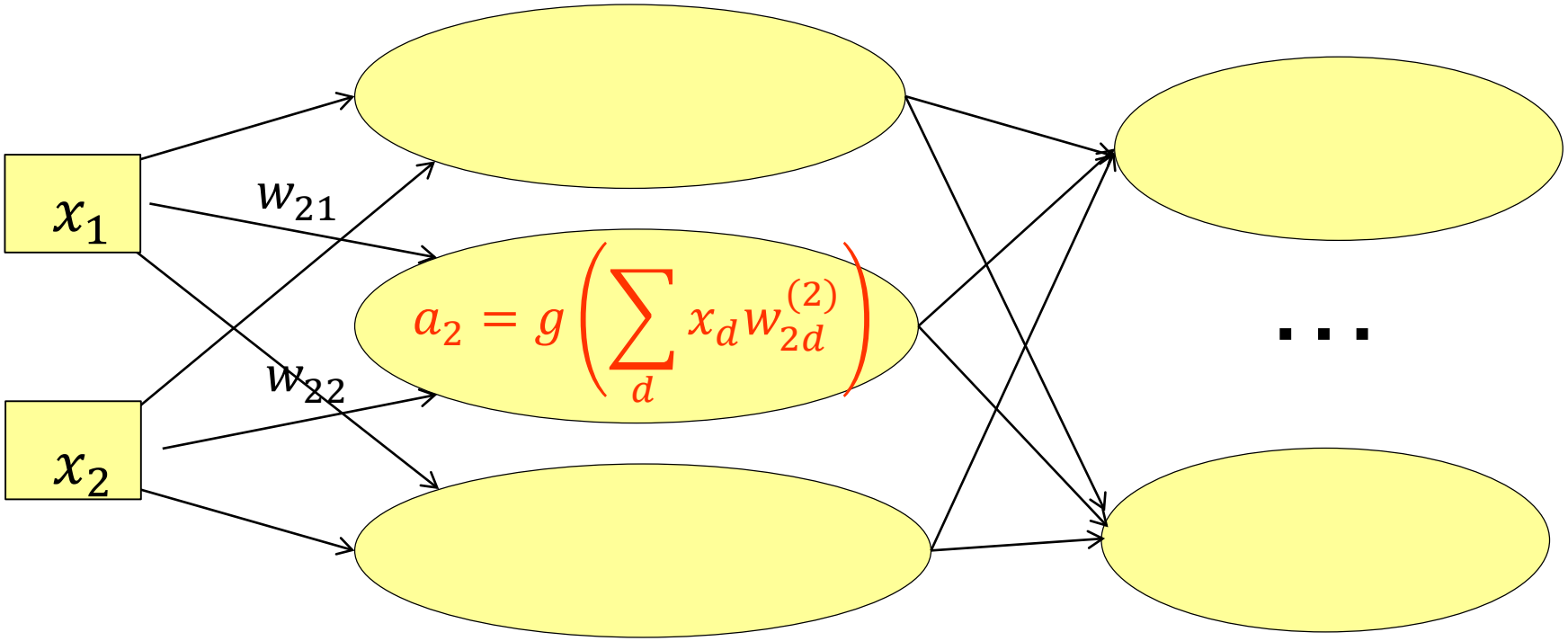
A single layer in neural network

- Let $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $a_1 = g\left(\begin{bmatrix} w_{11} & w_{12} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{10}\right) = g(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)$



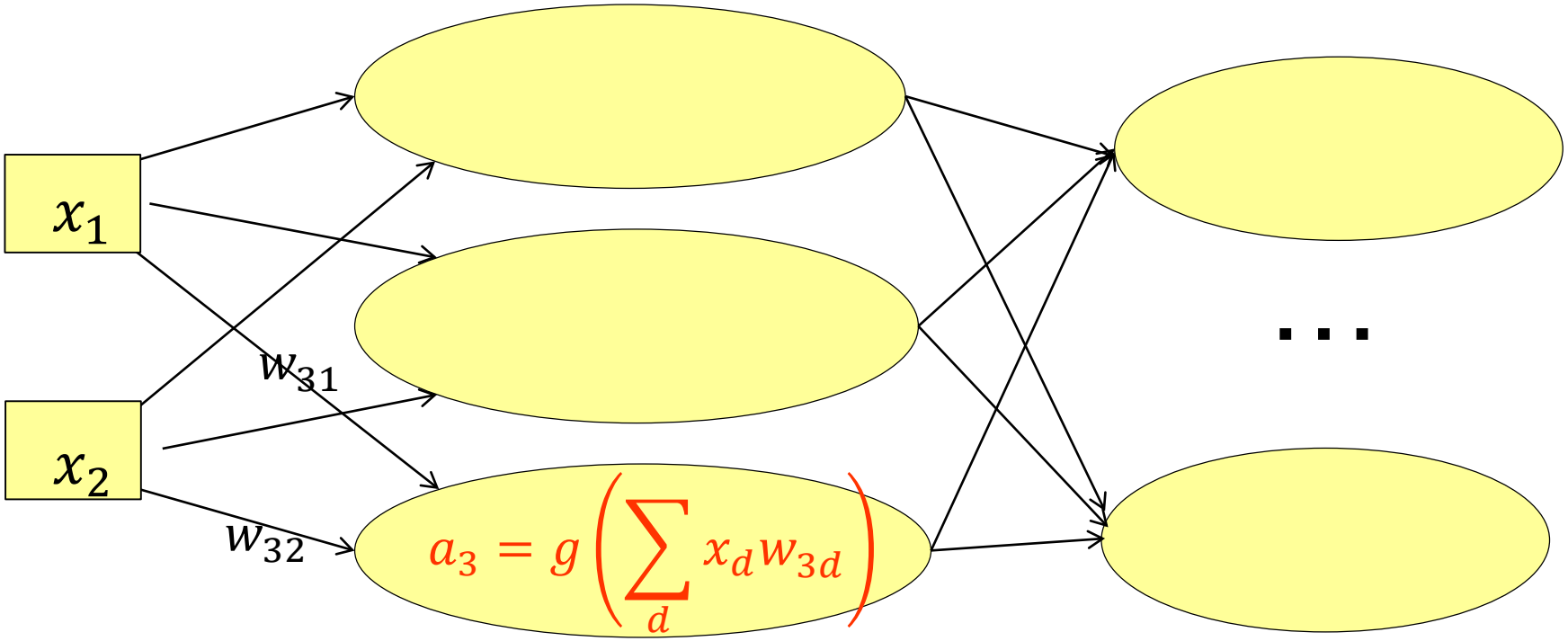
A single layer in neural network

- Let $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $a_2 = g\left(\begin{bmatrix} w_{21} & w_{22} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{20}\right) = g(\boldsymbol{w}_2^T \boldsymbol{x} + b_2)$



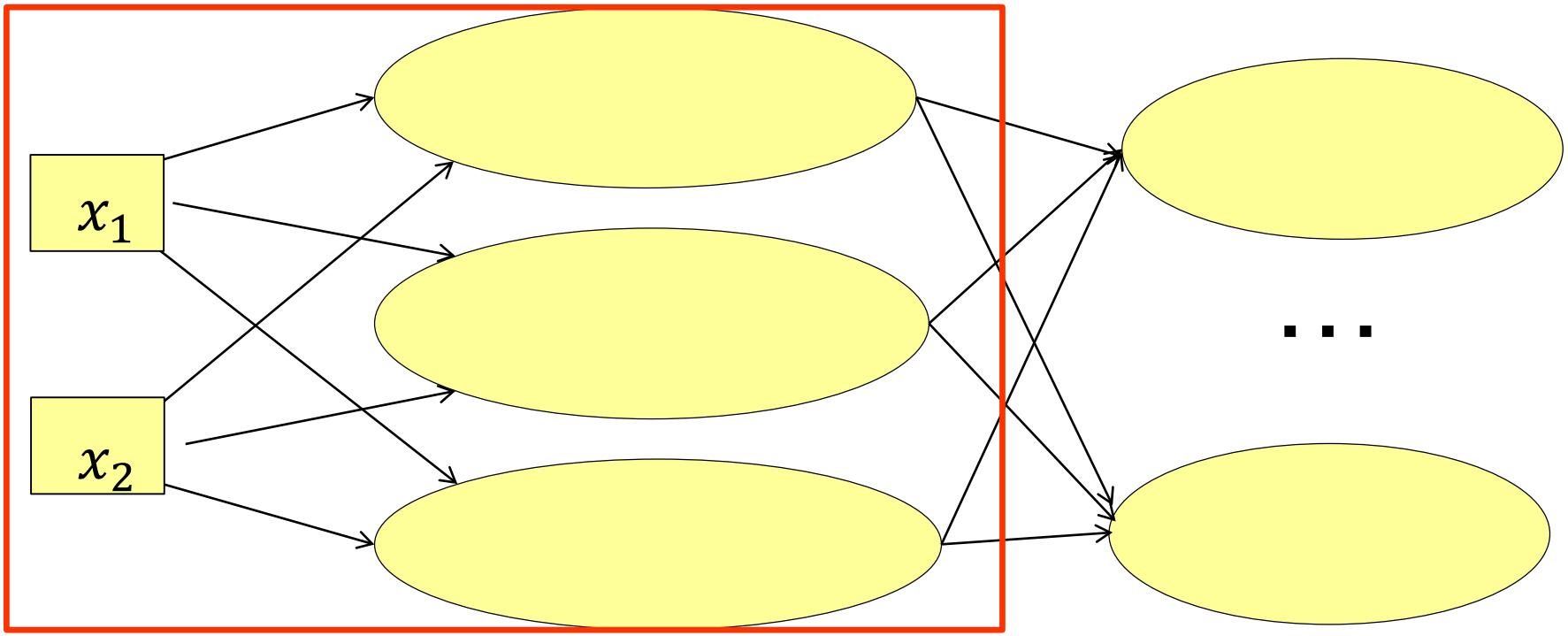
A single layer in neural network

- Let $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $a_3 = g\left(\begin{bmatrix} w_{31} & w_{32} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{30}\right) = g(\boldsymbol{w}_3^T \boldsymbol{x} + b_3)$



A single layer in neural network

- $a_1 = g\left([w_{11}\ \ w_{12}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{10}\right) = g(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)$
- $a_2 = g\left([w_{21}\ \ w_{22}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{20}\right) = g(\boldsymbol{w}_2^T \boldsymbol{x} + b_2)$
- $a_3 = g\left([w_{31}\ \ w_{32}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{30}\right) = g(\boldsymbol{w}_3^T \boldsymbol{x} + b_3)$



