10-315
Introduction to ML

Neural Networks

Instructor: Pat Virtue

Outline

Last time: Neural Networks

- Three-neuron network + optimization
- Neural network structure (adding more neurons)
- Forward pass and Backpropagation (scalar version)

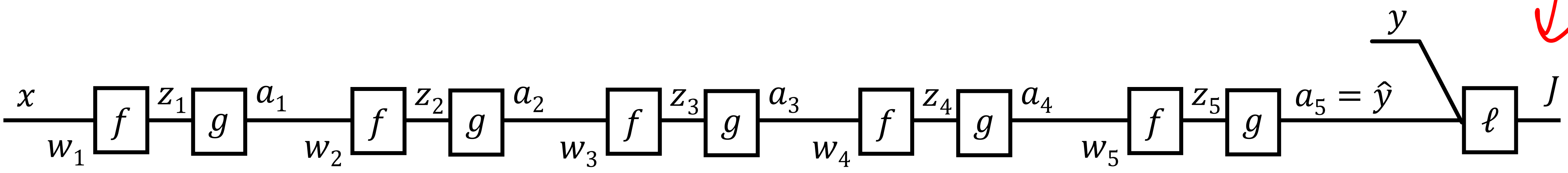
Today: Neural Networks

- Calculus: multi-variate chain rule
- Backpropagation (vector version)
- Optimization intuition (sliders)
- Neural Network Properties and Intuition

Forward Pass

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g(w_2 \cdot g(w_1 \cdot x))\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)



With a new data point (x, y), we have our current weight values but we don't have $\hat{y}$ (or any of the intermediate values z_ℓ and a_ℓ) or the value of our object function J

$$J(\theta) = \frac{1}{N} \sum J^{(i)}(\theta)$$

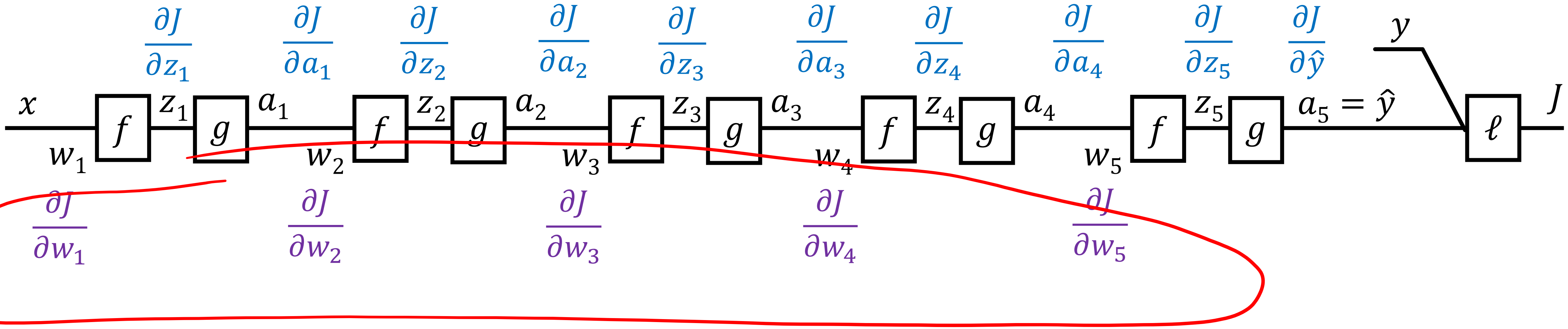
The forward pass propagates x (forward) through the network to give us these values

$$J^{(i)}(\theta) = \ell(y, \hat{y})$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g(w_2 \cdot g(w_1 \cdot x))\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)



To do gradient descent we need the partial derivative of the objective with respect to each parameter, $w_\ell \leftarrow w_\ell - \alpha\, \partial J / \partial w_\ell$

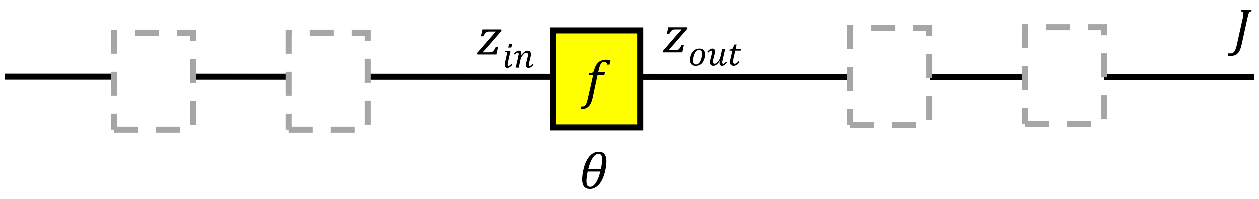
The backward pass propagates the change in the objective with respect to intermediate values ($\partial J / \partial z_\ell$ and $\partial J / \partial a_\ell$) back through the network to produce each $\partial J / \partial w_\ell$

Generic Layer Implementation (so-far)

Compute derivatives per layer, utilizing previous derivatives

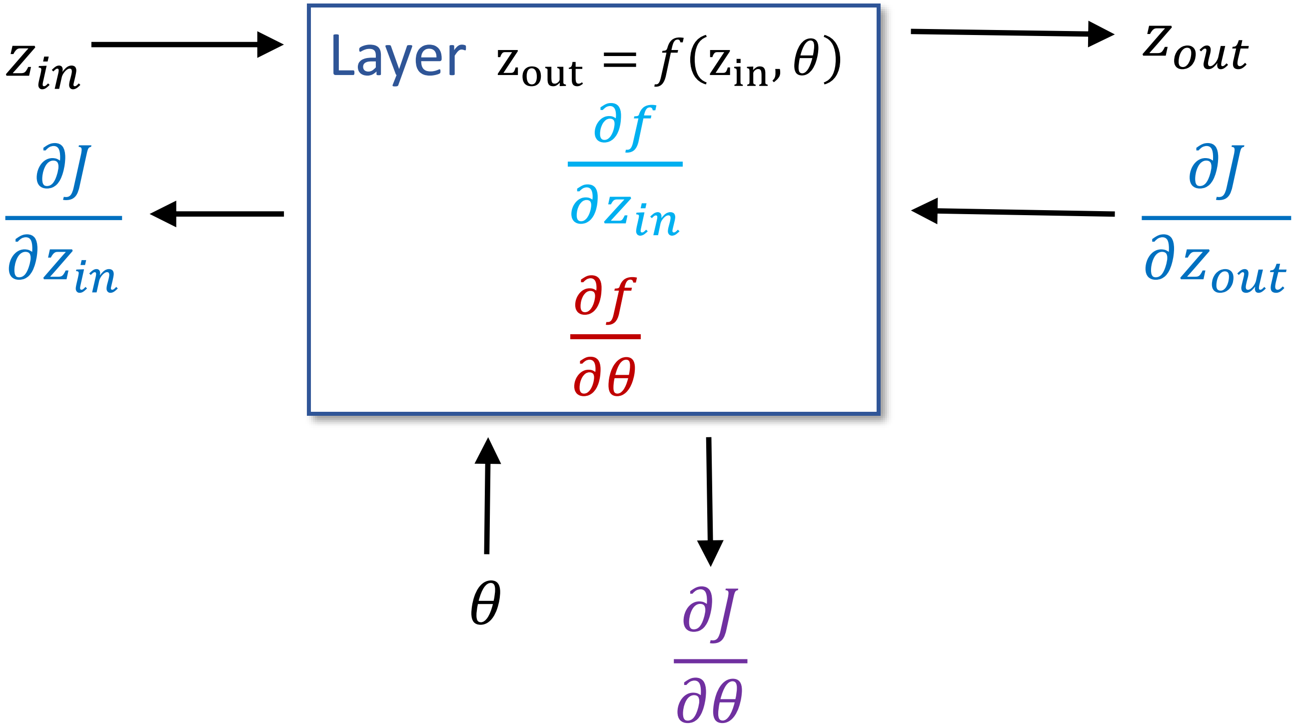
Objective: $J(\theta)$

Arbitrary layer: $\mathrm{z}_{\mathrm{out}} = f(\mathrm{z}_{\mathrm{in}}, \theta)$



Need:

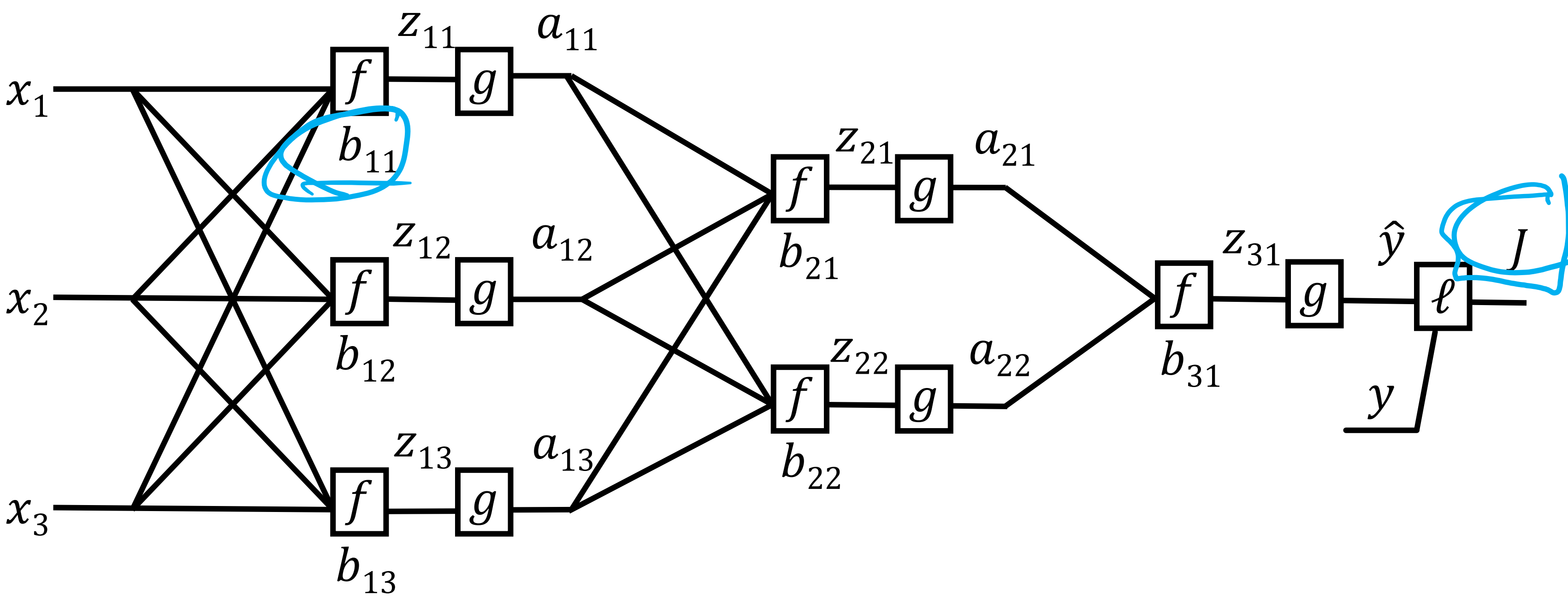
- $\dfrac{\partial J}{\partial z_{in}} = \dfrac{\partial J}{\partial z_{out}} \dfrac{\partial f}{\partial z_{in}}$
- $\dfrac{\partial J}{\partial \theta} = \dfrac{\partial J}{\partial z_{out}} \dfrac{\partial f}{\partial \theta}$



Optimization

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(b_{3,1} + \sum_k w_{3,1,k}\ g\left(b_{2,k} + \sum_i w_{2,k,i}\ g\left(b_{1,i} + \sum_j w_{1,i,j}\ x_j\right)\right)\right)$$

Tons of repeated partial derivatives



$$\frac{\partial J}{\partial b_{3,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial b_{3,1}}$$

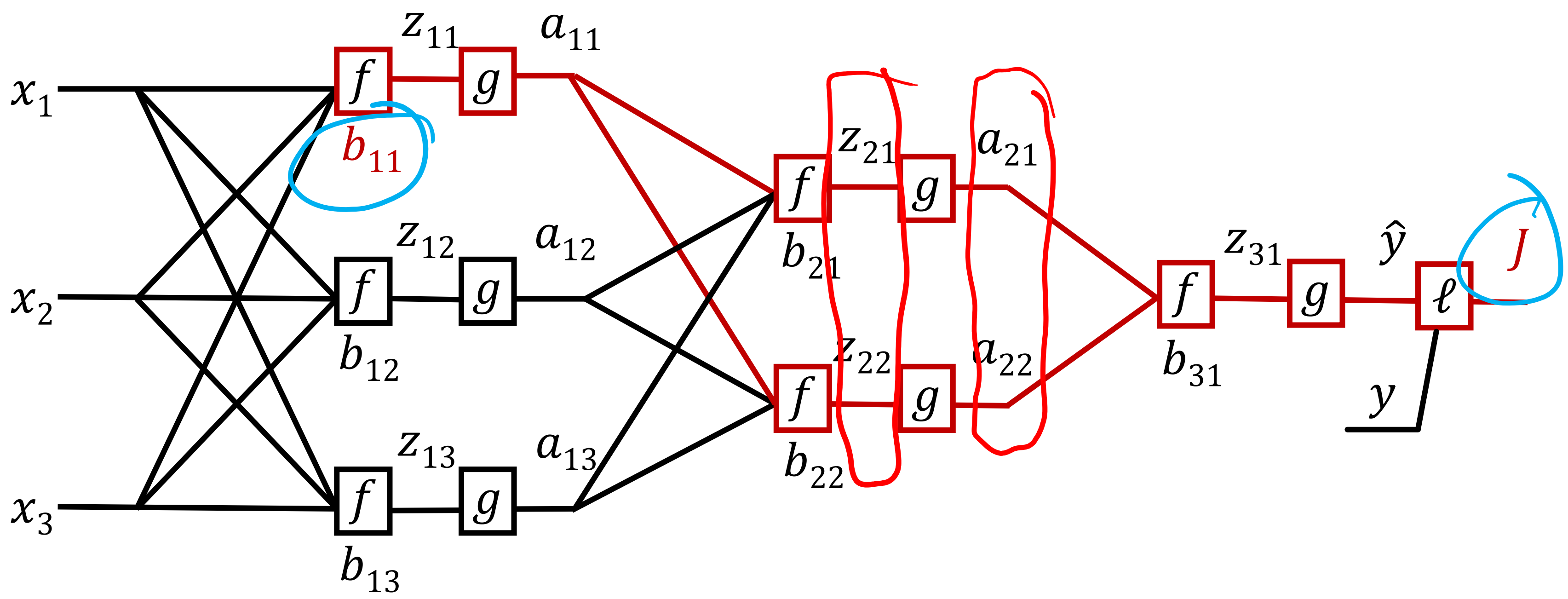
$$\frac{\partial J}{\partial b_{2,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial a_{2,i}} \frac{\partial a_{2,i}}{\partial z_{2,i}} \frac{\partial z_{2,i}}{\partial b_{2,i}}$$

$$\boxed{\frac{\partial J}{\partial b_{1,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \boxed{\frac{\partial z_{3,1}}{\partial \mathbf{a}_2} \frac{\partial \mathbf{a}_2}{\partial \mathbf{z}_2} \frac{\partial \mathbf{z}_2}{\partial a_{1,i}}} \frac{\partial a_{1,i}}{\partial z_{1,i}} \frac{\partial z_{1,i}}{\partial b_{1,i}}}$$

Optimization

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(b_{3,1} + \sum_k w_{3,1,k}\ g\left(b_{2,k} + \sum_i w_{2,k,i}\ g\left(b_{1,i} + \sum_j w_{1,i,j}\ x_j\right)\right)\right)$$

Tons of repeated partial derivatives



$$\frac{\partial J}{\partial b_{3,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial b_{3,1}}$$

$$\frac{\partial J}{\partial b_{2,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial a_{2,i}} \frac{\partial a_{2,i}}{\partial z_{2,i}} \frac{\partial z_{2,i}}{\partial b_{2,i}}$$

$$\frac{\partial J}{\partial b_{1,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \boxed{\frac{\partial z_{3,1}}{\partial \mathbf{a}_2} \frac{\partial \mathbf{a}_2}{\partial \mathbf{z}_2} \frac{\partial \mathbf{z}_2}{\partial a_{1,i}}} \frac{\partial a_{1,i}}{\partial z_{1,i}} \frac{\partial z_{1,i}}{\partial b_{1,i}}$$

Calculus Time-out

Multivariate-chain rule

Multivariable Chain Rule

$g_1(x) = 3x$

$g_2(x) = 5x$

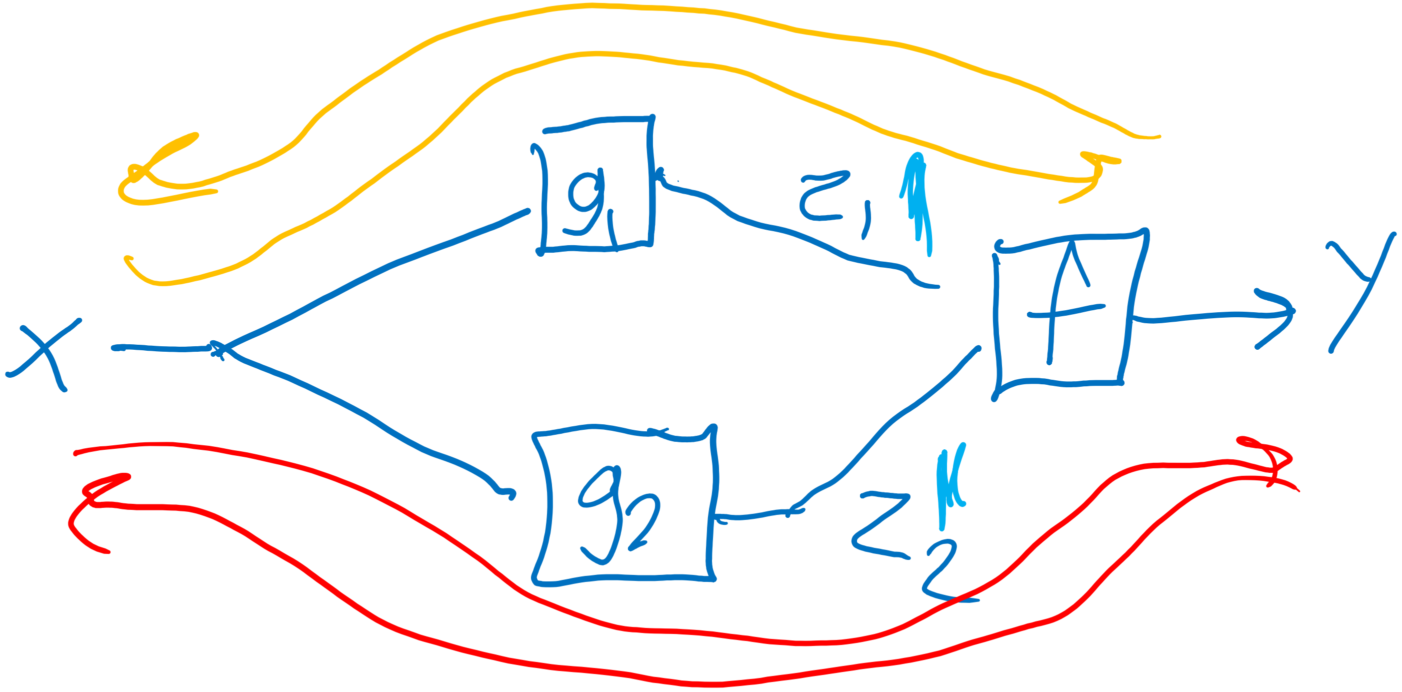
$f(z_1, z_2) = 2z_1 + 7z_2$

$y = f(g_1(x), g_2(x))$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z_1}\frac{\partial z_1}{\partial x} + \frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial x}$$

$$= (2)(3) + (7)(5)$$

$$= 41$$



Finite differences check

Let $x = 10$ and $\epsilon = 0.01$

$f(10) = 2(3 \cdot 10) + 7(5 \cdot 10) = 410$

$f(10+\epsilon) = 2(3 \cdot 10.01) + 7(5 \cdot 10.01) = 410.41$

$$\frac{f(10+\epsilon) - f(10)}{(10+\epsilon) - 10} = \frac{410.41 - 410}{0.01} = 41 \checkmark$$

