Foundations of Machine Learning (ECE 5984)

- Neural Networks -

Eunbyung Park

Assistant Professor

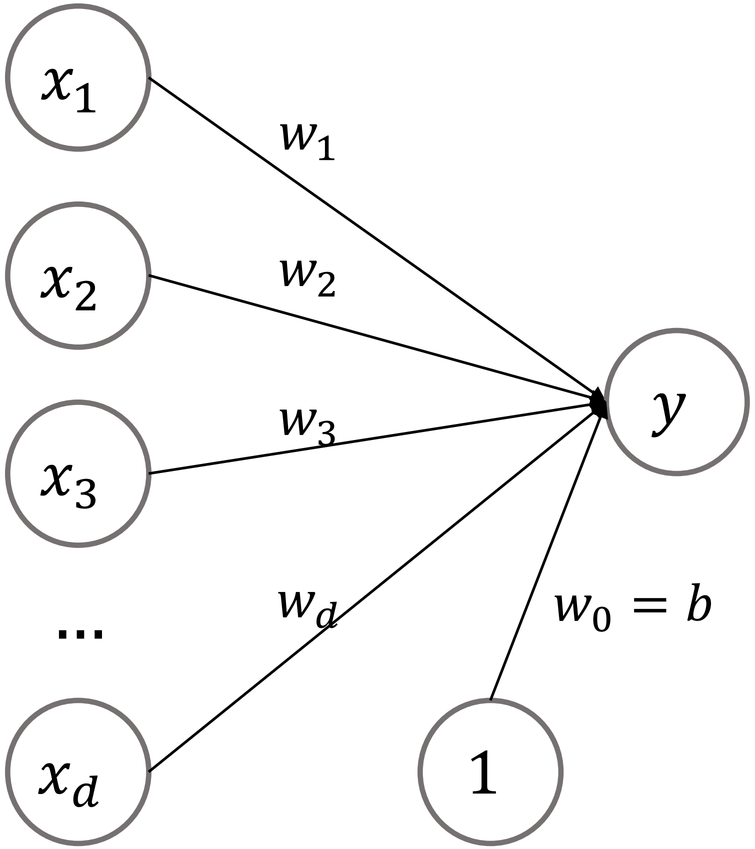
School of Electronic and Electrical Engineering

Eunbyung Park (silverbottlep.github.io)

Multi-Layer Perceptron

Linear Models as Shallow Neural Networks

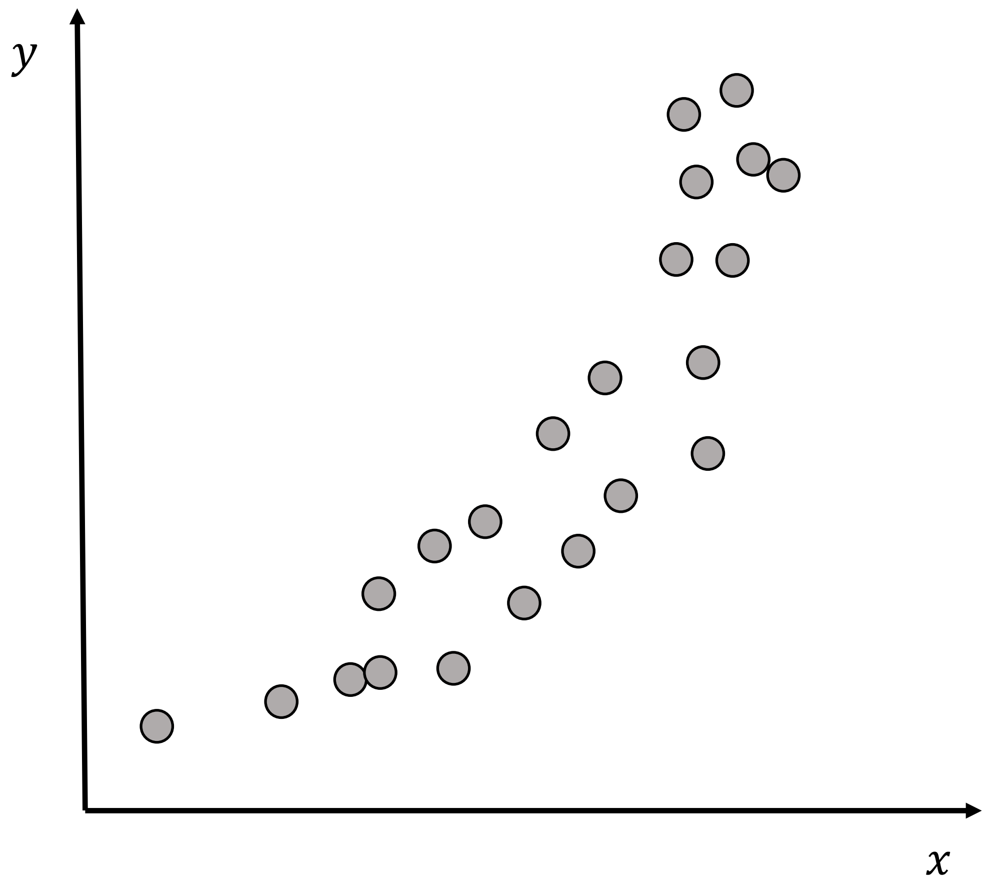
- It is a single layer neural network

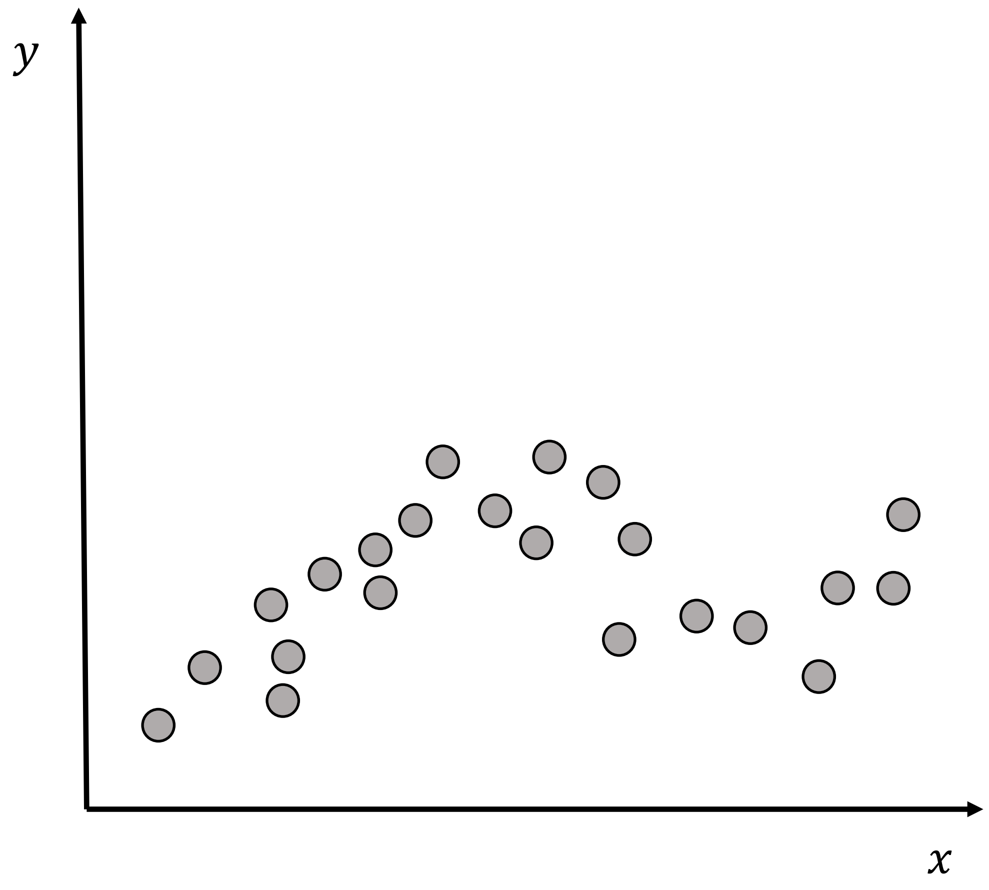


$$y = w^\top x$$

Linear Models

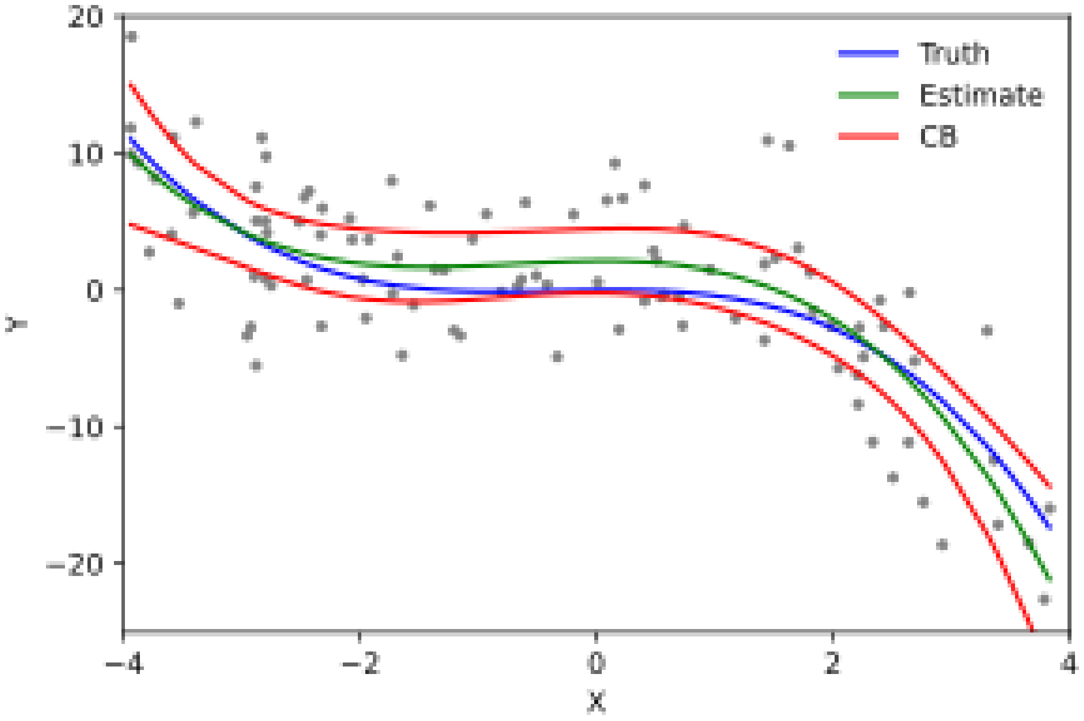
- Is linear model a good for all?





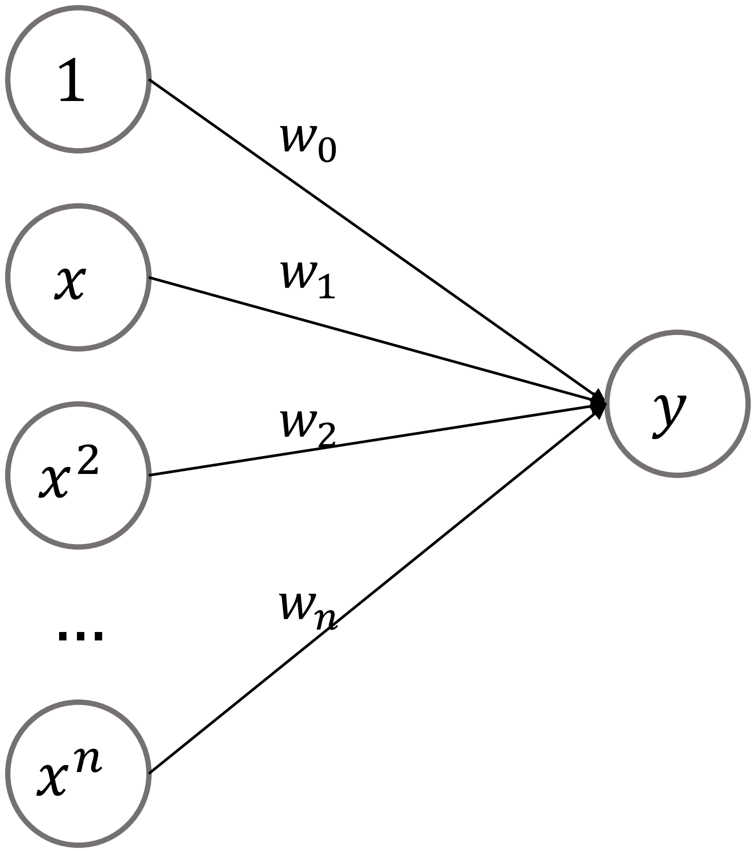
Nonlinear Models

- nth-degree Polynomial regression



$$f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_n x^n$$

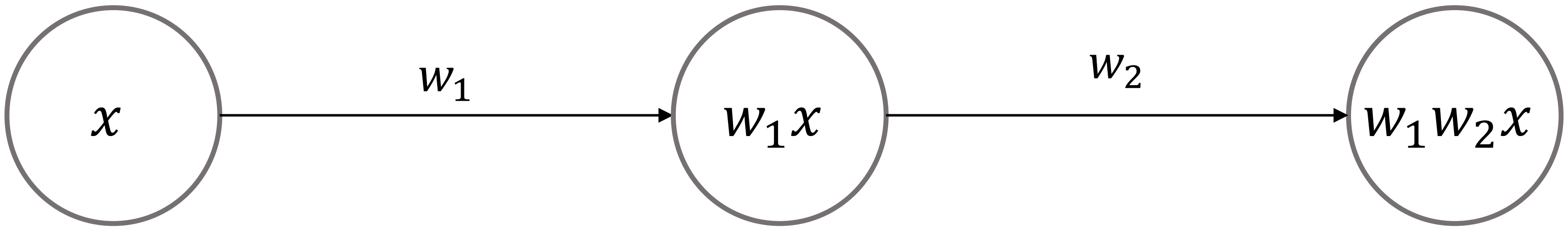
Polynomals as Neural Network



$$f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_n x^n$$

- Feature engineering is hard
- Can we make it non-linear w/o feature engineering?

Feed-Forward Neural Network

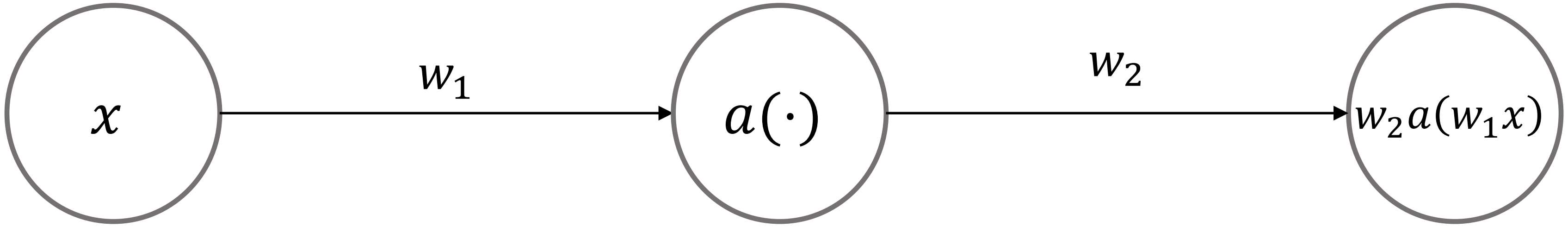


$$f(x) = w_1 w_2 x$$

Is it non-linear in x?