A single layer in neural network

- $a_1 = g\left([w_{11}\ \ w_{12}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{10}\right) = g(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)$
- $a_2 = g\left([w_{21}\ \ w_{22}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{20}\right) = g(\boldsymbol{w}_2^T \boldsymbol{x} + b_2)$
- $a_3 = g\left([w_{31}\ \ w_{32}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{30}\right) = g(\boldsymbol{w}_3^T \boldsymbol{x} + b_3)$

g: Any activation function that applies on every element of the input (element-wise operation)

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} g(\boldsymbol{w}_1^T \boldsymbol{x} + b_1) \\ g(\boldsymbol{w}_2^T \boldsymbol{x} + b_2) \\ g(\boldsymbol{w}_3^T \boldsymbol{x} + b_3) \end{bmatrix} = g(\begin{bmatrix} \boldsymbol{w}_1^T \boldsymbol{x} + b_1 \\ \boldsymbol{w}_2^T \boldsymbol{x} + b_2 \\ \boldsymbol{w}_3^T \boldsymbol{x} + b_3 \end{bmatrix})$$

A single layer in neural network

- $a_1 = g\left([w_{11}\ \ w_{12}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{10}\right) = g(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)$
- $a_2 = g\left([w_{21}\ \ w_{22}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{20}\right) = g(\boldsymbol{w}_2^T \boldsymbol{x} + b_2)$
- $a_3 = g\left([w_{31}\ \ w_{32}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{30}\right) = g(\boldsymbol{w}_3^T \boldsymbol{x} + b_3)$

Rearrange the linear operations

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = g\left(\begin{bmatrix} \boldsymbol{w}_1^T \boldsymbol{x} + b_1 \\ \boldsymbol{w}_2^T \boldsymbol{x} + b_2 \\ \boldsymbol{w}_3^T \boldsymbol{x} + b_3 \end{bmatrix}\right) = g\left(\begin{bmatrix} \boldsymbol{w}_1^T \\ \boldsymbol{w}_2^T \\ \boldsymbol{w}_3^T \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}\right)$$

A single layer in neural network

- $a_1 = g\left([w_{11}\ \ w_{12}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{10}\right) = g(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)$
- $a_2 = g\left([w_{21}\ \ w_{22}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{20}\right) = g(\boldsymbol{w}_2^T \boldsymbol{x} + b_2)$
- $a_3 = g\left([w_{31}\ \ w_{32}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{30}\right) = g(\boldsymbol{w}_3^T \boldsymbol{x} + b_3)$

Rearrange the linear operations

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = g\left(\begin{bmatrix} \boldsymbol{w}_1^T \\ \boldsymbol{w}_2^T \\ \boldsymbol{w}_3^T \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}\right) = g(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b})$$

A single layer in neural network

- $a_1 = g\left([w_{11}\ \ w_{12}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{10}\right) = g(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)$
- $a_2 = g\left([w_{21}\ \ w_{22}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{20}\right) = g(\boldsymbol{w}_2^T \boldsymbol{x} + b_2)$
- $a_3 = g\left([w_{31}\ \ w_{32}] * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{30}\right) = g(\boldsymbol{w}_3^T \boldsymbol{x} + b_3)$

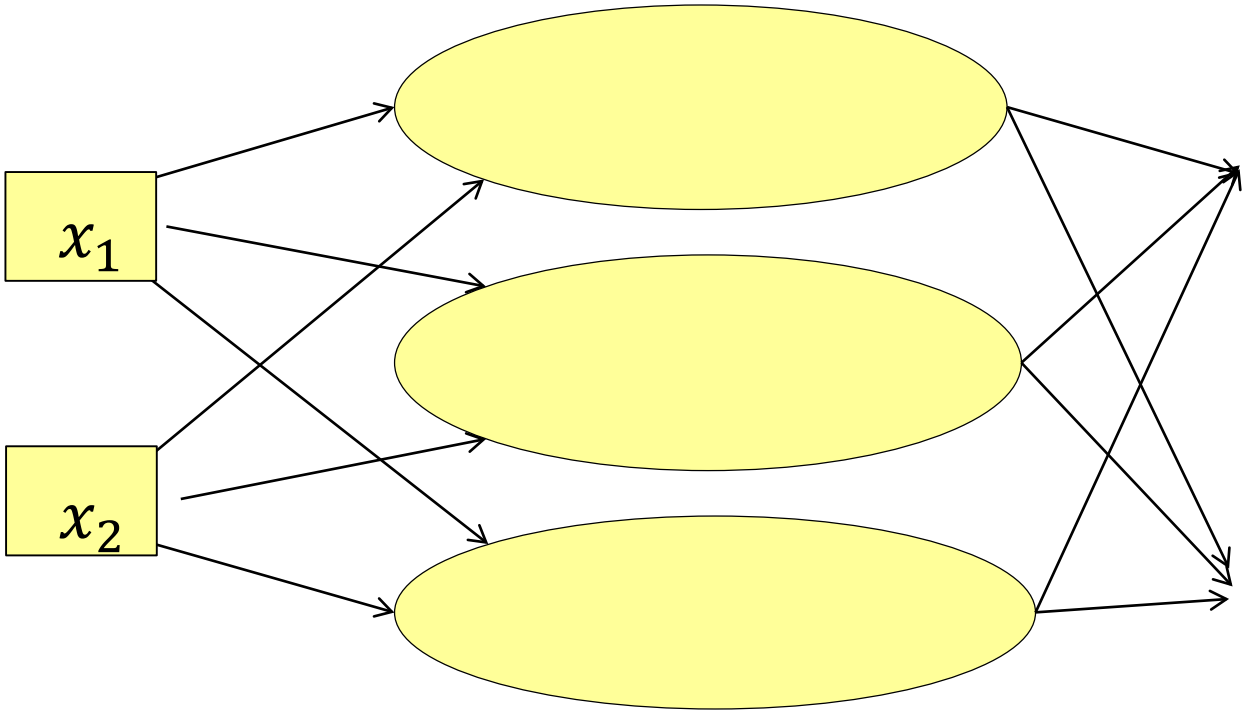
Here is what we have

$$\boldsymbol{a} = g\left(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}\right)$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \boldsymbol{W} = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} w_{10} \\ w_{20} \\ w_{30} \end{bmatrix}$$

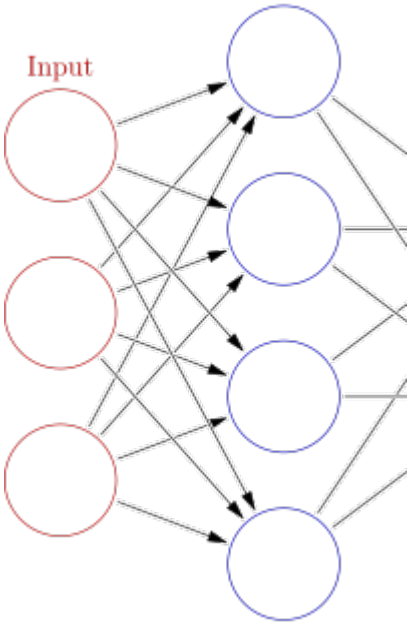
A single layer in neural network

- $\boldsymbol{a} = g(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b})$
- Work for any element-wise activation function g



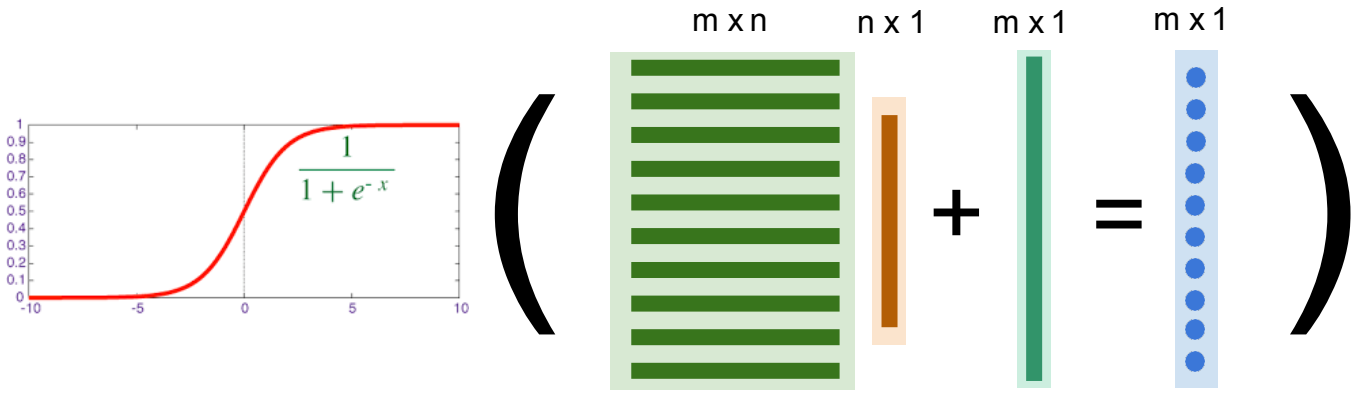
A single layer in neural network

- $\boldsymbol{a} = g(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b})$
- Work for any element-wise activation function g
- Work for any input / output dimensions
- Map an input $\boldsymbol{x} \in R^n$ to an output $\boldsymbol{a} \in R^m$
- $\boldsymbol{x} \in R^n,\ \boldsymbol{W} \in R^{n \times m},\ \boldsymbol{b} \in R^m,\ \boldsymbol{a} \in R^m$



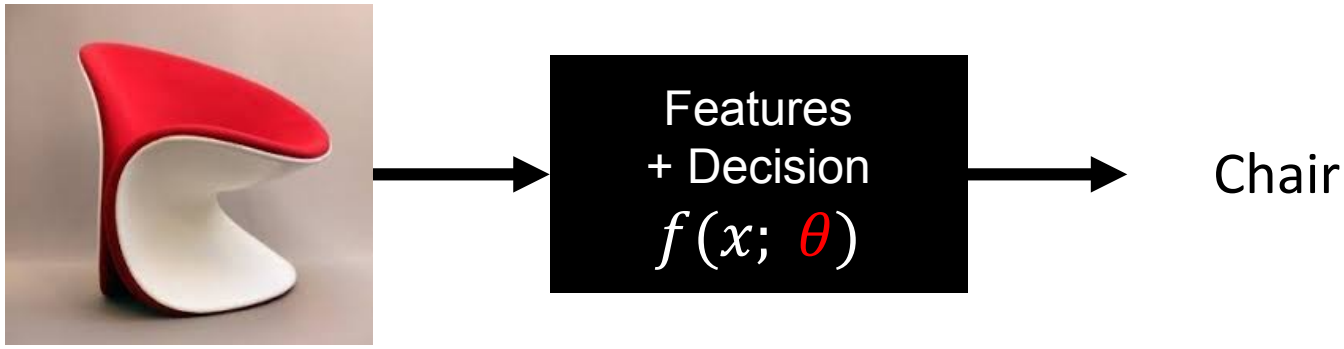
A single layer in neural network

- $\boldsymbol{a} = g(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b})$
- Work for any element-wise activation function g
- Work for any input / output dimensions
- Map an input $\boldsymbol{x} \in R^n$ to an output $\boldsymbol{a} \in R^m$
- $\boldsymbol{x} \in R^n,\ \boldsymbol{W} \in R^{n \times m},\ \boldsymbol{b} \in R^m,\ \boldsymbol{a} \in R^m$