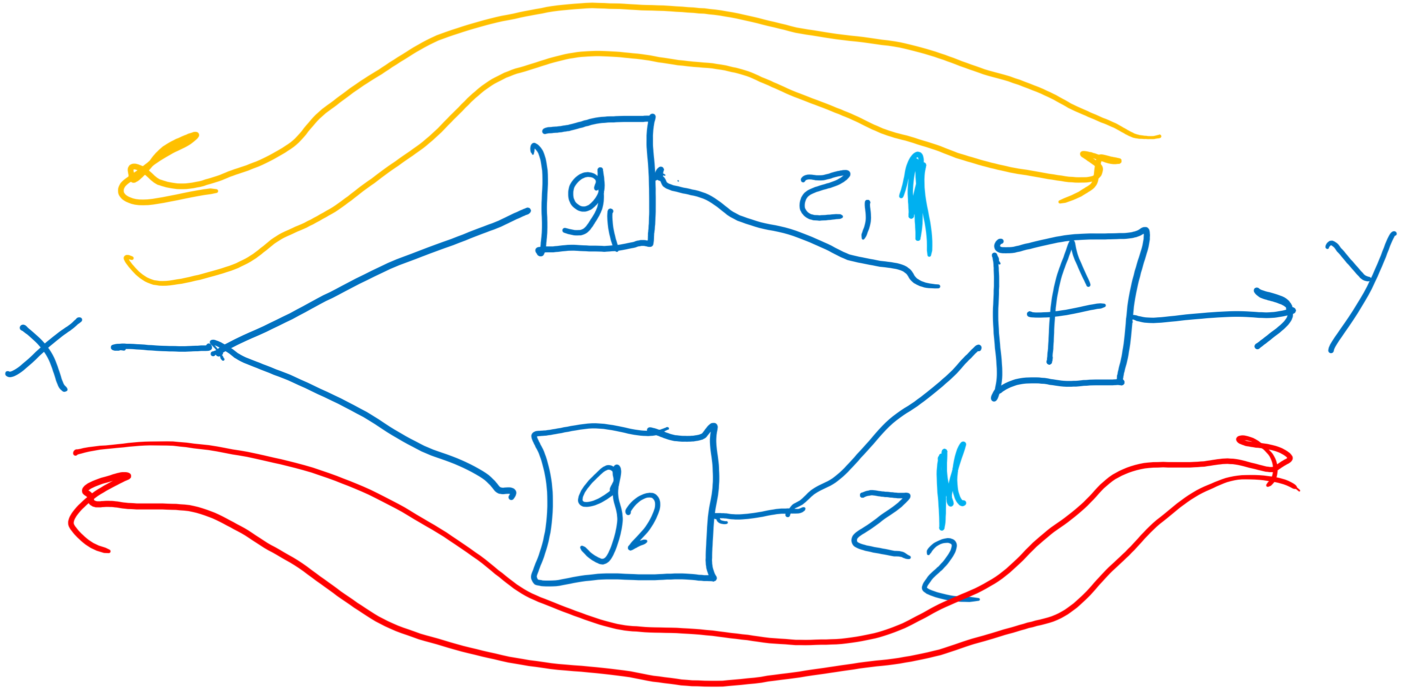
Multivariable Chain Rule

$$g_1(x) = 3x$$
$$g_2(x) = 5x$$
$$f(z_1, z_2) = 2z_1 + 7z_2$$
$$y = f(g_1(x), g_2(x))$$

$$\frac{\partial y}{\partial x} = \underbrace{\frac{\partial y}{\partial z_1}\frac{\partial z_1}{\partial x}} + \underbrace{\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial x}}$$

$$= (2)(3) + (7)(5)$$

$$= 41$$



Finite differences check

$$f(x) = 2(3x) + 7(5x) = 41x$$

$$f(x+\epsilon) = 2(3(x+\epsilon)) + 7(5(x+\epsilon))$$

$$= 6x + 6\epsilon + 35x + 35\epsilon = 41x + 41\epsilon$$

$$\frac{f(x+\epsilon) - f(x)}{(x+\epsilon) - x} = \frac{(\cancel{41x} + 41\epsilon) - \cancel{41x}}{\epsilon} = 41$$

Exercise: Multivariable Chain Rule

$$\boxed{\frac{df}{dx} = \frac{\partial f}{\partial g_1}\frac{\partial g_1}{\partial x} + \frac{\partial f}{\partial g_2}\frac{\partial g_2}{\partial x}}$$

$z_1 = g_1(x) = \sin(x)$ $\qquad \partial g_1/\partial x = \cos x$

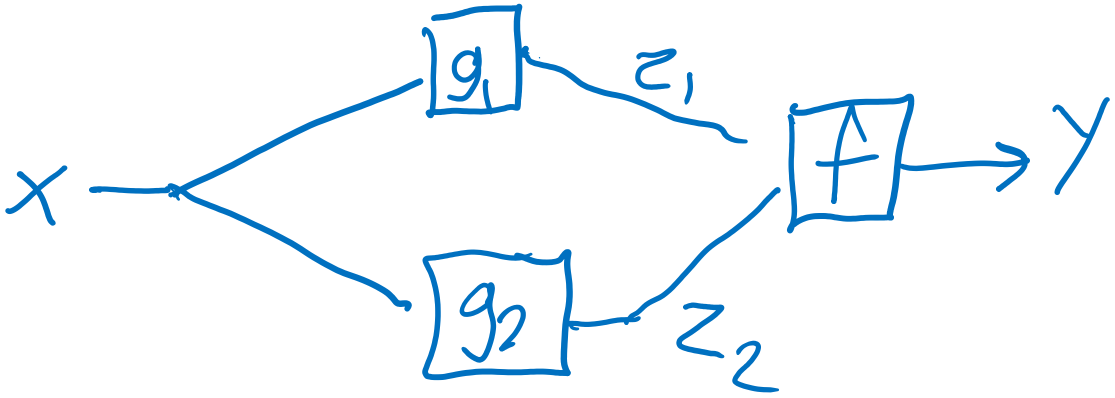
$z_2 = g_2(x) = x^3$ $\qquad \partial g_2/\partial x = 3x^2$

$y = f(z_1, z_2) = z_1^4 e^{z_2} + 5z_1 + 7z_2$

$$\frac{\partial f}{\partial z_1} = 4z_1^3 e^{z_2} + 5 + 0$$

$$\frac{\partial f}{\partial z_2} = z_1^4 e^{z_2} + 0 + 7$$

$$\frac{df}{dx} = (4z_1^3 e^{z_2} + 5)\cos x + (z_1^4 e^{z_2} + 7)\cdot 3x^2$$



Multivariable Chain Rule

$$\boxed{\frac{df}{dx} = \frac{\partial f}{\partial g_1}\frac{\partial g_1}{\partial x} + \frac{\partial f}{\partial g_2}\frac{\partial g_2}{\partial x}}$$

$z_1 = g_1(x) = \sin(x)$

$z_2 = g_2(x) = x^3$

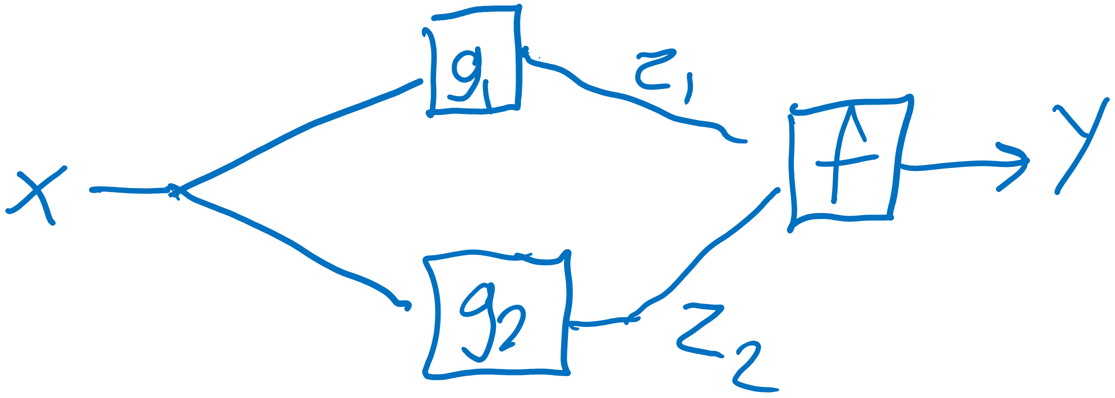
$y = f(z_1, z_2) = z_1 z_2$

$\partial g_1 / \partial x = \cos x$

$\partial g_2 / \partial x = 3x^2$

$$\frac{\partial f}{\partial z_1} = z_2$$

$$\frac{\partial f}{\partial z_2} = z_1$$



$$\frac{df}{dx} = z_2 \frac{dz_1}{dx} + z_1 \frac{dz_2}{dx}$$

Product Rule!

Calculus Chain Rule

$x \xrightarrow{g} z \xrightarrow{f} y$

Scalar:

$$y = f(z)$$

$$z = g(x)$$

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

Multivariate:

$$y = f(\mathbf{z})$$

$$\mathbf{z} = g(x)$$

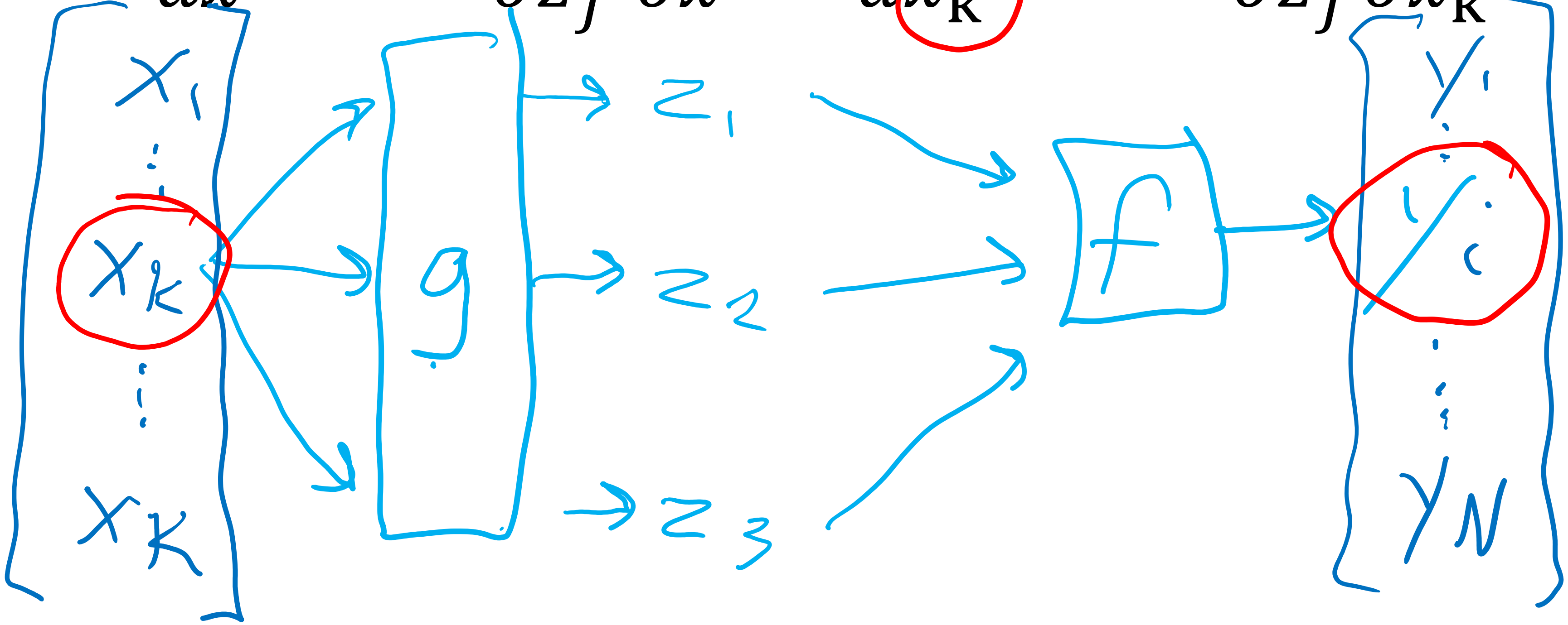
$$\frac{dy}{dx} = \sum_j \frac{\partial y}{\partial z_j}\frac{\partial z_j}{\partial x}$$

Multivariate:

$$\mathbf{y} = f(\mathbf{z})$$

$$\mathbf{z} = g(\mathbf{x})$$

$$\frac{dy_{\mathrm{i}}}{dx_{\mathrm{k}}} = \sum_j \frac{\partial y_i}{\partial z_j}\frac{\partial z_j}{\partial x_{\mathrm{k}}}$$



Multivariable Chain Rule

$\dfrac{\partial \vec{y}}{\partial \vec{x}}$

	Numerator layout	Denominator layout	Notes
$\frac{d}{dt}\ f\left(g(t), h(t)\right)$	$\frac{df}{dg}\frac{dg}{dt} + \frac{df}{dh}\frac{dh}{dt}$	Same	$f : (\mathbb{R}\times\mathbb{R}) \to \mathbb{R},\ t \in \mathbb{R}$ $g : \mathbb{R} \to \mathbb{R},\ h : \mathbb{R} \to \mathbb{R}$
$\frac{d}{dt}\ f\left(g_1(t), \ldots, g_N(t)\right)$	$\sum_{i=1}^{N} \frac{df}{dg_i}\frac{dg_i}{dt}$	Same	$h : (\mathbb{R}\times\cdots\times\mathbb{R}) \to \mathbb{R},\ t \in \mathbb{R}$ $f_i : \mathbb{R} \to \mathbb{R}\ \ \forall i \in \{1, \ldots, N\}$
$\frac{d}{dt}\ f(\mathbf{g}(t))$	$\frac{\partial f}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial t}$	$\frac{\partial \mathbf{g}}{\partial t}\frac{\partial f}{\partial \mathbf{g}}$	$f : \mathbb{R}^N \to \mathbb{R},\ \mathbf{g} : \mathbb{R} \to \mathbb{R}^N$ $t \in \mathbb{R}$
$\frac{\partial}{\partial \mathbf{v}}\ f(\mathbf{g}(\mathbf{v}))$	$\frac{\partial f}{\partial \mathbf{g}}\boxed{\frac{\partial \mathbf{g}}{\partial \mathbf{v}}}$	$\boxed{\frac{\partial \mathbf{g}}{\partial \mathbf{v}}}\frac{\partial f}{\partial \mathbf{g}}$	$f : \mathbb{R}^N \to \mathbb{R},\ \mathbf{g} : \mathbb{R}^M \to \mathbb{R}^N$ $\mathbf{v} \in \mathbb{R}^M$
$\frac{\partial}{\partial \mathbf{v}}\ f(\mathbf{g}(\mathbf{v}), \mathbf{h}(\mathbf{v}))$	$\frac{\partial f}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial \mathbf{v}} + \frac{\partial f}{\partial \mathbf{h}}\frac{\partial \mathbf{h}}{\partial \mathbf{v}}$	$\frac{\partial \mathbf{g}}{\partial \mathbf{v}}\frac{\partial f}{\partial \mathbf{g}} + \frac{\partial \mathbf{h}}{\partial \mathbf{v}}\frac{\partial f}{\partial \mathbf{h}}$	$h : \mathbb{R}^K \times \mathbb{R}^N \to \mathbb{R},\ \mathbf{v} \in \mathbb{R}^M$ $f : \mathbb{R}^M \to \mathbb{R}^K,\ \mathbf{g} : \mathbb{R}^M \to \mathbb{R}^N$

Backpropagation (vector version)

Network Optimization

$$J(\boldsymbol{w}) = z_4$$

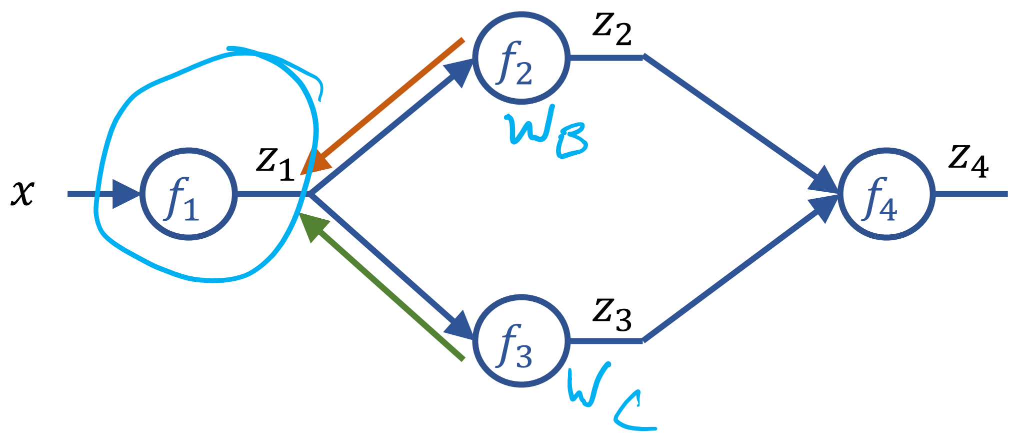
$$z_4 = f_4(w_D, w_E, z_2, z_3)$$

$$z_3 = f_3(w_C, z_1)$$

$$z_2 = f_2(w_B, z_1)$$

$$z_1 = f_1(w_A, x)$$

Need multivariate chain rule!



$$\frac{\partial J}{\partial w_E} = \frac{\partial J}{\partial z_4}\frac{\partial z_4}{\partial w_E}$$

$$\frac{\partial J}{\partial w_D} = \frac{\partial J}{\partial z_4}\frac{\partial z_4}{\partial w_D}$$

$$\frac{\partial J}{\partial z_3} = \frac{\partial J}{\partial z_4}\frac{\partial z_4}{\partial z_3}$$

$$\frac{\partial J}{\partial z_2} = \frac{\partial J}{\partial z_4}\frac{\partial z_4}{\partial z_2}$$

$$\boxed{\begin{aligned}\frac{\partial J}{\partial w_C} &= \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial w_C}\\ \frac{\partial J}{\partial w_B} &= \frac{\partial J}{\partial z_2}\frac{\partial z_2}{\partial w_B}\end{aligned}}$$

$$\frac{\partial J}{\partial z_1} = \frac{\partial J}{\partial z_2}\frac{\partial z_2}{\partial z_1} + \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial z_1}$$

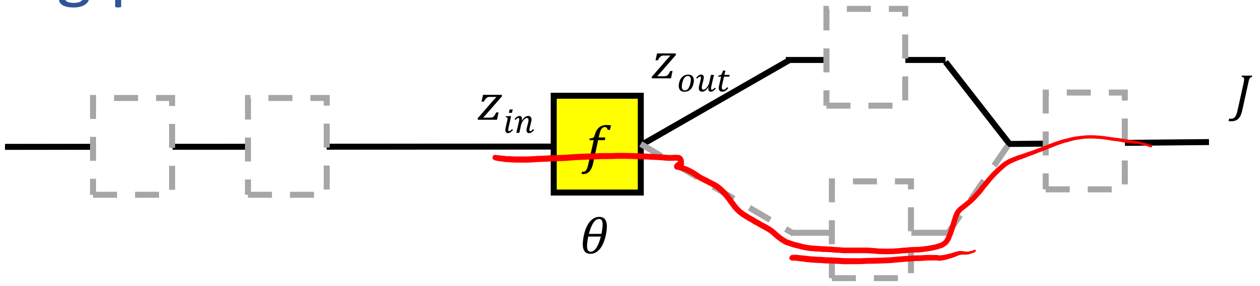
$$\frac{\partial J}{\partial w_A} = \frac{\partial J}{\partial z_1}\frac{\partial z_1}{\partial w_A}$$

Generic Layer Implementation (updated)

Compute derivatives per layer, utilizing previous derivatives

Objective: $J(\theta)$

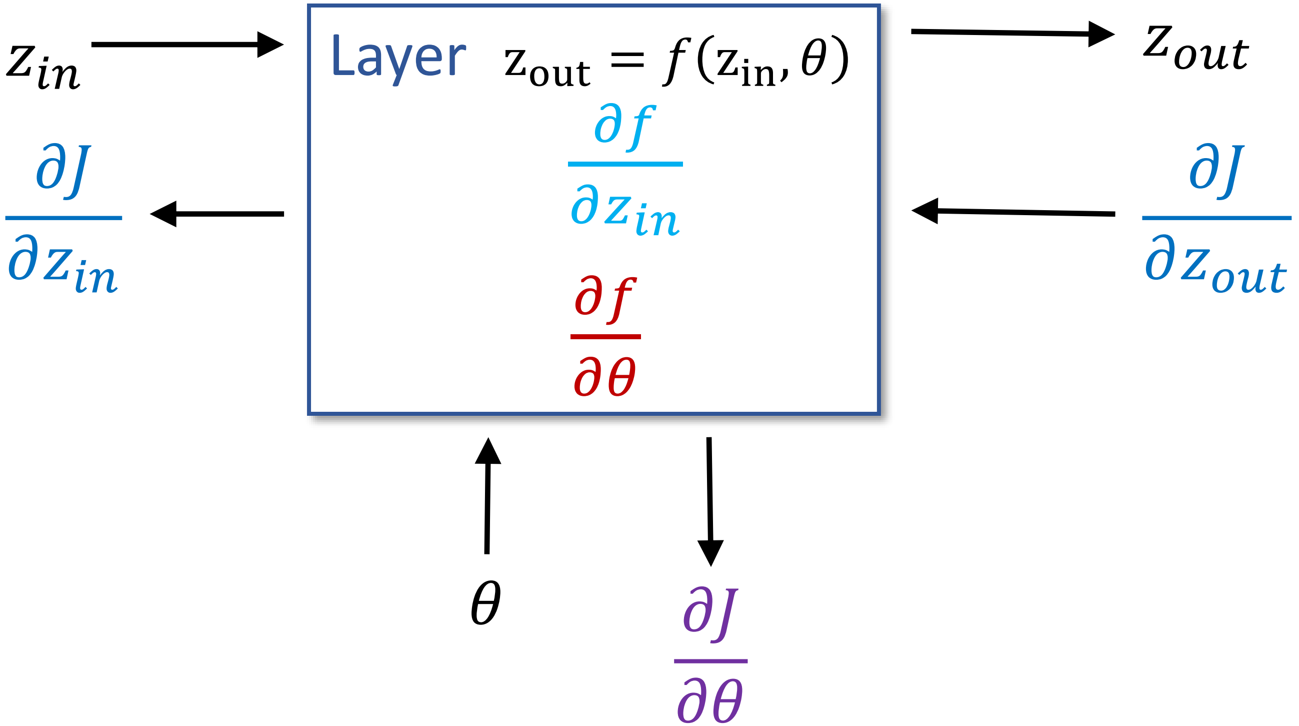
Arbitrary layer: $z_{out} = f(z_{in}, \theta)$



Need:

- $\frac{\partial J}{\partial z_{in}} += \frac{\partial J}{\partial z_{out}} \frac{\partial f}{\partial z_{in}}$
- $\frac{\partial J}{\partial \theta} += \frac{\partial J}{\partial z_{out}} \frac{\partial f}{\partial \theta}$

init to zero

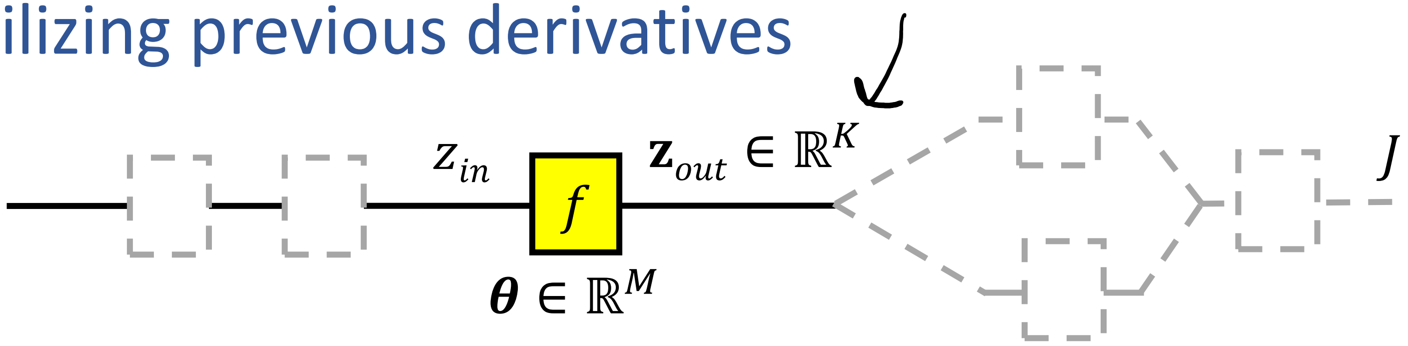


Generic Layer Implementation (vector output version)

Compute derivatives per layer, utilizing previous derivatives

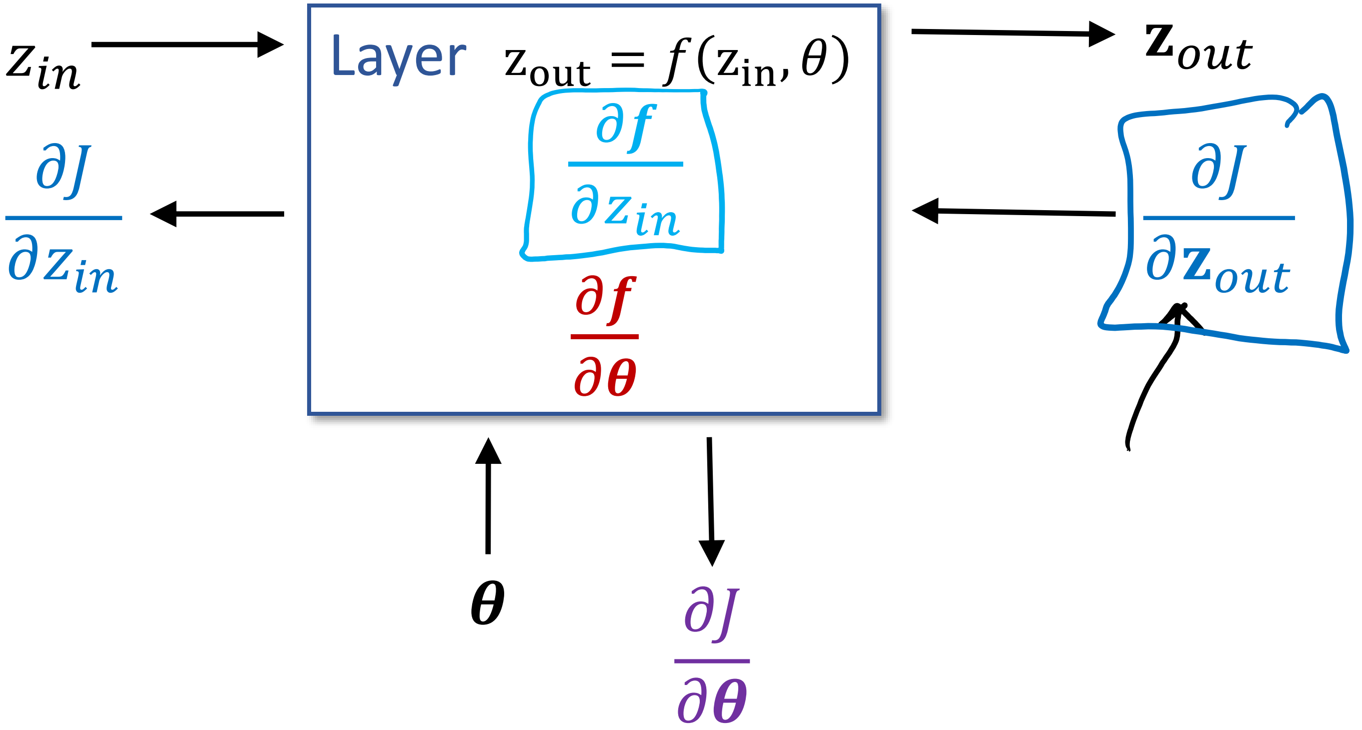
Objective: $J(\theta)$

Arbitrary layer: $\mathrm{z}_{\text{out}} = f(\mathrm{z}_{\text{in}}, \theta)$



Need:

- $\dfrac{\partial J}{\partial z_{in}} = \sum_{k=1}^{K} \boxed{\dfrac{\partial J}{\partial z_k}} \boxed{\dfrac{\partial z_k}{\partial z_{in}}}$
- $\dfrac{\partial J}{\partial \theta_j} = \sum_{k=1}^{K} \dfrac{\partial J}{\partial z_k} \dfrac{\partial z_k}{\partial \theta_j}$

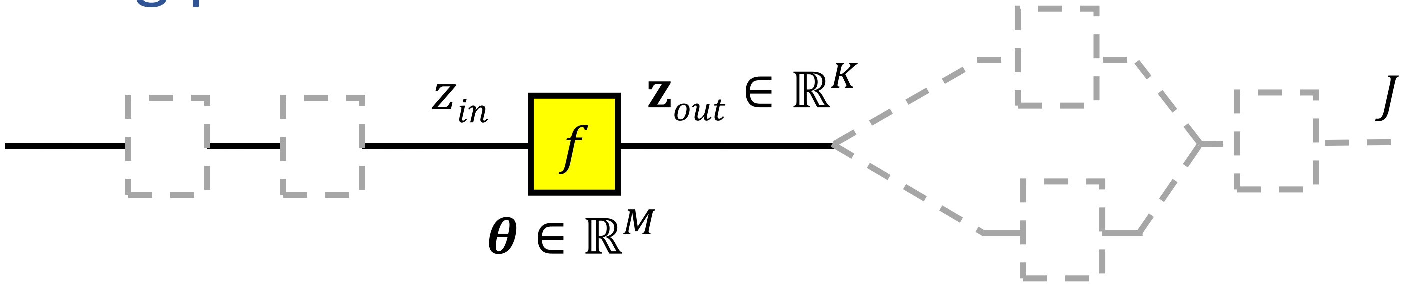


Generic Layer Implementation (vector output version)

Compute derivatives per layer, utilizing previous derivatives

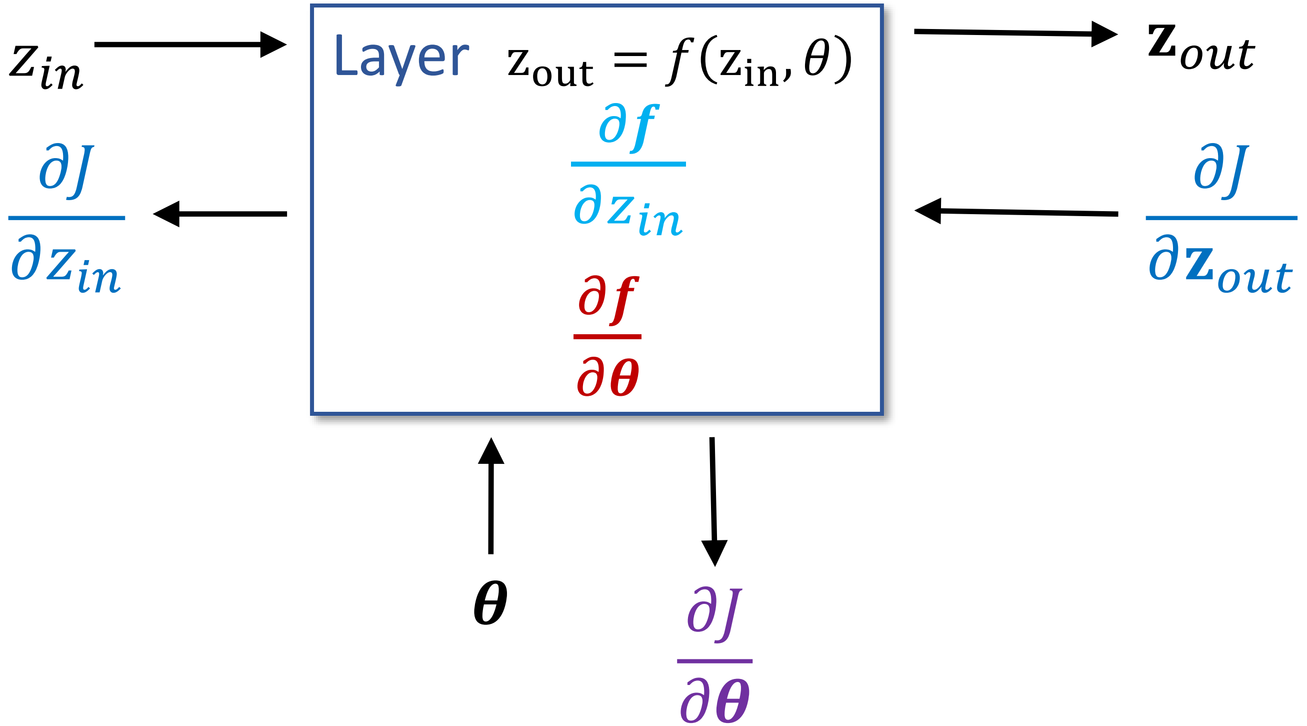
Objective: $J(\theta)$

Arbitrary layer: $\mathrm{z}_{\mathrm{out}} = f(\mathrm{z}_{\mathrm{in}}, \theta)$



Need:

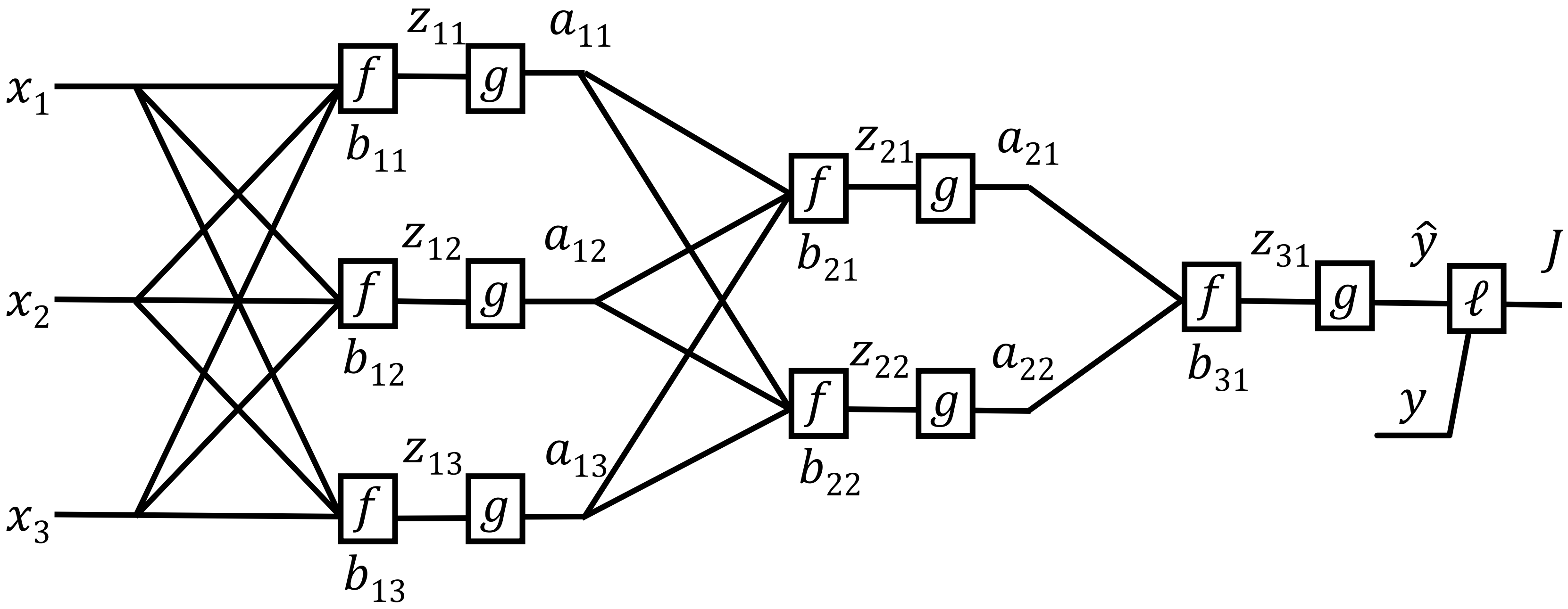
- $\dfrac{\partial J}{\partial z_{in}} = \sum_k \dfrac{\partial J}{\partial z_{out,k}} \dfrac{\partial z_{out,k}}{\partial z_{in}}$
- $\dfrac{\partial J}{\partial \theta_j} = \sum_k \dfrac{\partial J}{\partial z_{out,k}} \dfrac{\partial f}{\partial \theta_j}$



Optimization

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(b_{3,1} + \sum_k w_{3,1,k}\ g\left(b_{2,k} + \sum_i w_{2,k,i}\ g\left(b_{1,i} + \sum_j w_{1,i,j}\ x_j\right)\right)\right)$$

Where do we need chain rule?

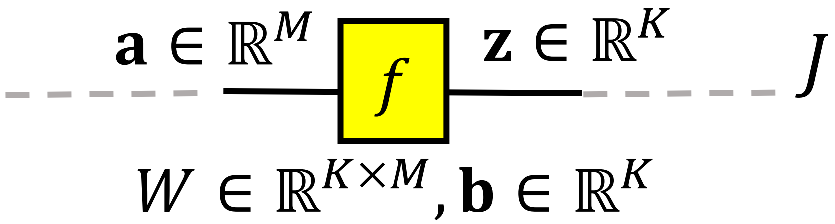


$$\frac{\partial J}{\partial b_{3,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial b_{3,1}}$$

$$\frac{\partial J}{\partial b_{2,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial a_{2,i}} \frac{\partial a_{2,i}}{\partial z_{2,i}} \frac{\partial z_{2,i}}{\partial b_{2,i}}$$

$$\frac{\partial J}{\partial b_{1,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial \mathbf{a}_2} \frac{\partial \mathbf{a}_2}{\partial \mathbf{z}_2} \frac{\partial \mathbf{z}_2}{\partial a_{1,i}} \frac{\partial a_{1,i}}{\partial z_{1,i}} \frac{\partial z_{1,i}}{\partial b_{1,i}}$$

Linear Layer Implementation



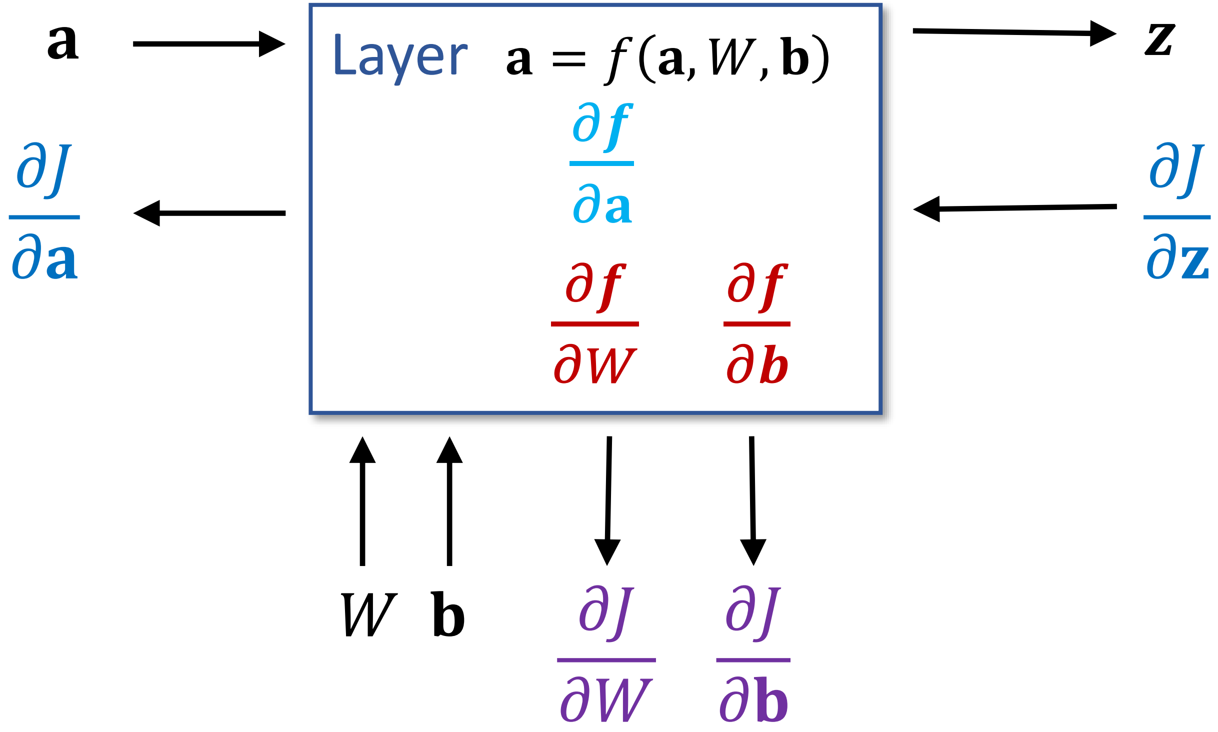
Compute derivatives per layer

Objective: $J(\theta)$

Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$

Scalar version (no formatting)

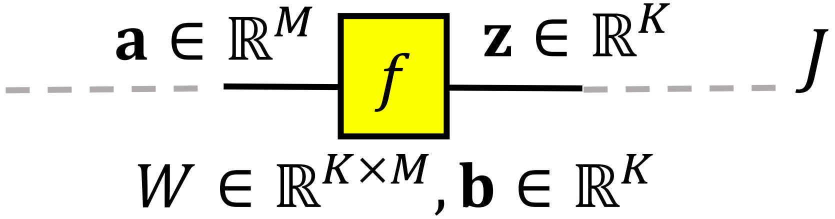
- $\frac{\partial J}{\partial a_j} =$
- $\frac{\partial J}{\partial W_{i,j}} =$
- $\frac{\partial J}{\partial b_i} =$



$$\mathbf{z} = W\mathbf{a} + \mathbf{b}$$

$$z_k = b_k + \sum_j W_{k,j} a_j$$

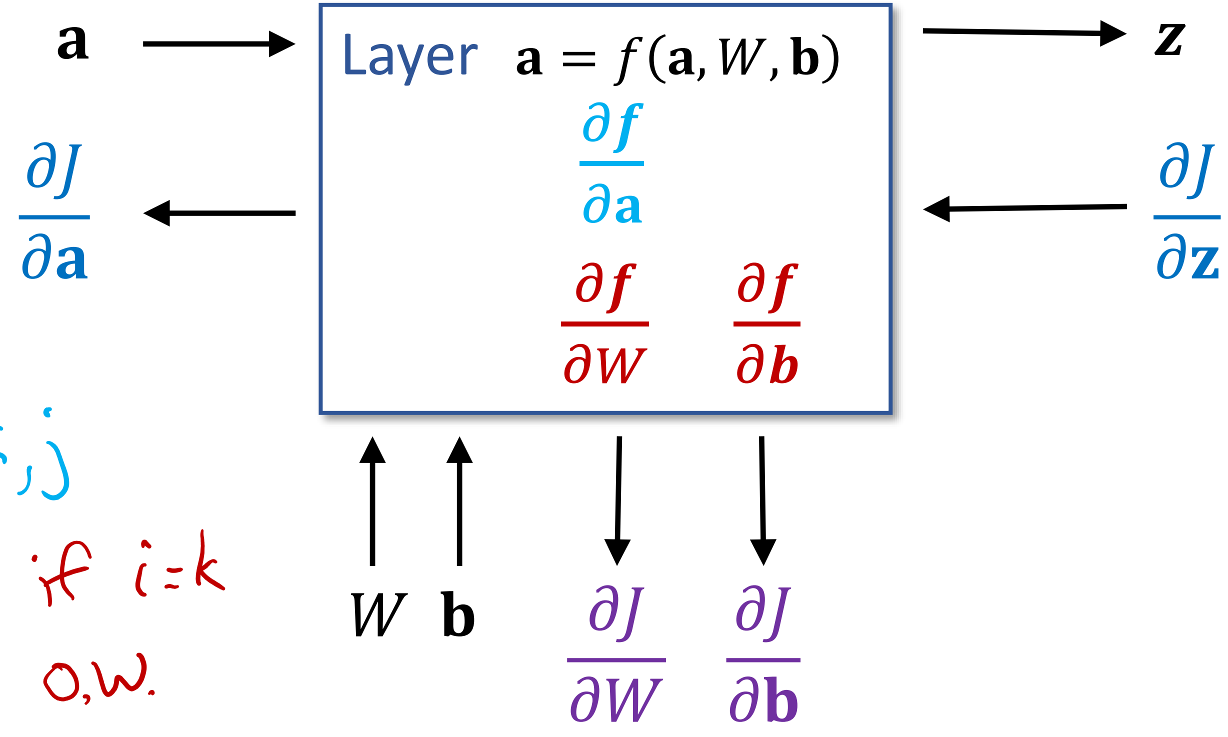
Linear Layer Implementation



Compute derivatives per layer

Objective: $J(\theta)$

Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$



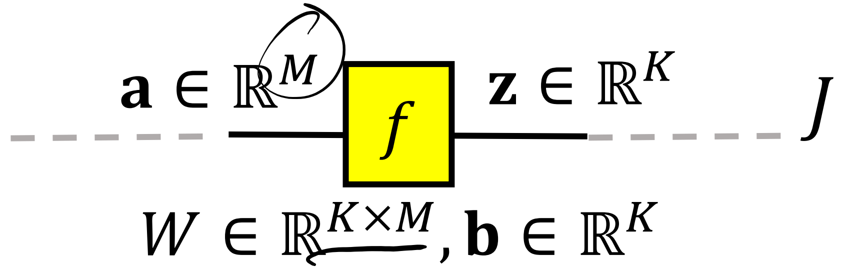
Scalar version (no formatting)