Feed-Forward Neural Network

- Using non-linear activation function



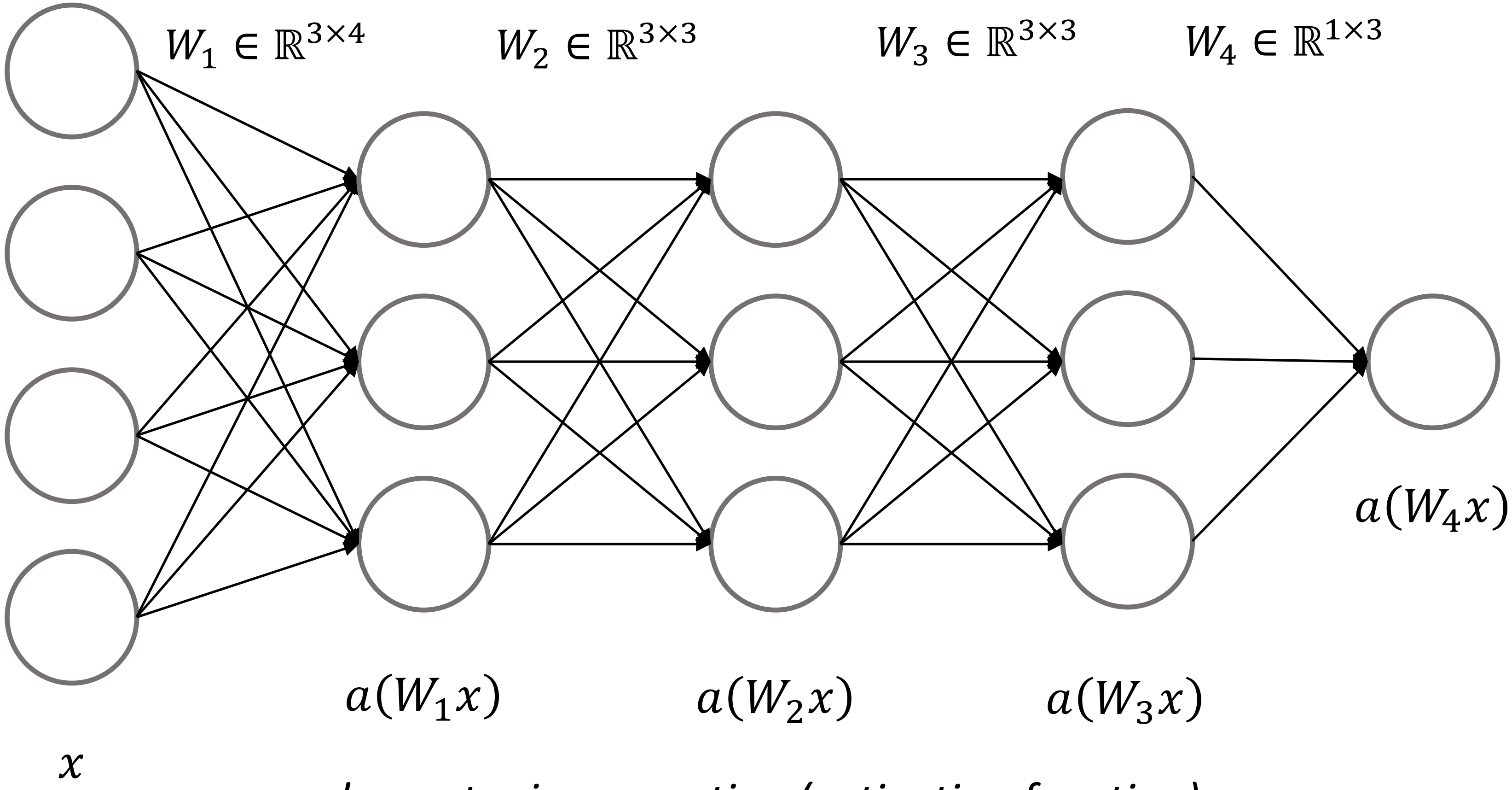
$$f(x) = w_2 a(w_1 x)$$

$$a(x) = \max(0, x)$$ (Rectifier Linear Unit)

$$a(x) = \frac{1}{1 + e^{-x}}$$ (Sigmoid)

Feed-Forward Neural Network

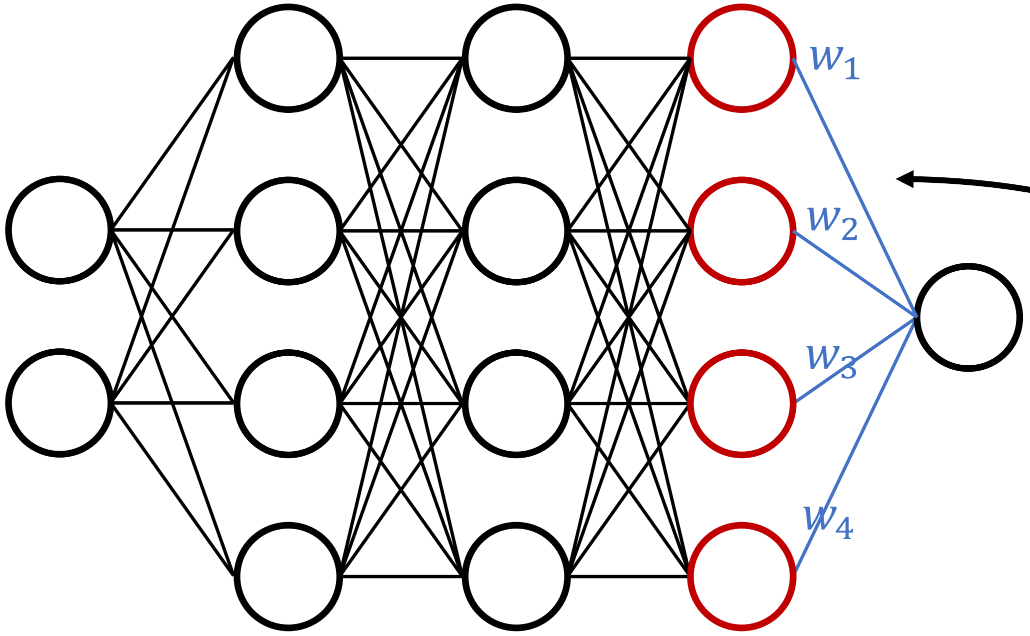
- AKA, Multi-Layer Perceptron



a: element-wise operation (activation function)

Feed-Forward Neural Network

- Connection to the Kernel Method

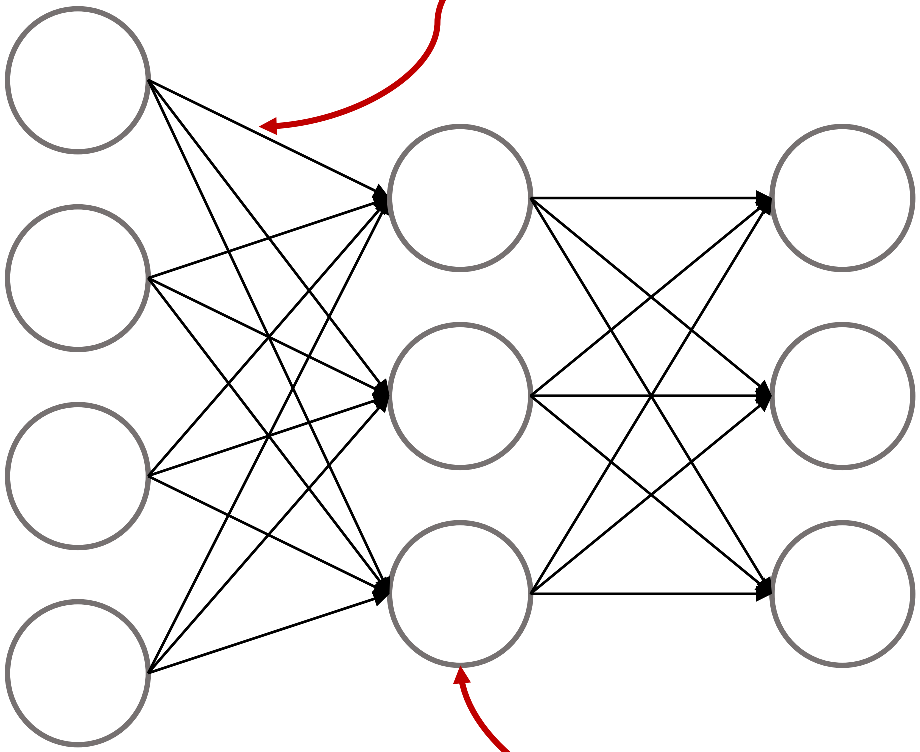


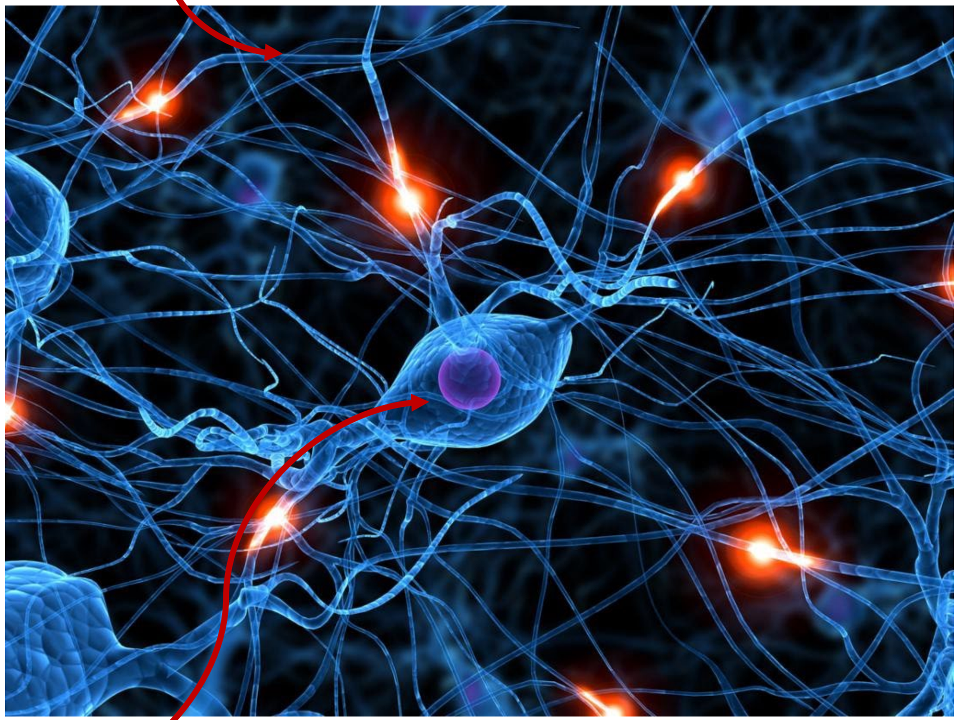
$$\sum_{j=1}^{d} w_j \phi_j(x)$$

Feed-Forward Neural Network

- Biology Inspired

Weights ≈ Synapses

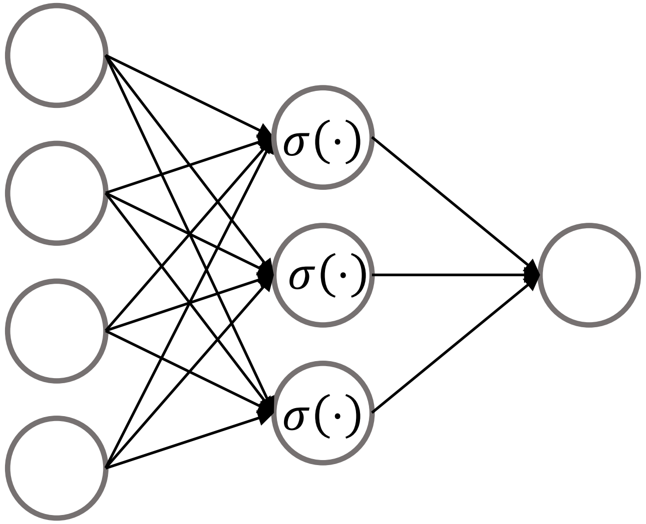




Units ≈ Neurons

Feed-Forward Neural Network

- Regression with two layers MLP



$D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$

$x^{(i)} \in \mathbb{R}^d, y^{(i)} \in \mathbb{R}, X \in \mathbb{R}^{N \times d}, Y \in \mathbb{R}^N$

$\theta = \{W_1, W_2\}, W_1 \in \mathbb{R}^{h \times d}, W_2 \in \mathbb{R}^{1 \times h}$

$f_\theta(x) = W_2 \sigma(W_1 x)$

$f_\theta: \mathbb{R}^d \to \mathbb{R}$

$$L(\theta) = \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - f_\theta(x^{(i)})\right)^2 = \frac{1}{2} (Y - \sigma(W_1 X^\top)^\top W_2^\top)^\top (Y - \sigma(W_1 X^\top)^\top W_2^\top)$$

Feed-Forward Neural Network

- Regression with two layers MLP

$$D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$$

$$x^{(i)} \in \mathbb{R}^d, y^{(i)} \in \mathbb{R}, X \in \mathbb{R}^{N\times d}, Y \in \mathbb{R}^N$$

$$\theta = \{W_1, W_2\}, W_1 \in \mathbb{R}^{h\times d}, W_2 \in \mathbb{R}^{1\times h}$$

$$f_\theta(x) = W_2 \sigma(W_1 x)$$

$$f_\theta : \mathbb{R}^d \to \mathbb{R}$$

1. Can you take the gradients w.r.t θ?
2. Does it have a closed form solution?
3. Is it a convex function?

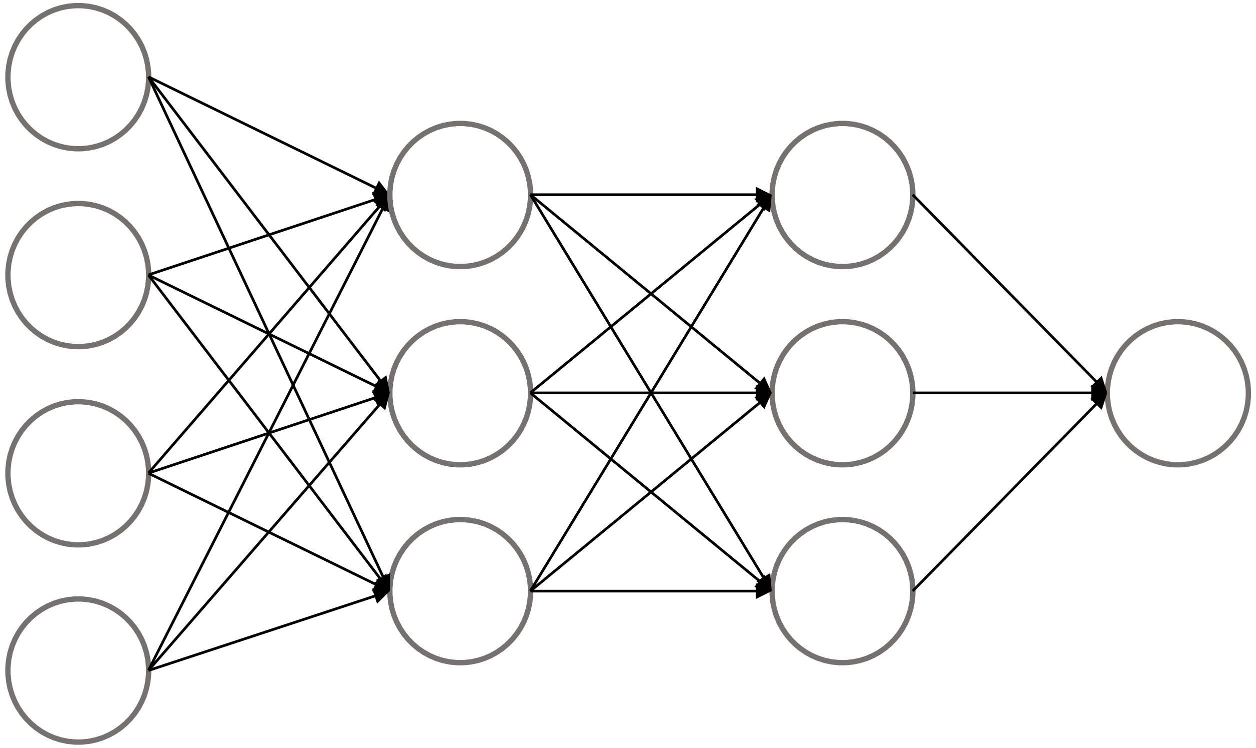
$$L(\theta) = \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - f_\theta(x^{(i)})\right)^2 = \frac{1}{2}(Y - \sigma(W_1 X^\top)^\top W_2^\top)^\top (Y - \sigma(W_1 X^\top)^\top W_2^\top)$$

Gradient Descent

- We are using *gradient descent* for training deep neural networks

$$W := W - \alpha \left(\frac{\partial L}{\partial W} \right)$$

(descent) (step-size) (gradient)



The Universal Approximator

The Universal Approximation Theorem

- A single hidden layer neural network can approximate any continuous function arbitrarily well, given enough hidden units.
- This holds for many different activation functions, e.g. sigmoid, tanh, ReLU, etc.

Cybenko Theorem

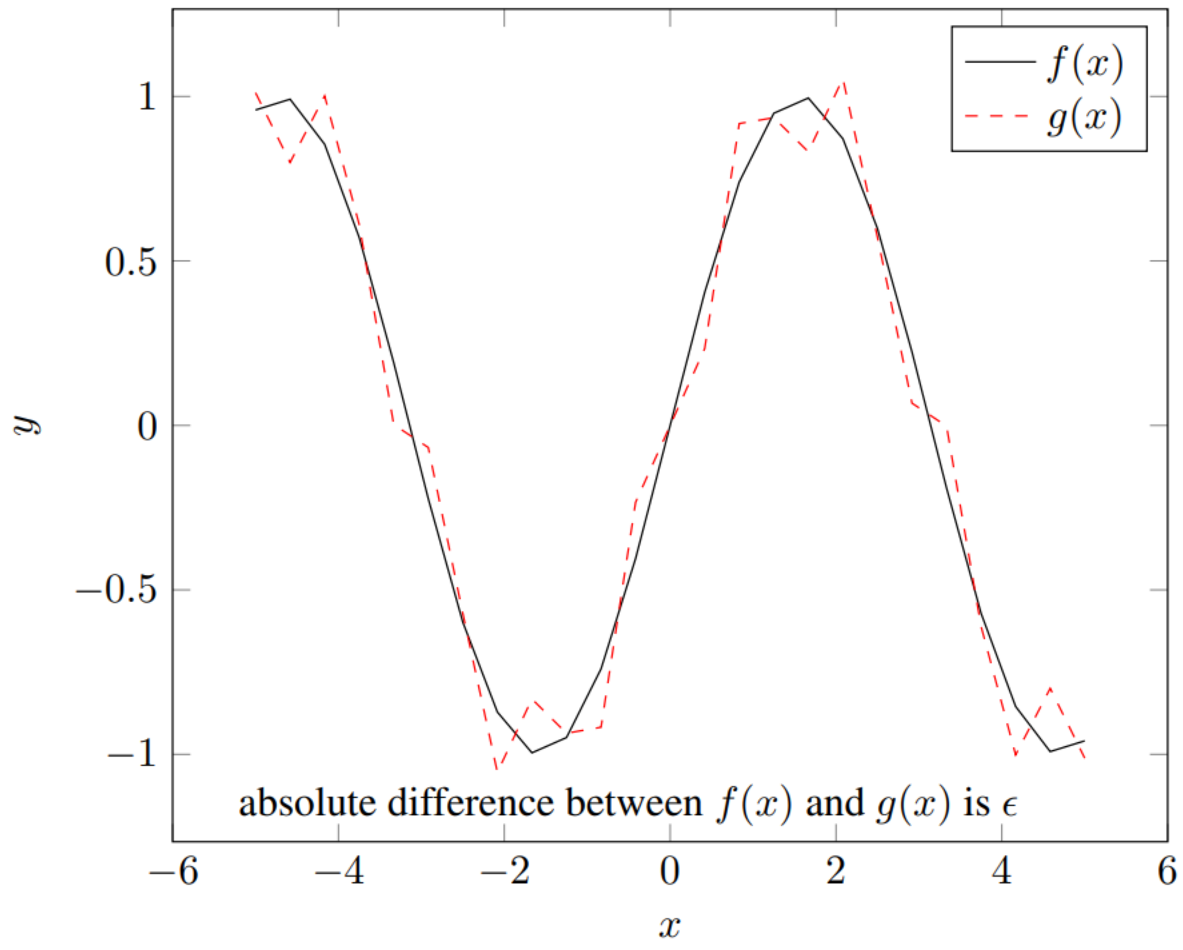
- Cybenko Approximation by Superposition of Sigmoidal Function