Neural Networks / Deep Learning

- What type of functions shall we consider for *f?*



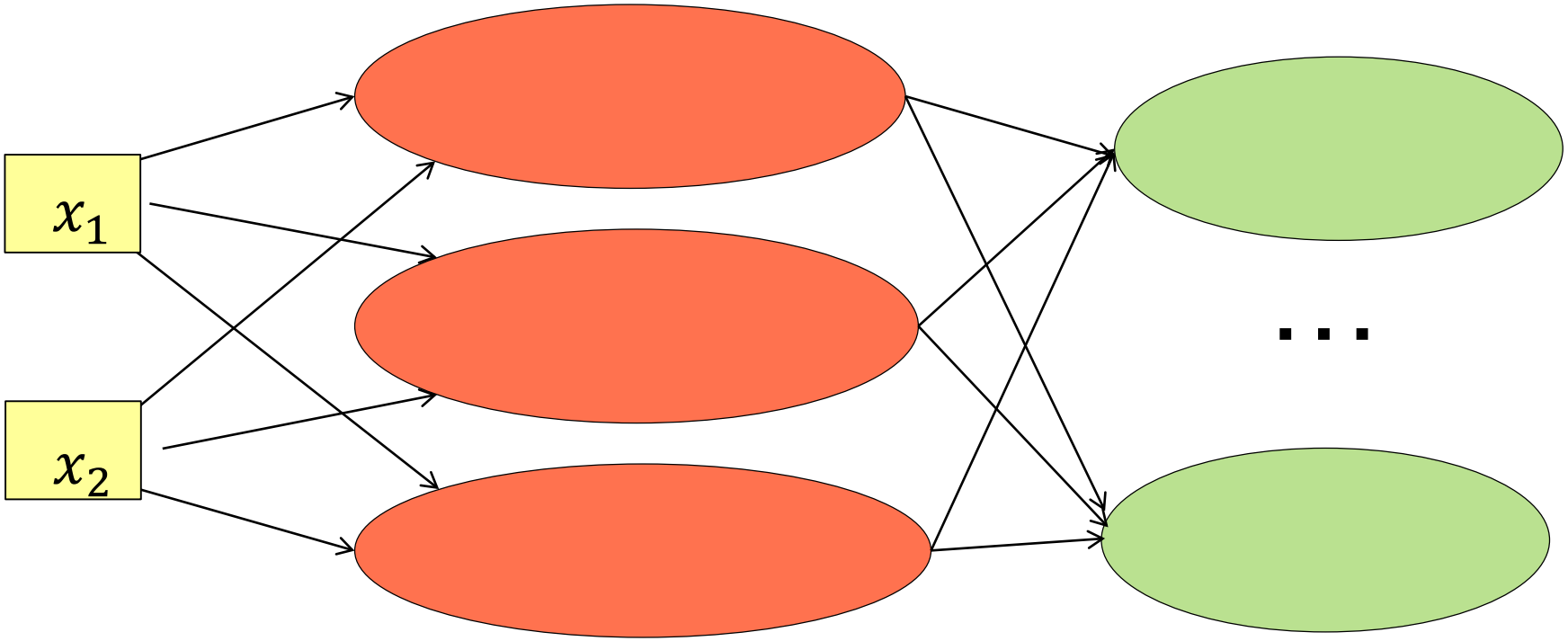
Proposal: Composing a set of (nonlinear) functions *g*

$$f(\boldsymbol{x};\boldsymbol{\theta}) = g_1(\dots g_{n-1}(g_n(\boldsymbol{x};\boldsymbol{\theta_n}),\boldsymbol{\theta_{n-1}})\dots,\boldsymbol{\theta_1})$$

Example: $\boldsymbol{a} = sigmoid(\boldsymbol{W^T x} + \boldsymbol{b}) = g(\boldsymbol{x};\boldsymbol{W},\boldsymbol{b})$

An example neural network

- Let $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $f(\boldsymbol{x}; \boldsymbol{\theta}) = g_1(g_2(\boldsymbol{x}; \boldsymbol{\theta_2}), \boldsymbol{\theta_1}))$

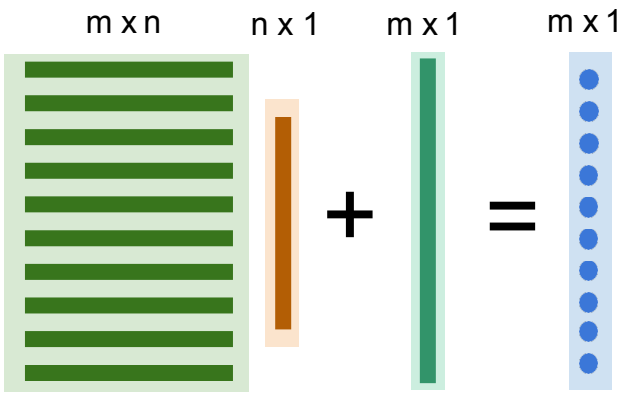


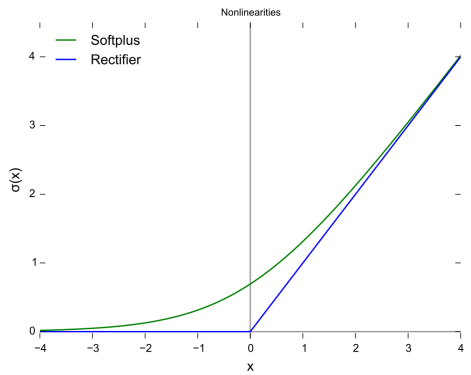
Neural Networks / Deep Learning

- (Deep) Neural Network:
  Composing a set of (nonlinear) functions *g*

$$f(\boldsymbol{x};\boldsymbol{\theta}) = g_1(\ldots g_{n-1}(g_n(\boldsymbol{x};\boldsymbol{\theta_n}),\boldsymbol{\theta_{n-1}})\ldots,\boldsymbol{\theta_1})$$

- Key Elements:
  Linear operations + Nonlinear activations



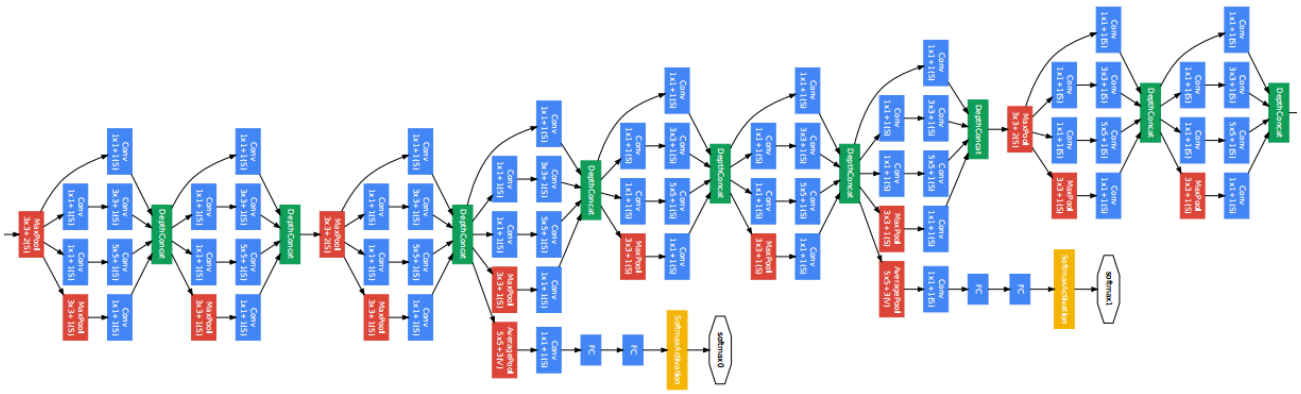


Neural Networks / Deep Learning

- (Deep) Neural Network:
  Composing a set of (nonlinear) functions g

$$f(\boldsymbol{x};\boldsymbol{\theta}) = g_1(\dots g_{n-1}(g_n(\boldsymbol{x};\boldsymbol{\theta_n}),\boldsymbol{\theta_{n-1}})\dots,\boldsymbol{\theta_1})$$

- Key Challenge:
  How to present the composition of nonlinear functions & make the computation possible and efficient?

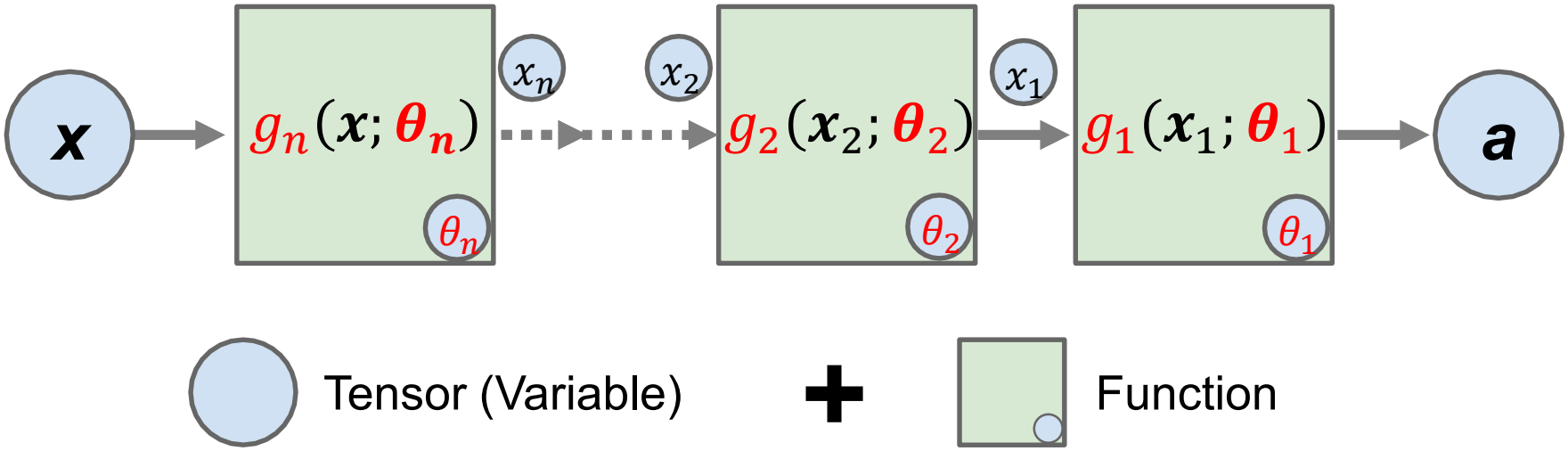


Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Neural network as a stack of layers