- $\frac{\partial J}{\partial a_j} = \sum_k \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial a_j} = \sum_k \frac{\partial J}{\partial z_k} W_{k,j}$
- $\frac{\partial J}{\partial W_{i,j}} = \sum_k \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial W_{i,j}} = \sum_k \frac{\partial J}{\partial z_k} \begin{cases} a_j & \text{if } i=k \\ 0 & \text{o.w.} \end{cases}$
- $\frac{\partial J}{\partial b_i} = \sum_k \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial b_i} = \sum_k \frac{\partial J}{\partial z_k} \begin{cases} 1 & \text{if } i=k \\ 0 & \text{o.w} \end{cases}$

$\mathbf{z} = W\mathbf{a} + \mathbf{b}$

$\longrightarrow z_k = b_k + \sum_j W_{k,j} a_j$ ✓

Linear Layer Implementation

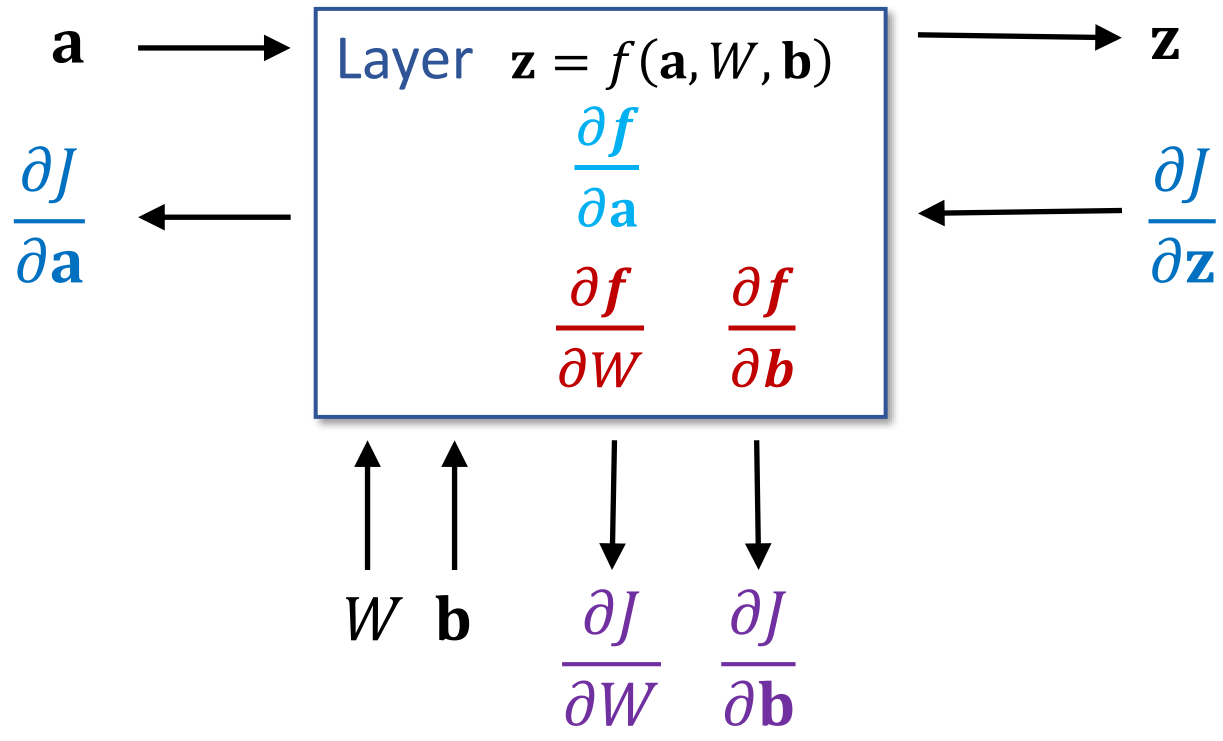


Compute derivatives per layer

Objective: $J(\theta)$

Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$

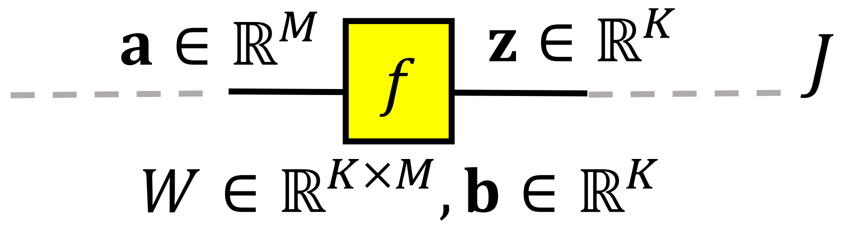
(Numerator format)



$[1 \times M]$ $\qquad\qquad\qquad\qquad$ $[1 \times K]$ $[K \times M]$

- $\frac{\partial J}{\partial \mathbf{a}} = \frac{\partial J}{\partial \vec{z}} \frac{\partial \vec{z}}{\partial \vec{a}} = \frac{\partial J}{\partial \vec{z}} W$
- $\frac{\partial J}{\partial W}$
- $\frac{\partial J}{\partial \mathbf{b}}$

Linear Layer Implementation

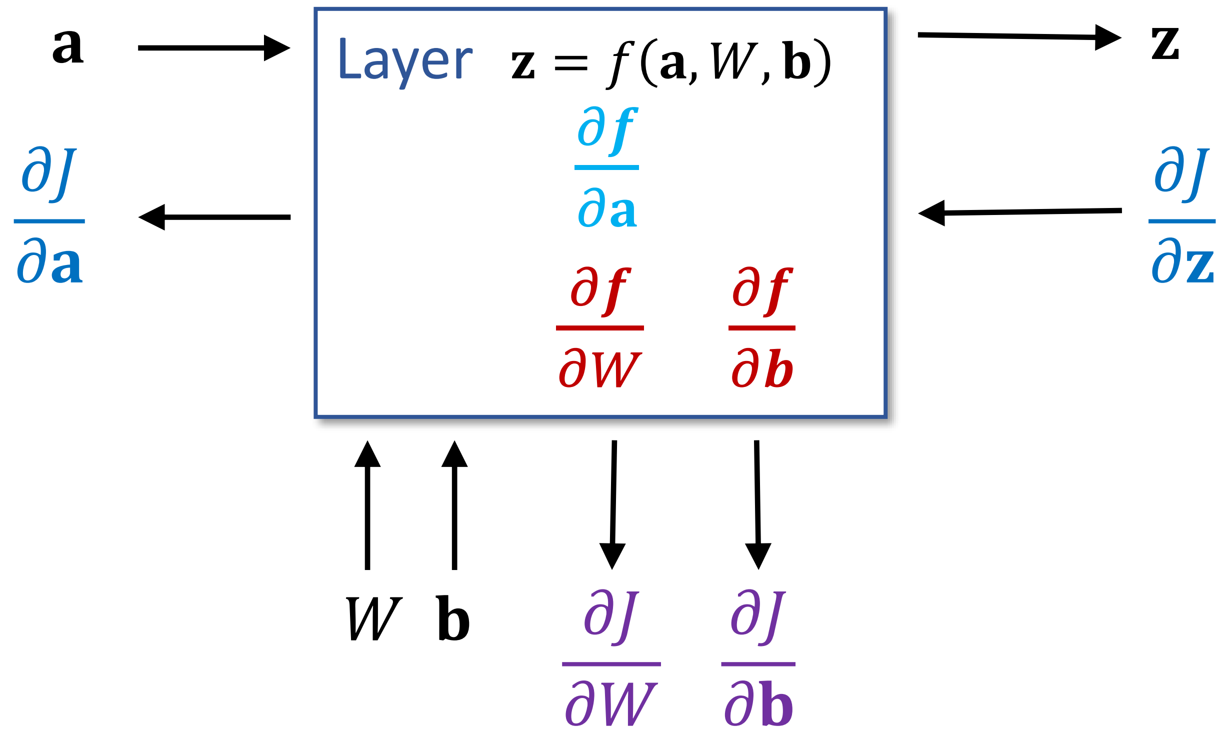


Compute derivatives per layer

Objective: $J(\theta)$

Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$

(Numerator format)



- $\underset{[1 \times M]}{\frac{\partial J}{\partial \mathbf{a}}} = \underset{[1 \times K][K \times M]}{\frac{\partial J}{\partial \mathbf{z}} \frac{\partial \boldsymbol{f}}{\partial \mathbf{a}}} = \underset{[1 \times K][K \times M]}{\frac{\partial J}{\partial \mathbf{z}} W}$

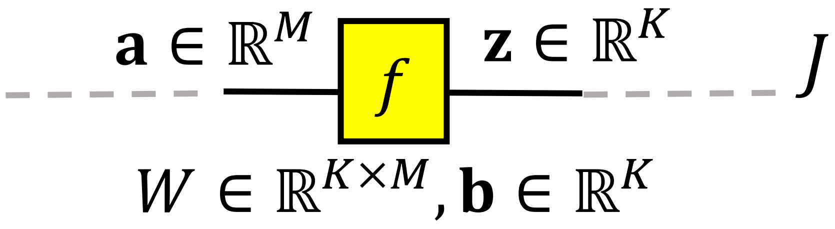
Fixed dimensions to be outer product

- $\underset{[M \times K]}{\frac{\partial J}{\partial W}} = \underset{[1 \times K][\,? \times ?\,]}{\frac{\partial J}{\partial \mathbf{z}} \frac{\partial \boldsymbol{f}}{\partial W}} = \underset{[1 \times K][M \times 1]}{\frac{\partial J}{\partial \mathbf{z}} \mathbf{a}} = \underset{[M \times 1][1 \times K]}{\mathbf{a} \frac{\partial J}{\partial \mathbf{z}}}$

  Broken!

- $\underset{[1 \times M]}{\frac{\partial J}{\partial \mathbf{b}}} = \underset{[1 \times K][K \times K]}{\frac{\partial J}{\partial \mathbf{z}} \frac{\partial \boldsymbol{f}}{\partial \mathbf{b}}} = \underset{[1 \times K][K \times K]}{\frac{\partial J}{\partial \mathbf{z}} I_K}$

Linear Layer Implementation



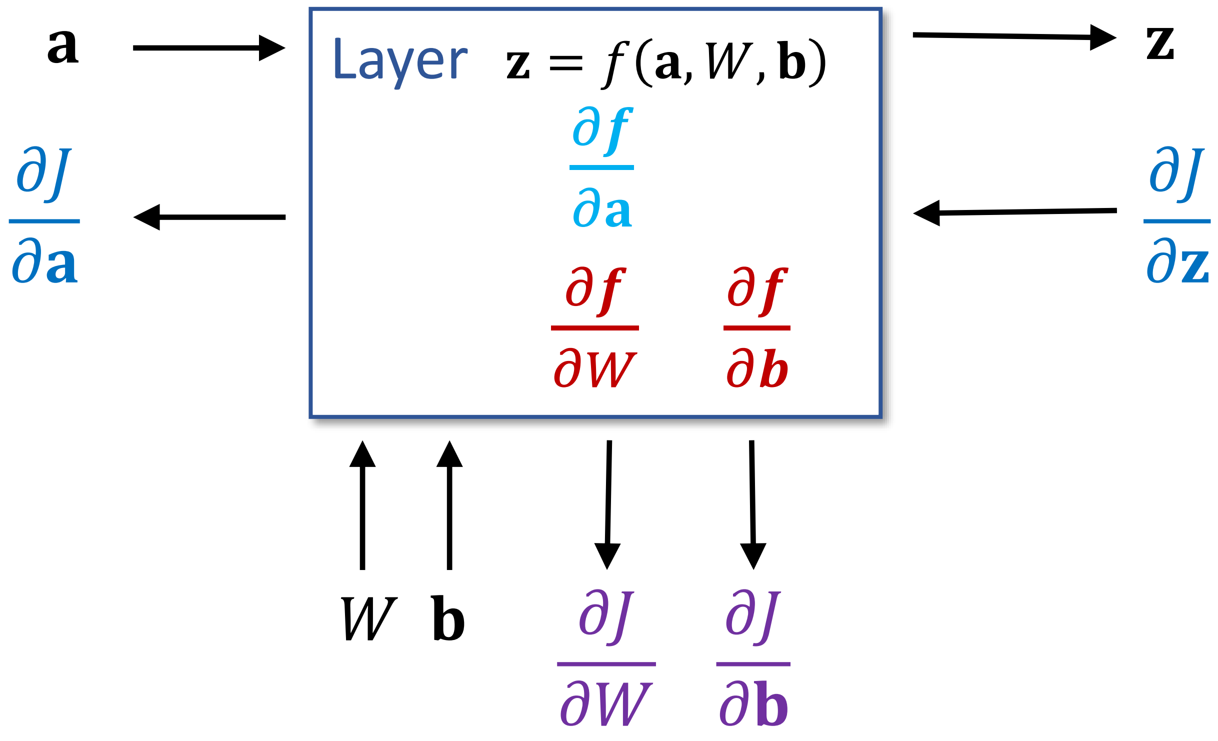
Compute derivatives per layer

Objective: $J(\theta)$

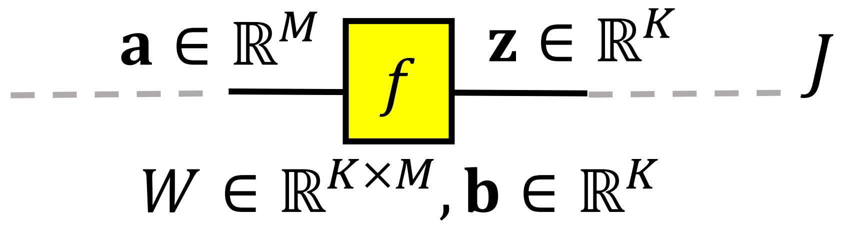
Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$

(Denominator format)

- $\frac{\partial J}{\partial \mathbf{a}}$
- $\frac{\partial J}{\partial W}$
- $\frac{\partial J}{\partial \mathbf{b}}$



Linear Layer Implementation

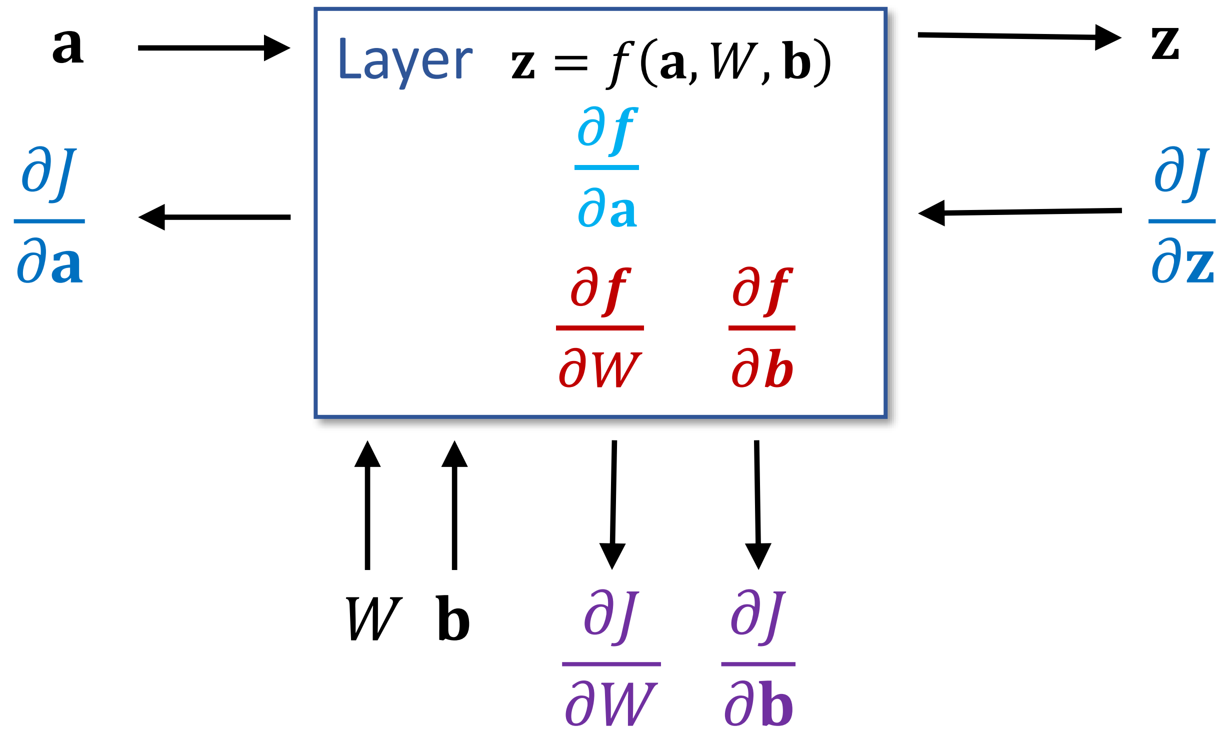


Compute derivatives per layer

Objective: $J(\theta)$

Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$

(Denominator format)



- $$\underset{[M \times 1]}{\frac{\partial J}{\partial \mathbf{a}}} = \underset{[M \times K][K \times 1]}{\frac{\partial \boldsymbol{f}}{\partial \mathbf{a}} \frac{\partial J}{\partial \mathbf{z}}} = \underset{[M \times K][K \times 1]}{W^\top \frac{\partial J}{\partial \mathbf{z}}}$$

Broken!

Fixed dimensions to be outer product

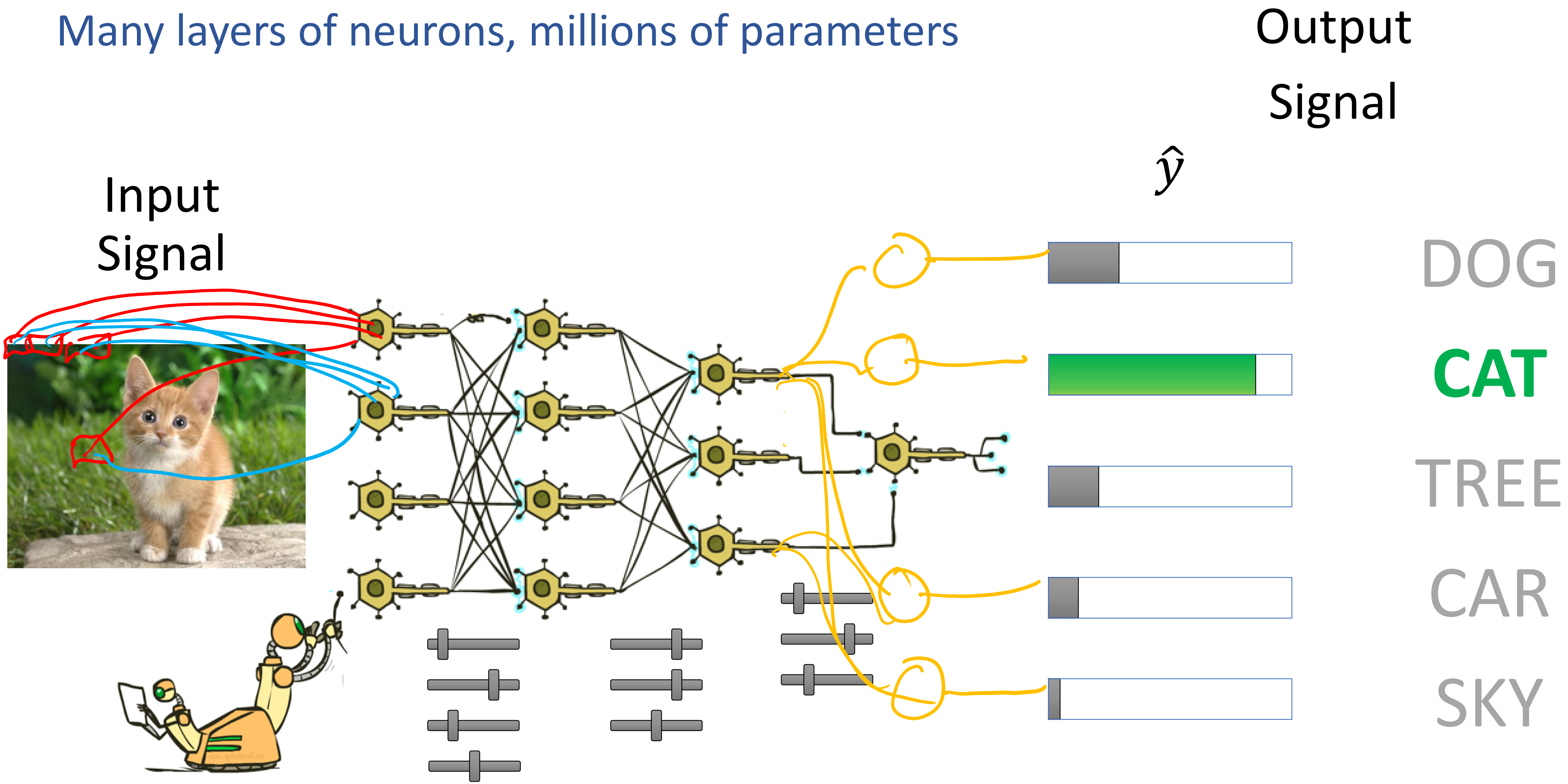
- $$\underset{[K \times M]}{\frac{\partial J}{\partial W}} = \underset{[\,? \times ?\,][K \times 1]}{\frac{\partial \boldsymbol{f}}{\partial W} \frac{\partial J}{\partial \mathbf{z}}} = \underset{[M \times 1][K \times 1]}{\mathbf{a} \frac{\partial J}{\partial \mathbf{z}}} = \underset{[K \times 1][1 \times M]}{\frac{\partial J}{\partial \mathbf{z}} \mathbf{a}^\top}$$