Let $C([0,1]^n)$ *denote the set of all continuous function* $[0,1]^n \to \mathbb{R}$, *let* σ *be any sigmoidal activation function then the finite sum of the form* $f(x) = \sum_{i=1}^{N} \alpha_i \, \sigma(w_i^\top x + b_i)$ *is dense in* $C([0,1]^n)$

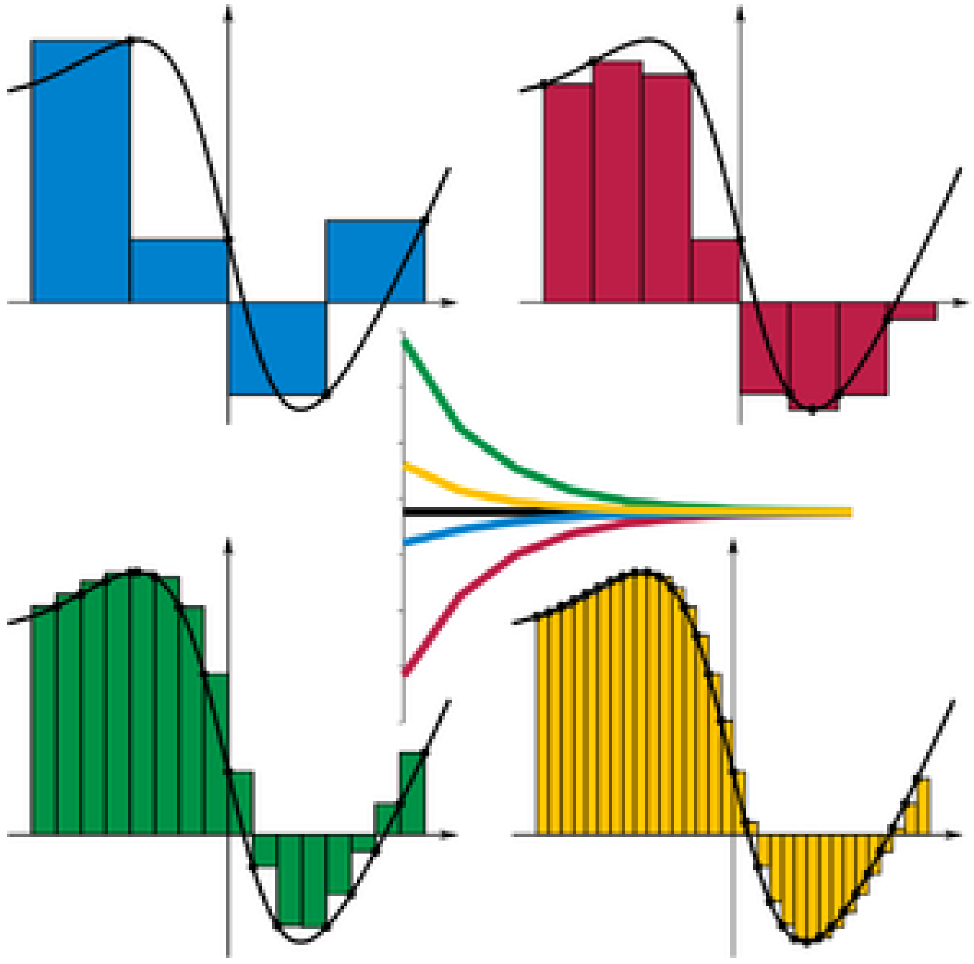
For any $g \subset C([0,1]^n)$ *and any* $\epsilon > 0$, *there exists* $f: x \to \sum_{i=1}^{N} \alpha_i \, \sigma\left(w_i^\top x + b_i\right)$, *such that* $|f(x) - g(x)| < \epsilon$ *for all* $x \subset [0,1]^n$.

Cybenko Theorem

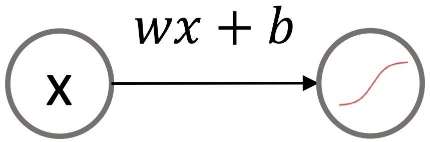
- Cybenko Approximation by Superposition of Sigmoidal Function



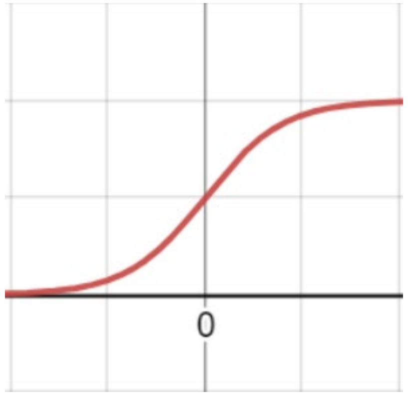
The Universal Approximation Theorem



The Universal Approximation Theorem



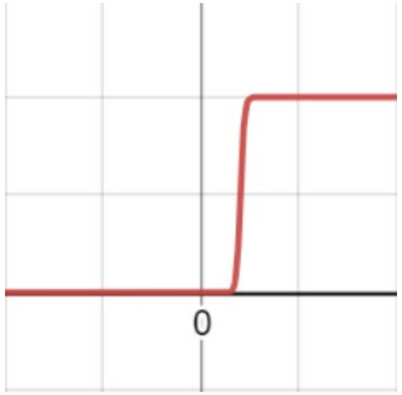
$w = 5, b = 0$



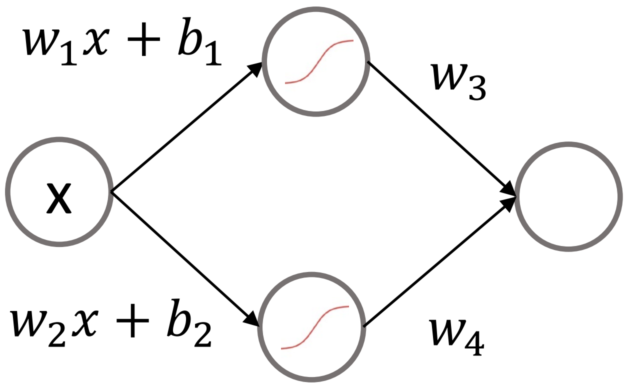
$w = 5, b = 3$

$w = 10, b = -7$

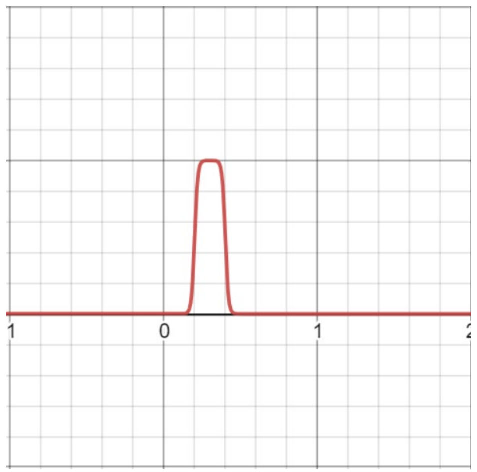
$w = 100, b = -20$



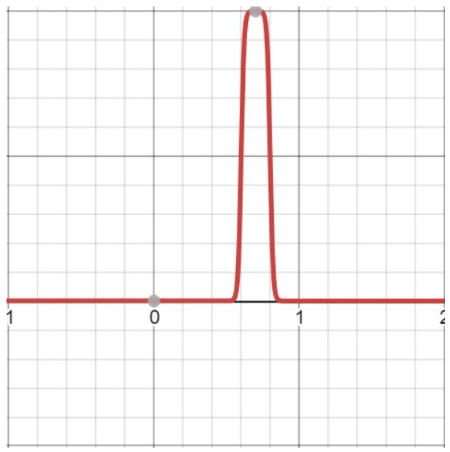
The Universal Approximation Theorem



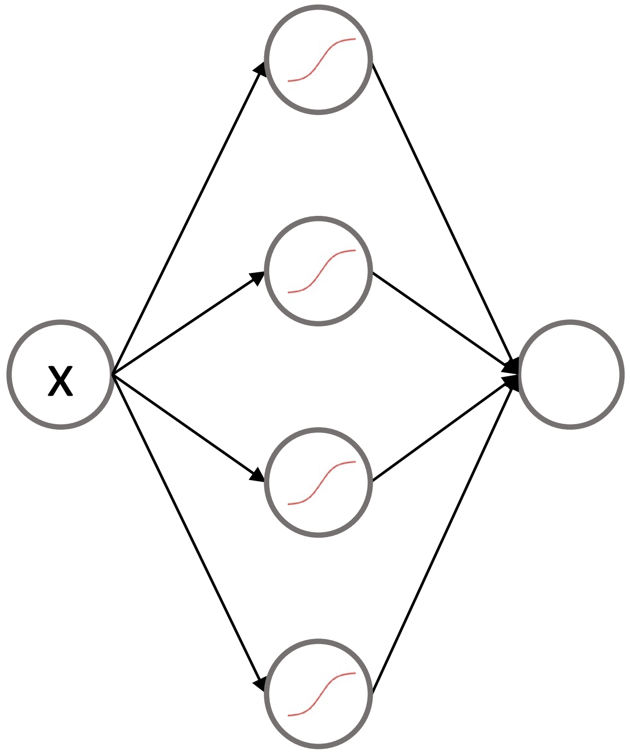
$w_1 = 100, b_1 = -20$
$w_2 = 100, b_2 = -40$
$w_3 = 1, w_4 = -1$