- (Deep) Neural Network:
  Composing a set of (nonlinear) functions *g*

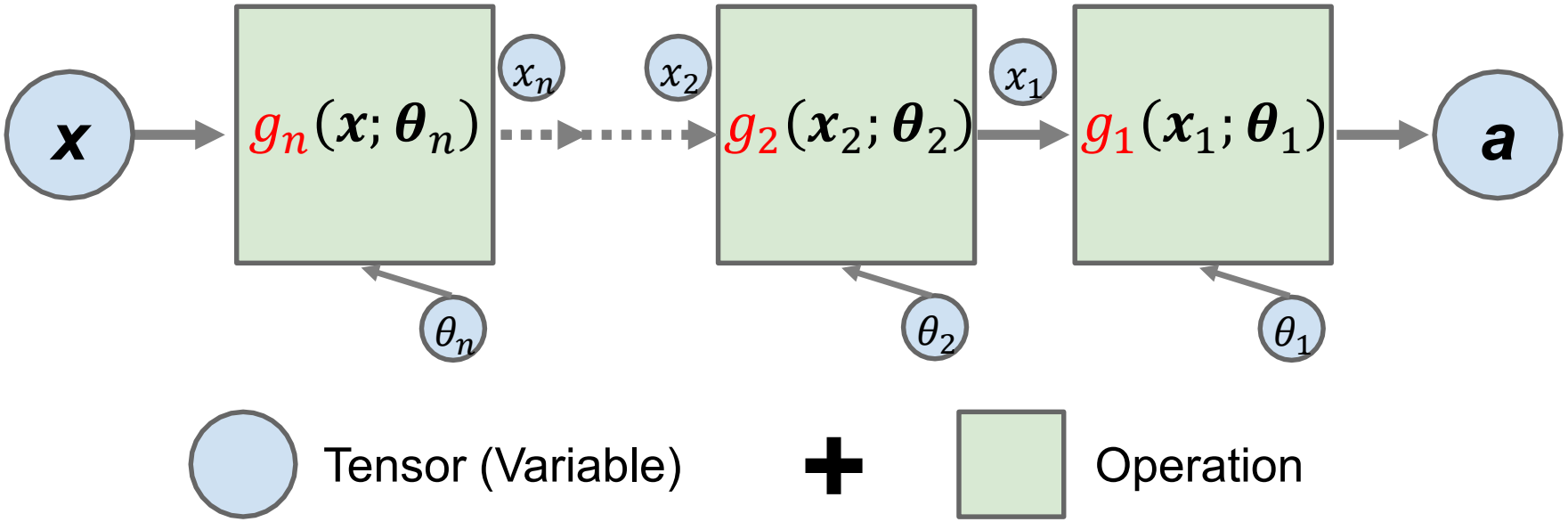
$$f(\boldsymbol{x};\boldsymbol{\theta}) = g_1(\dots g_{n-1}(g_n(\boldsymbol{x};\boldsymbol{\theta_n}),\boldsymbol{\theta_{n-1}})\dots,\boldsymbol{\theta_1})$$



Neural network as variables + operations

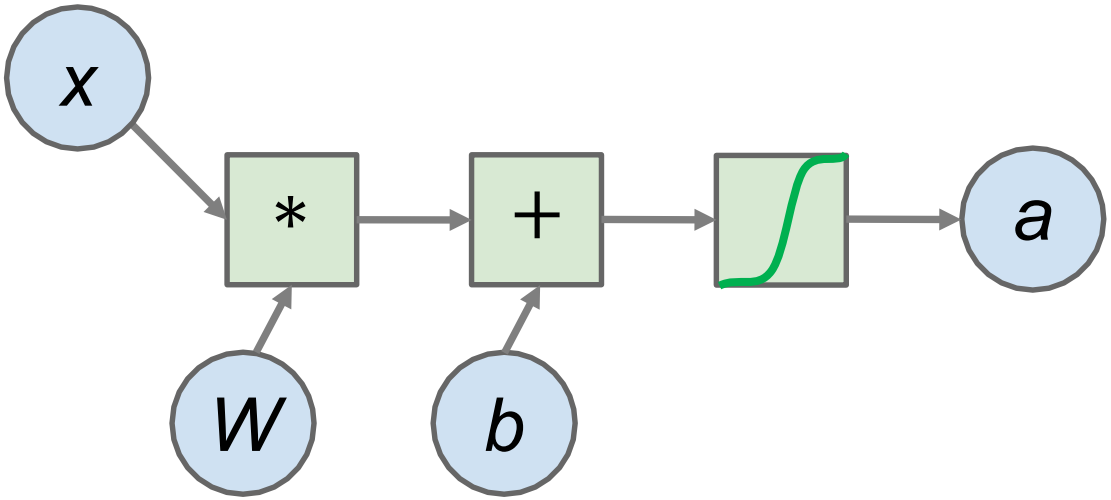
- (Deep) Neural Network:
  Composing a set of (nonlinear) functions *g*

$$f(\boldsymbol{x};\boldsymbol{\theta}) = g_1(\ldots g_{n-1}(g_n(\boldsymbol{x};\boldsymbol{\theta_n}),\boldsymbol{\theta_{n-1}})\ldots,\boldsymbol{\theta_1})$$



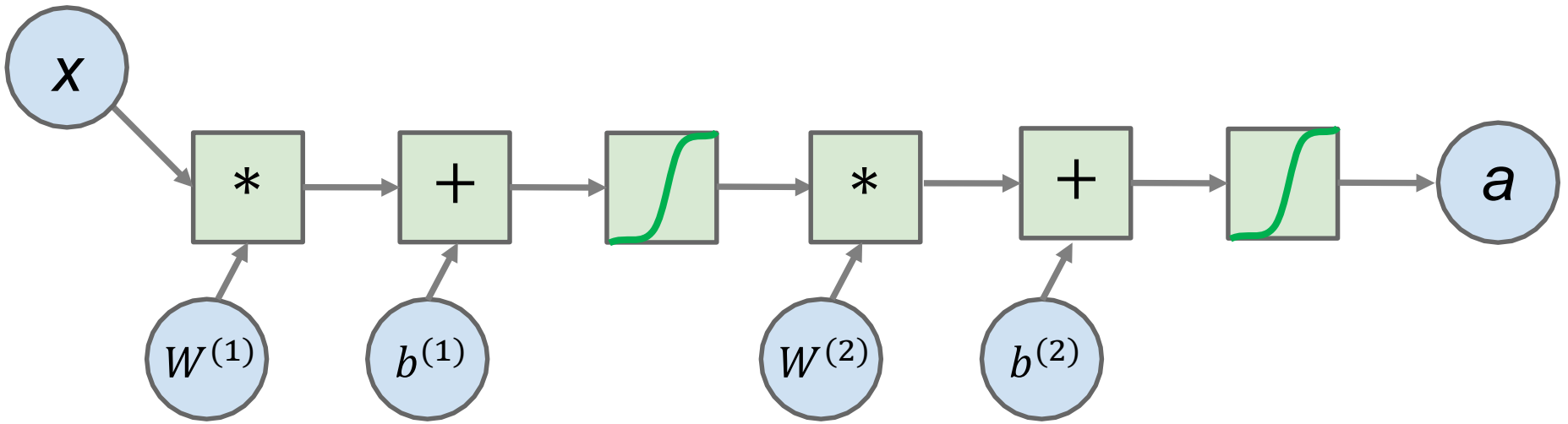
Neural network as variables + operations

- $\boldsymbol{a} = sigmoid(\boldsymbol{W^T x} + \boldsymbol{b})$
- Decompose functions into atomic operations
- Separate data (variables) and computing (operations)
- Known as a computational graph



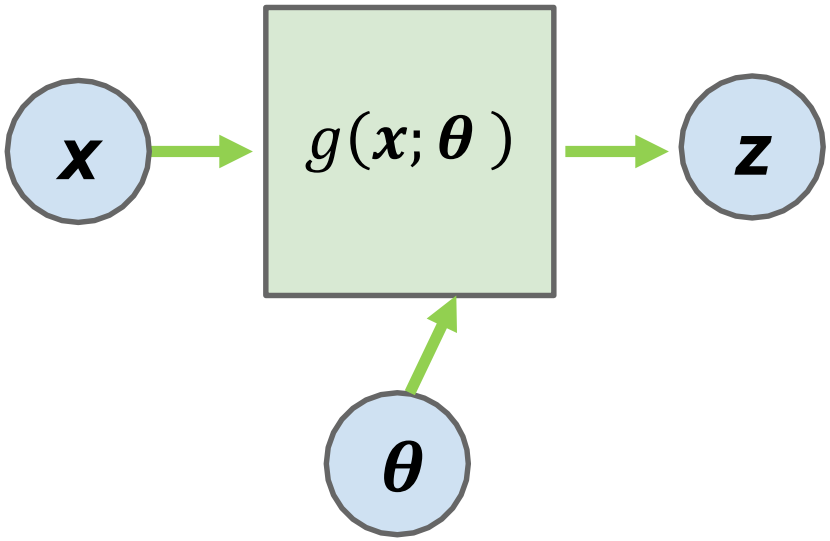
Neural network as variables + operations