- $$\underset{[M \times 1]}{\frac{\partial J}{\partial \mathbf{b}}} = \underset{[K \times K][K \times 1]}{\frac{\partial \boldsymbol{f}}{\partial \mathbf{b}} \frac{\partial J}{\partial \mathbf{z}}} = \underset{[K \times K][K \times 1]}{I_K \frac{\partial J}{\partial \mathbf{z}}}$$

Optimization Intuition

Neural Networks

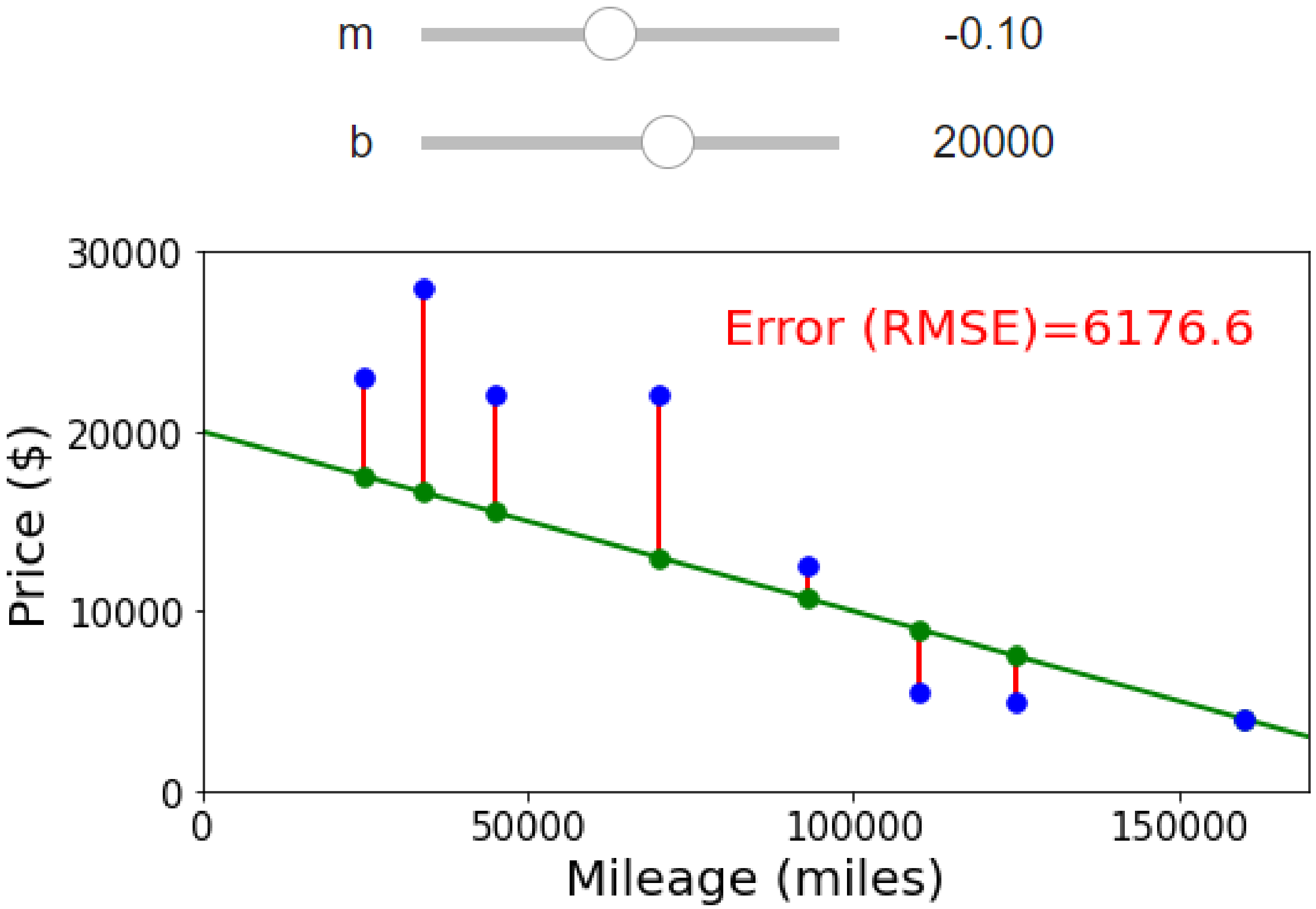
Many layers of neurons, millions of parameters



Neural Networks

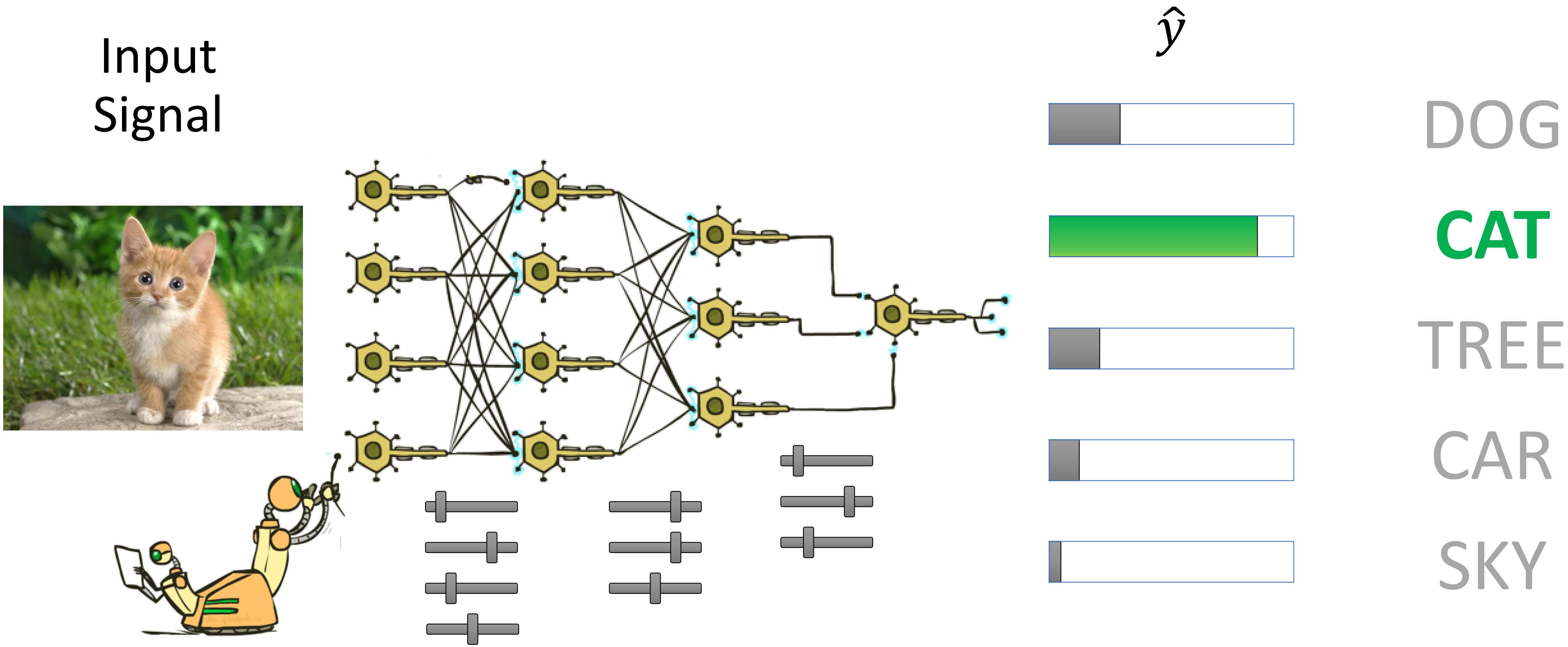
Building on optimization for linear and logistic regression

- Selling my car



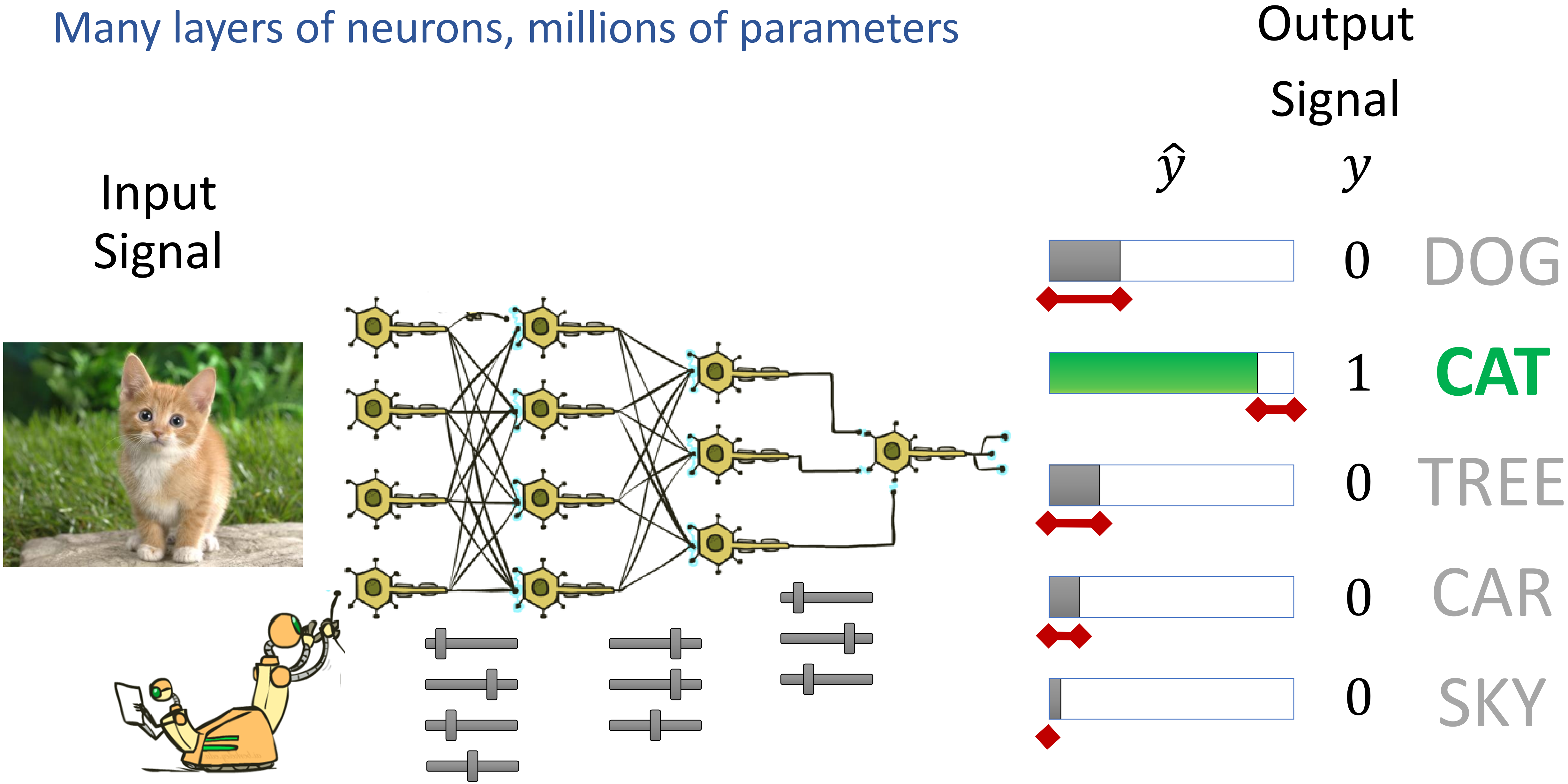
Neural Networks

Many layers of neurons, millions of parameters



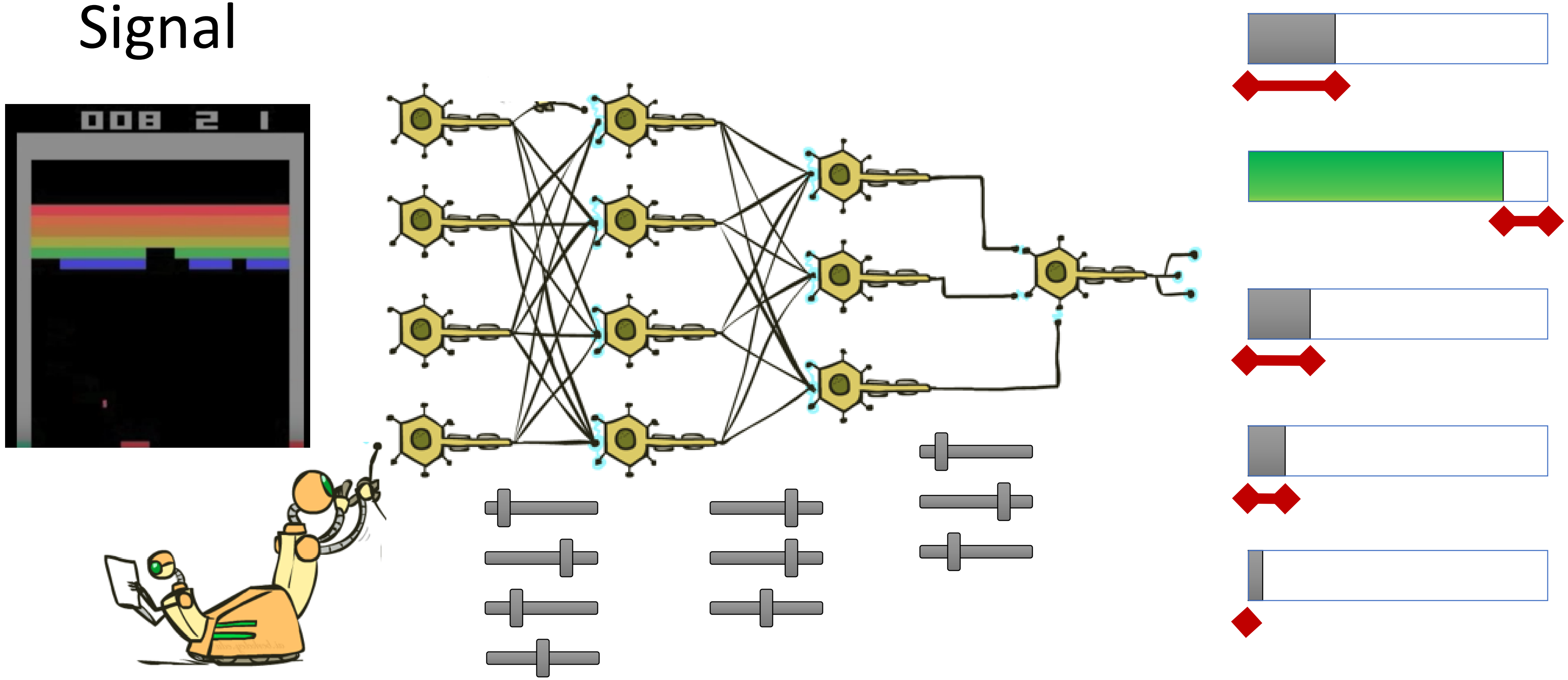
Neural Networks

Many layers of neurons, millions of parameters



Neural Networks

Many layers of neurons, millions of parameters



Outline

Last time: Neural Networks

- Three-neuron network + optimization
- Neural network structure (adding more neurons)
- Forward pass and Backpropagation (scalar version)

Today: Neural Networks

- Calculus: multi-variate chain rule
- Backpropagation (vector version)
- Optimization intuition (sliders)
- Neural Network Properties and Intuition

Neural Network Properties

Neural Networks Properties

Practical considerations

- Large number of neurons
  - Danger for overfitting
- Modeling assumptions vs data assumptions trade-off
- Gradient descent can easily get stuck local optima

lin reg
custom

vs

net

What if there are no non-linear activations?

- A deep neural network with only linear layers can be reduced to an exactly equivalent single linear layer

Universal Approximation Theorem:

- A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy.

Neural Networks Properties

Non-linearity

Fitting complex functions

Fitting any! function (universal approximation theorem)

Overfitting

Neural Network Properties

Non-linearity

Non-linearity

Neural network prediction function, $\hat{y} = h(x)$, is definitely not linear. The non-linear activation functions provide this non-linearity

What if there are no non-linear activations?

- A deep neural network with only linear layers can be reduced to an exactly equivalent single linear layer

Example

What happens with you add to linear functions together?

- Adding two lines? $h(x) = f_1(x, w_1, b_1) + f_2(x, w_2, b_2)$ $= w_1 x + b_1 + w_2 x + b_2$

$= (w_1 + w_2)x + (b_1 + b_2)$

- Adding two planes?

Objective Function is Not Convex

Objective function for...	Convex?	Closed-form solution?
Linear regression	✓	✓
Logistic regression	✓	☹
Neural networks	☹	☹

Optimization

Convex function

If $f(\boldsymbol{x})$ is convex, then:

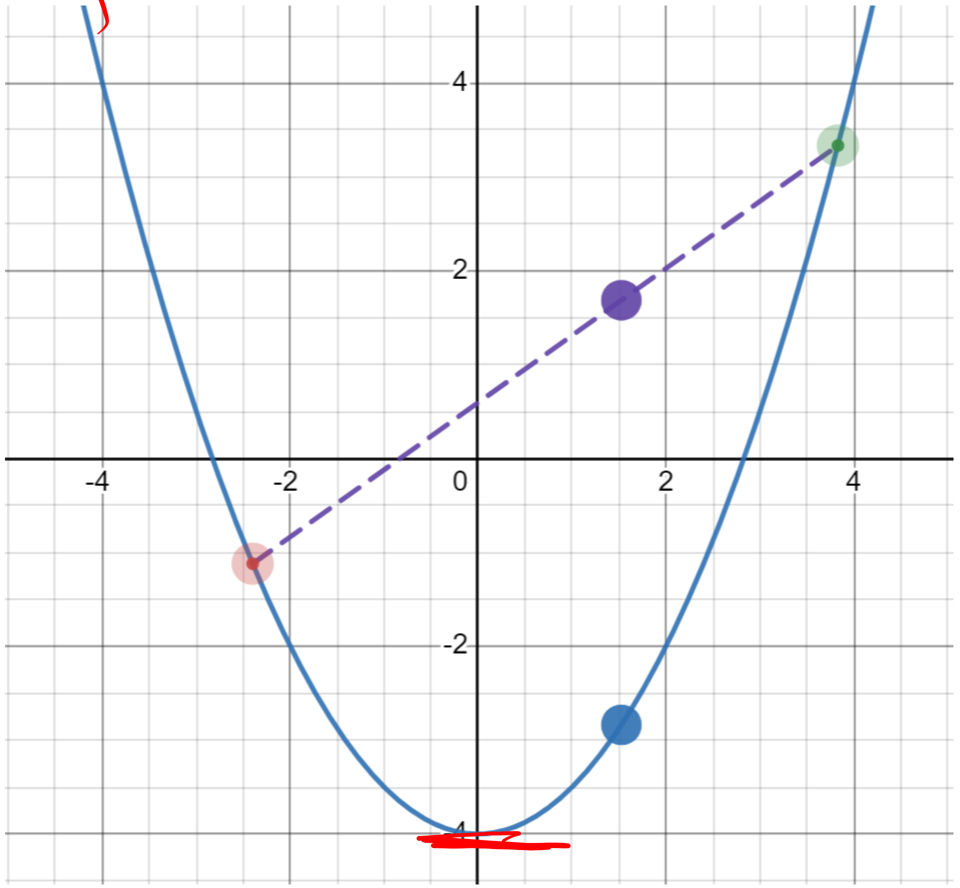
- $f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{z}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{z})$
  $\forall\, 0 \leq \alpha \leq 1$

Convex optimization

If second derivative is ≥ 0 everywhere then function is convex

If $f(\boldsymbol{x})$ is convex, then:

- Every local minimum is also a global minimum ☺



Non-linearity

Prediction function

Neural network prediction function, $\hat{y} = h(x)$, is definitely not linear. The non-linear activation functions provide this non-linearity

Objective function

Neural network prediction function, $\hat{y} = h(x)$, is definitely not linear. The non-linear activation functions provide this non-linearity

Just because it can fit any function will it?

- Objective function is non-convex

  → it will get stuck in local minima

- *Stochastic* gradient decent take pseudorandom steps

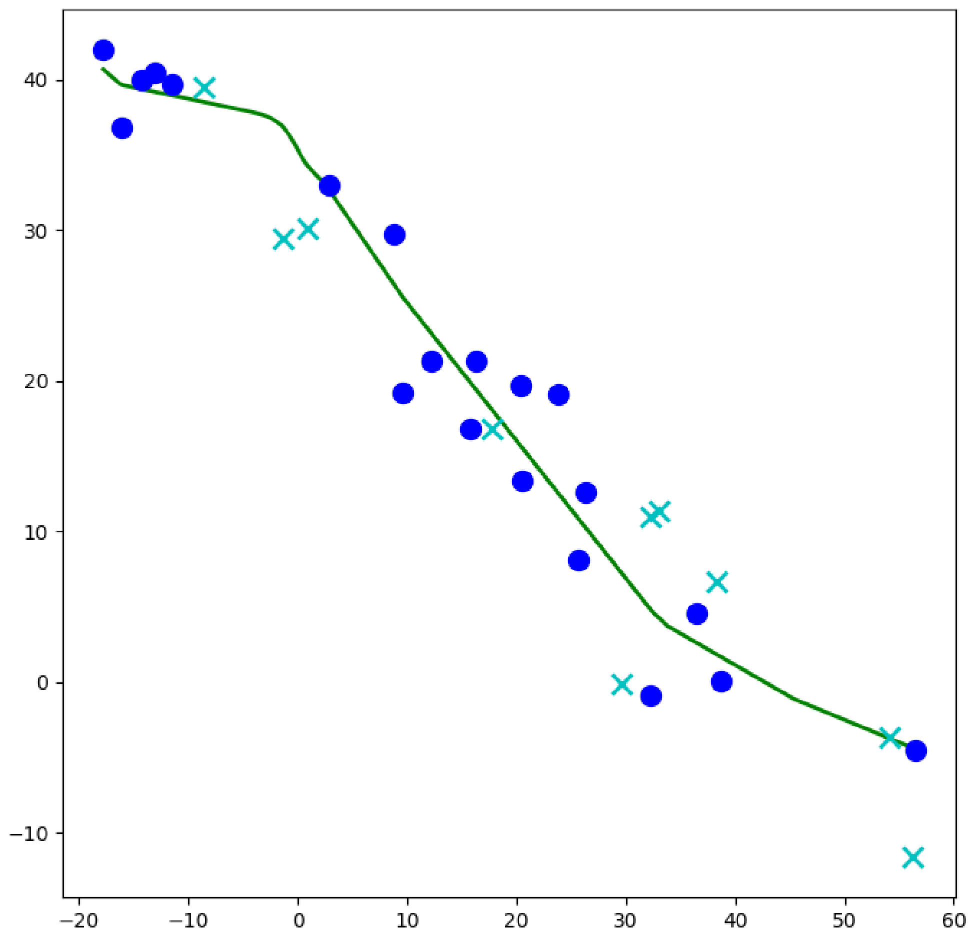
  → Helps pop out of local minima

- Is getting stuck at a local minima necessarily a bad thing??