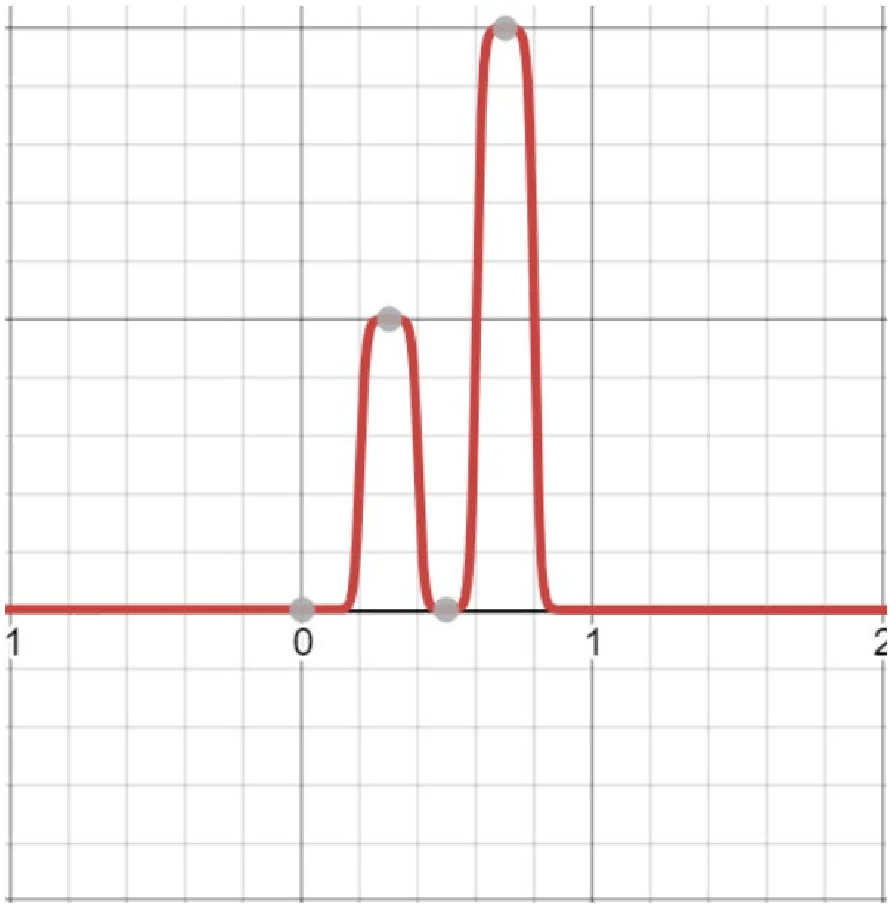


$w_1 = 100, b_1 = -60$
$w_2 = 100, b_2 = -80$
$w_3 = 2, w_4 = -2$

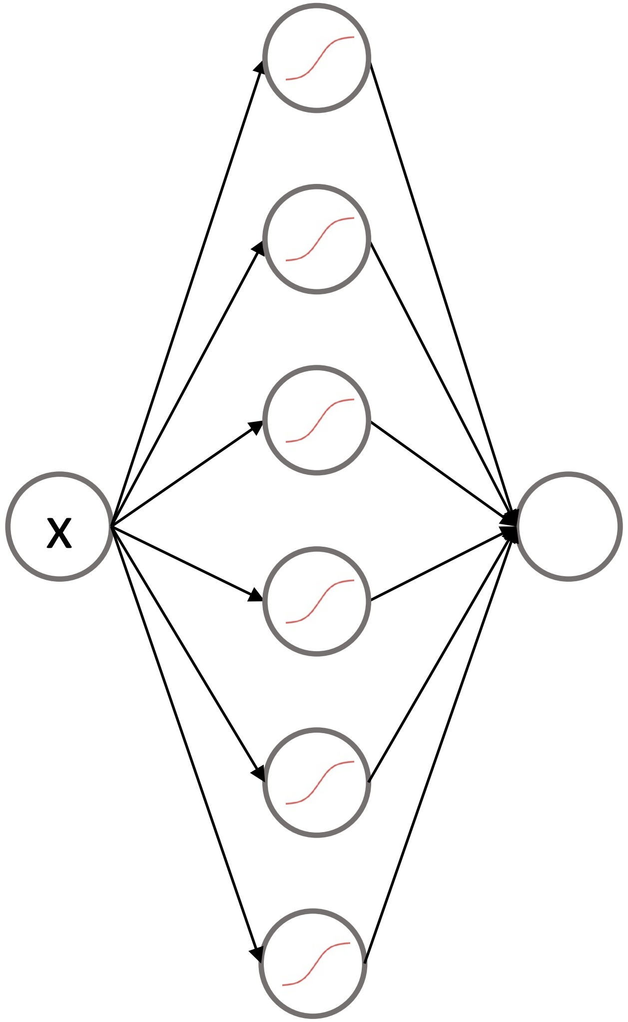


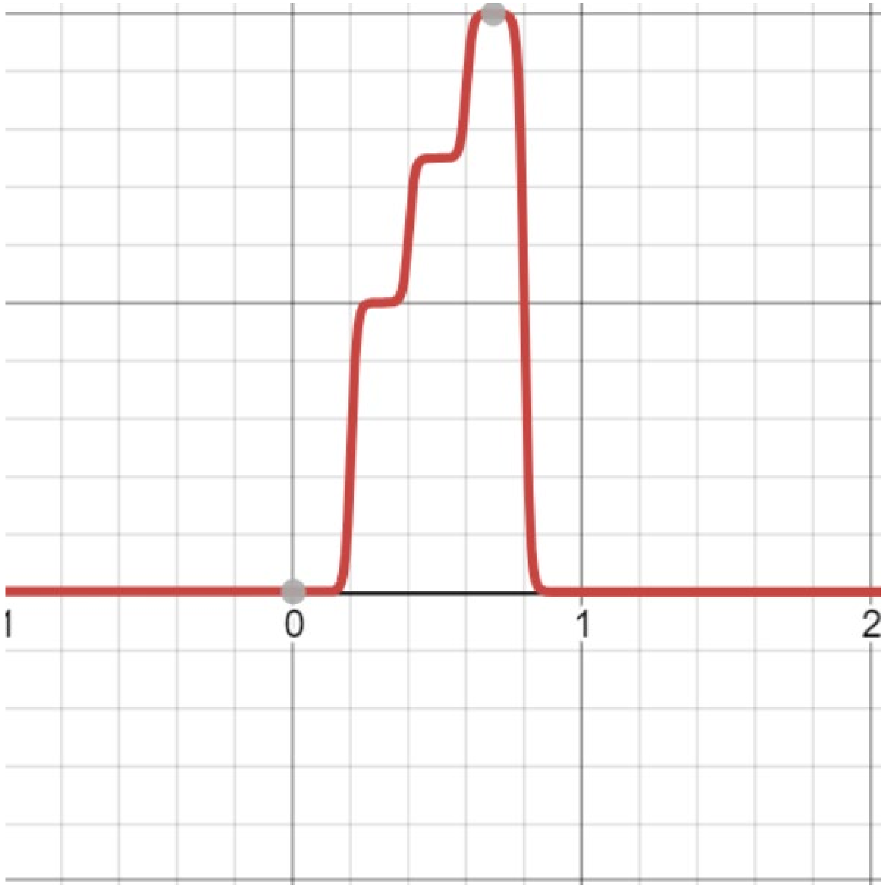
The Universal Approximation Theorem



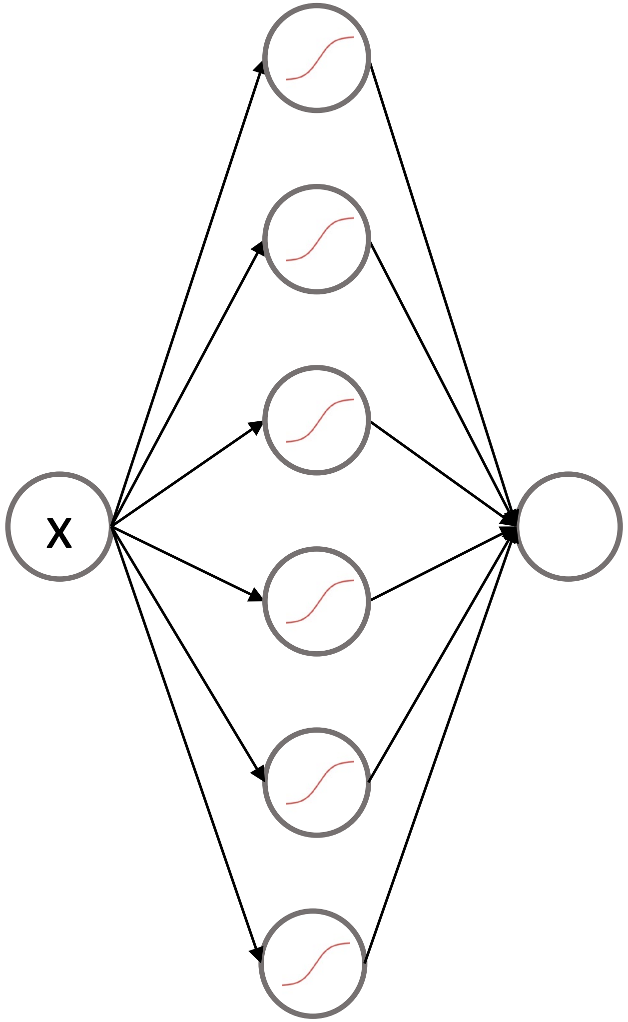


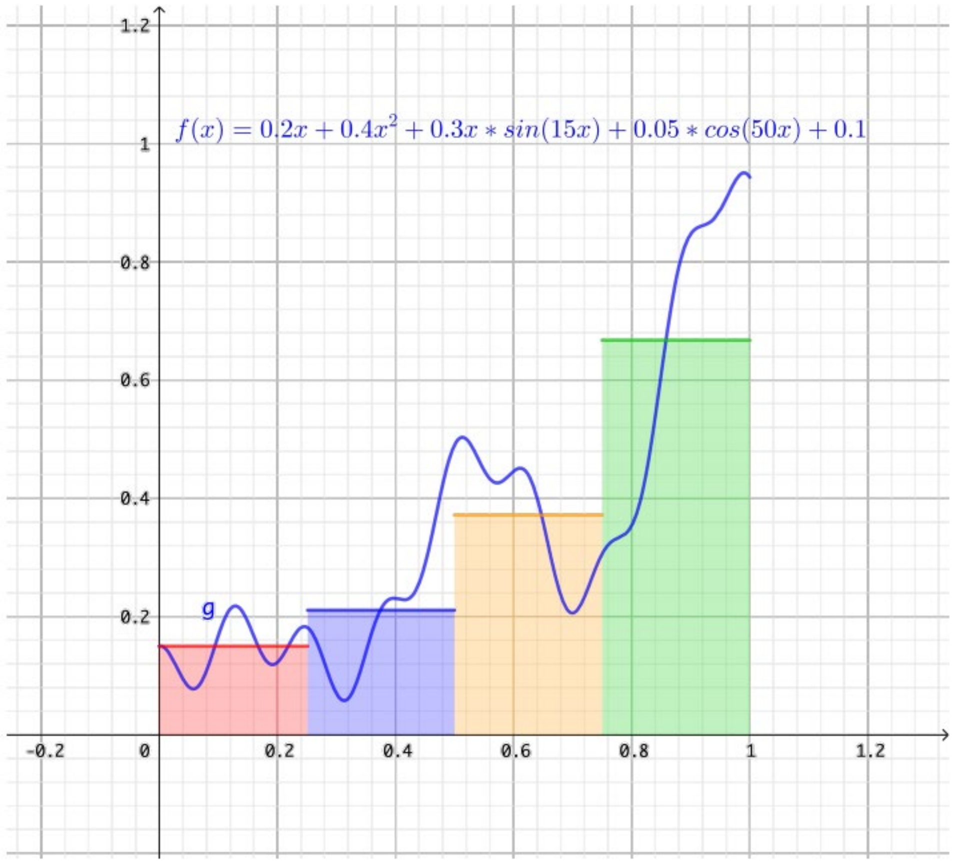
The Universal Approximation Theorem



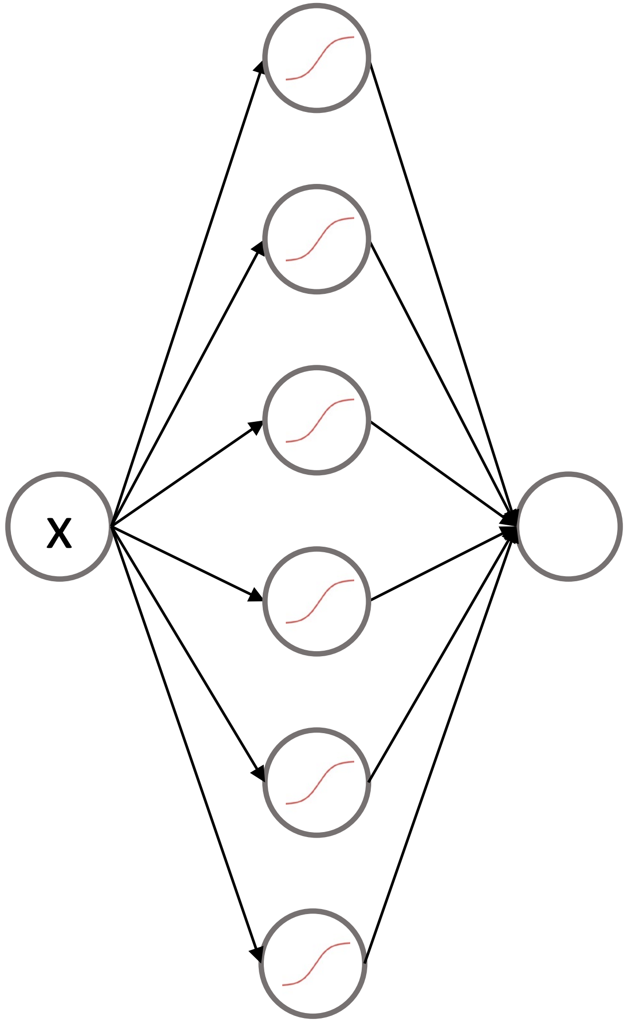


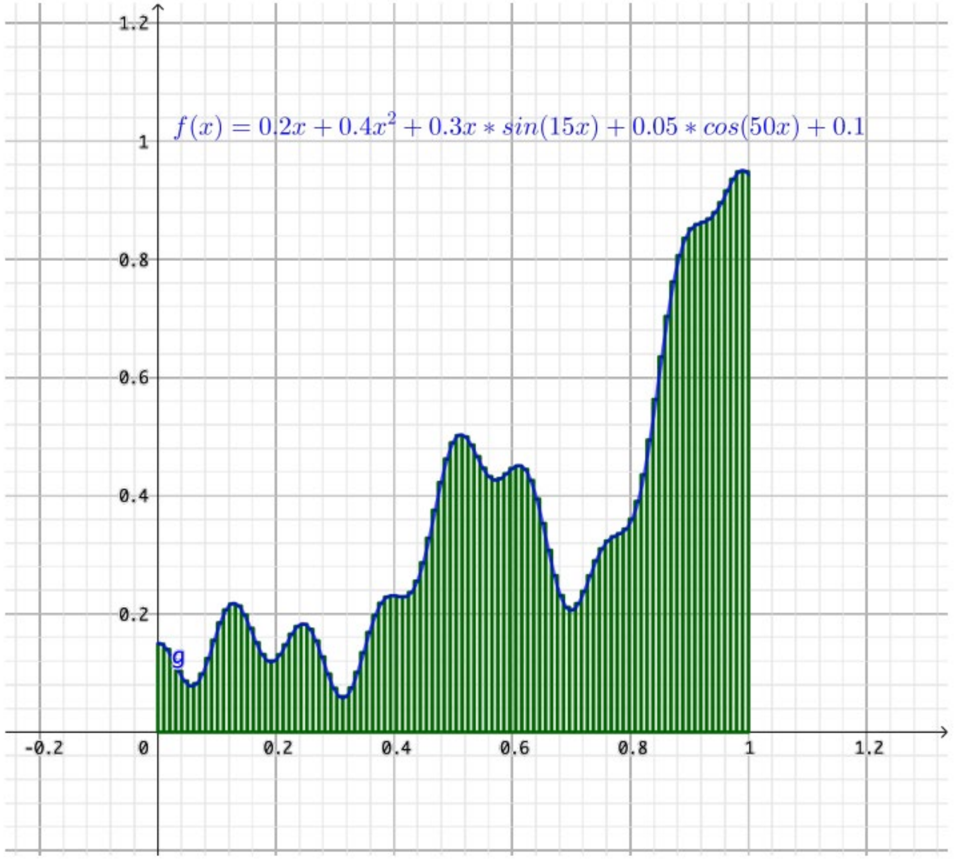
The Universal Approximation Theorem



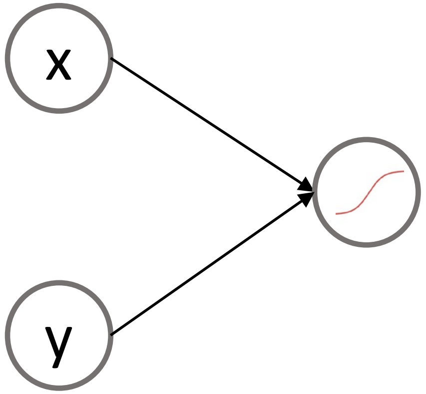


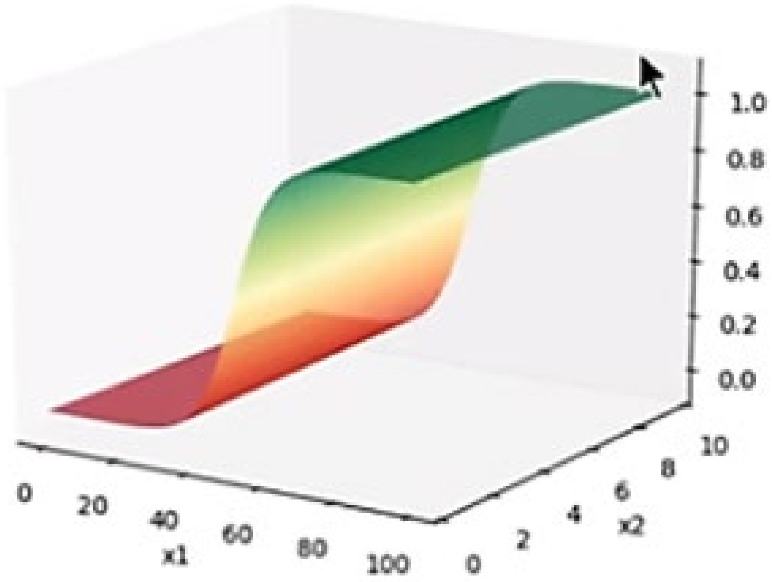
The Universal Approximation Theorem



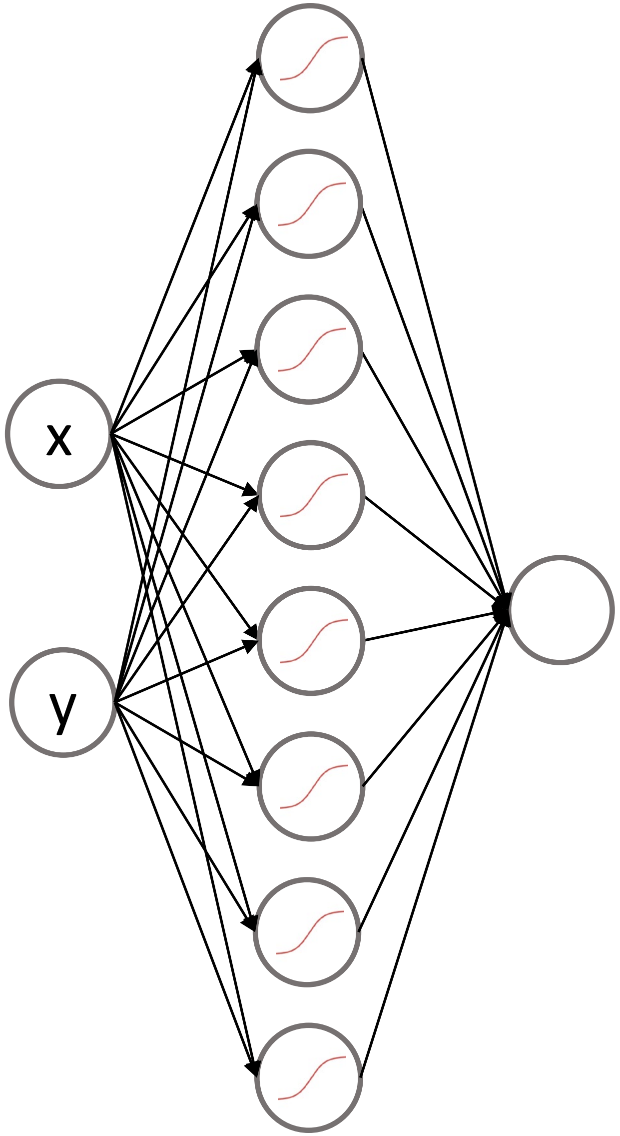


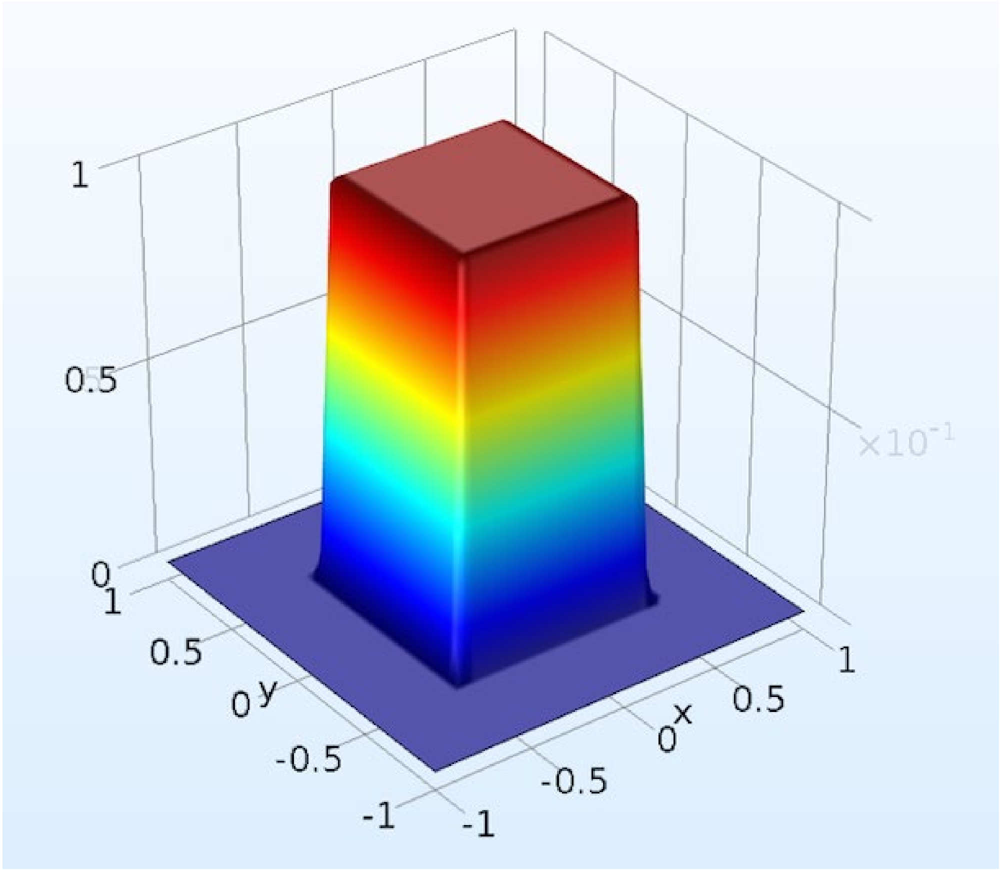
The Universal Approximator in 2D



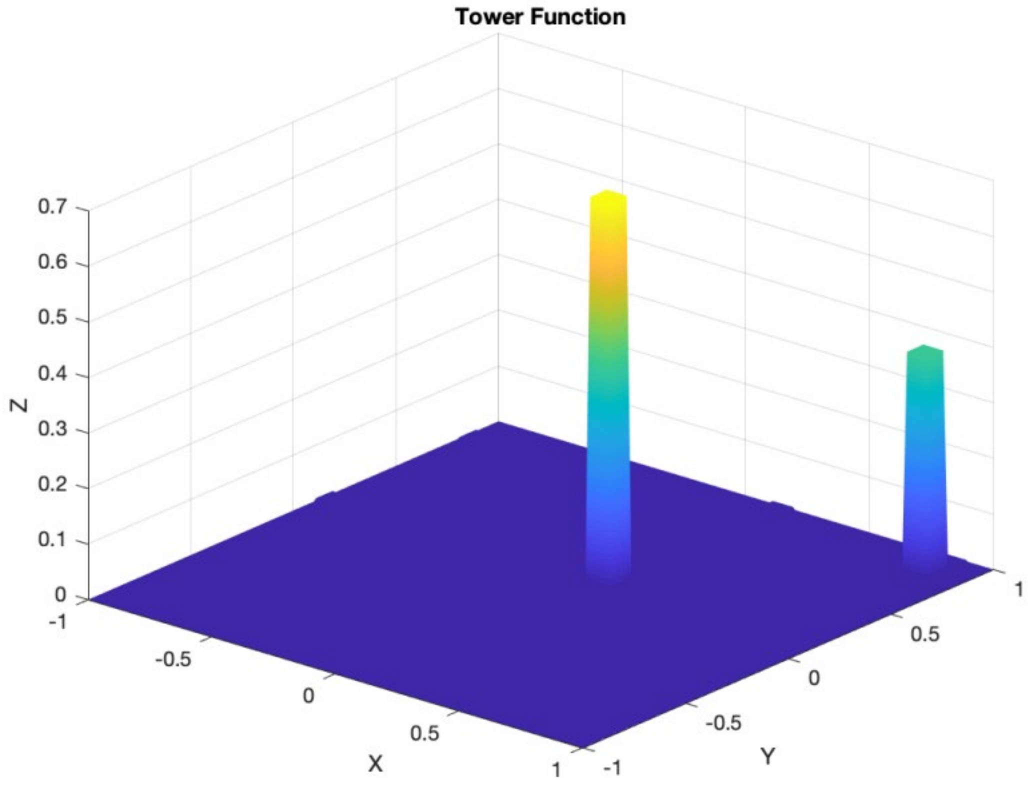


The Universal Approximator in 2D

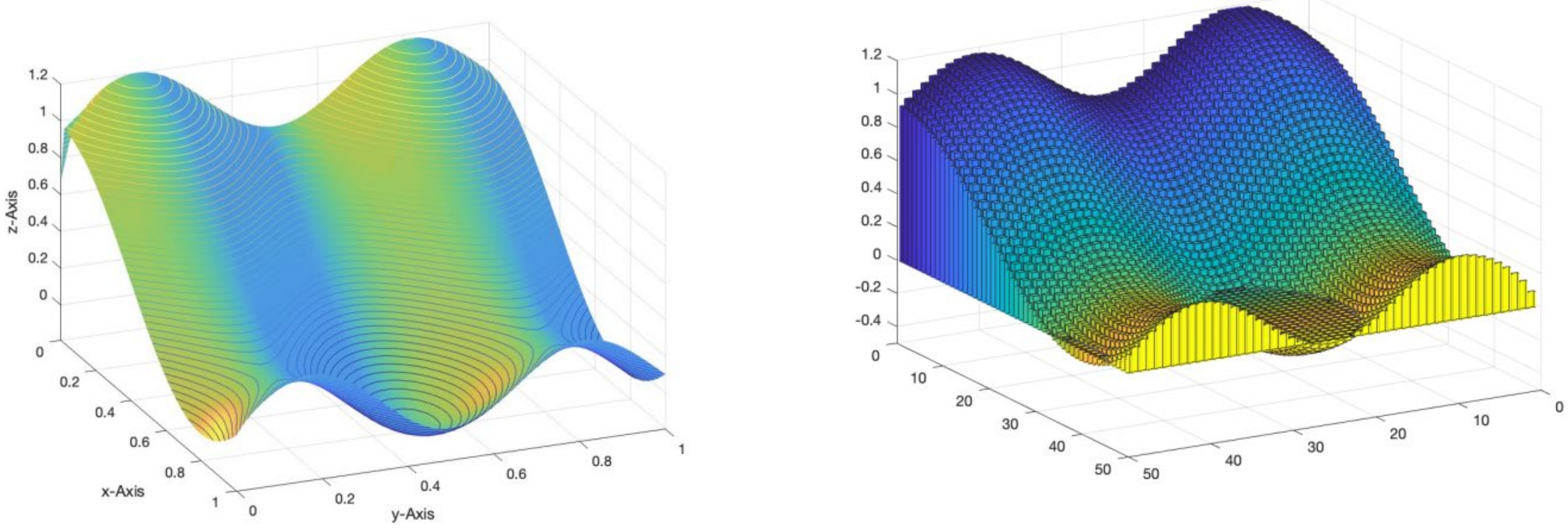




The Universal Approximator in 2D



The Universal Approximator in 2D



The Universal Approximation Theorem

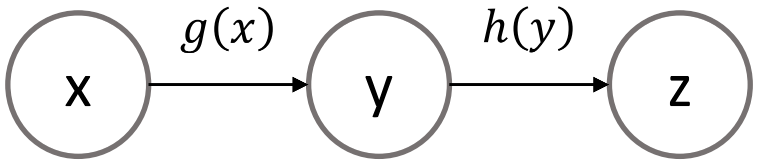
- Single layer might be enough, but it requires ‘enough’ neurons.
- Informally, ‘shallower and wider’ networks require exponentially more hidden units to compute ‘narrower and deeper’ neural networks
  - Lecture 2 | The Universal Approximation Theorem - YouTube

The Chain Rule

The Chain Rule

- A single variable chain rule

$$f, g, h: \mathbb{R} \to \mathbb{R}$$



$$f: h \circ g$$

$$f'(x) = h'\big(g(x)\big)g'(x)$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

$$y = g(x), z = h(y)$$

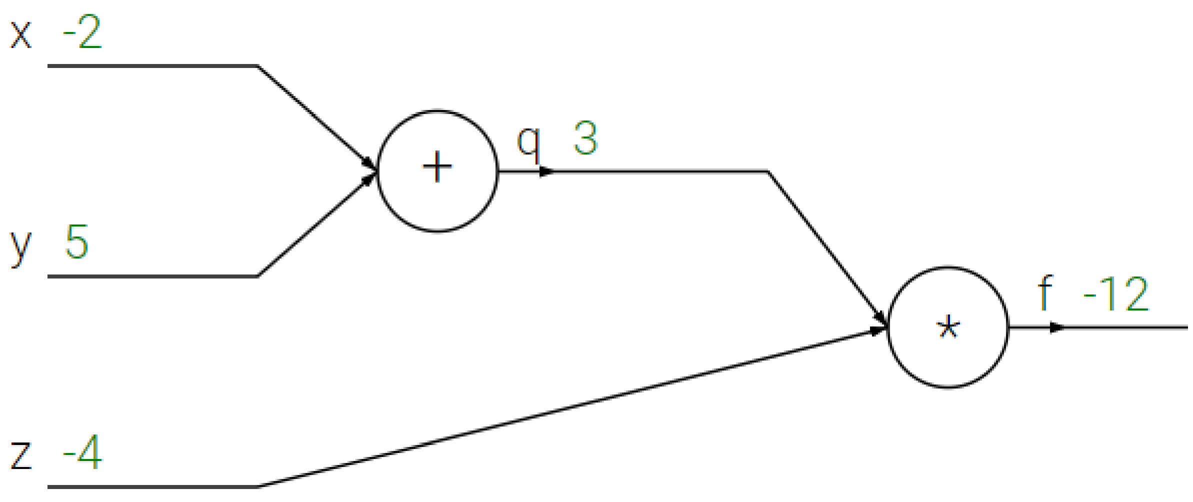
Simple Example

$$f(x, y, z) = (x + y)z$$

$$q = x + y, \qquad f = qz$$

$$\frac{\partial q}{\partial x} = 1, \qquad \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \qquad \frac{\partial f}{\partial z} = q$$