- A two-layer neural network



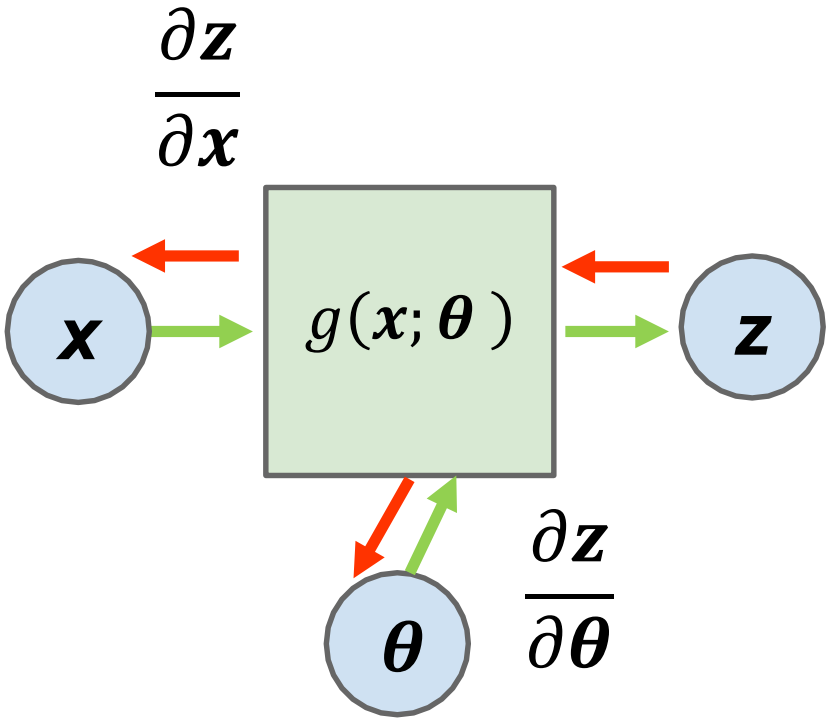
Neural network as variables + operations

- Differentiable operations
- Forward / backward



Neural network as variables + operations

- Differentiable operations
- Forward / backward

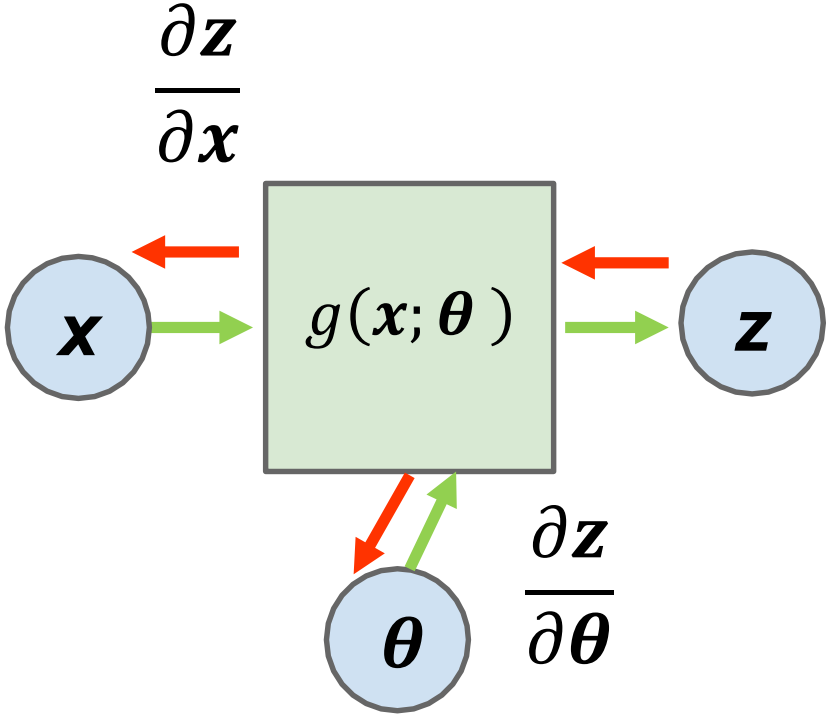


Neural network as variables + operations

- Differentiable operations
- Examples

$$z = g(\boldsymbol{x}) = sigmoid(\boldsymbol{x})$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \boldsymbol{z}(1 - \boldsymbol{z})$$



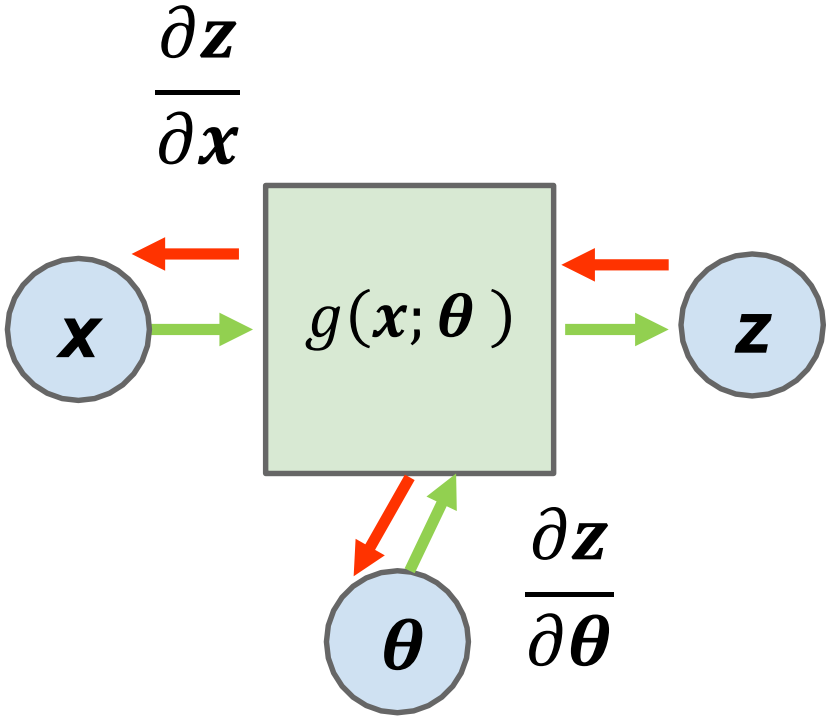
Neural network as variables + operations

- Differentiable operations
- Examples

$$z = g(\boldsymbol{x}; \boldsymbol{b}) = \boldsymbol{x} + \boldsymbol{b}$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \mathbf{1}$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}} = \mathbf{1}$$



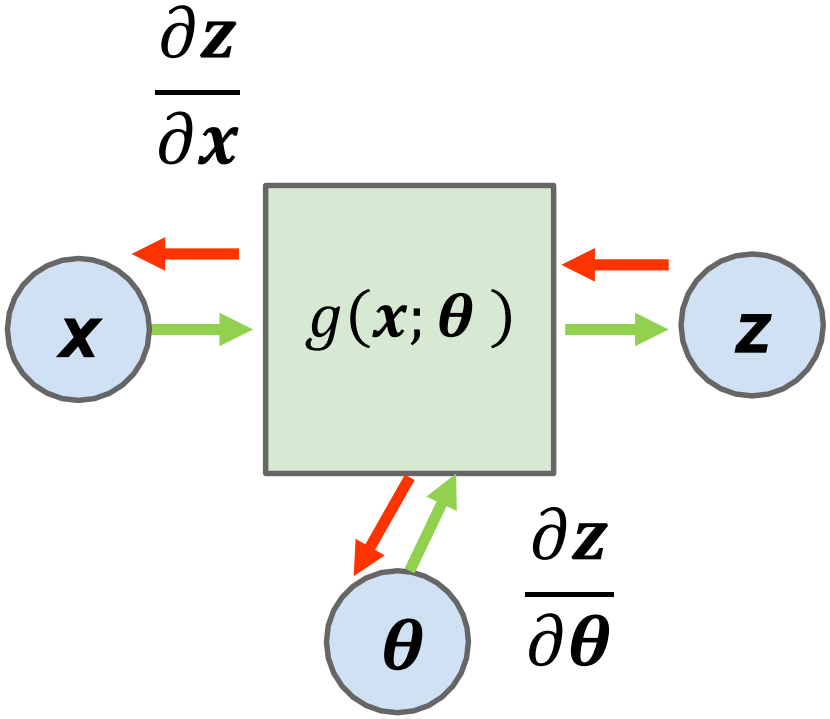
Neural network as variables + operations

- Differentiable operations
- Examples

$$g(\boldsymbol{x}; \boldsymbol{\theta} = \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \boldsymbol{w}$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{w}} = \boldsymbol{x}$$



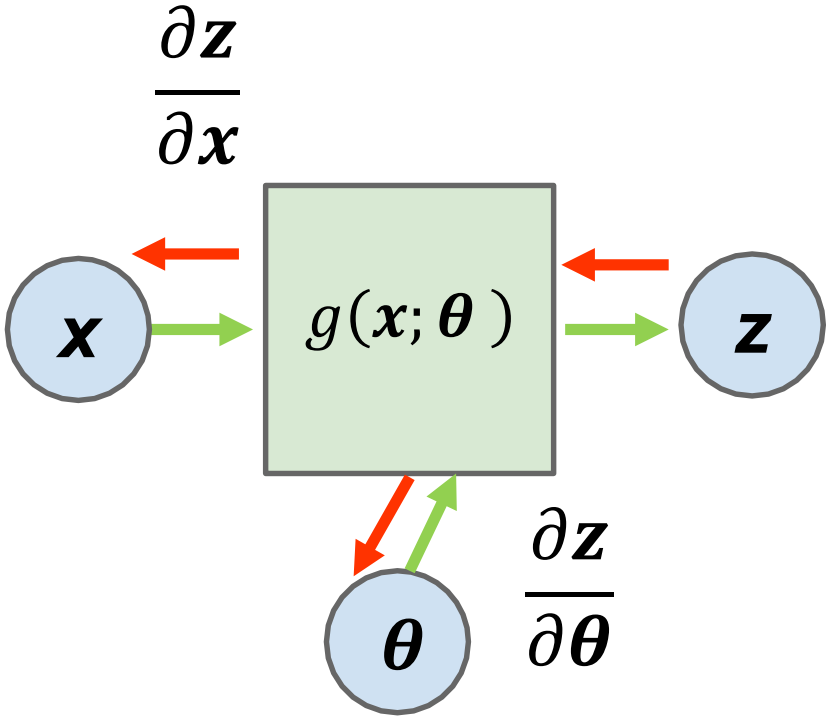
Neural network as variables + operations

- Differentiable operations
- Examples

$$g(\boldsymbol{x}; \boldsymbol{\theta} = \boldsymbol{W}) = \boldsymbol{W}^T \boldsymbol{x}$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = ? \qquad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}} = ?$$

$$\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_1^T \\ \boldsymbol{w}_2^T \\ \boldsymbol{w}_3^T \end{bmatrix} \boldsymbol{x}$$



Neural network as variables + operations

- Differentiable operations
- Examples

$$g(\boldsymbol{x}; \boldsymbol{\theta} = \boldsymbol{W}) = \boldsymbol{W}^T \boldsymbol{x}$$