Neural Network Properties

Fitting complex functions (with 1-D input)

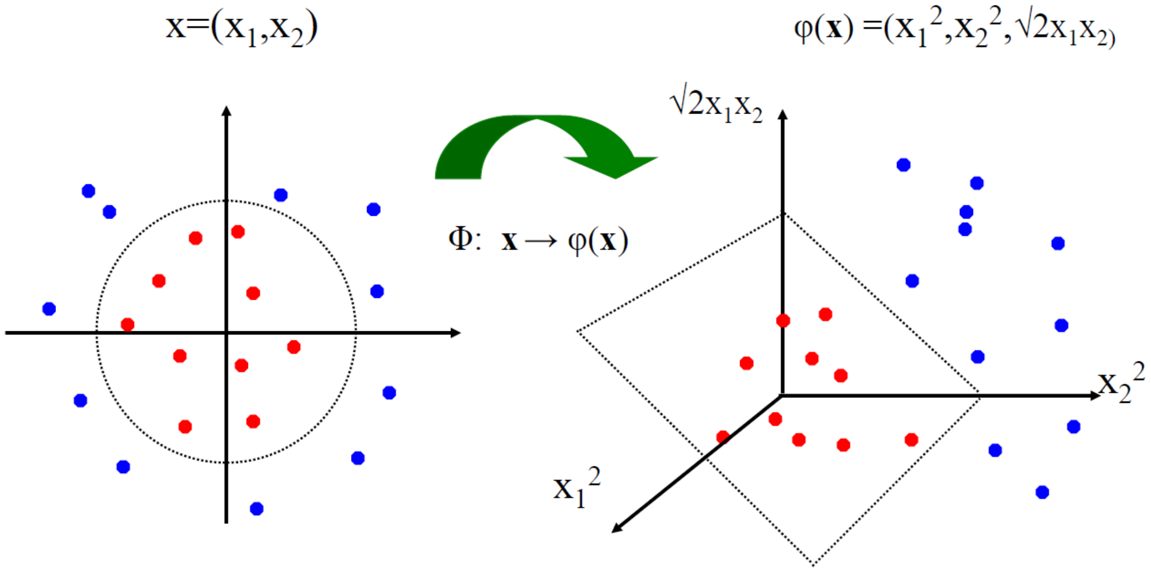
Network to Approximate a 1-D Function



Linear Classifiers for Nonlinear Data

Linear classifiers have linear decision boundaries

Feature mapping can convert nonlinear data to higher dimensions

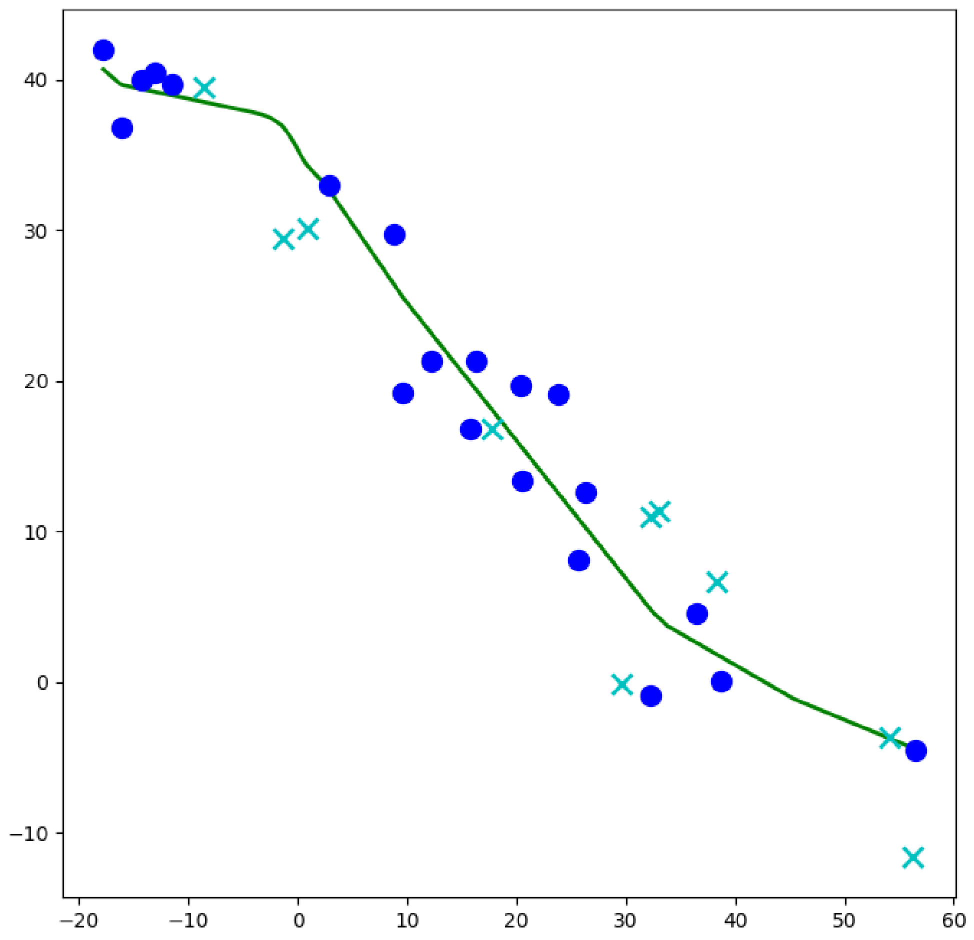


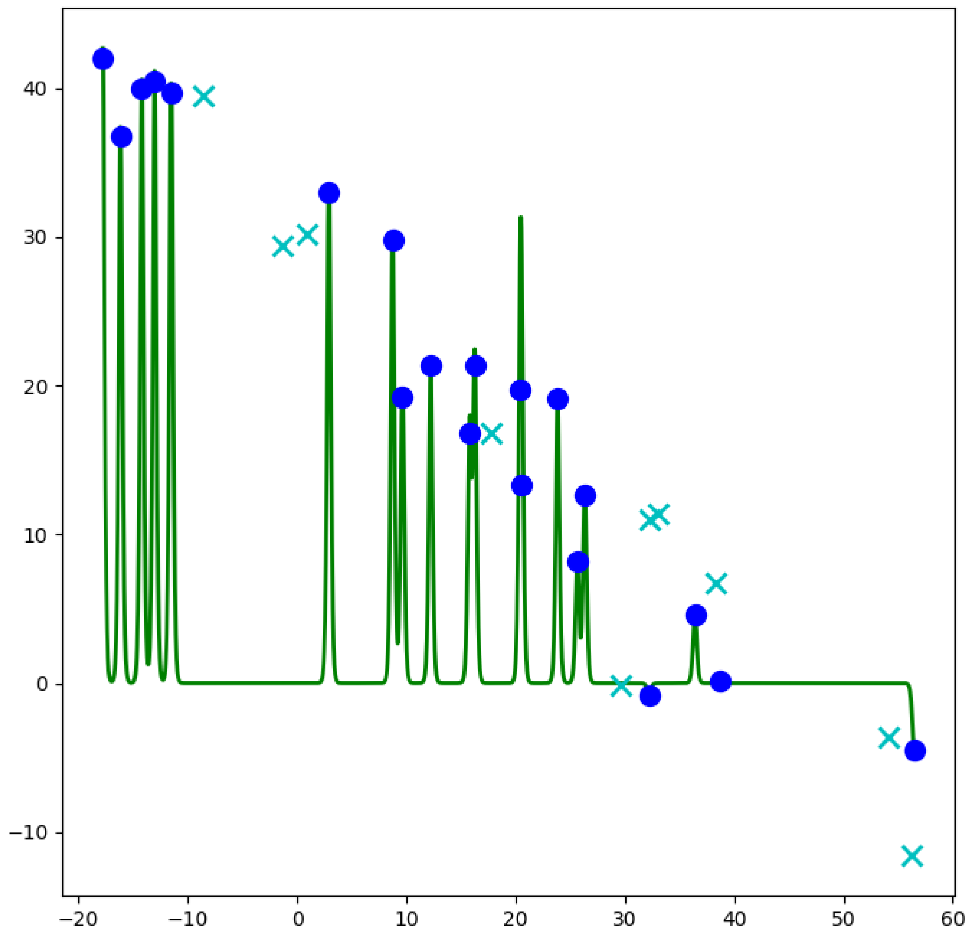
This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Today, instead of choosing a feature mapping function, we'll use neural networks to learn nonlinear decision boundaries.

- We'll quickly shift this to doing nonlinear regression too!

Network to Approximate a 1-D Function

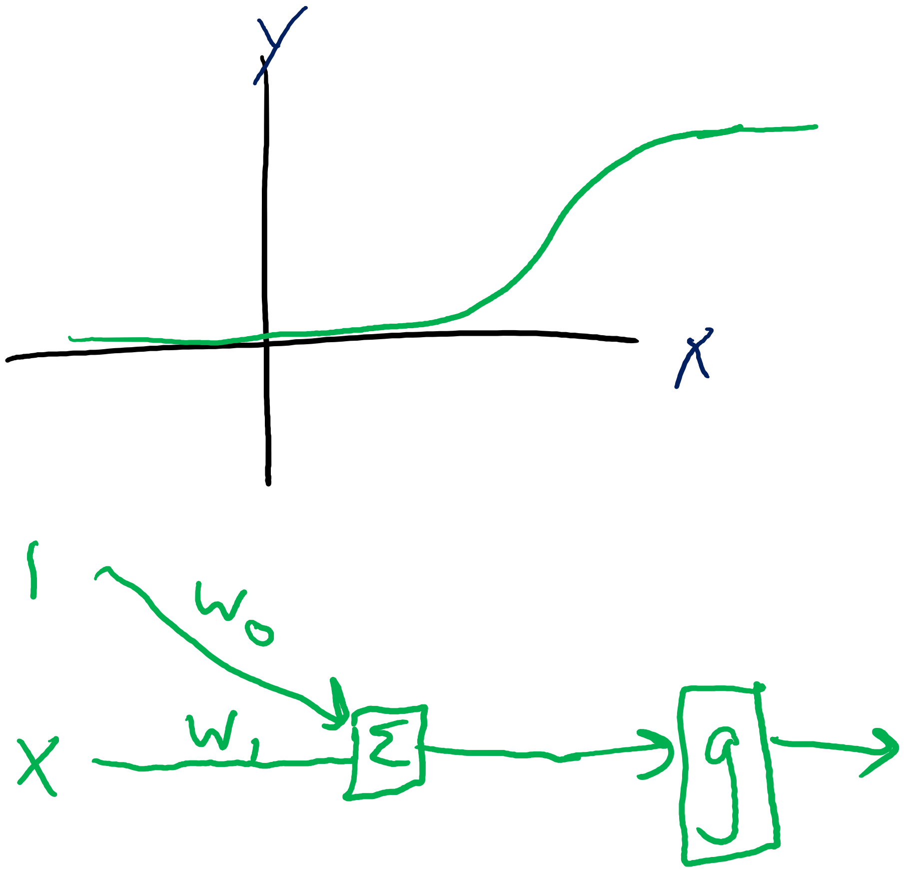


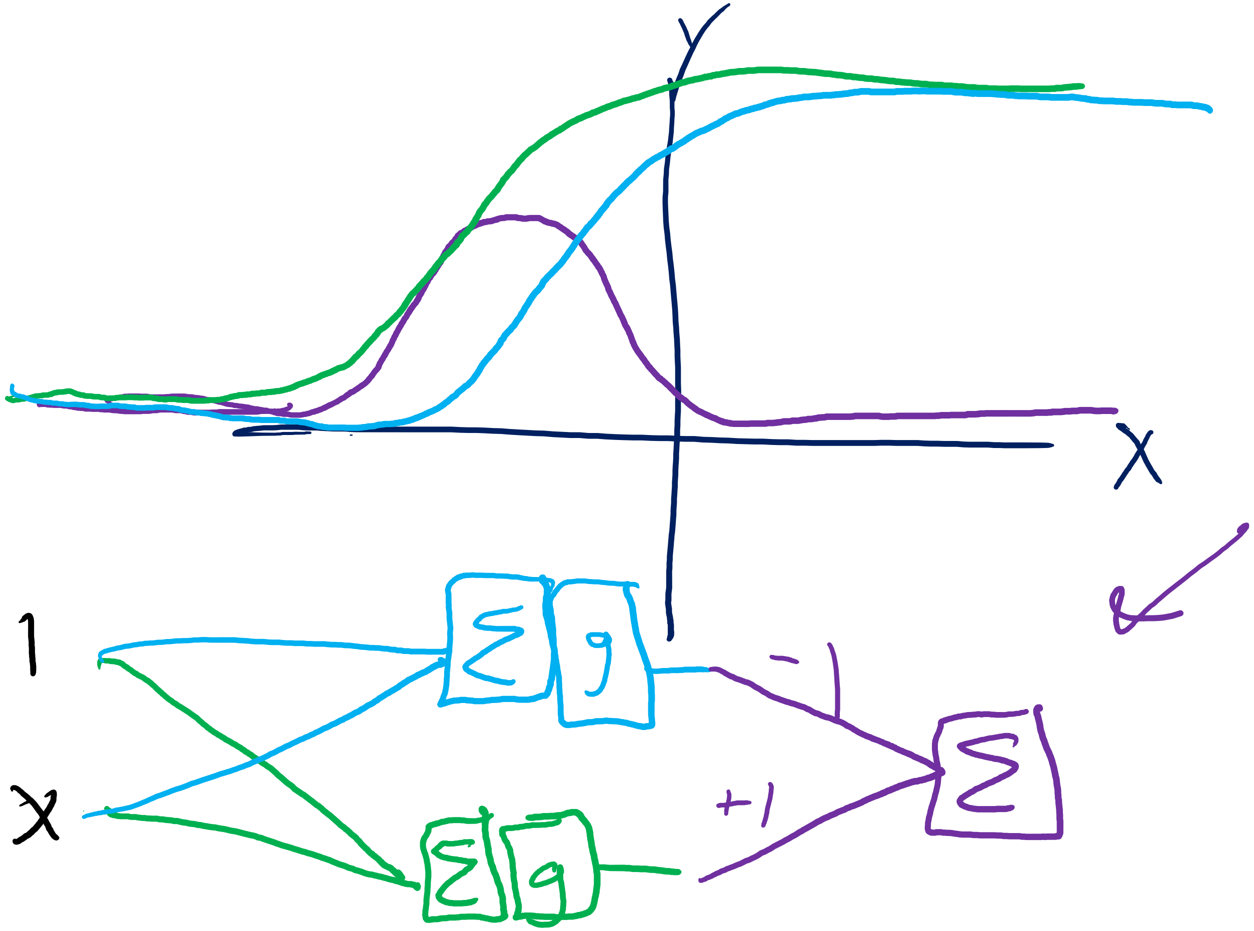


Network to Approximate a 1-D Function

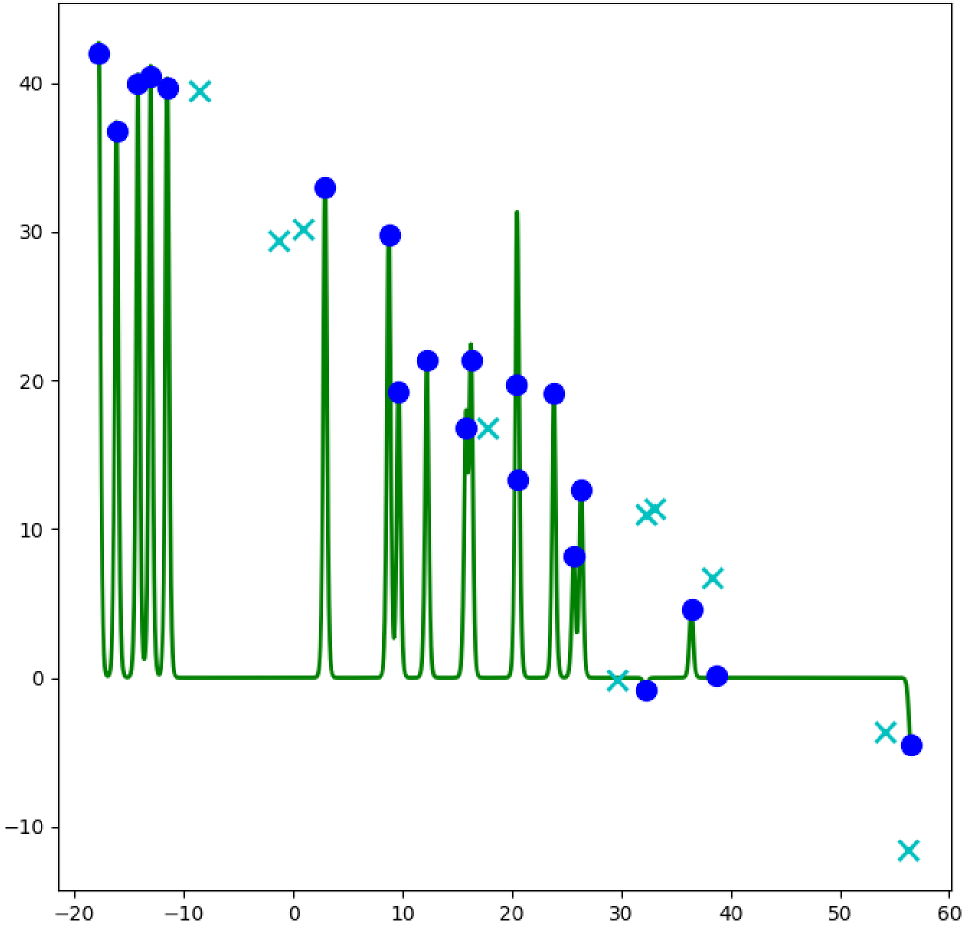
Design a network to approximate this function using:

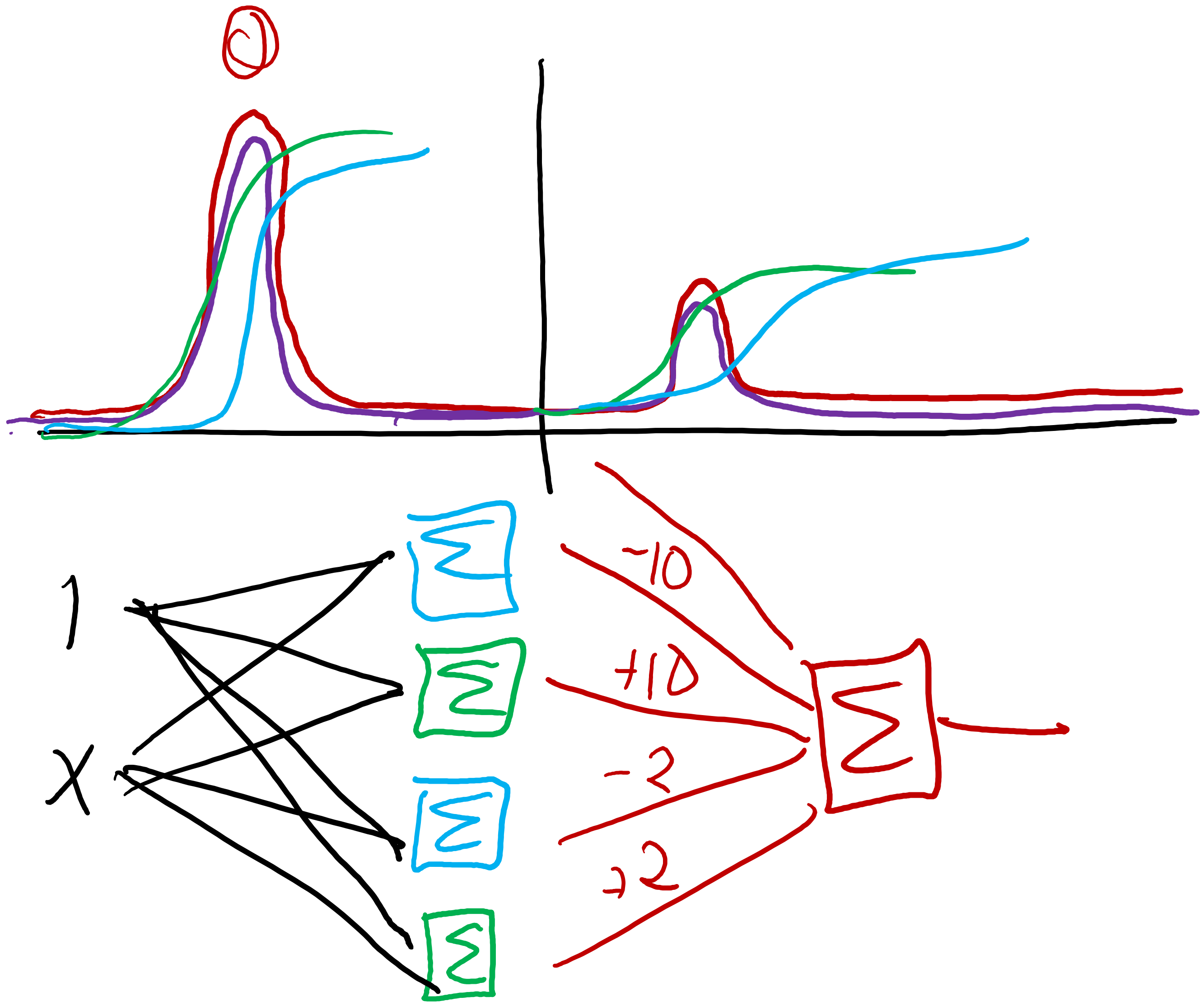
Linear, Sigmoid, Step, or ReLU





Network to Approximate a 1-D Function





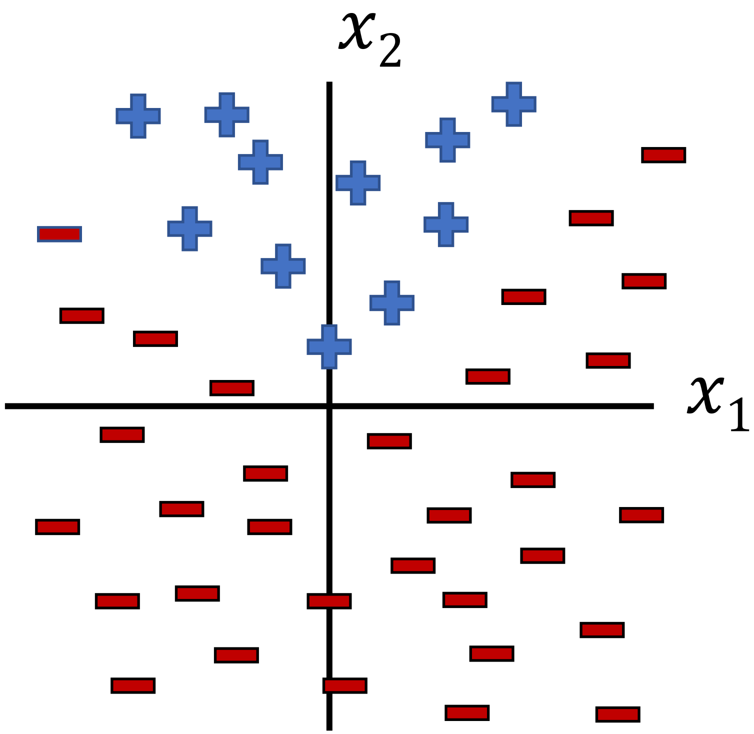
Neural Network Properties

Fitting complex functions (with 2-D input)

Classification Design Challenge

How could you configure three specific perceptrons to classify this data?

$$h_A(\boldsymbol{x}) = sign(\boldsymbol{w}_A^T\boldsymbol{x} + b_A)$$
$$h_B(\boldsymbol{x}) = sign(\boldsymbol{w}_B^T\boldsymbol{x} + b_B)$$
$$h_C(\boldsymbol{x}) = sign(\boldsymbol{w}_C^T\boldsymbol{x} + b_C)$$

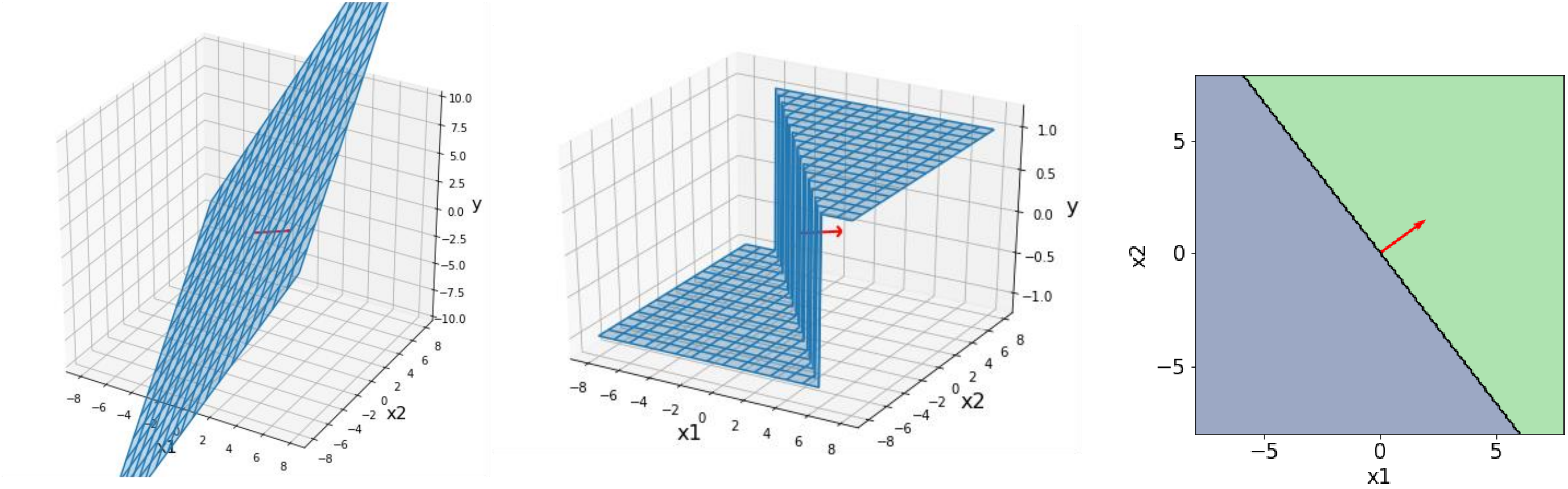


Perceptron

$$sign(\boldsymbol{z}) = \begin{cases} 1, & if\ z \geq 0 \\ -1, & if\ z < 0 \end{cases}$$

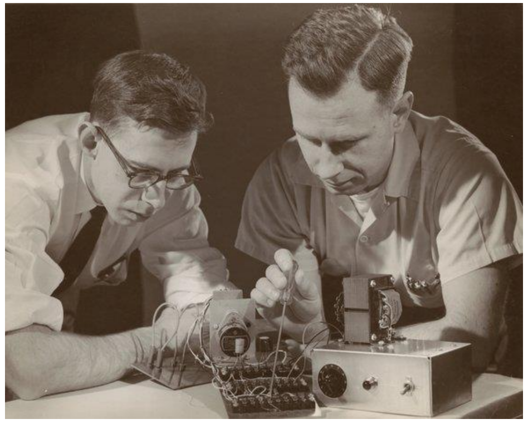
Classification: Hard threshold on linear model

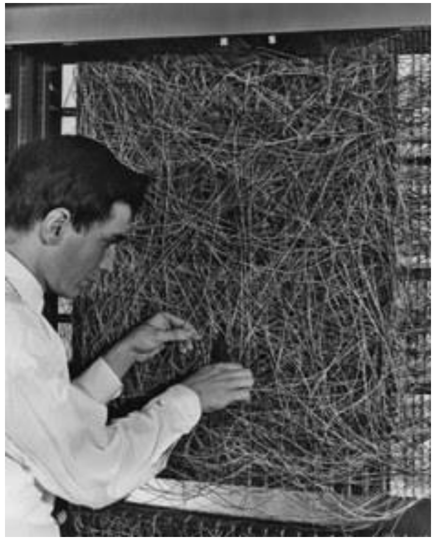
$$h(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x} + b)$$



Perceptron History

Frank Rosenblatt, 1957





The New Yorker, December 6, 1958 P. 44

Talk story about the perceptron, a new electronic brain which hasn't been built, but which has been successfully simulated on the I.B.M. 704. Talk with Dr. Frank Rosenblatt, of the Cornell Aeronautical Laboratory, who is one of the two men who developed the prodigy; the other man is Dr. Marshall C. Yovits, of the Office of Naval Research, in Washington. Dr. Rosenblatt defined the perceptron as the first non-biological object which will achieve an organization o its external environment in a meaningful way. It interacts with its environment, forming concepts that have not been made ready for it by a human agent. If a triangle is held up, the perceptron's eye picks up the image & conveys it along a random succession of lines to the response units, where the image is registered. It can tell the difference betw. a cat and a dog, although it wouldn't be able to tell whether the dog was to theleft or right of the cat. Right now it is of no practical use, Dr. Rosenblatt conceded, but he said that one day it might be useful to send one into outer space to take in impressions for us.

Exercise

$$h(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x} + b)$$

Which of the following perceptron parameters will perfectly classify this data?

$$sign(\boldsymbol{z}) = \begin{cases} 1, & if\ z \geq 0 \\ -1, & if\ z < 0 \end{cases}$$

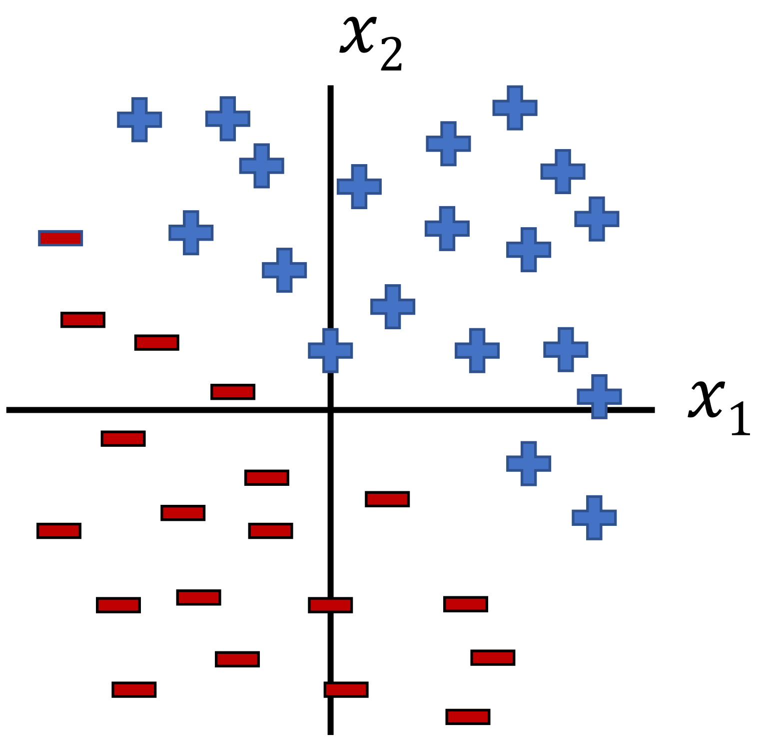
A. $\boldsymbol{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b = 0$

B. $\boldsymbol{w} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, b = 0$

C. $\boldsymbol{w} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, b = 0$

D. $\boldsymbol{w} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, b = 0$

E. None of the above



Poll

$$h(\boldsymbol{x}) = sign(\boldsymbol{w}^T \boldsymbol{x} + b)$$

Which of the following perceptron parameters will perfectly classify this data?

$$sign(\boldsymbol{z}) = \begin{cases} 1, & if\ z \geq 0 \\ -1, & if\ z < 0 \end{cases}$$

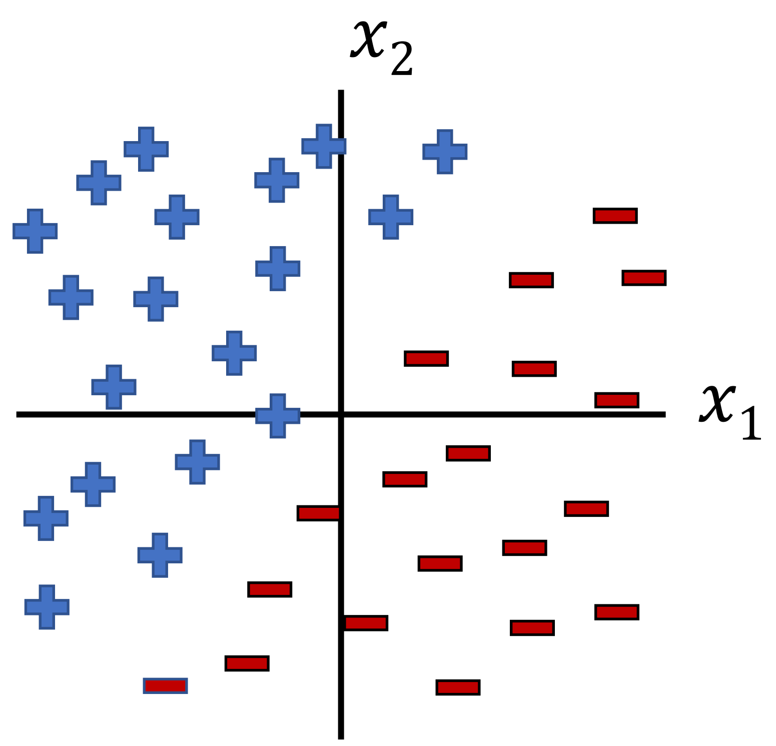
A. $\boldsymbol{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b = 0$

B. $\boldsymbol{w} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, b = 0$

C. $\boldsymbol{w} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, b = 0$

D. $\boldsymbol{w} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, b = 0$

E. None of the above



Poll

$$h_C(\boldsymbol{z}) = sign\left(\boldsymbol{w}_C^T\boldsymbol{z} + b_C\right)$$

Which of the following parameters of $h_C(\boldsymbol{z})$ will perfectly classify this data?

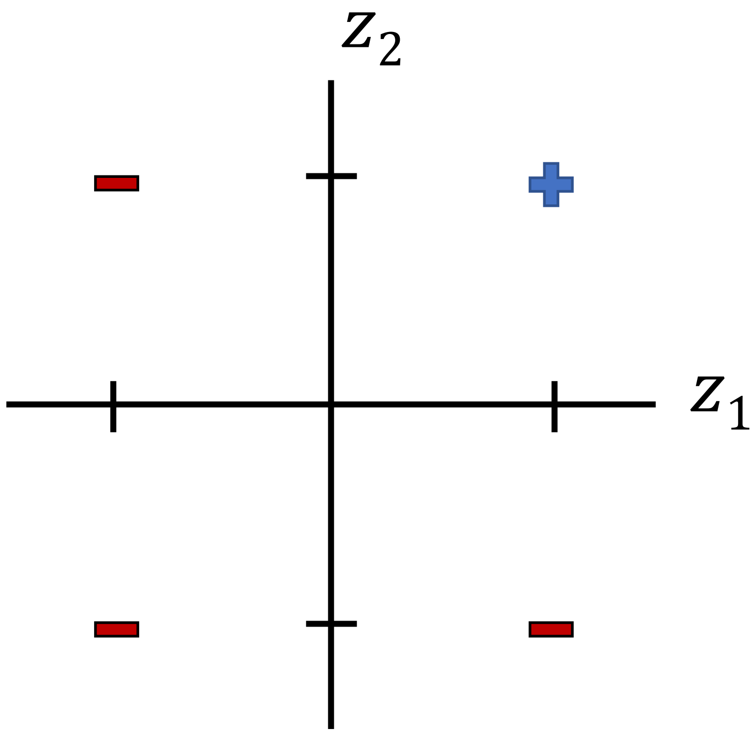
$$sign(\boldsymbol{x}) = \begin{cases} 1, & if\ x \geq 0 \\ -1, & if\ x < 0 \end{cases}$$

A. $\boldsymbol{w}_C = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_C = 0$

B. $\boldsymbol{w}_C = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_C = 1$

C. $\boldsymbol{w}_C = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_C = -1$

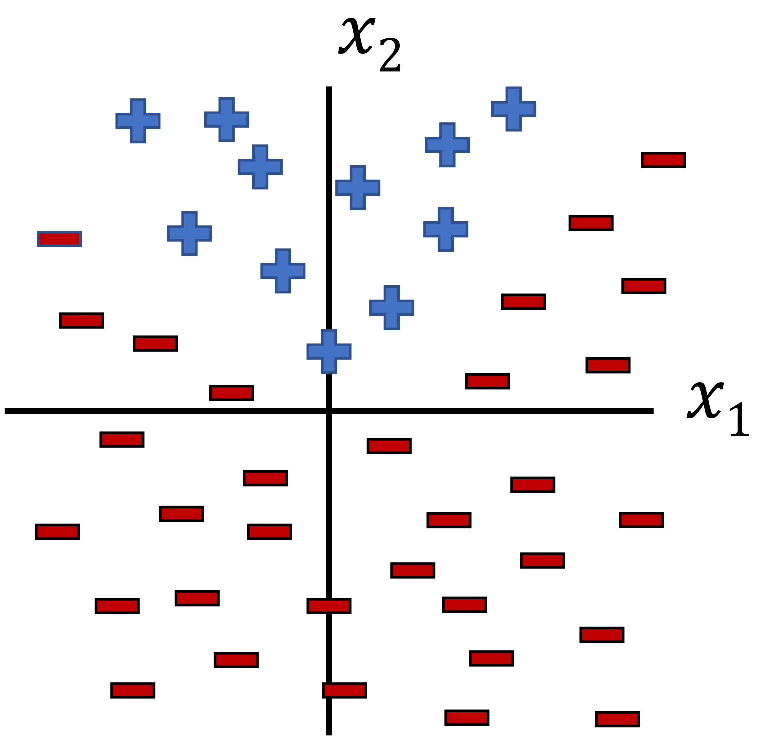
D. None of the above



Classification Design Challenge

$$h_A(\boldsymbol{x}) = sign(\boldsymbol{w}_A^T\boldsymbol{x} + b_A)$$
$$h_B(\boldsymbol{x}) = sign(\boldsymbol{w}_B^T\boldsymbol{x} + b_B)$$
$$h_C(\boldsymbol{x}) = sign(\boldsymbol{w}_C^T\boldsymbol{x} + b_C)$$

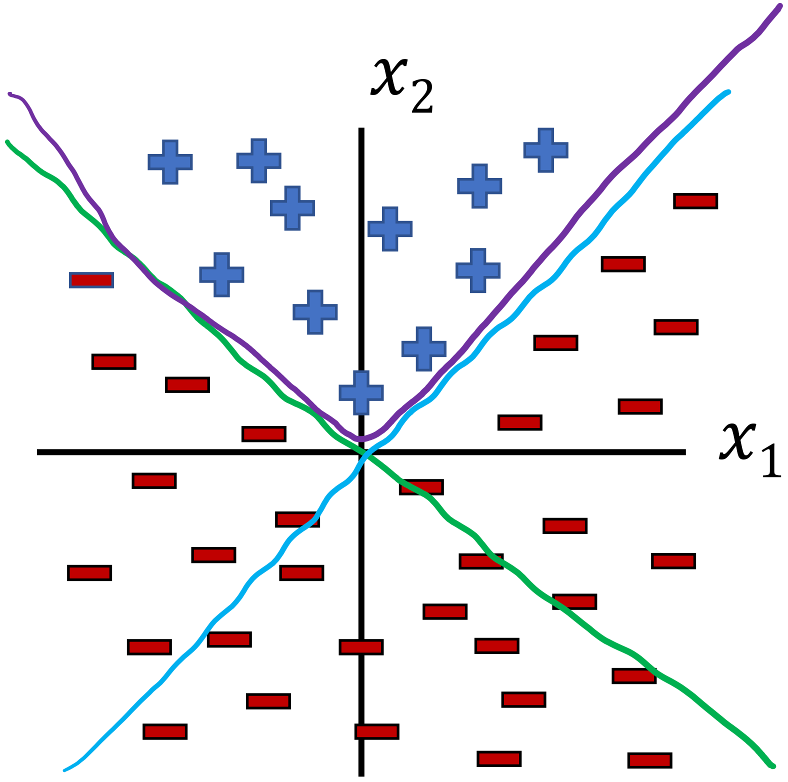
How could you configure three specific perceptrons to classify this data?