Simple Example

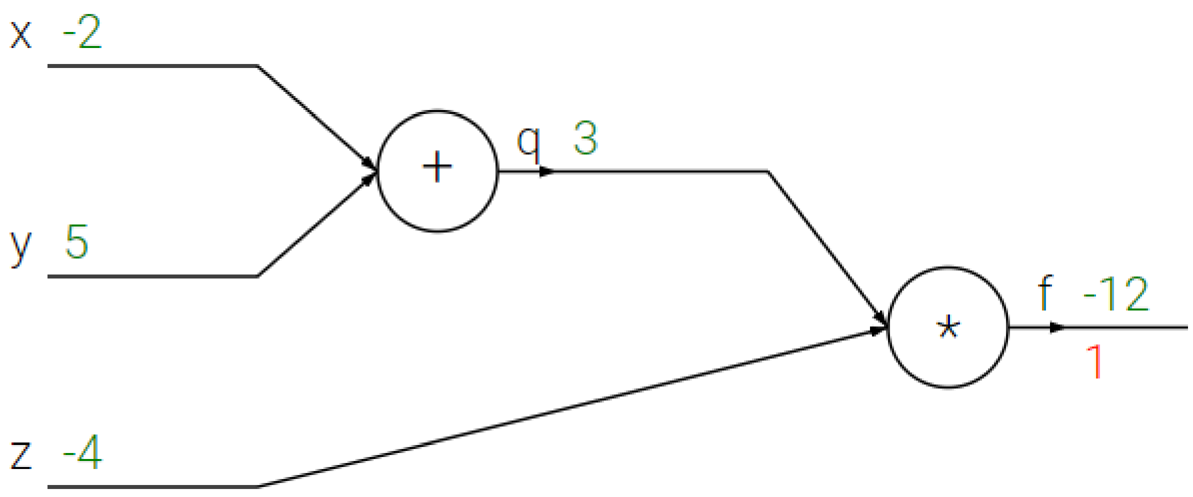
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

$$f(x, y, z) = (x + y)z$$

$$q = x + y, \qquad f = qz$$

$$\frac{\partial q}{\partial x} = 1, \qquad \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \qquad \frac{\partial f}{\partial z} = q$$



CS231n Convolutional Neural Networks for Visual Recognition

Simple Example

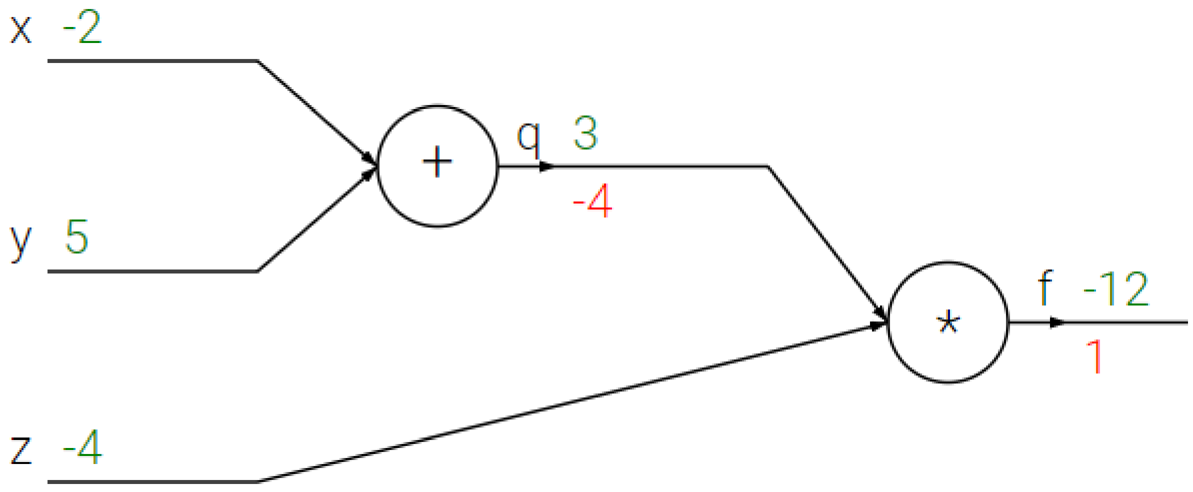
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

$$f(x, y, z) = (x + y)z$$

$$q = x + y, \qquad f = qz$$

$$\frac{\partial q}{\partial x} = 1, \qquad \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \qquad \frac{\partial f}{\partial z} = q$$



Simple Example

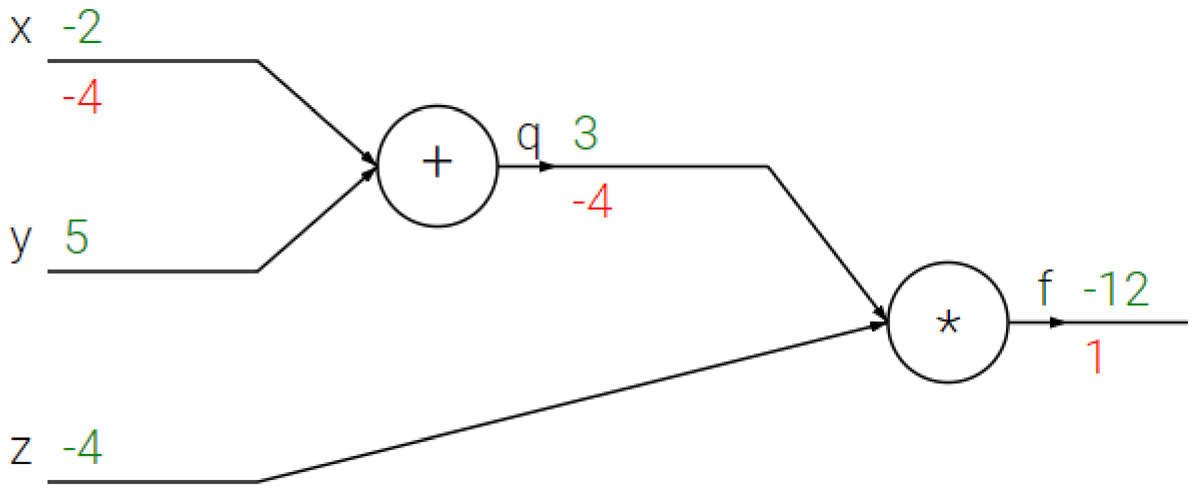
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

$$f(x, y, z) = (x + y)z$$

$$q = x + y, \qquad f = qz$$

$$\frac{\partial q}{\partial x} = 1, \qquad \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \qquad \frac{\partial f}{\partial z} = q$$



[CS231n Convolutional Neural Networks for Visual Recognition]

Simple Example

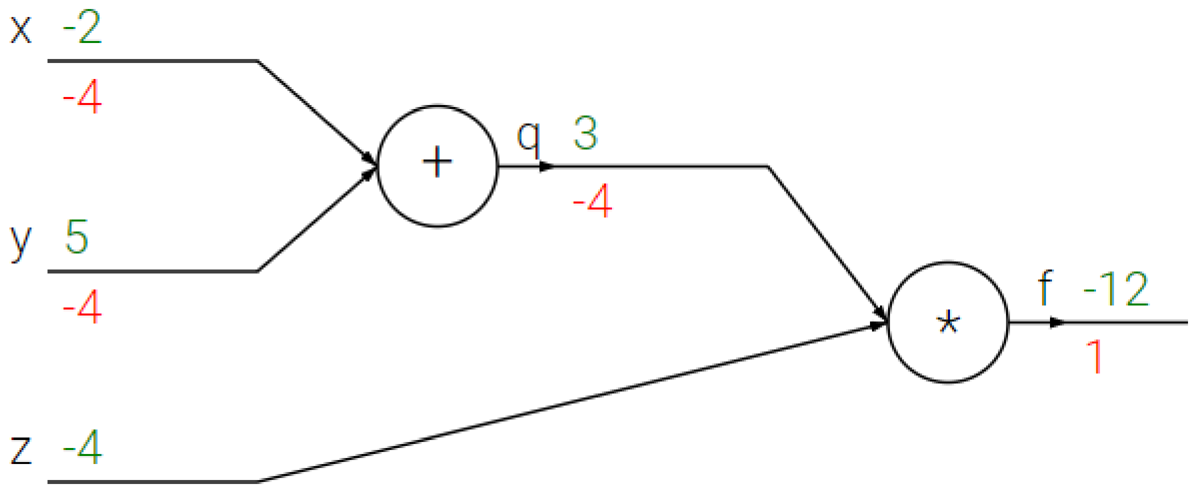
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial q}\frac{\partial q}{\partial y}$$

$$f(x, y, z) = (x + y)z$$

$$q = x + y, \qquad f = qz$$

$$\frac{\partial q}{\partial x} = 1, \qquad \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \qquad \frac{\partial f}{\partial z} = q$$



CS231n Convolutional Neural Networks for Visual Recognition

Simple Example

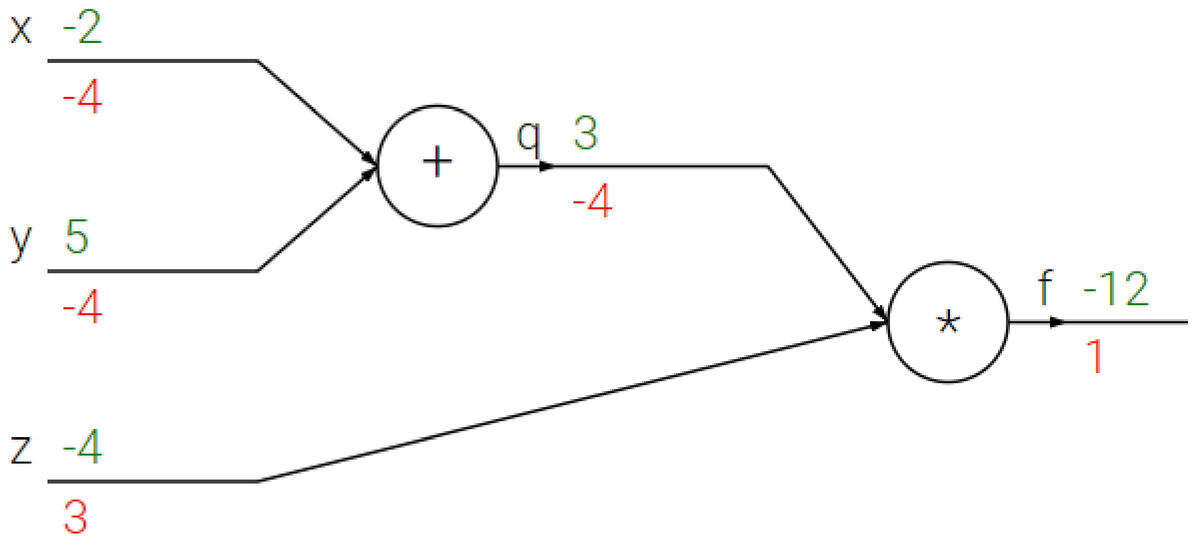
$$f(x, y, z) = (x + y)z$$

$$q = x + y, \qquad f = qz$$

$$\frac{\partial q}{\partial x} = 1, \qquad \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \qquad \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial z}$$



CS231n Convolutional Neural Networks for Visual Recognition

Sigmoid Example

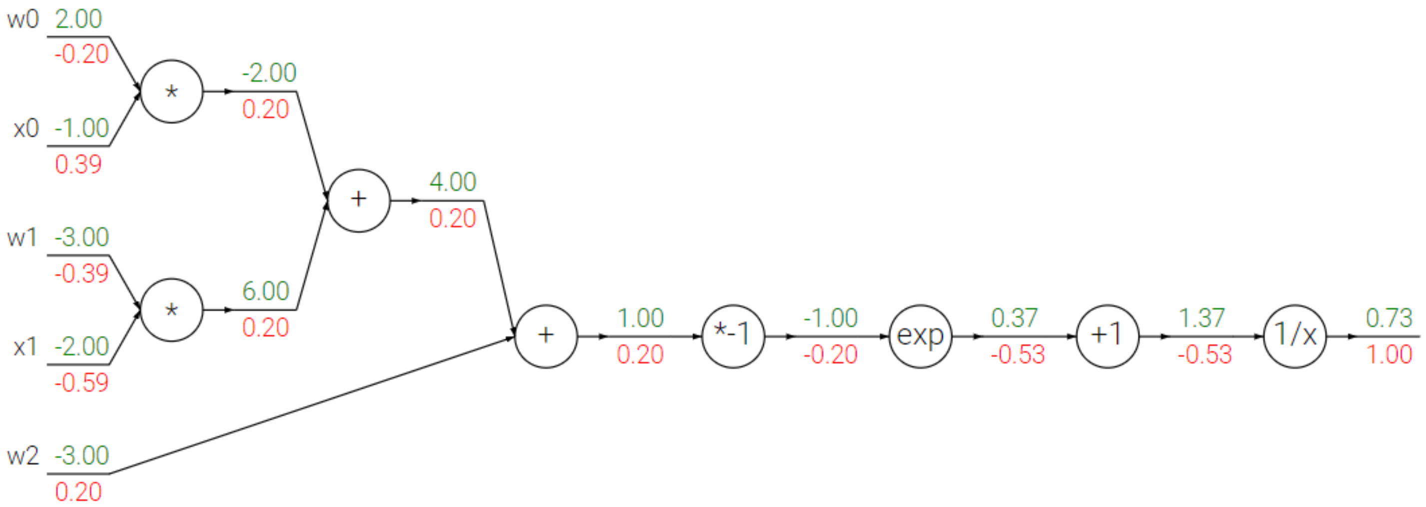
$$\sigma(x, w) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

$$f(x) = \frac{1}{x}, \qquad g(x) = 1 + x, \qquad h(x) = e^{-x}, \qquad i(x) = w_0 x_0 + w_1 x_1 + w_2$$

Sigmoid Example

$$\sigma(x, w) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

$$f(x) = \frac{1}{x}, \qquad g(x) = 1 + x, \qquad h(x) = e^{-x}, \qquad i(x) = w_0 x_0 + w_1 x_1 + w_2$$



[CS231n Convolutional Neural Networks for Visual Recognition]

Backpropagation Algorithm

Gradient

- In vector calculus, the *gradient* of a *scalar-valued* differentiable function $f: \mathbb{R}^n \to \mathbb{R}$ at the point x

$$\nabla f: \mathbb{R}^n \to \mathbb{R}^n \qquad \nabla f = \frac{\partial f}{\partial x} = \left[\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n}\right]$$

Jacobian

- In vector calculus, the *Jacobian* of a *vector-valued* differentiable function is the matrix of all its first-order partial derivatives.

$$f: \mathbb{R}^n \to \mathbb{R}^m$$

$$\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j} \qquad \mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Matrix Calculus

$X \in \mathbb{R}^{n \times m}, y \in \mathbb{R}$

$f: \mathbb{R}^{n \times m} \to \mathbb{R}$

$y = f(x)$

$$\frac{\partial y}{\partial X} = \begin{bmatrix} \frac{\partial y}{\partial X_{11}} & \cdots & \frac{\partial y}{\partial X_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial X_{n1}} & \cdots & \frac{\partial y}{\partial X_{nm}} \end{bmatrix} \in \mathbb{R}^{n \times m}$$

Matrix Calculus

$X \in \mathbb{R}^{n \times m}, y \in \mathbb{R}^{l}$

$f: \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^{l}$

$y = f(x)$

$$\frac{\partial y_1}{\partial X} = \begin{bmatrix} \frac{\partial y_1}{\partial X_{11}} & \cdots & \frac{\partial y_1}{\partial X_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial X_{n1}} & \cdots & \frac{\partial y_1}{\partial X_{nm}} \end{bmatrix} \in \mathbb{R}^{n \times m}$$

$\frac{\partial y}{\partial X} \in \mathbb{R}^{l \times n \times m}$ (3 dim tensor)

Finite Difference

- Numerical method to compute the gradients based on the definition of gradients

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$\frac{df}{dx} \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

Forward difference

$$\frac{df}{dx} \approx \frac{f(x) - f(x-\Delta x)}{\Delta x}$$

Backward difference

$$\frac{df}{dx} \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$$

Central difference

Finite Difference

- Numerical method to compute the gradients based on the definition of gradients

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$\frac{df}{dx} \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

Forward difference

$$\frac{df}{dx} \approx \frac{f(x) - f(x-\Delta x)}{\Delta x}$$

Backward difference

$$\frac{df}{dx} \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$$

Central difference

What's wrong with this approach?

The Chain Rule

- Multi-variable chain rule

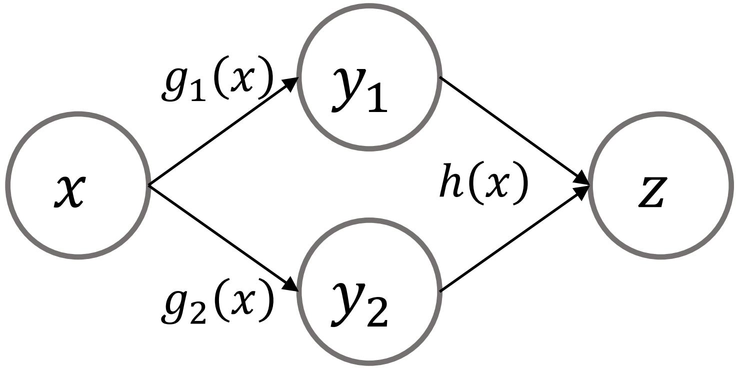
$$f, g_1, g_2 : \mathbb{R} \to \mathbb{R}, \qquad h : \mathbb{R}^2 \to \mathbb{R}$$

$$y_1 = g_1(x), \qquad y_2 = g_2(y)$$

$$z = h(y_1, y_2)$$

$$\frac{dz}{dx} = \frac{dz}{dy_1}\frac{dy_1}{dx} + \frac{dz}{dy_2}\frac{dy_2}{dx}$$

(Total derivative)



The Chain Rule

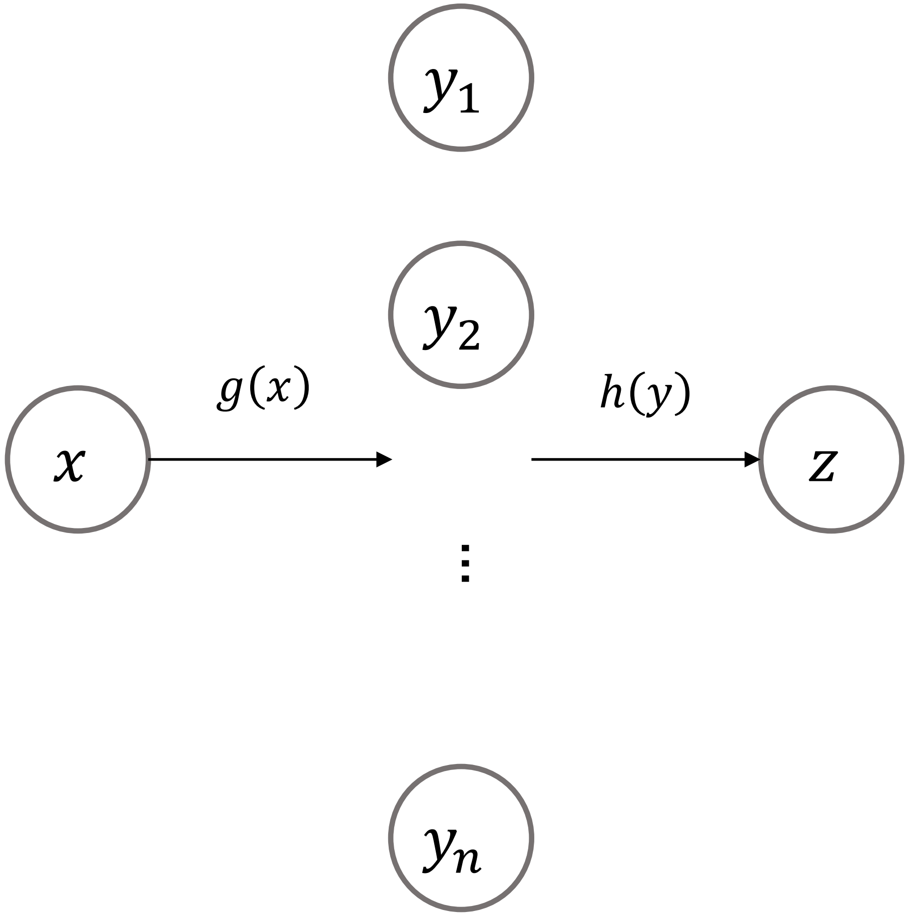
- Multi-variable chain rule

$$x \in \mathbb{R}, y \in \mathbb{R}^n, z \in \mathbb{R}$$

$$g: \mathbb{R} \to \mathbb{R}^n, \qquad y = g(x)$$

$$h: \mathbb{R}^n \to \mathbb{R}, \qquad z = h(y)$$

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i}\frac{dy_i}{dx} = \underbrace{\frac{\partial z}{\partial y}}_{\in \mathbb{R}^{1\times n}} \overbrace{\frac{\partial y}{\partial x}}^{\in \mathbb{R}^{n\times 1}}$$



The Chain Rule

- Multi-variable chain rule

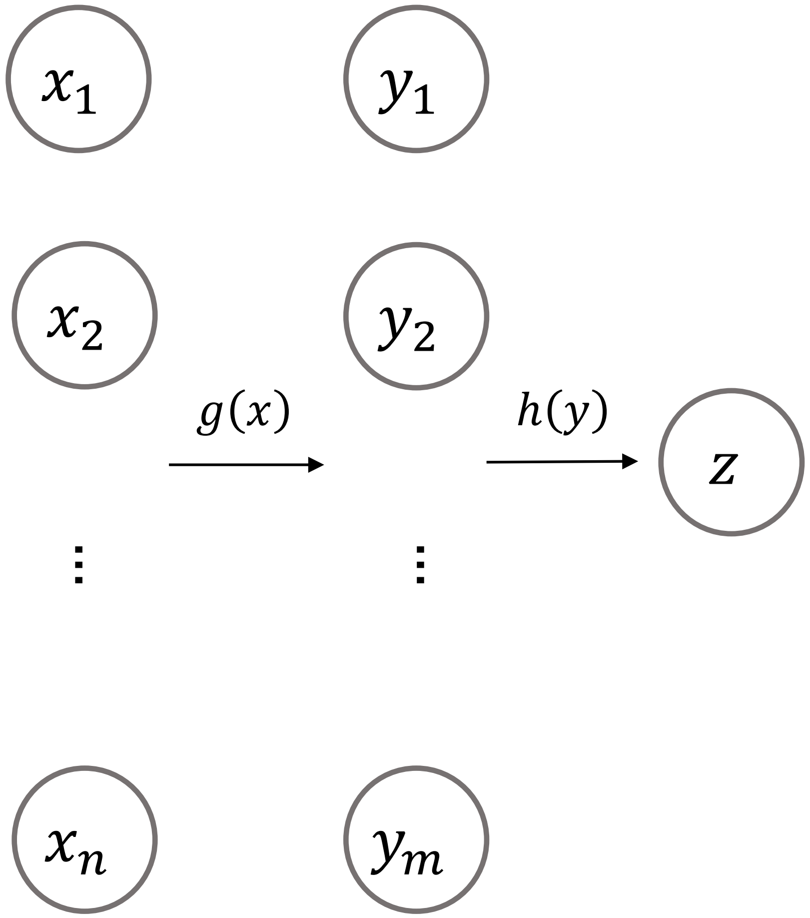
$$x \in \mathbb{R}^n, y \in \mathbb{R}^m, z \in \mathbb{R}$$

$$g: \mathbb{R}^n \to \mathbb{R}^m, \qquad y = g(x)$$

$$h: \mathbb{R}^m \to \mathbb{R}, \qquad z = h(y)$$

$$\frac{\partial z}{\partial x_j} = \sum_{i=1}^{m} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x_j} = \underset{\in \mathbb{R}^{1\times m}}{\frac{\partial z}{\partial y}} \overset{\in \mathbb{R}^{m\times 1}}{\frac{\partial y}{\partial x_j}}$$

$$\frac{\partial z}{\partial x} = \left[\sum_{i=1}^{m} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x_1}, \cdots, \sum_{i=1}^{m} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x_n}\right] = \underset{\in \mathbb{R}^{1\times m}}{\frac{\partial z}{\partial y}} \overset{\in \mathbb{R}^{m\times n}}{\frac{\partial y}{\partial x}}$$



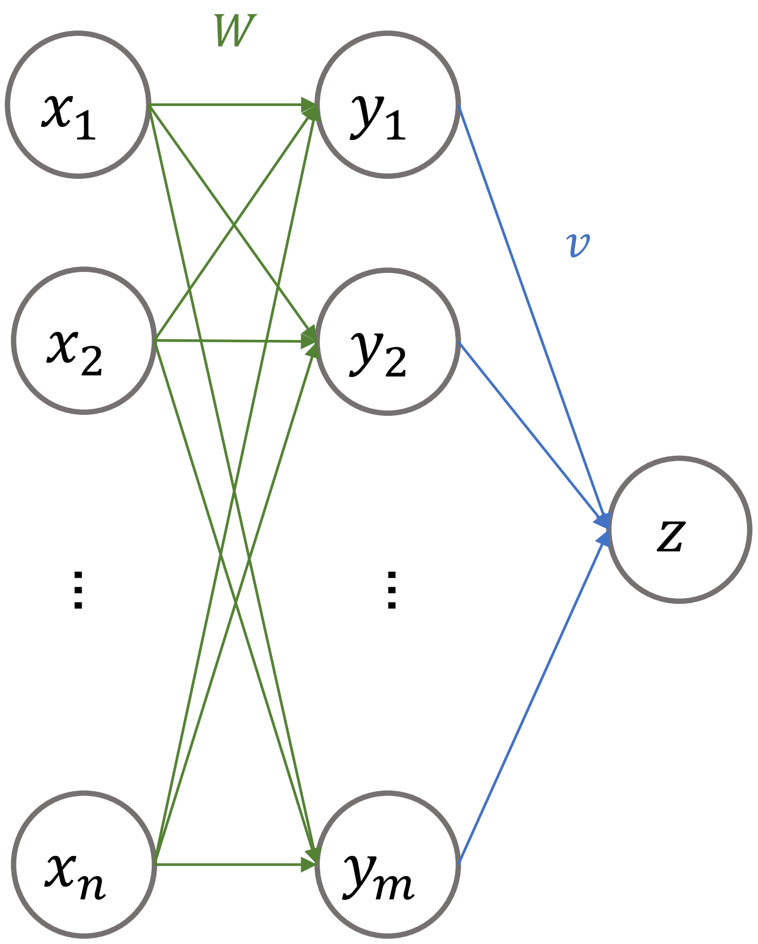
Two Layers MLP

$x \in \mathbb{R}^n, y \in \mathbb{R}^m, z \in \mathbb{R}, W \in \mathbb{R}^{m\times n}, v \in \mathbb{R}^m$

$$y = Wx \qquad z = \sum_{i=1}^{m} v_i y_i = v^\top y$$

$$\frac{\partial z}{\partial x_j} = \sum_{i=1}^{m} \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x_j} = \underbrace{\frac{\partial z}{\partial y}}_{\in \mathbb{R}^{1\times m}} \overbrace{\frac{\partial y}{\partial x_j}}^{\in \mathbb{R}^{m\times 1}}$$

$$\frac{\partial z}{\partial x} = \left[\sum_{i=1}^{m} \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x_1}, \cdots, \sum_{i=1}^{m} \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x_n}\right] = \underbrace{\frac{\partial z}{\partial y}}_{\in \mathbb{R}^{1\times m}} \overbrace{\frac{\partial y}{\partial x}}^{\in \mathbb{R}^{m\times n}} = v^\top W$$



Derivatives of Linear Layer

$$y = Wx \qquad z = \sum_{i=1}^{m} v_i y_i = v^\top y$$

$$\frac{\partial z}{\partial y} \qquad\qquad \frac{\partial y}{\partial x}$$

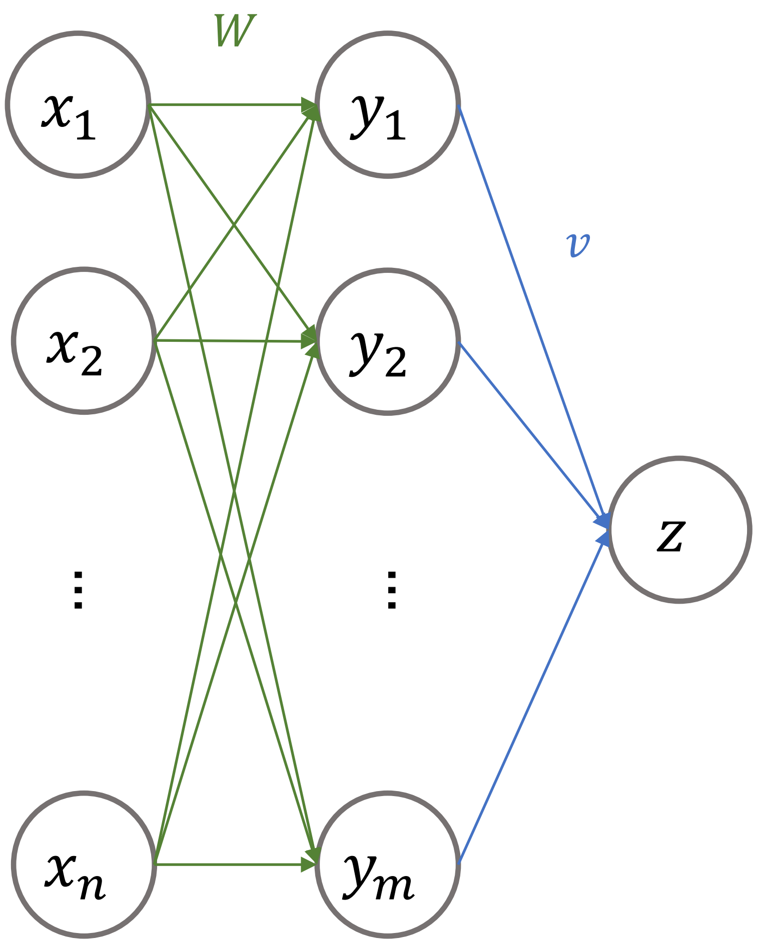
Two Layers MLP

$x \in \mathbb{R}^n, y \in \mathbb{R}^m, z \in \mathbb{R}, W \in \mathbb{R}^{m \times n}, v \in \mathbb{R}^m$

$$y = Wx \qquad z = \sum_{i=1}^{m} v_i y_i = v^\top y$$

$$\underbrace{\frac{\partial z}{\partial W}}_{\in \mathbb{R}^{m \times n}} = \underbrace{\frac{\partial z}{\partial y}}_{\in \mathbb{R}^{1 \times m}} \underbrace{\frac{\partial y}{\partial W}}_{\in \mathbb{R}^{m \times m \times n}}$$

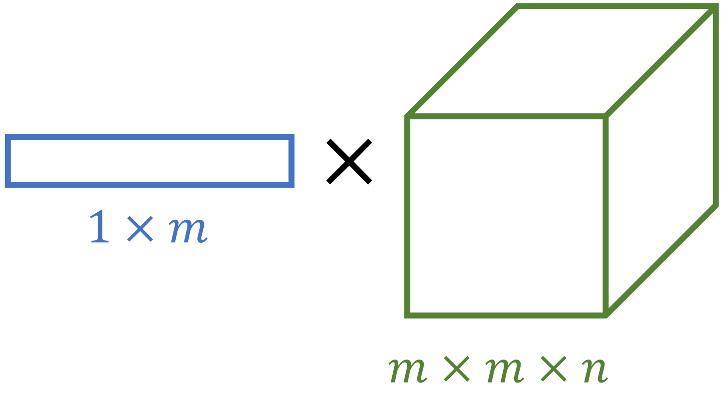
Tensor Product
(n-mode product)

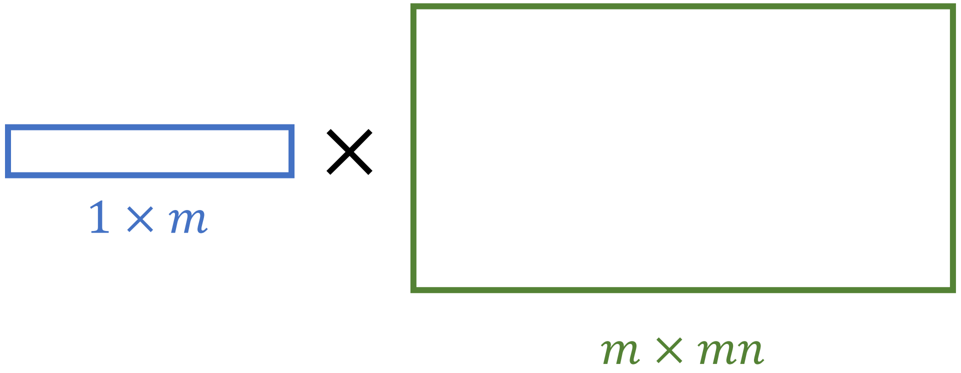


Tensor Product

- N-mode product
  - Matricization -> matrix multiplication

$$\underbrace{\frac{\partial z}{\partial W}}_{\in \mathbb{R}^{m\times n}} = \underbrace{\frac{\partial z}{\partial y}}_{\in \mathbb{R}^{1\times m}} \underbrace{\frac{\partial y}{\partial W}}_{\in \mathbb{R}^{m\times m\times n}}$$





Vector Jacobian Product (VJP)

- Jacobian is very sparse and explicit formation of it is too expensive

$x \in \mathbb{R}^n, y \in \mathbb{R}^m, z \in \mathbb{R}, W \in \mathbb{R}^{m \times n}, v \in \mathbb{R}^m$

$y = Wx$

$$\underset{\in \mathbb{R}^{m\times n}}{\frac{\partial y_1}{\partial W}} = \begin{bmatrix} \frac{\partial y_1}{\partial W_{11}} & \cdots & \frac{\partial y_1}{\partial W_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial W_{m1}} & \cdots & \frac{\partial y_1}{\partial W_{mn}} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial W_{11}} & \cdots & \frac{\partial y_1}{\partial W_{1n}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\underset{\in \mathbb{R}^{m\times n}}{\frac{\partial y_2}{\partial W}} = \begin{bmatrix} \frac{\partial y_2}{\partial W_{11}} & \cdots & \frac{\partial y_2}{\partial W_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_2}{\partial W_{m1}} & \cdots & \frac{\partial y_2}{\partial W_{mn}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{\partial y_2}{\partial W_{21}} & \cdots & \frac{\partial y_2}{\partial W_{2n}} \\ 0 & 0 & 0 \end{bmatrix}$$

Vector Jacobian Product (VJP)

- Jacobian is very sparse and explicit formation of it is too expensive

$x \in \mathbb{R}^n, y \in \mathbb{R}^m, z \in \mathbb{R}, W \in \mathbb{R}^{m\times n}, v \in \mathbb{R}^m$

$y = Wx$

$$\underset{\in \mathbb{R}^{m\times mn}}{\mathrm{reshape}\left(\frac{\partial y}{\partial W}\right)} = \begin{bmatrix} \frac{\partial y_1}{\partial W_{11}} & \cdots & \frac{\partial y_1}{\partial W_{1n}} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \frac{\partial y_2}{\partial W_{21}} & \cdots & \frac{\partial y_2}{\partial W_{2n}} & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial y_3}{\partial W_{31}} & \cdots & \frac{\partial y_3}{\partial W_{3n}} & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \end{bmatrix}$$

Vector Jacobian Product (VJP)

- Jacobian is very sparse and explicit formation of it is too expensive

$$\frac{\partial z}{\partial y}\mathrm{reshape}\left(\frac{\partial y}{\partial W}\right) =$$

$$\begin{bmatrix} \frac{\partial z}{\partial y_1} & \frac{\partial z}{\partial y_2} & \cdots & \frac{\partial z}{\partial y_m} \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial W_{11}} & \cdots & \frac{\partial y_1}{\partial W_{1n}} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \frac{\partial y_2}{\partial W_{21}} & \cdots & \frac{\partial y_2}{\partial W_{2n}} & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial y_3}{\partial W_{31}} & \cdots & \frac{\partial y_3}{\partial W_{3n}} & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial W_{11}} & \cdots & \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial W_{1n}} & \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial W_{21}} & \cdots & \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial W_{2n}} & \cdots \end{bmatrix}$$

Vector Jacobian Product (VJP)

- Jacobian is very sparse and explicit formation of it is too expensive

$$\frac{\partial z}{\partial W} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial W} = \mathrm{reshape}\left(\frac{\partial z}{\partial y}\,\mathrm{reshape}\left(\frac{\partial y}{\partial W}\right)\right)$$

$$= \mathrm{reshape}\left(\left[\frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial W_{11}} \quad \cdots \quad \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial W_{1n}} \quad \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial W_{21}} \quad \cdots \quad \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial W_{2n}} \quad \cdots\right]\right)$$

$$= \begin{bmatrix} \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial W_{11}} & \cdots & \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial W_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial z}{\partial y_m}\frac{\partial y_m}{\partial W_{m1}} & \cdots & \frac{\partial z}{\partial y_m}\frac{\partial y_m}{\partial W_{mn}} \end{bmatrix} = \begin{bmatrix} \frac{\partial z}{\partial y_1}x_1 & \cdots & \frac{\partial z}{\partial y_1}x_n \\ \vdots & \ddots & \vdots \\ \frac{\partial z}{\partial y_m}x_1 & \cdots & \frac{\partial z}{\partial y_m}x_n \end{bmatrix} = \left(\frac{\partial z}{\partial y}\right)^\top x^\top$$

Vector Jacobian Product (VJP)

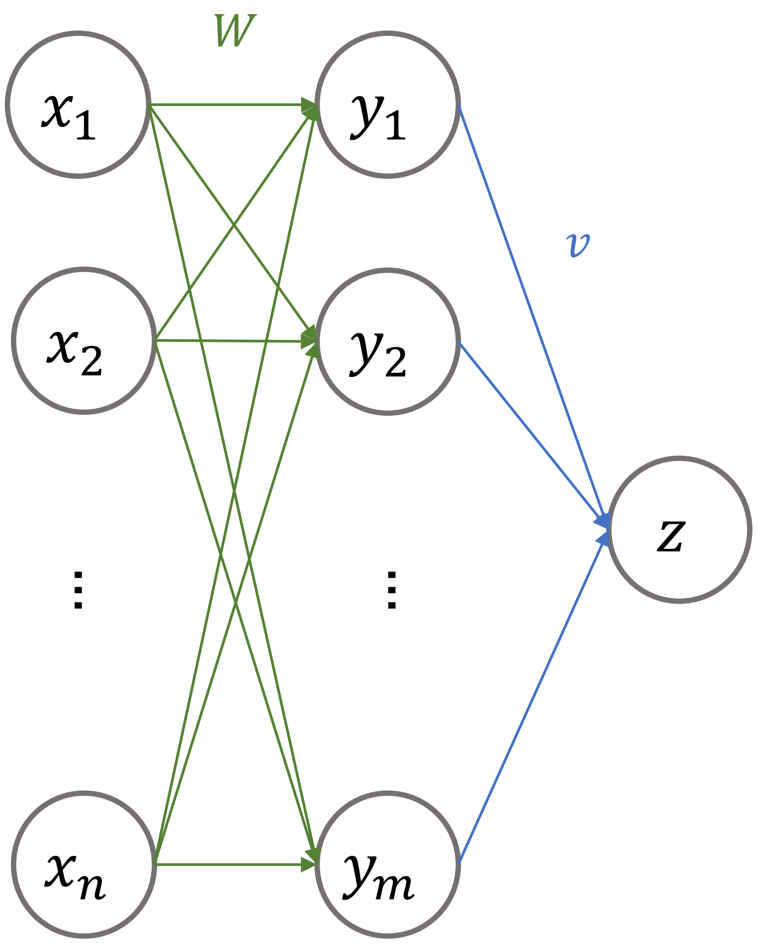
- Explicit formation of Jacobian is too expensive

$$x \in \mathbb{R}^n, y \in \mathbb{R}^m, z \in \mathbb{R}, W \in \mathbb{R}^{m\times n}, v \in \mathbb{R}^m$$

$$y = Wx \qquad z = \sum_{i=1}^{m} v_i y_i = v^\top y$$

$$\underbrace{\frac{\partial z}{\partial W}}_{\in \mathbb{R}^{m\times n}} = \underbrace{\frac{\partial z}{\partial y}}_{\in \mathbb{R}^{1\times m}} \underbrace{\frac{\partial y}{\partial W}}_{\in \mathbb{R}^{m\times m\times n}}$$

$$\underbrace{\frac{\partial z}{\partial W}}_{\in \mathbb{R}^{m\times n}} = \underbrace{\left(\frac{\partial z}{\partial y}\right)^\top}_{\in \mathbb{R}^{m\times 1}} \underbrace{x^\top}_{\in \mathbb{R}^{1\times n}}$$



We almost never explicitly construct Jacobians ($\frac{\partial y}{\partial W}$). We instead directly compute vector-Jacobian product (VJP, $\frac{\partial z}{\partial y}\frac{\partial y}{\partial W}$) in more efficient way ($\left(\frac{\partial z}{\partial y}\right)^\top x^\top$)

Vector Jacobian Product (VJP)

- Elementwise activation functions

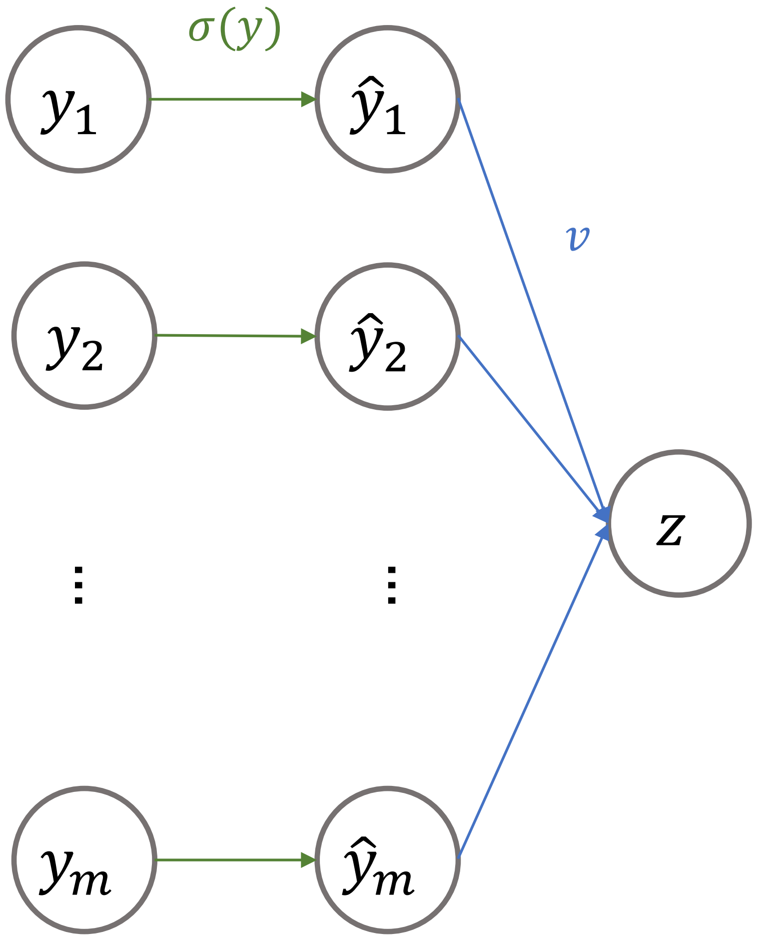
$$y \in \mathbb{R}^m, \hat{y} \in \mathbb{R}^m$$

$$\hat{y} = \sigma(y) \qquad z = \sum_{i=1}^{m} v_i \hat{y}_i = v^\top \hat{y}$$

$$\underbrace{\frac{\partial z}{\partial y}}_{\in \mathbb{R}^{1\times m}} = \underbrace{\frac{\partial z}{\partial \hat{y}}}_{\in \mathbb{R}^{1\times m}} \underbrace{\frac{\partial \hat{y}}{\partial y}}_{\in \mathbb{R}^{m\times m}}$$

$$\frac{\partial \hat{y}}{\partial y} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial y_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{\partial \hat{y}_m}{\partial y_m} \end{bmatrix}$$

$$= \begin{bmatrix} \sigma(y_1)(1-\sigma(y_1)) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma(y_m)(1-\sigma(y_m)) \end{bmatrix}$$



Vector Jacobian Product (VJP)

- Elementwise activation functions

$$\underbrace{\frac{\partial z}{\partial y}}_{\in \mathbb{R}^{1\times m}} = \underbrace{\frac{\partial z}{\partial \hat{y}}}_{\in \mathbb{R}^{1\times m}} \underbrace{\frac{\partial \hat{y}}{\partial y}}_{\in \mathbb{R}^{m\times m}}$$

$$\frac{\partial \hat{y}}{\partial y} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial y_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{\partial \hat{y}_m}{\partial y_m} \end{bmatrix} = \begin{bmatrix} \sigma(y_1)(1-\sigma(y_1)) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma(y_m)(1-\sigma(y_m)) \end{bmatrix}$$

Element-wise product

$$\frac{\partial z}{\partial y} = \underbrace{\frac{\partial z}{\partial \hat{y}}}_{\in \mathbb{R}^{1\times m}} \odot \overbrace{\Big(\sigma(y)\big(1-\sigma(y)\big)\Big)^{\top}}^{\in \mathbb{R}^{1\times m}}$$

Automatic Differentiation

Automatic Differentiation (AD)

- A procedure for automatic evaluation of derivatives of arbitrary algebraic functions
- Backpropagation == reverse-mode AD

Reverse-Mode AD (a.k.a Backpropagation)

$f: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$g: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$h: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$i: \mathbb{R}^{n_4} \to \mathbb{R}$

Loss function:
scalar function

$$
\begin{aligned}
b &= f(a) \\
c &= g(b) \\
d &= h(c) \\
e &= i(d)
\end{aligned}
$$

$$\frac{\partial e}{\partial a}?$$

Reverse-Mode AD (a.k.a Backpropagation)

$f: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$g: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$h: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$i: \mathbb{R}^{n_4} \to \mathbb{R}$

Loss function:
scalar function

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = i(d)$$

$$\frac{\partial e}{\partial d} = \underbrace{\frac{\partial e}{\partial e}}_{\in \mathbb{R}^{1\times 1}} \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{1\times n_4}} = 1 \frac{\partial e}{\partial d}$$

Reverse-Mode AD (a.k.a Backpropagation)

$f: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$g: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$h: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$i: \mathbb{R}^{n_4} \to \mathbb{R}$

Loss function:
scalar function

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = i(d)$$

$$\frac{\partial e}{\partial d} = \underbrace{\frac{\partial e}{\partial e}}_{\in \mathbb{R}^{1\times 1}} \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{1\times n_4}} = 1 \frac{\partial e}{\partial d}$$

$$\frac{\partial e}{\partial c} = \frac{\partial e}{\partial e} \frac{\partial e}{\partial d} \frac{\partial d}{\partial c} = \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{1\times n_4}} \underbrace{\frac{\partial d}{\partial c}}_{\in \mathbb{R}^{n_4\times n_3}}$$

Reverse-Mode AD (a.k.a Backpropagation)

$f: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$g: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$h: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$i: \mathbb{R}^{n_4} \to \mathbb{R}$

Loss function:
scalar function

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = i(d)$$

$$\frac{\partial e}{\partial d} = \underbrace{\frac{\partial e}{\partial e}}_{\in \mathbb{R}^{1\times 1}} \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{1\times n_4}} = 1 \frac{\partial e}{\partial d}$$

$$\frac{\partial e}{\partial c} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c} = \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{1\times n_4}} \underbrace{\frac{\partial d}{\partial c}}_{\in \mathbb{R}^{n_4\times n_3}}$$

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b} = \underbrace{\frac{\partial e}{\partial c}}_{\in \mathbb{R}^{1\times n_3}} \underbrace{\frac{\partial c}{\partial b}}_{\in \mathbb{R}^{n_3\times n_2}}$$

Reverse-Mode AD (a.k.a Backpropagation)

$f: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$g: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$h: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$i: \mathbb{R}^{n_4} \to \mathbb{R}$

Loss function:
scalar function

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = i(d)$$

$$\frac{\partial e}{\partial d} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d} = \underbrace{1}_{\in \mathbb{R}^{1\times 1}} \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{1\times n_4}}$$

$$\frac{\partial e}{\partial c} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c} = \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{1\times n_4}} \underbrace{\frac{\partial d}{\partial c}}_{\in \mathbb{R}^{n_4\times n_3}}$$

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b} = \underbrace{\frac{\partial e}{\partial c}}_{\in \mathbb{R}^{1\times n_3}} \underbrace{\frac{\partial c}{\partial b}}_{\in \mathbb{R}^{n_3\times n_2}}$$

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b}\frac{\partial b}{\partial a} = \underbrace{\frac{\partial e}{\partial b}}_{\in \mathbb{R}^{1\times n_2}} \underbrace{\frac{\partial b}{\partial a}}_{\in \mathbb{R}^{n_2\times n_1}}$$

$\boxed{\text{Vector-Jacobian Product (VJP)}}$

Reverse-Mode AD (a.k.a Backpropagation)

$f: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$g: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$h: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$i: \mathbb{R}^{n_4} \to \mathbb{R}^{3}$

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = i(d)$$

What if i is vector-valued function?

$$\frac{\partial e}{\partial d} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\in \mathbb{R}^{3\times 3}} \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{3\times n_4}}$$

3 X Computation

$$\frac{\partial e}{\partial c} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c} = \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{3\times n_4}} \underbrace{\frac{\partial d}{\partial c}}_{\in \mathbb{R}^{n_4\times n_3}}$$

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b} = \underbrace{\frac{\partial e}{\partial c}}_{\in \mathbb{R}^{3\times n_3}} \underbrace{\frac{\partial c}{\partial b}}_{\in \mathbb{R}^{n_3\times n_2}}$$

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b}\frac{\partial b}{\partial a} = \underbrace{\frac{\partial e}{\partial b}}_{\in \mathbb{R}^{3\times n_2}} \underbrace{\frac{\partial b}{\partial a}}_{\in \mathbb{R}^{n_2\times n_1}}$$

Reverse-Mode AD (a.k.a Backpropagation)

$f: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$g: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$h: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$i: \mathbb{R}^{n_4} \to \mathbb{R}^{3}$

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = i(d)$$

$$\frac{\partial e}{\partial d} = \frac{\partial e}{\partial e}\frac{\partial e}{\partial d} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\in \mathbb{R}^{3\times 3}} \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{3\times n_4}}$$

$$\frac{\partial e_1}{\partial d}$$

Forward-Mode AD

single variable

$f: \mathbb{R} \to \mathbb{R}^{n_1}$

$g: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$h: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$L: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = L(d)$$

$$\frac{\partial b}{\partial a} = \underbrace{\frac{\partial b}{\partial a}}^{\in \mathbb{R}^{n_1 \times 1}} \overbrace{\frac{\partial a}{\partial a}}^{\in \mathbb{R}^{1\times 1}} = \frac{\partial b}{\partial a} 1$$

Forward-Mode AD

single variable

$f: \mathbb{R} \to \mathbb{R}^{n_1}$

$g: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$h: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$L: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = L(d)$$

$$\frac{\partial b}{\partial a} = \underbrace{\frac{\partial b}{\partial a}}^{\in \mathbb{R}^{n_1 \times 1}} \overbrace{\frac{\partial a}{\partial a}}^{\in \mathbb{R}^{1 \times 1}} = \frac{\partial b}{\partial a} 1$$

$$\frac{\partial c}{\partial a} = \frac{\partial c}{\partial b} \frac{\partial b}{\partial a} \frac{\partial a}{\partial a} = \overbrace{\frac{\partial c}{\partial b}}^{\in \mathbb{R}^{n_2 \times n_1}} \overbrace{\frac{\partial b}{\partial a}}^{\in \mathbb{R}^{n_1 \times 1}}$$

Forward-Mode AD

single variable

$f: \mathbb{R} \to \mathbb{R}^{n_1}$

$g: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$h: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$L: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = L(d)$$

$$\frac{\partial b}{\partial a} = \underbrace{\frac{\partial b}{\partial a}}_{\in \mathbb{R}^{n_1 \times 1}} \underbrace{\frac{\partial a}{\partial a}}_{\in \mathbb{R}^{1 \times 1}} = \frac{\partial b}{\partial a} 1$$

$$\frac{\partial c}{\partial a} = \frac{\partial c}{\partial b} \frac{\partial b}{\partial a} \frac{\partial a}{\partial a} = \underbrace{\frac{\partial c}{\partial b}}_{\in \mathbb{R}^{n_2 \times n_1}} \underbrace{\frac{\partial b}{\partial a}}_{\in \mathbb{R}^{n_1 \times 1}}$$

$$\frac{\partial d}{\partial a} = \frac{\partial d}{\partial c} \frac{\partial c}{\partial b} \frac{\partial b}{\partial a} \frac{\partial a}{\partial a} = \underbrace{\frac{\partial d}{\partial c}}_{\in \mathbb{R}^{n_3 \times n_2}} \underbrace{\frac{\partial c}{\partial a}}_{\in \mathbb{R}^{n_2 \times 1}}$$

Forward-Mode AD

single variable

$f: \mathbb{R} \to \mathbb{R}^{n_1}$

$g: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$

$h: \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$

$L: \mathbb{R}^{n_3} \to \mathbb{R}^{n_4}$

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = L(d)$$

$$\frac{\partial b}{\partial a} = \frac{\partial b}{\partial a}\frac{\partial a}{\partial a} = \underbrace{\frac{\partial b}{\partial a}}_{\in \mathbb{R}^{n_1 \times 1}} \underbrace{1}_{\in \mathbb{R}^{1 \times 1}}$$

$$\frac{\partial c}{\partial a} = \frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial a} = \underbrace{\frac{\partial c}{\partial b}}_{\in \mathbb{R}^{n_2 \times n_1}} \underbrace{\frac{\partial b}{\partial a}}_{\in \mathbb{R}^{n_1 \times 1}}$$

$$\frac{\partial d}{\partial a} = \frac{\partial d}{\partial c}\frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial a} = \underbrace{\frac{\partial d}{\partial c}}_{\in \mathbb{R}^{n_3 \times n_2}} \underbrace{\frac{\partial c}{\partial a}}_{\in \mathbb{R}^{n_2 \times 1}}$$

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial a} = \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{n_4 \times n_3}} \underbrace{\frac{\partial d}{\partial a}}_{\in \mathbb{R}^{n_3 \times 1}}$$

Jacobian-Vector Product (JVP)

Forward-Mode AD

$f: \mathbb{R}^3 \rightarrow \mathbb{R}^{n_1}$

$g: \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_2}$

$h: \mathbb{R}^{n_2} \rightarrow \mathbb{R}^{n_3}$

$L: \mathbb{R}^{n_3} \rightarrow \mathbb{R}^{n_4}$

$$b = f(a)$$
$$c = g(b)$$
$$d = h(c)$$
$$e = L(d)$$

What if input is Multi-variables?

$$\frac{\partial b}{\partial a} = \frac{\partial b}{\partial a}\frac{\partial a}{\partial a} = \underbrace{\frac{\partial b}{\partial a}}_{\in \mathbb{R}^{n_1 \times 3}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\in \mathbb{R}^{3 \times 3}}$$

3 X Computation

$$\frac{\partial c}{\partial a} = \frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial a} = \underbrace{\frac{\partial c}{\partial b}}_{\in \mathbb{R}^{n_2 \times n_1}}\underbrace{\frac{\partial b}{\partial a}}_{\in \mathbb{R}^{n_1 \times 3}}$$

$$\frac{\partial d}{\partial a} = \frac{\partial d}{\partial c}\frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial a} = \underbrace{\frac{\partial d}{\partial c}}_{\in \mathbb{R}^{n_3 \times n_2}}\underbrace{\frac{\partial c}{\partial a}}_{\in \mathbb{R}^{n_2 \times 3}}$$

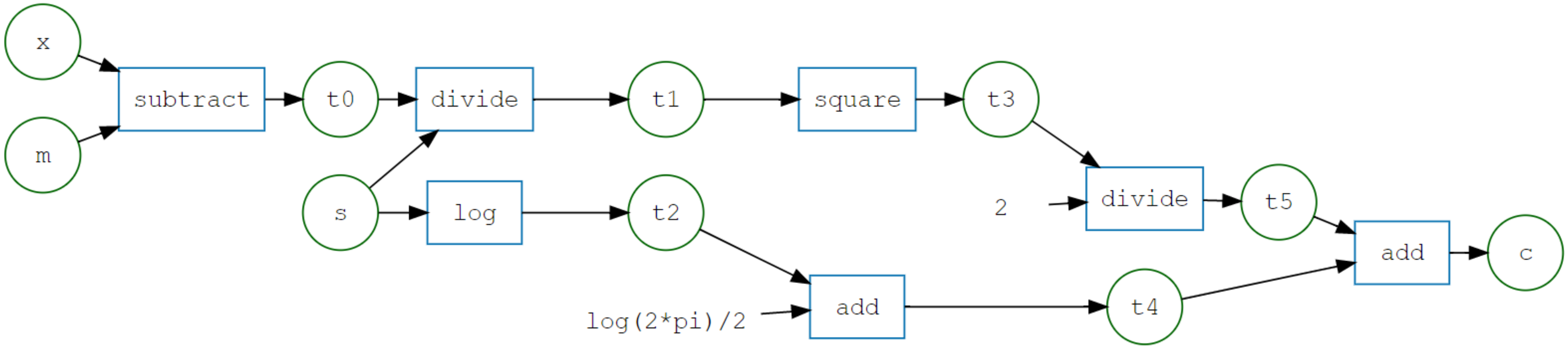
$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial a} = \underbrace{\frac{\partial e}{\partial d}}_{\in \mathbb{R}^{n_4 \times n_3}}\underbrace{\frac{\partial d}{\partial a}}_{\in \mathbb{R}^{n_3 \times 3}}$$

Automatic Differentiation (AD)

- For low dimensional outputs and high dimensional inputs
  - Objective function w/ deep neural networks
  - reverse-mode AD
- For high dimensional outputs and low dimensional inputs
  - Forward-mode AD

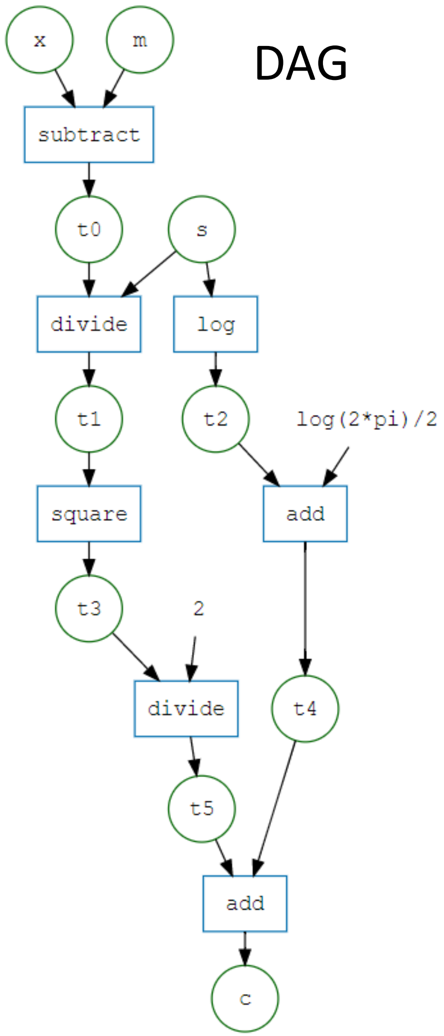
Computational Graph

```
t0 = x - m
t1 = t0 / s
t2 = np.log(s)
t3 = t1**2
t4 = t2 + np.log(2 * np.pi) / 2
t5 = t3 / 2
c = t4 + t5
```

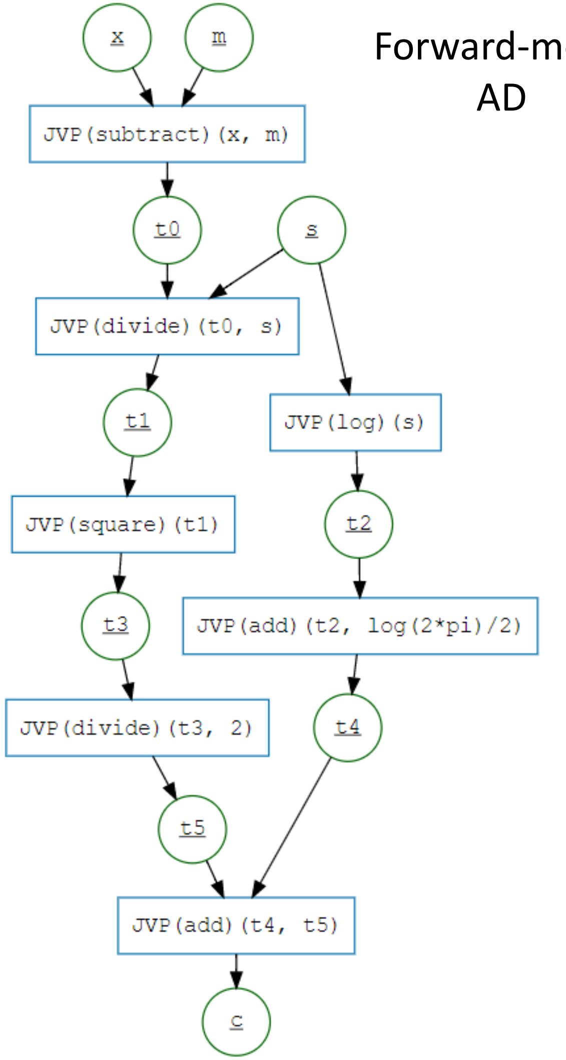


https://matt-graham.github.io/

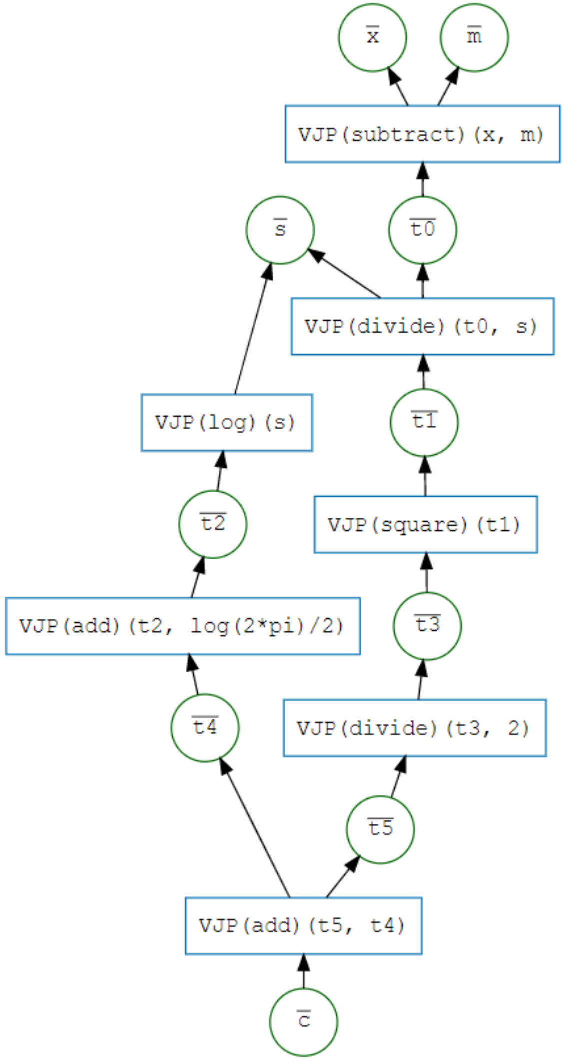
Automatic Differentiation



DAG



Forward-mode AD



Reverse-mode AD

https://matt-graham.github.io/

References

- mattjj/autodidact: A pedagogical implementation of Autograd (github.com)
- [1502.05767] Automatic differentiation in machine learning: a survey (arxiv.org)
- CSC321 Lecture 10: Automatic Differentiation (toronto.edu)