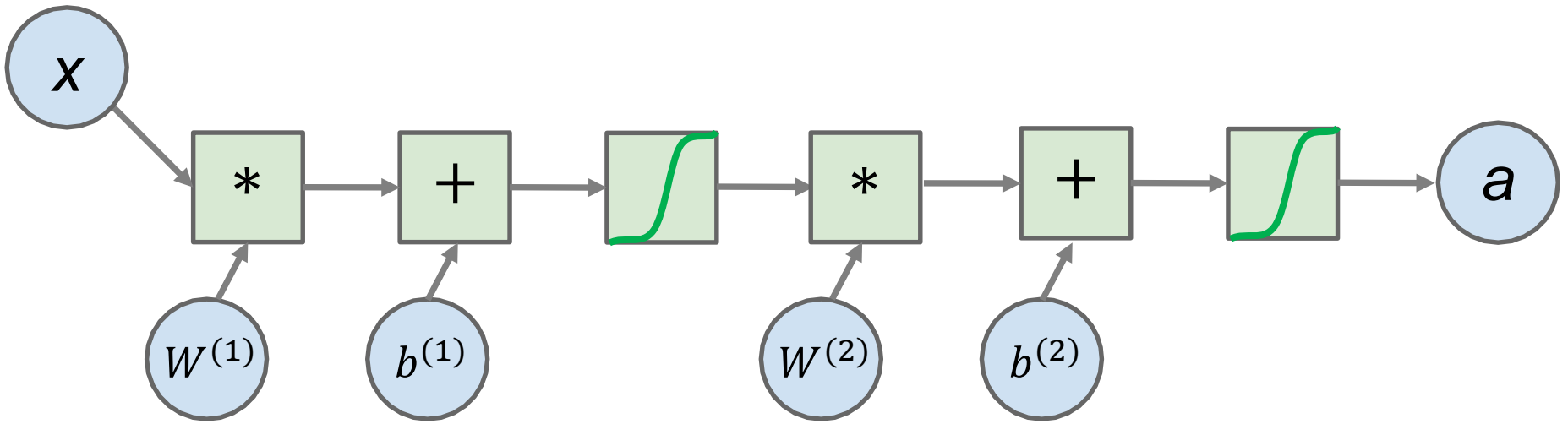
$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = ? \qquad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}} = ?$$

$$\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_1^T \\ \boldsymbol{w}_2^T \\ \boldsymbol{w}_3^T \end{bmatrix} \boldsymbol{x}$$

- Loop over each output value z_i
- Compute $\frac{\partial z_i}{\partial \boldsymbol{x}} = \boldsymbol{w_i}$ and $\frac{\partial z_i}{\partial \boldsymbol{w_i}} = \boldsymbol{x}$
- Rearrange them into the gradient matrix

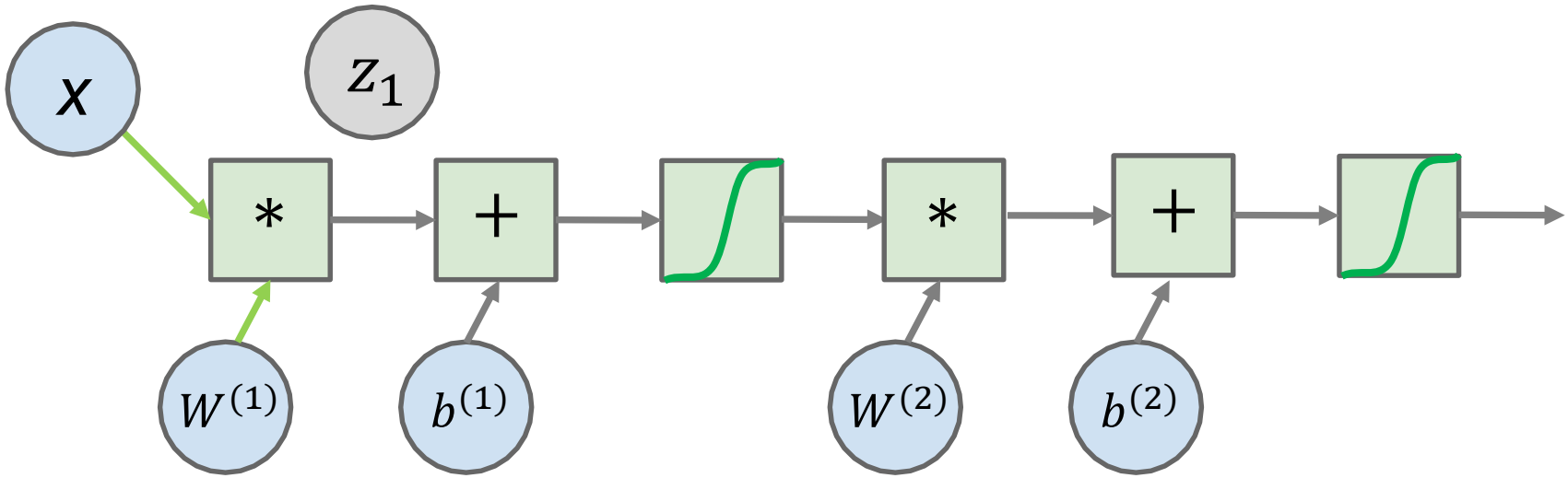
Neural network as a computational graph

- A two-layer neural network
- Forward propagation vs. backward propagation



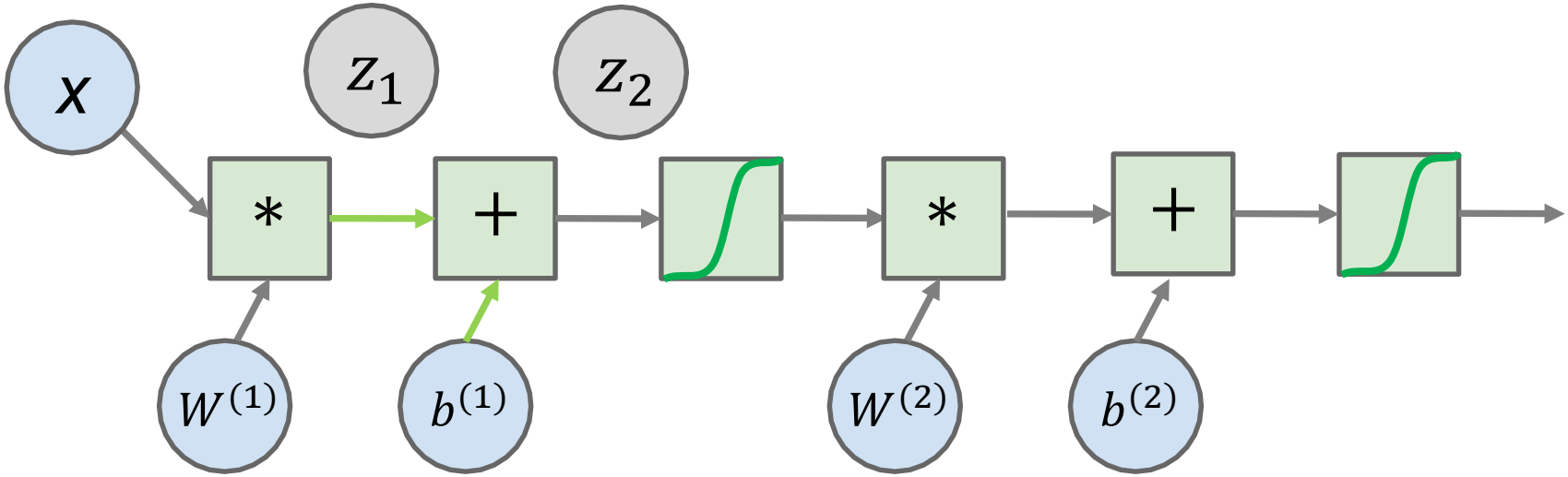
Neural network: forward propagation

- A two-layer neural network
- Intermediate variables Z



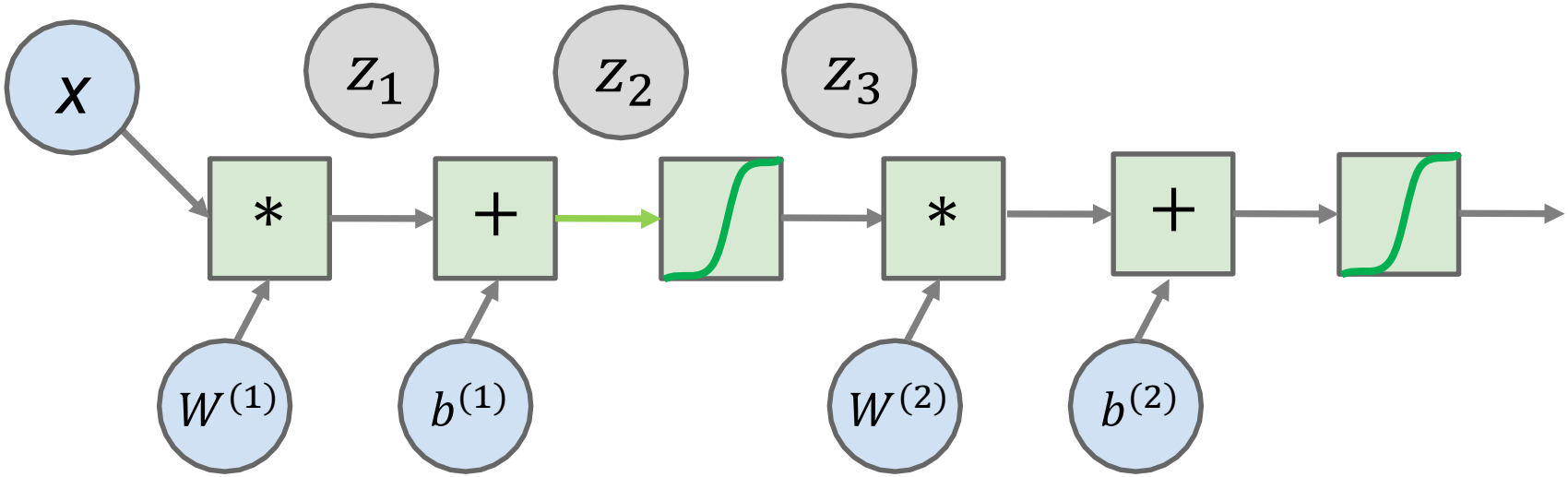
Neural network: forward propagation

- A two-layer neural network
- Intermediate variables Z



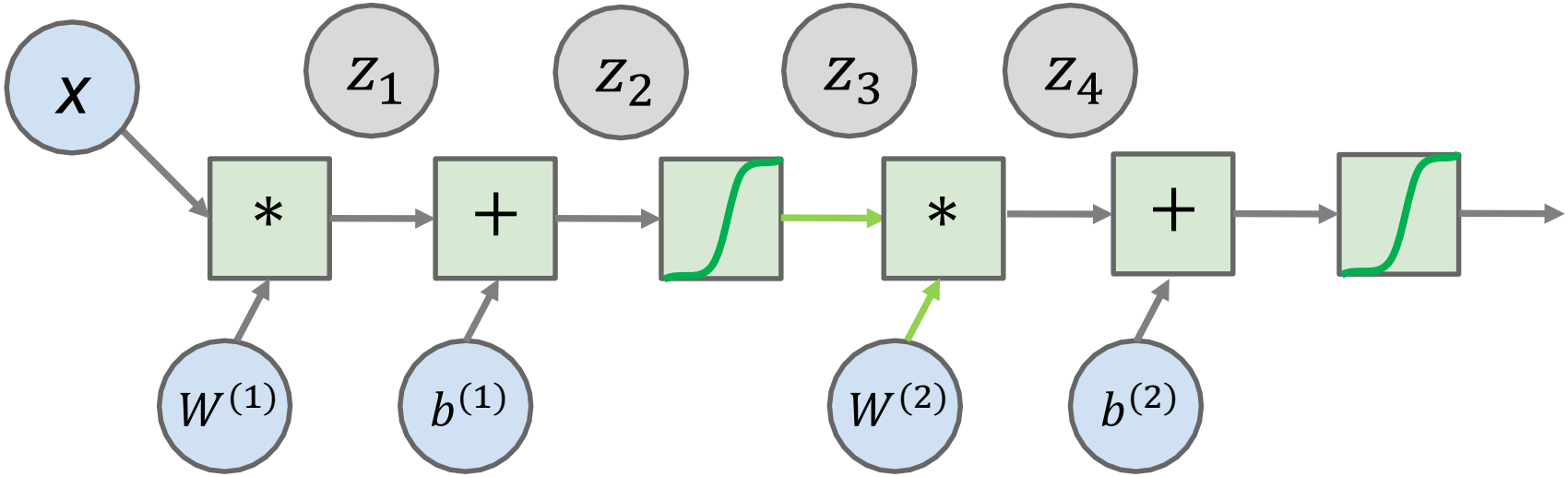
Neural network: forward propagation

- A two-layer neural network
- Intermediate variables *Z*



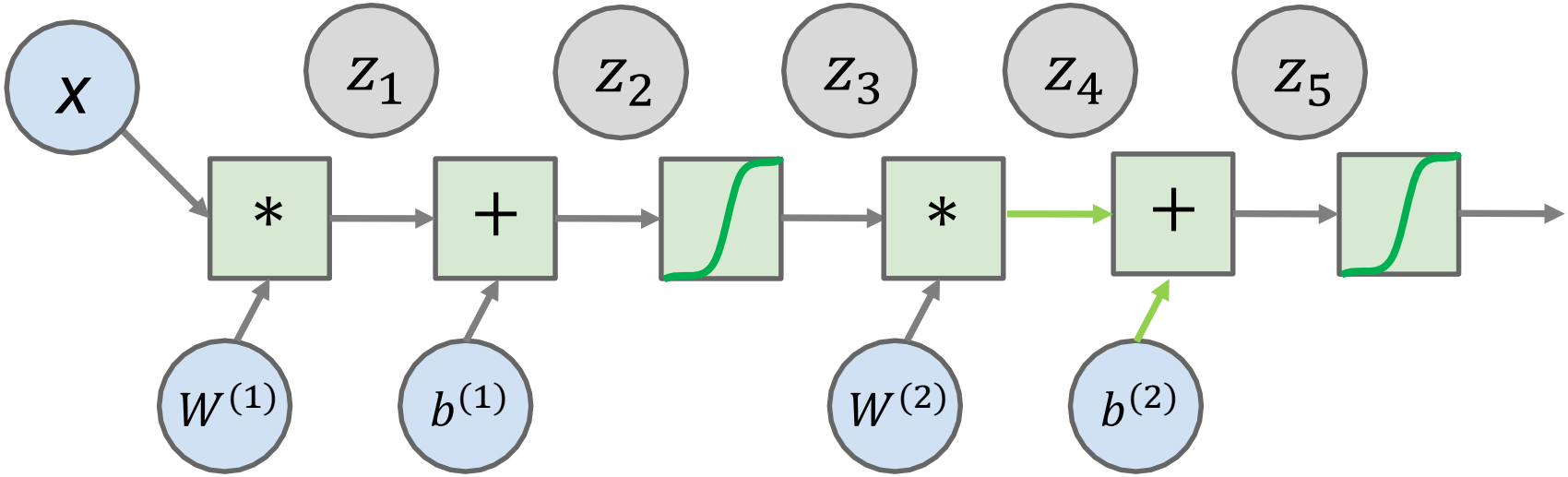
Neural network: forward propagation

- A two-layer neural network
- Intermediate variables *Z*



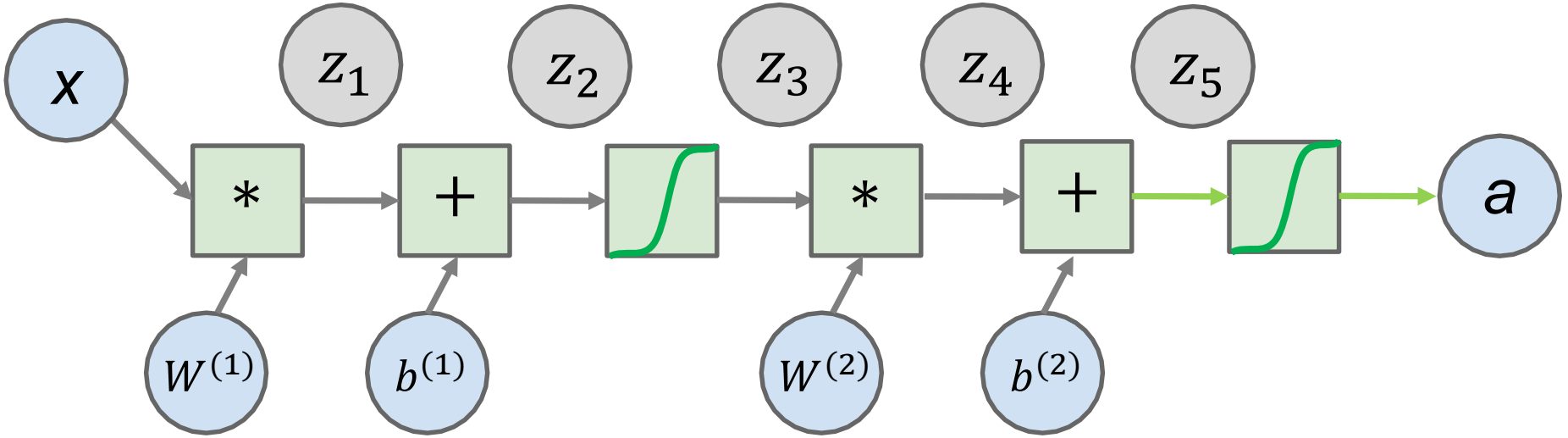
Neural network: forward propagation

- A two-layer neural network
- Intermediate variables *Z*



Neural network: forward propagation

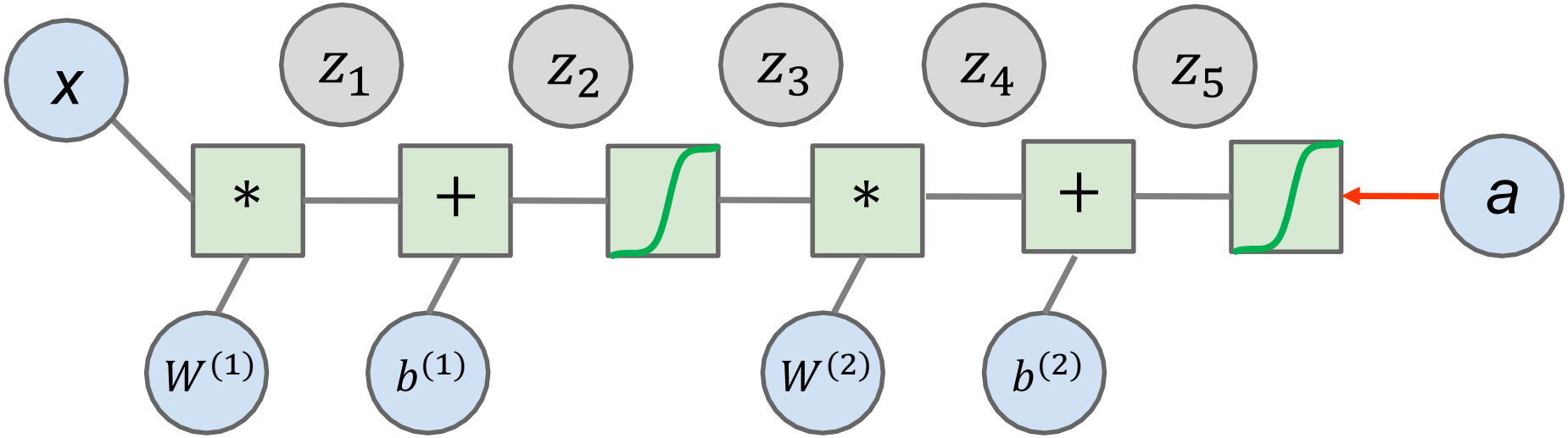
- A two-layer neural network
- Intermediate variables *Z*
- Forward propagation computes the output *a*



Neural network: backward propagation

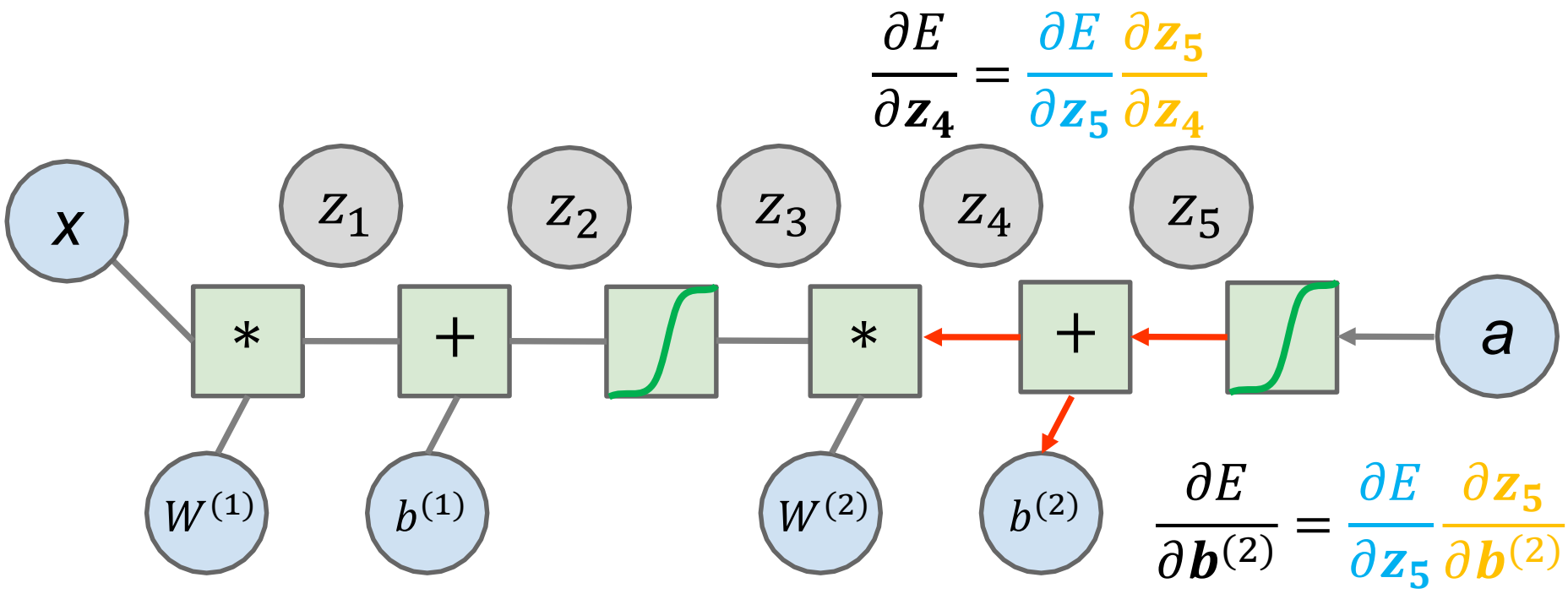
- A two-layer neural network
- Assuming forward propagation is done
- Have all intermediate variables *Z*
- Minimize a loss function *E*

$$\frac{\partial E}{\partial \boldsymbol{z_5}} = \frac{\partial E}{\partial \boldsymbol{a}} \frac{\partial \boldsymbol{a}}{\partial z_5}$$



Neural network: backward propagation

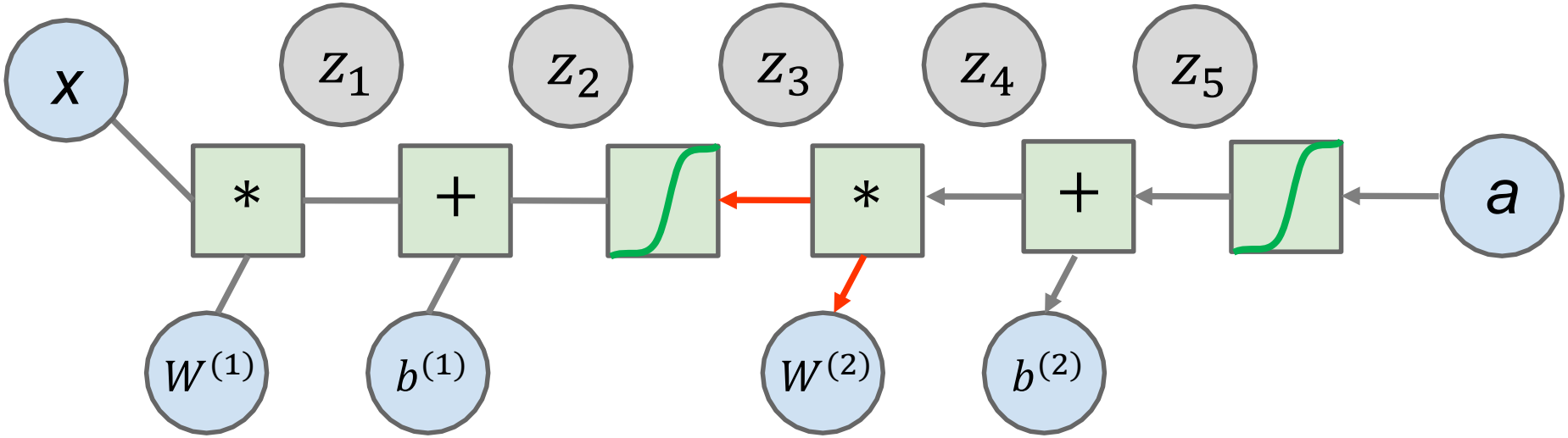
- A two-layer neural network
- Assuming forward propagation is done



Neural network: backward propagation

- A two-layer neural network
- Assuming forward propagation is done

$$\frac{\partial E}{\partial \boldsymbol{z_3}} = \frac{\partial E}{\partial \boldsymbol{z_4}} \frac{\partial \boldsymbol{z_4}}{\partial \boldsymbol{z_3}}$$



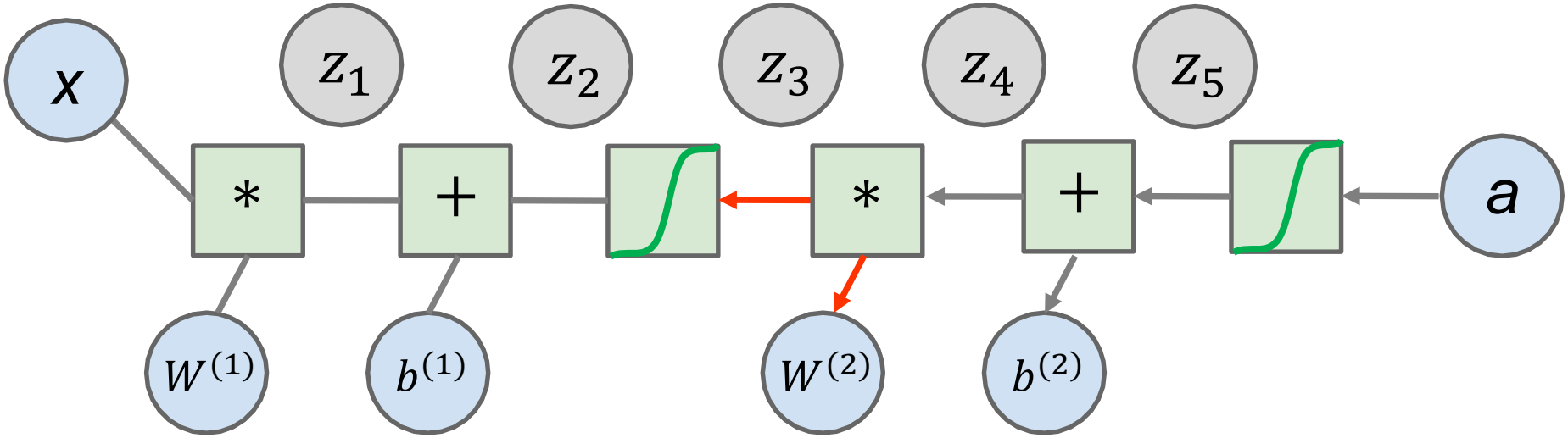
$$\frac{\partial E}{\partial \boldsymbol{W}^{(2)}} = \frac{\partial E}{\partial \boldsymbol{z_4}} \frac{\partial \boldsymbol{z_4}}{\partial \boldsymbol{W}^{(2)}}$$

Neural network: backward propagation

- Gradient to a variable =

gradient on the top x gradient from the current operation

$$\frac{\partial E}{\partial \boldsymbol{z_3}} = \frac{\partial E}{\partial \boldsymbol{z_4}} \frac{\partial \boldsymbol{z_4}}{\partial \boldsymbol{z_3}}$$



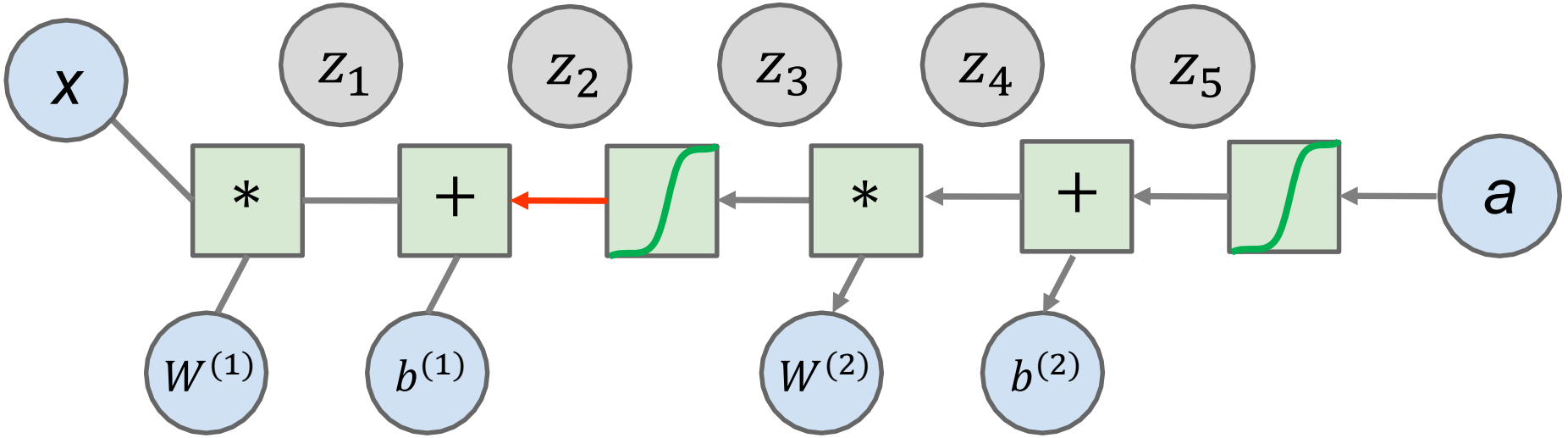
$$\frac{\partial E}{\partial \boldsymbol{W}^{(2)}} = \frac{\partial E}{\partial \boldsymbol{z_4}} \frac{\partial \boldsymbol{z_4}}{\partial \boldsymbol{W}^{(2)}}$$

Neural network: backward propagation

- Gradient to a variable =

gradient on the top x gradient from the current operation

$$\frac{\partial E}{\partial \mathbf{z_2}} = \frac{\partial E}{\partial \mathbf{z_3}} \frac{\partial \mathbf{z_3}}{\partial \mathbf{z_2}}$$



Back propagation: a modern treatment

- Define a neural network as a computational graph
  - Must be a *directed* graph
  - Nodes as *variables* and *operations*
  - All operations must be differentiable

Back propagation: a modern treatment

- Define a neural network as a computational graph
- Run topological sort to determine the order of execution of the operations (input -> output)
- Forward propagation to compute the output
  - Also store all intermediate outputs
- Compare the output with target label (the loss function)
- Backward propagation to compute the gradients w.r.t. the model parameters
  - gradient on the top x gradient from the current op
- Use gradient descent to update the parameters