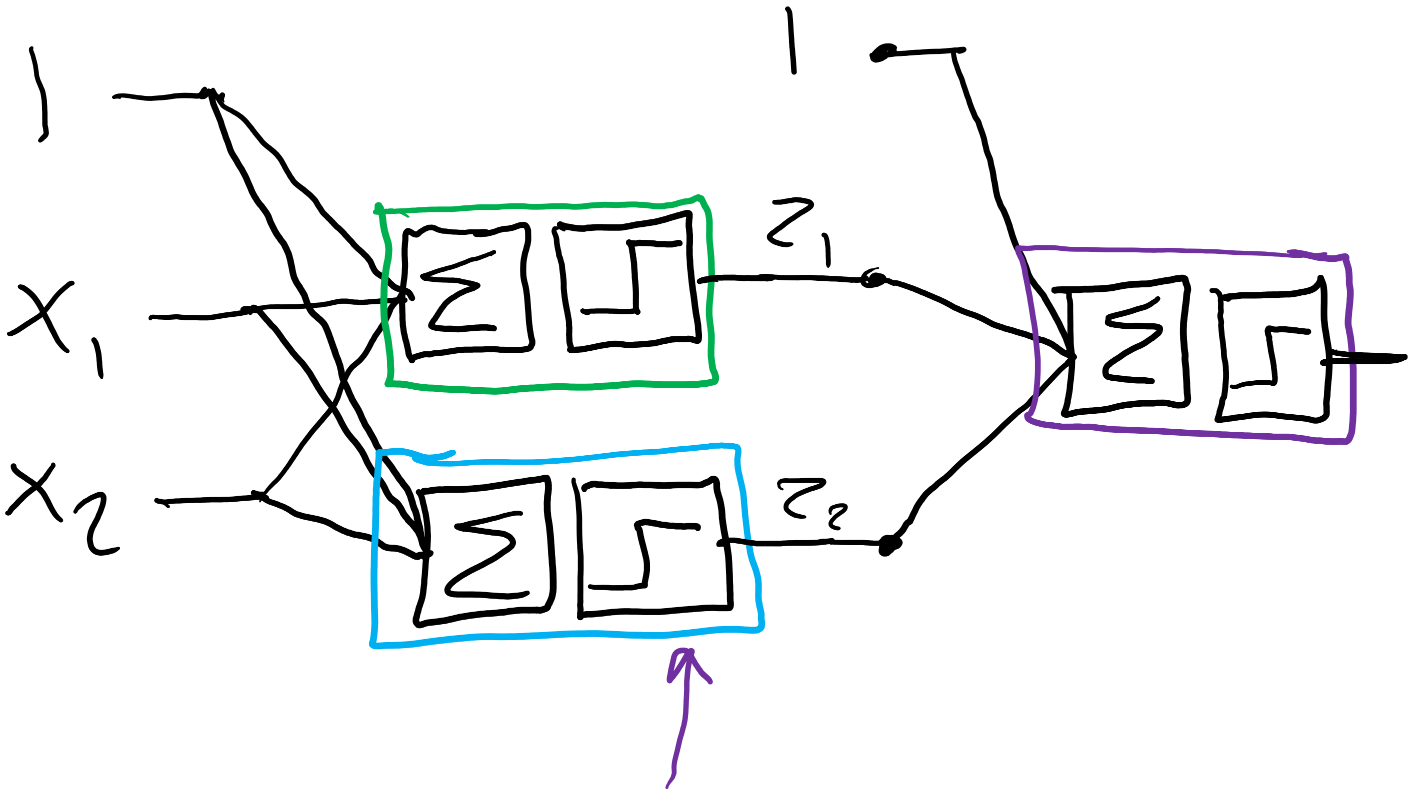


Classification Design Challenge

$$h_A(\boldsymbol{x}) = sign(\boldsymbol{w}_A^T\boldsymbol{x} + b_A)$$
$$h_B(\boldsymbol{x}) = sign(\boldsymbol{w}_B^T\boldsymbol{x} + b_B)$$
$$h_C(\underset{z}{\cancel{\boldsymbol{x}}}) = sign(\boldsymbol{w}_C^T\underset{z}{\cancel{\boldsymbol{x}}} + b_C)$$

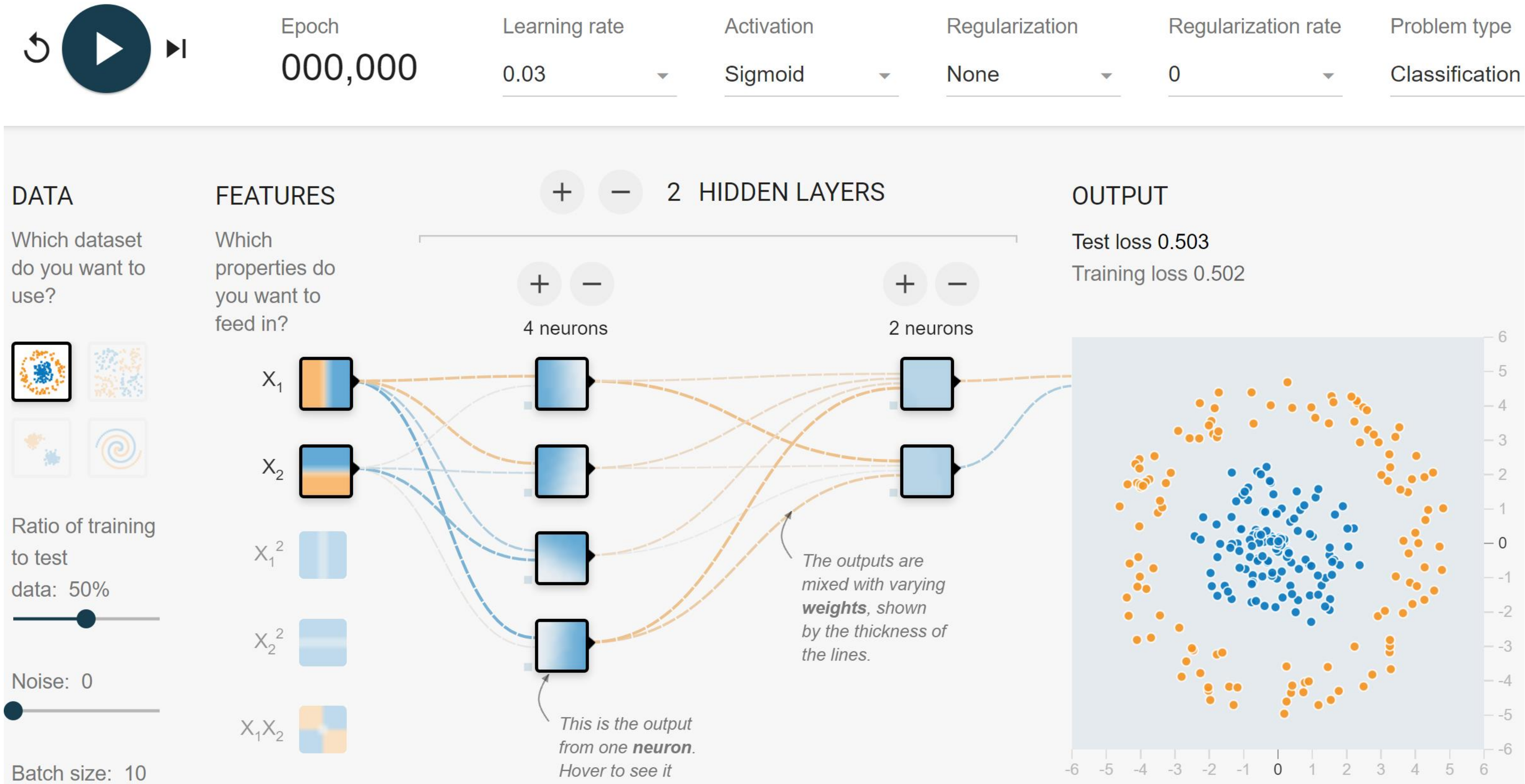
How could you configure three specific perceptrons to classify this data?





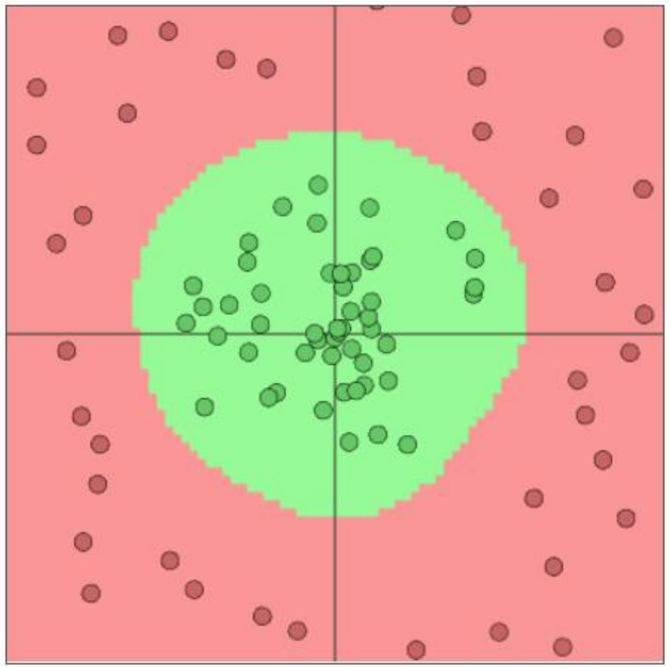
Network to Approximate Binary Classification

https://playground.tensorflow.org/#activation=sigmoid



Network to Approximate Binary Classification

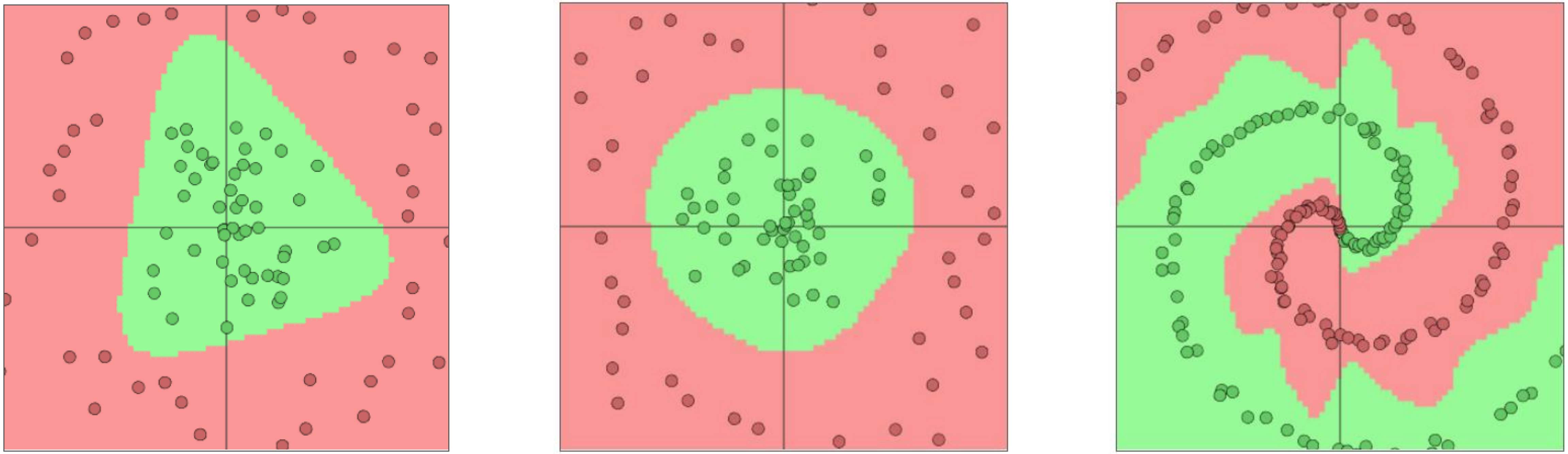
Approximate arbitrary decision boundary



https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Network to Approximate Binary Classification

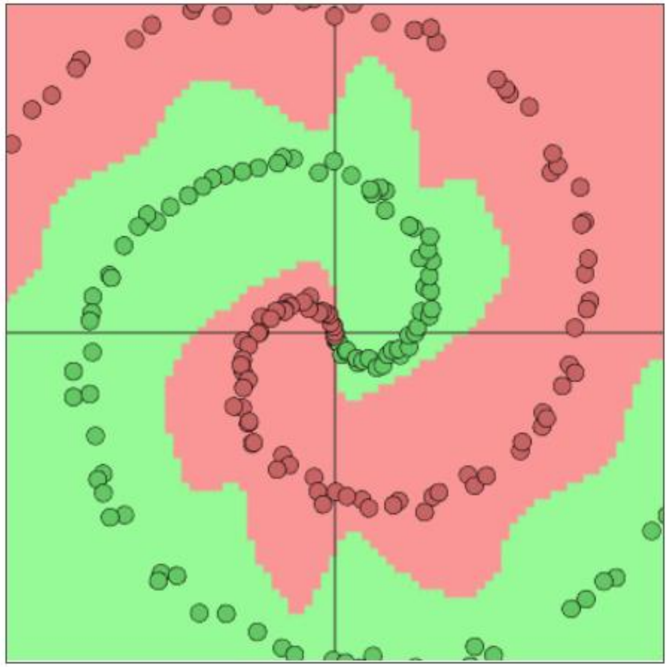
Approximate arbitrary decision boundary



https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Network to Approximate Binary Classification

Approximate arbitrary decision boundary



https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Neural Network Properties

(Over?) Fitting any function

Neural Networks Properties

Universal Approximation Theorem

- A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy

Reminder

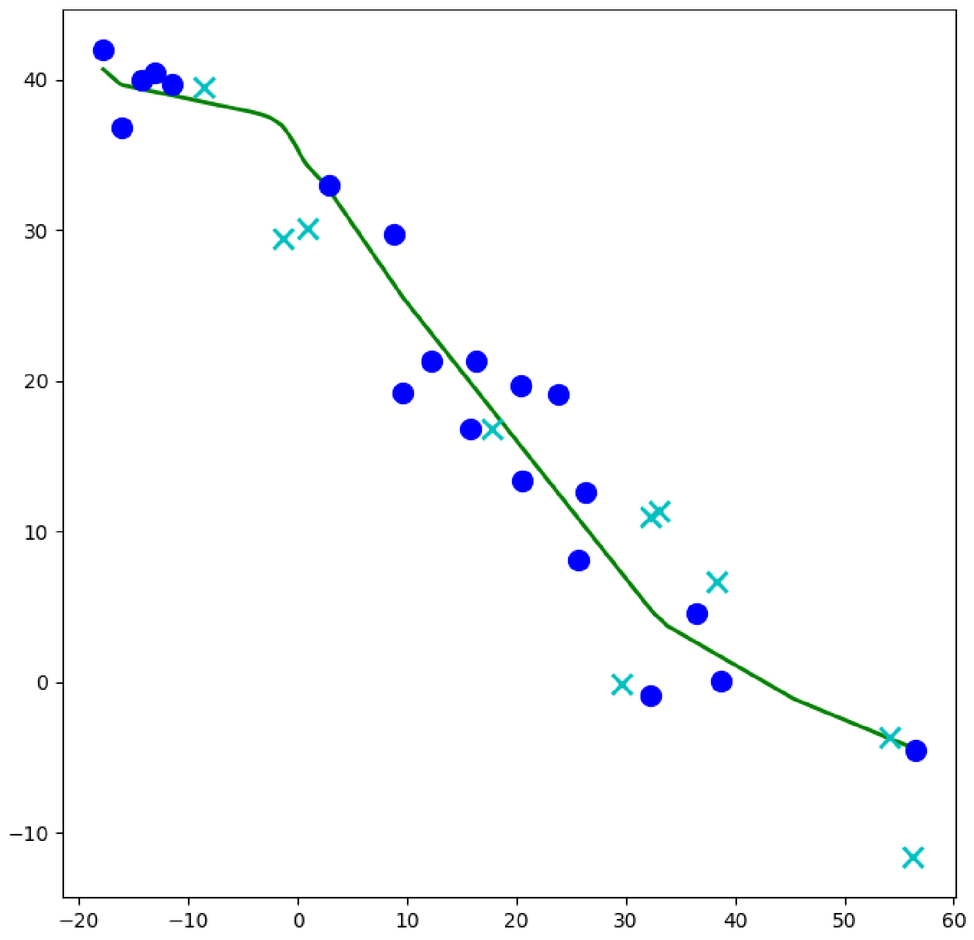
Just because it can fit any function will it?

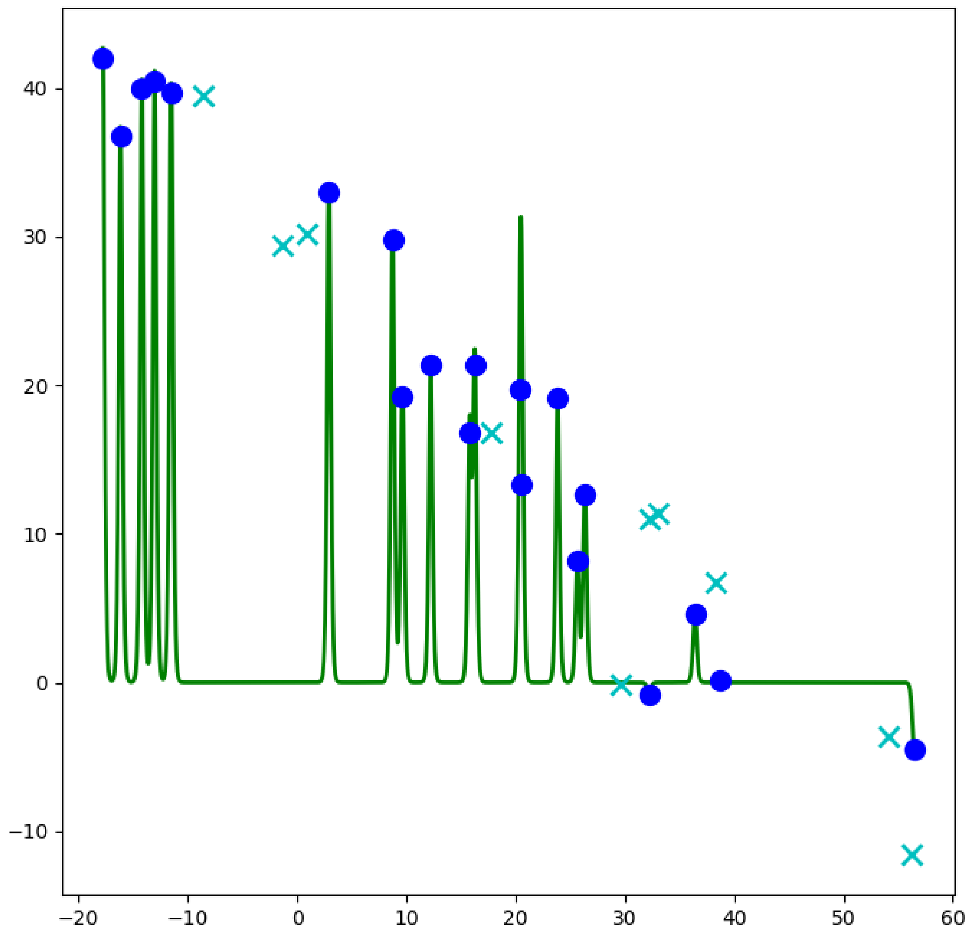
- Objective function is non-convex

  → it will get stuck in local minima
- *Stochastic* gradient decent take pseudorandom steps

  → Helps pop out of local minima
- Is getting stuck at a local minima necessarily a bad thing??

Network to Approximate a 1-D Function





Debugging Overfitting and Underfitting

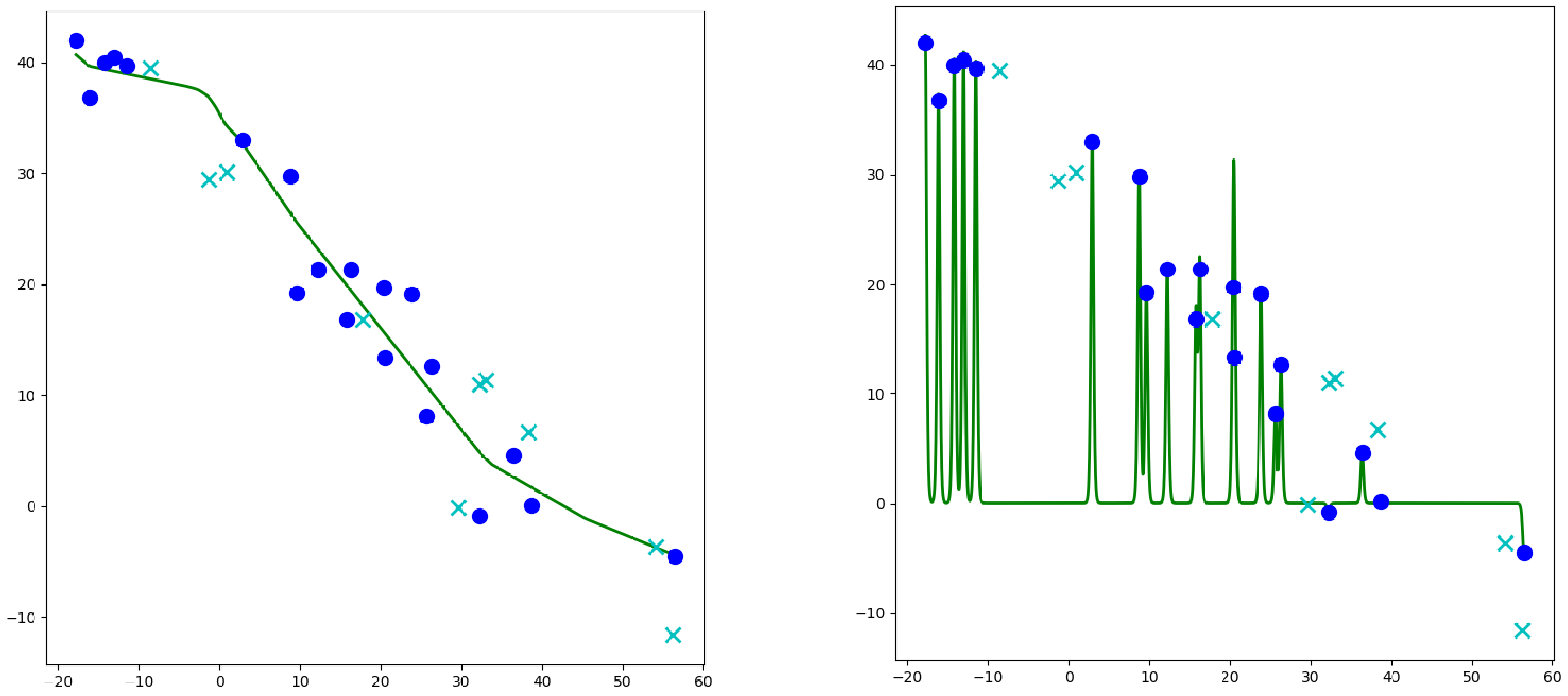
Underfitting (check this first!)

- Evidence: poor training loss (and poor validation loss)
  - Note: Compare with human performance as a baseline, i.e., make sure the task isn't impossible for a human (with unlimited resources)
- Try: Adding more capacity to network (wider or deeper)
  - But how do we choose a new network structure?

Overfitting

- Evidence: good training loss, but poor validation loss
- Evidence: really large parameter values
- Try: Regularization (we'll learn about this soon!)
- Try: Adding more data

Network to Approximate a 1-D Function



Summary: Neural Networks Properties

Practical considerations

- Large number of neurons
  - Danger for overfitting
- Modeling assumptions vs data assumptions trade-off
- Gradient descent can easily get stuck local optima

What if there are no non-linear activations?

- A deep neural network with only linear layers can be reduced to an exactly equivalent single linear layer

Universal Approximation Theorem:

- A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy.