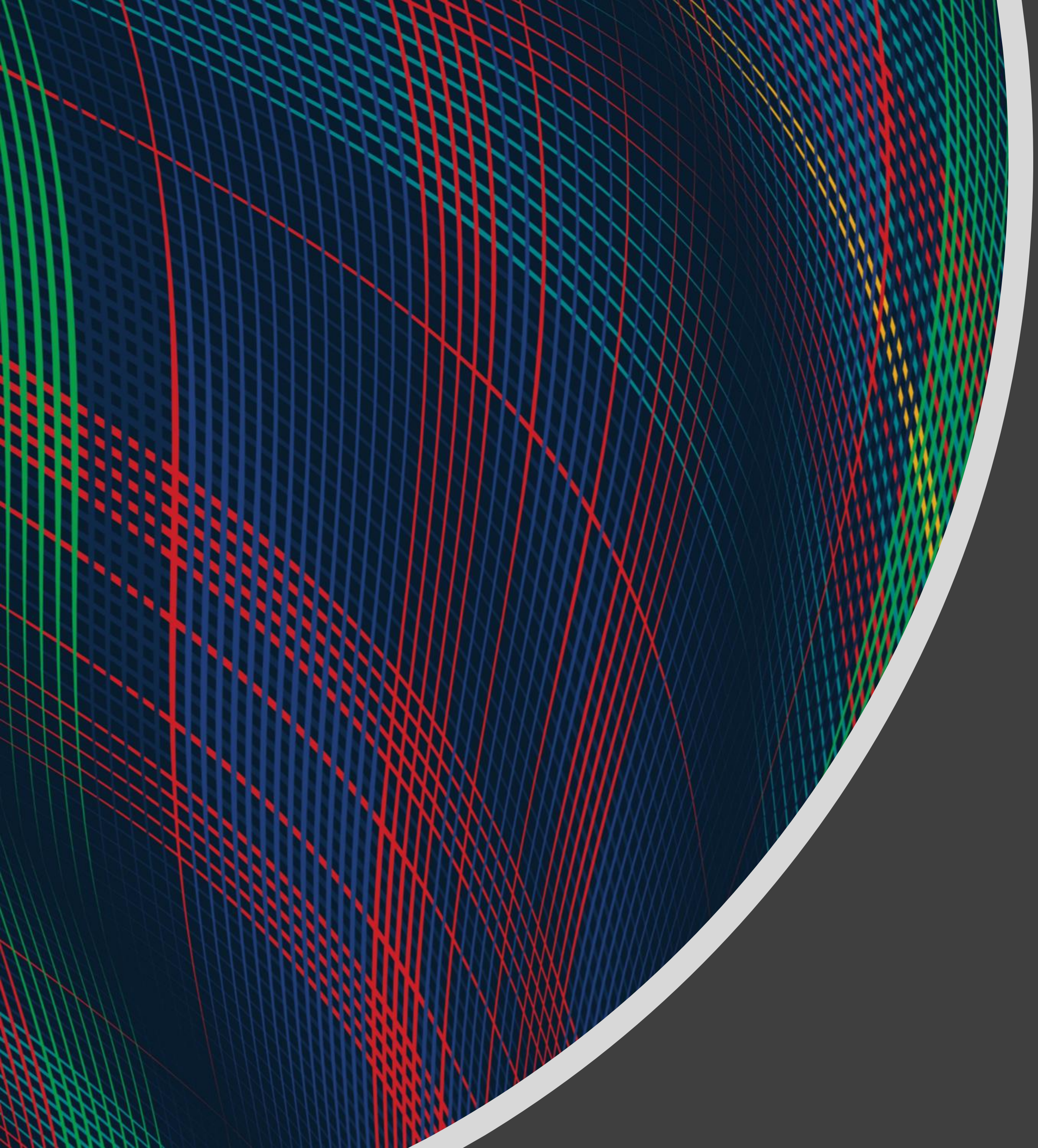


10-315 Introduction to ML

Neural Networks

Instructor: Pat Virtue

Poll 1

Logistic regression for 28x28=784 pixel hand-written digit images into 10 classes:

How many parameters (including bias terms)?

A. 10

B. 10+784

C. 10*784

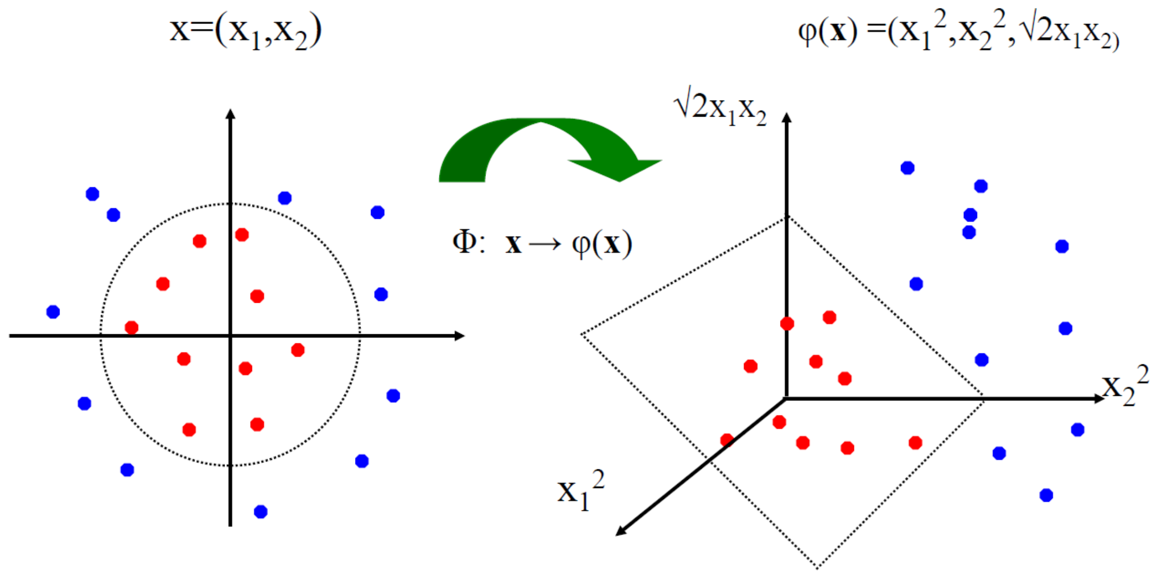
D. 10*784 + 10

E. 10*784 + 784

Linear Classifiers for Nonlinear Data

Linear classifiers have linear decision boundaries

Feature mapping can convert nonlinear data to higher dimensions

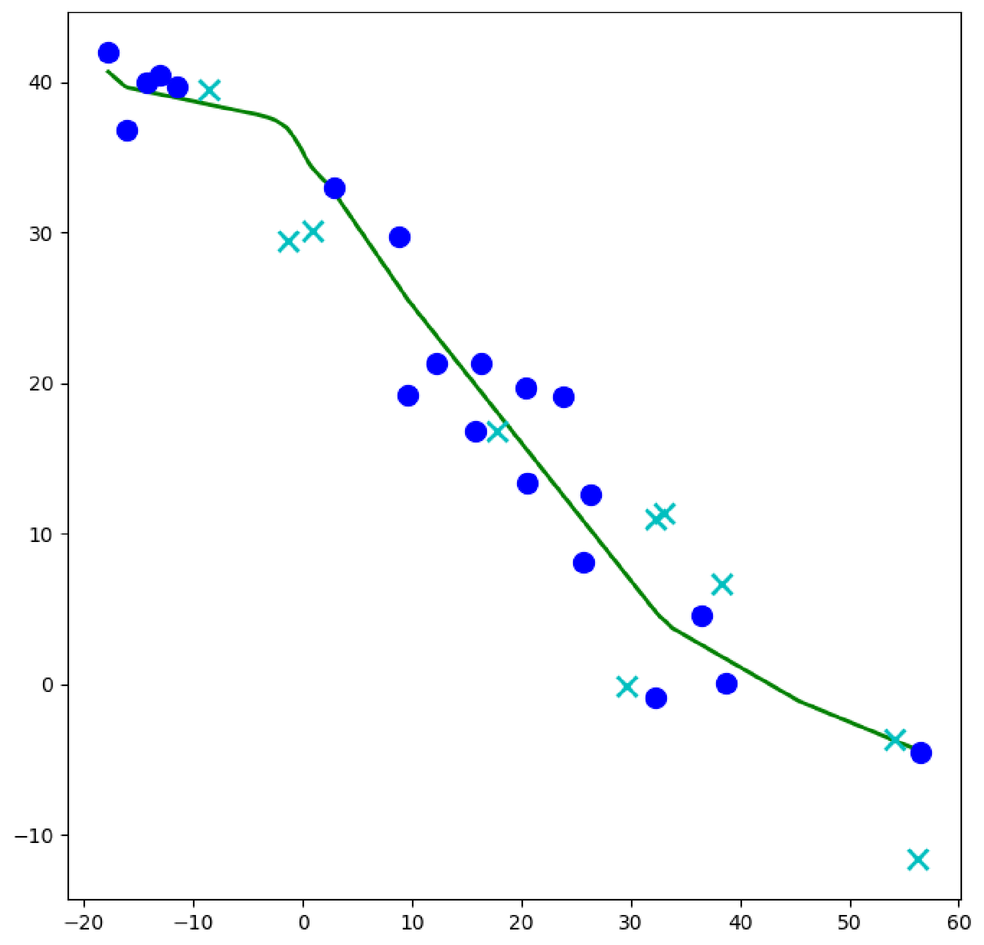

$x=(x_1,x_2)$
$\varphi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$
$\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$
$\sqrt{2}x_1x_2$
x_1^2
x_2^2

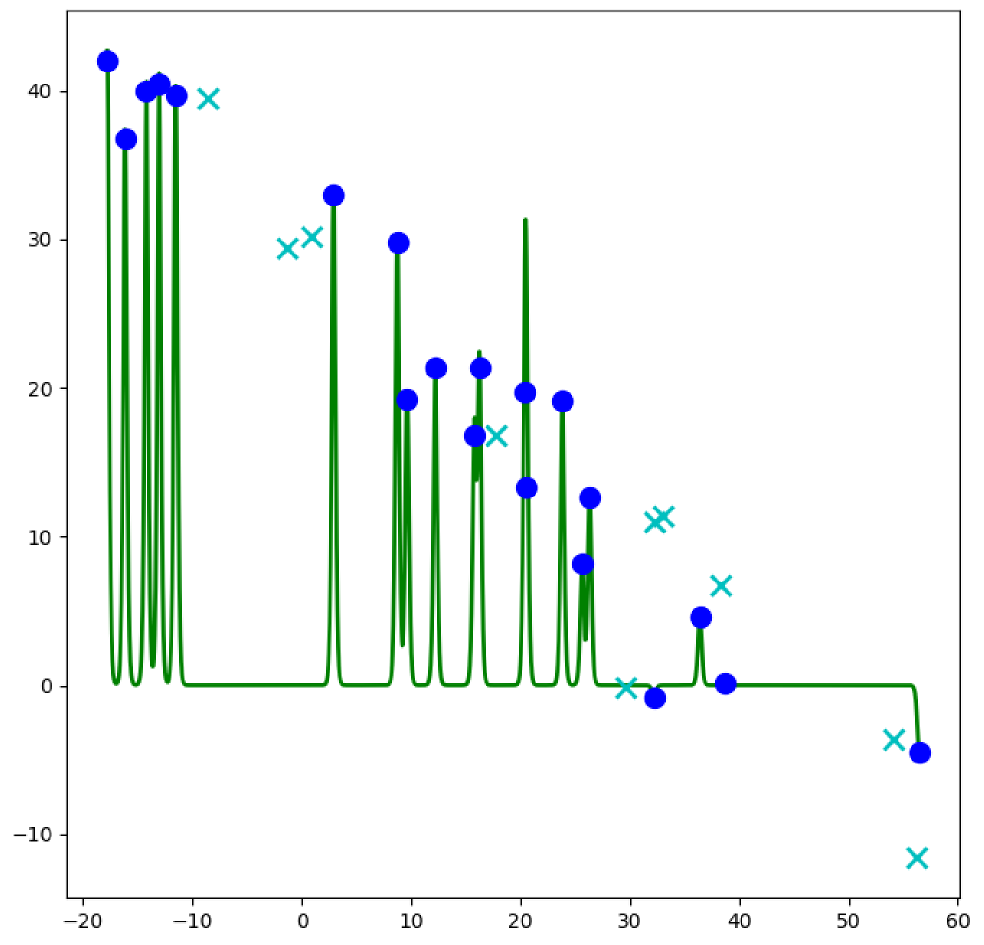
This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

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:

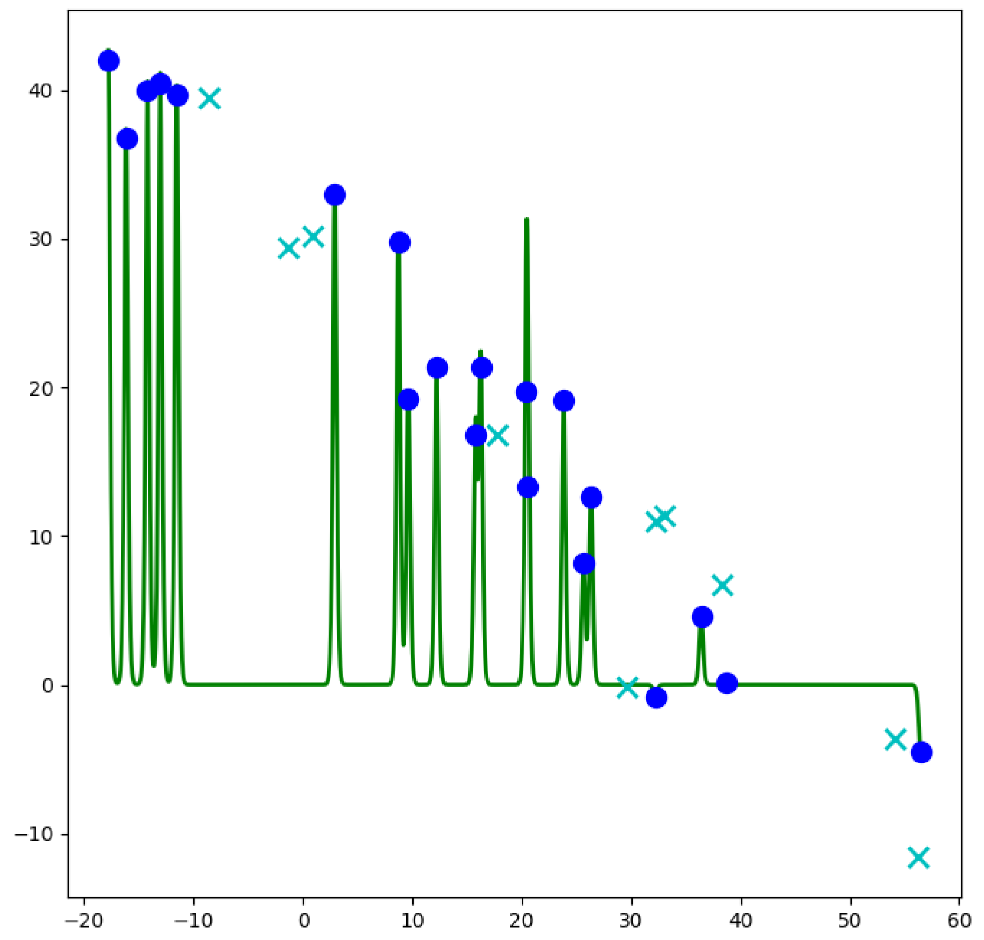
Linear, Sigmoid, Step, or ReLU

Network to Approximate a 1-D Function

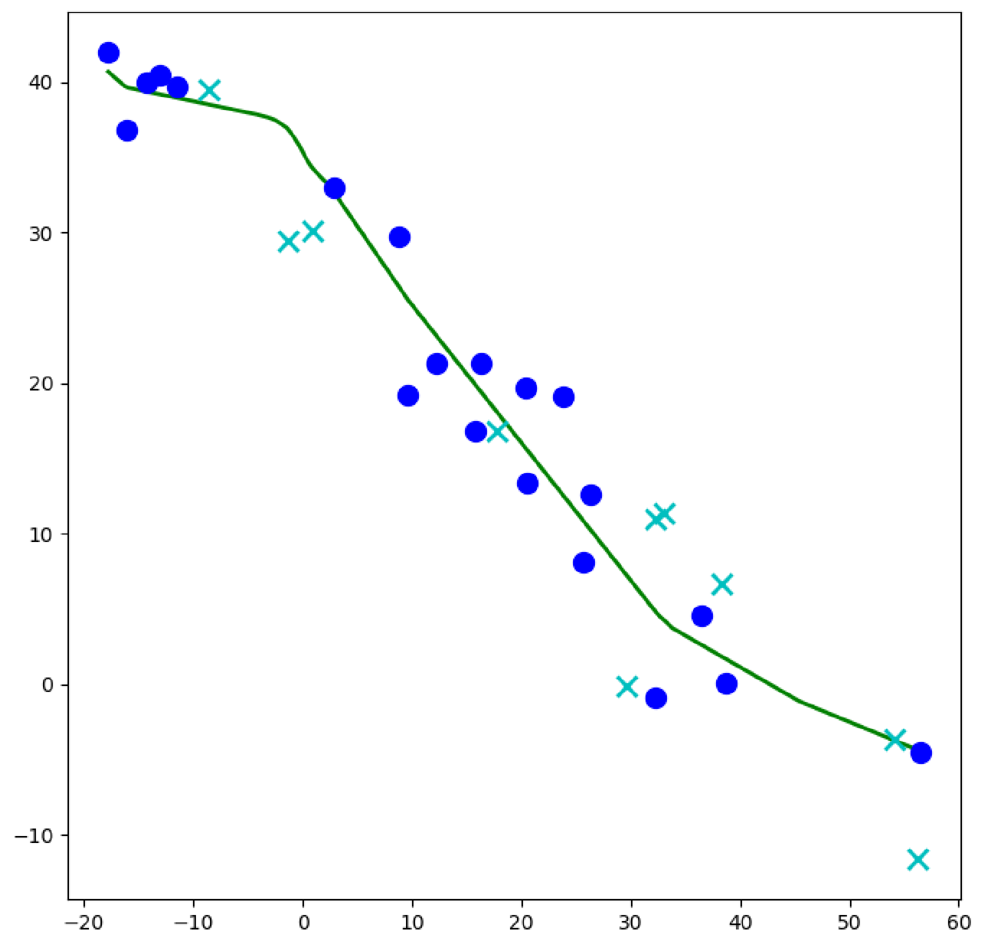
Design a network to approximate this function using:

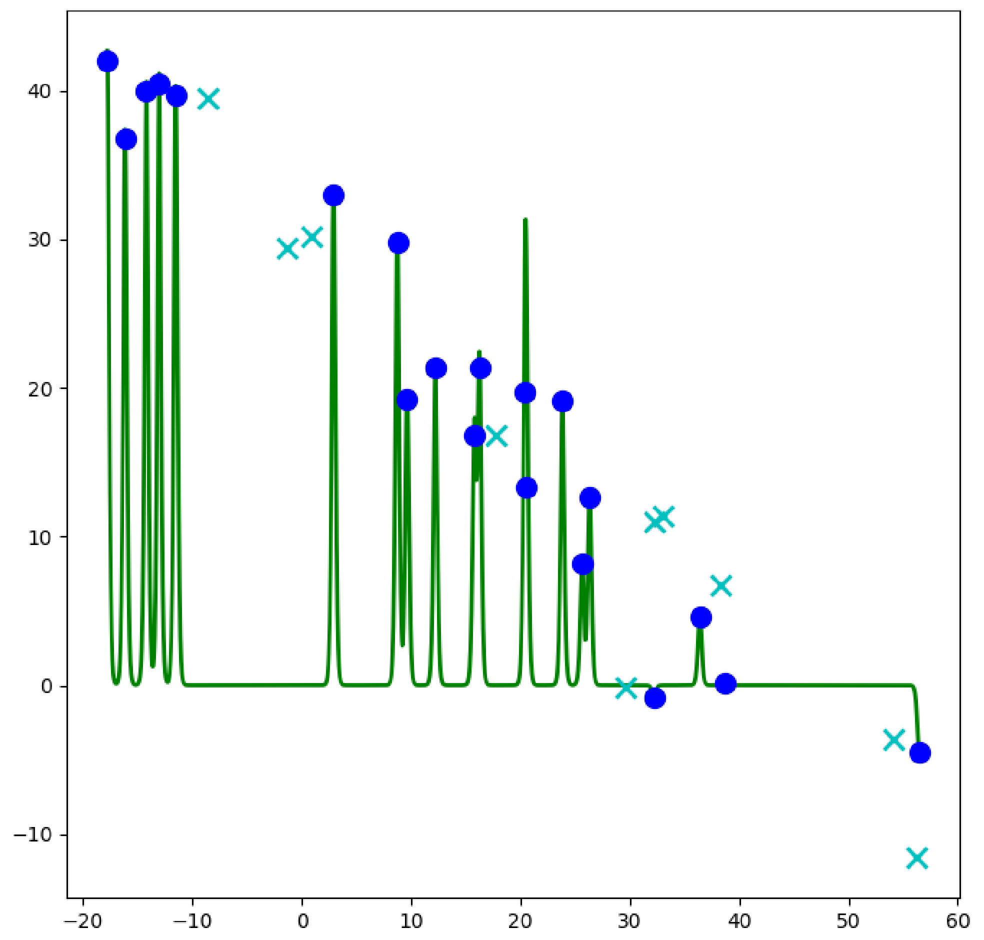
Linear, Sigmoid, Step, or ReLU

Network to Approximate a 1-D Function



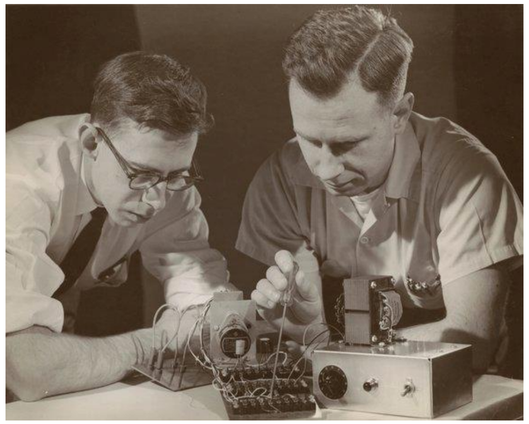
Network to Approximate a 1-D Function

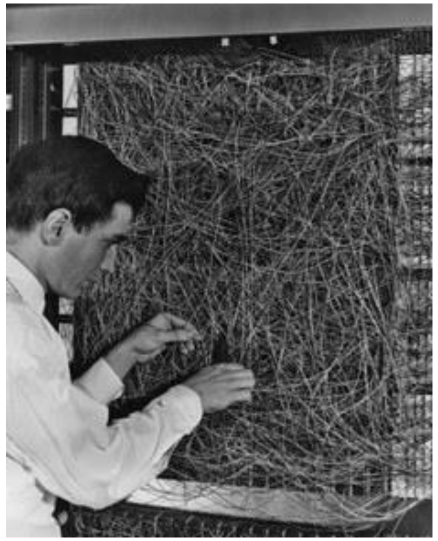




Perceptron History

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?

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

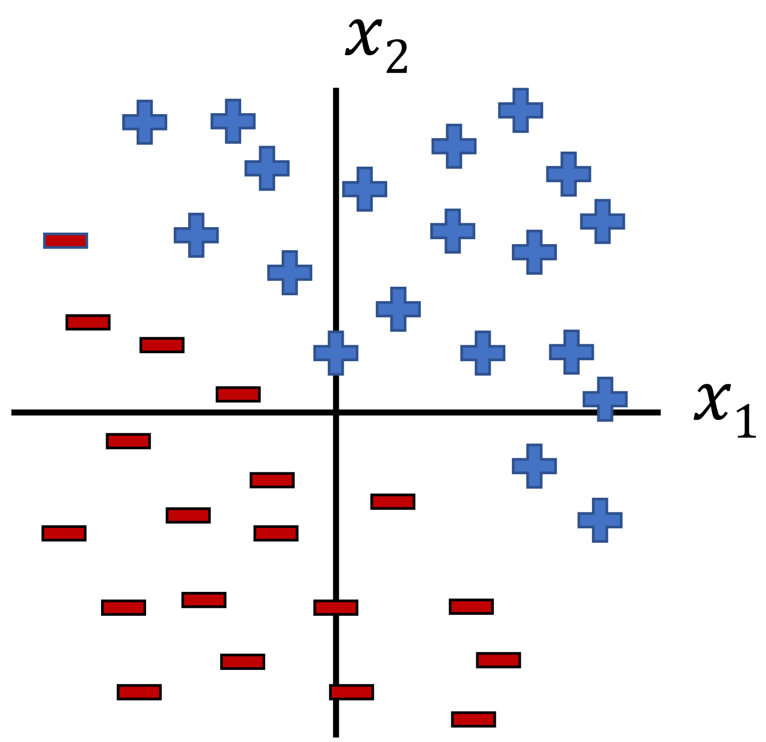
A. $\boldsymbol{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b = 0$

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 2

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

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

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

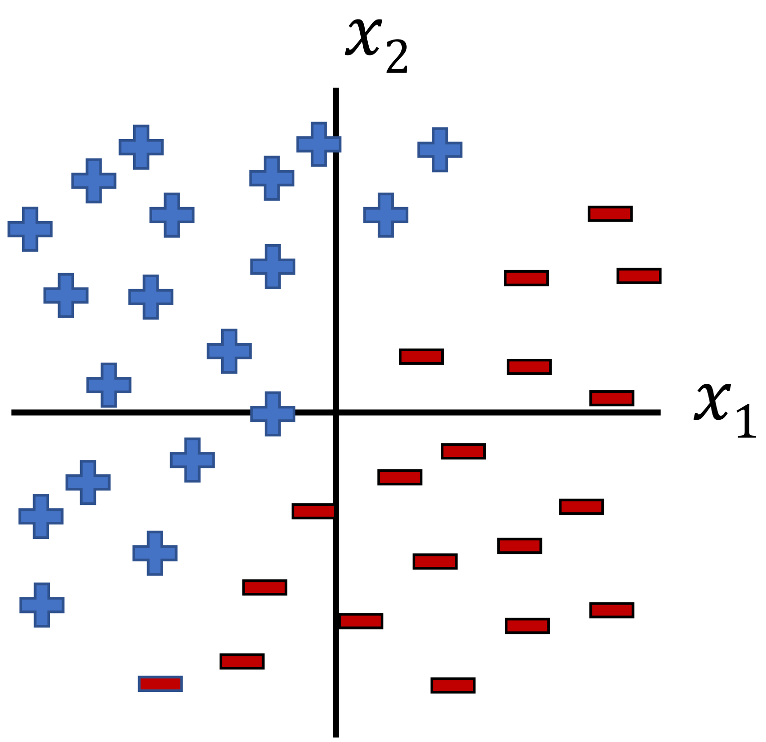
A. $\boldsymbol{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b = 0$

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 3

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

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

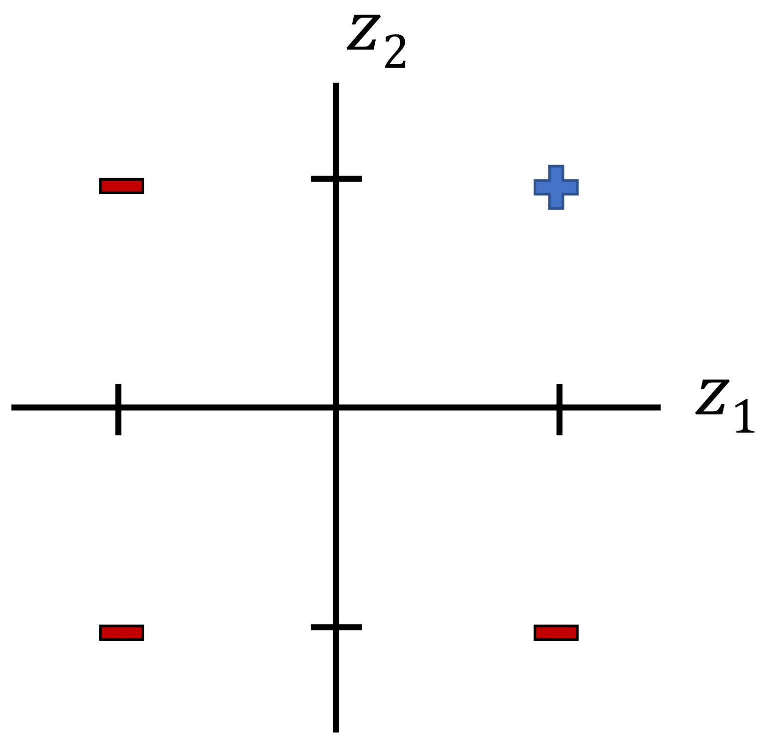
$$sign(\boldsymbol{x}) = \begin{cases} 1, & if\ x \geq 0 \\ -1, & if\ x < 0 \end{cases}$$

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

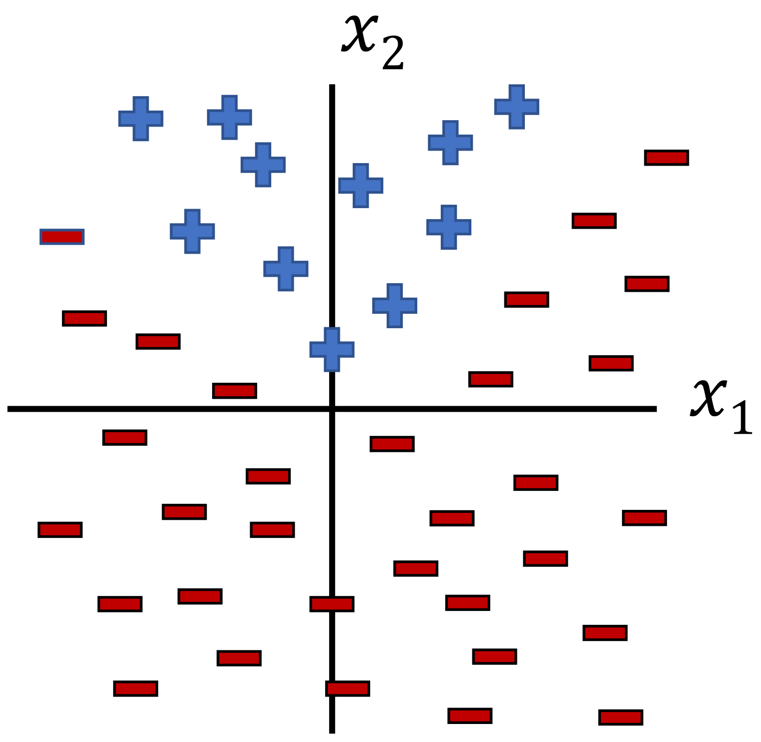
D. None of the above



Classification Design Challenge

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

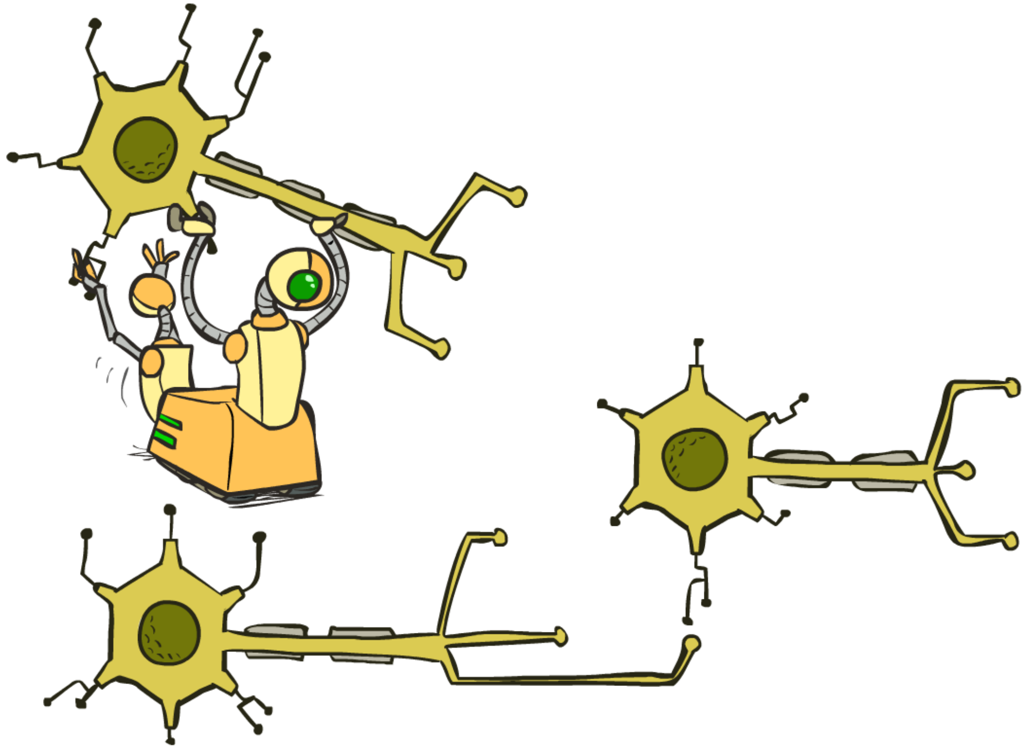
$$h_A(\boldsymbol{x}) = sign\left(\boldsymbol{w}_A^T\boldsymbol{x} + b_A\right)$$
$$h_B(\boldsymbol{x}) = sign\left(\boldsymbol{w}_B^T\boldsymbol{x} + b_B\right)$$
$$h_C(\boldsymbol{x}) = sign\left(\boldsymbol{w}_C^T\boldsymbol{x} + b_C\right)$$



Multilayer Perceptrons

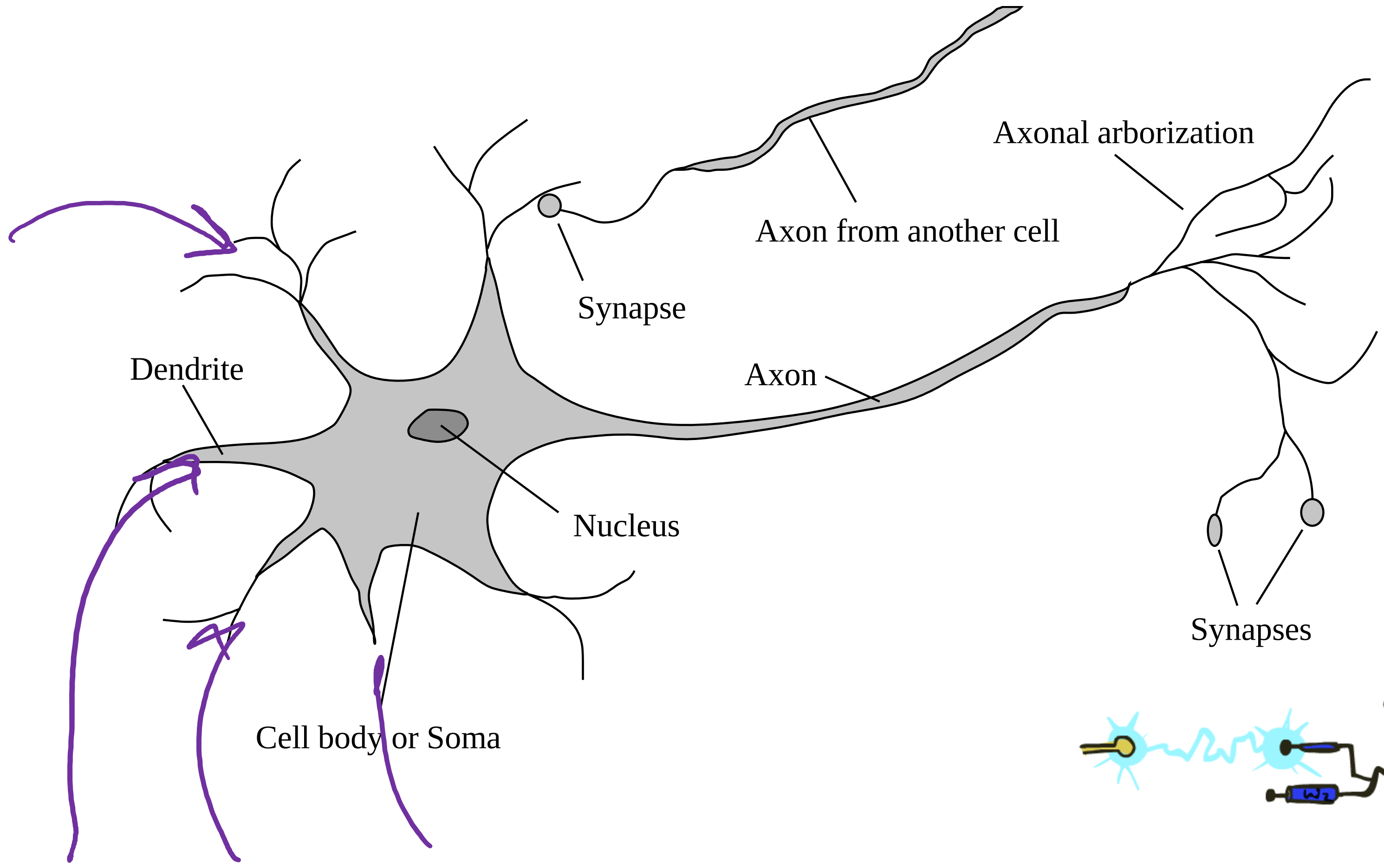
A ***multilayer perceptron*** is a feedforward neural network with at least one ***hidden layer*** (nodes that are neither inputs nor outputs)

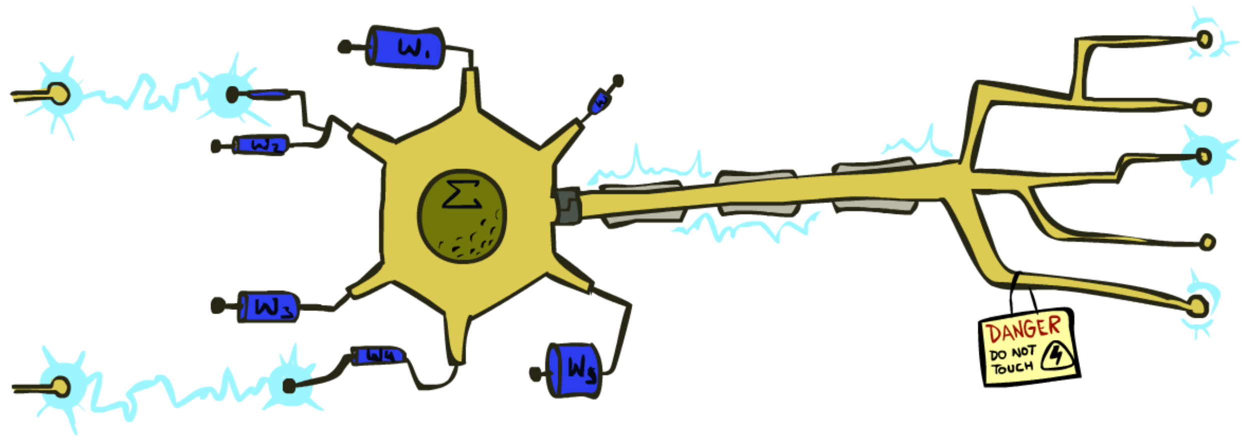
MLPs with enough hidden nodes can represent any function



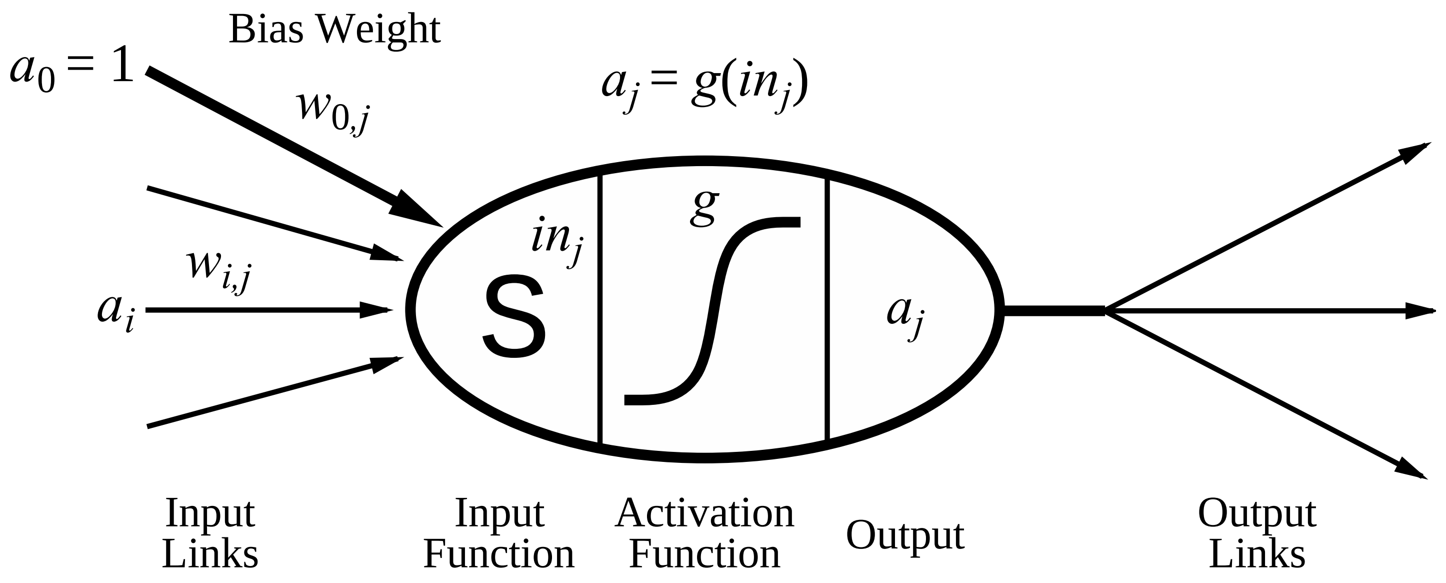
Slide from UC Berkeley AI

Very Loose Inspiration: Human Neurons





Simple Model of a Neuron (McCulloch & Pitts, 1943)



Inputs a_i come from the output of node i to this node j (or from "outside")

Each input link has a ***weight*** $w_{i,j}$

There is an additional fixed input a_0 with ***bias*** weight $w_{0,j}$

The total input is $in_j = \Sigma_i \, w_{i,j} \, a_i$

The output is $a_j = g(in_j) = g(\Sigma_i \, w_{i,j} \, a_i) = g(\mathbf{w.a})$

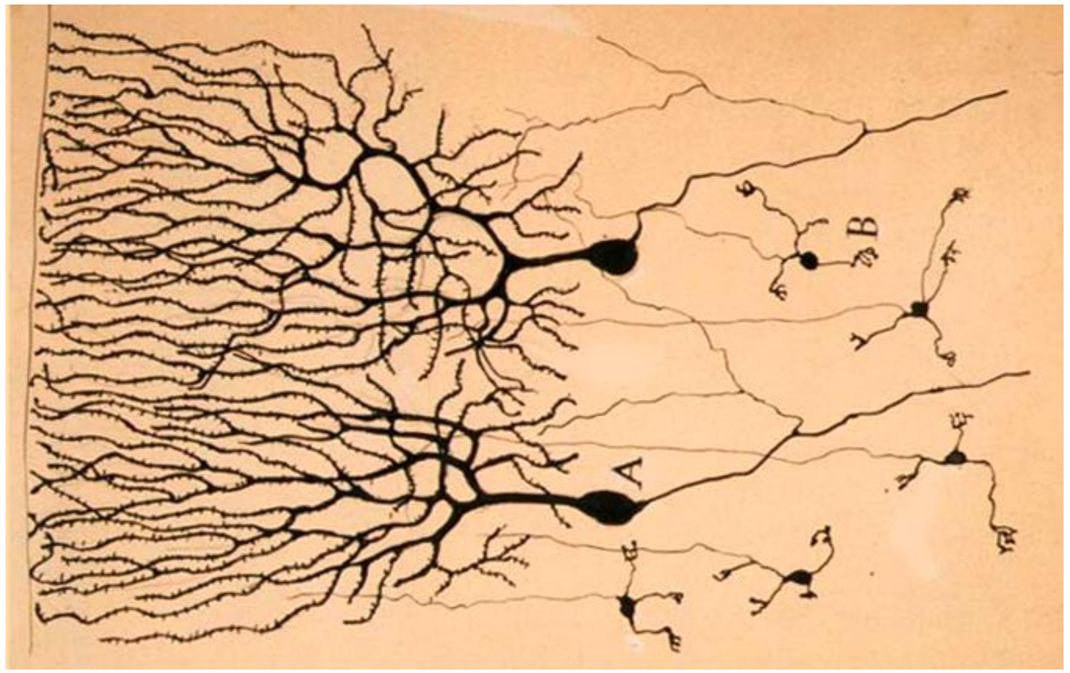
Slide from UC Berkeley AI

Neural Networks

Inspired by actual human brain

Input Signal

Output Signal



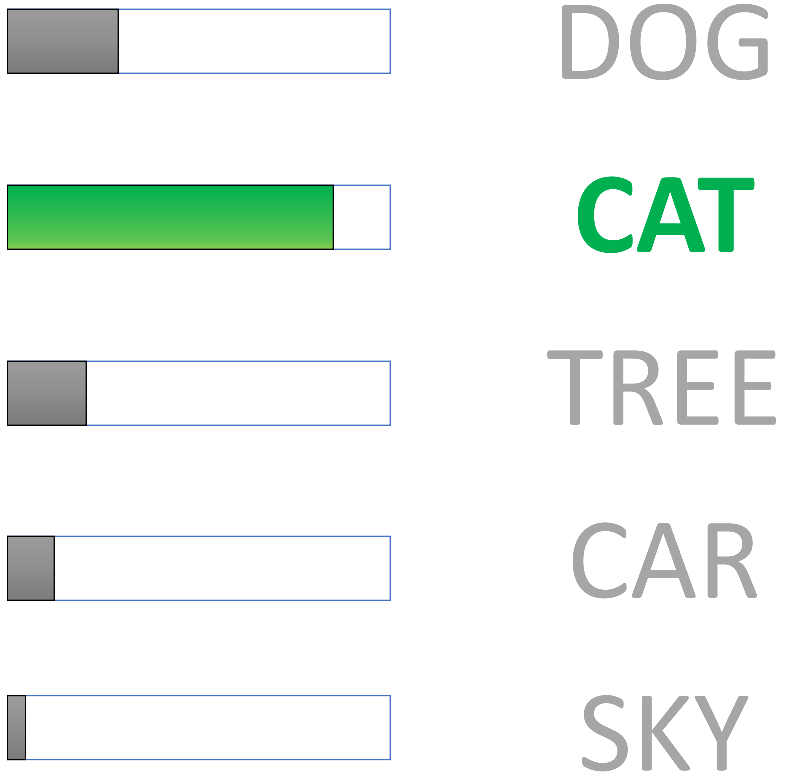
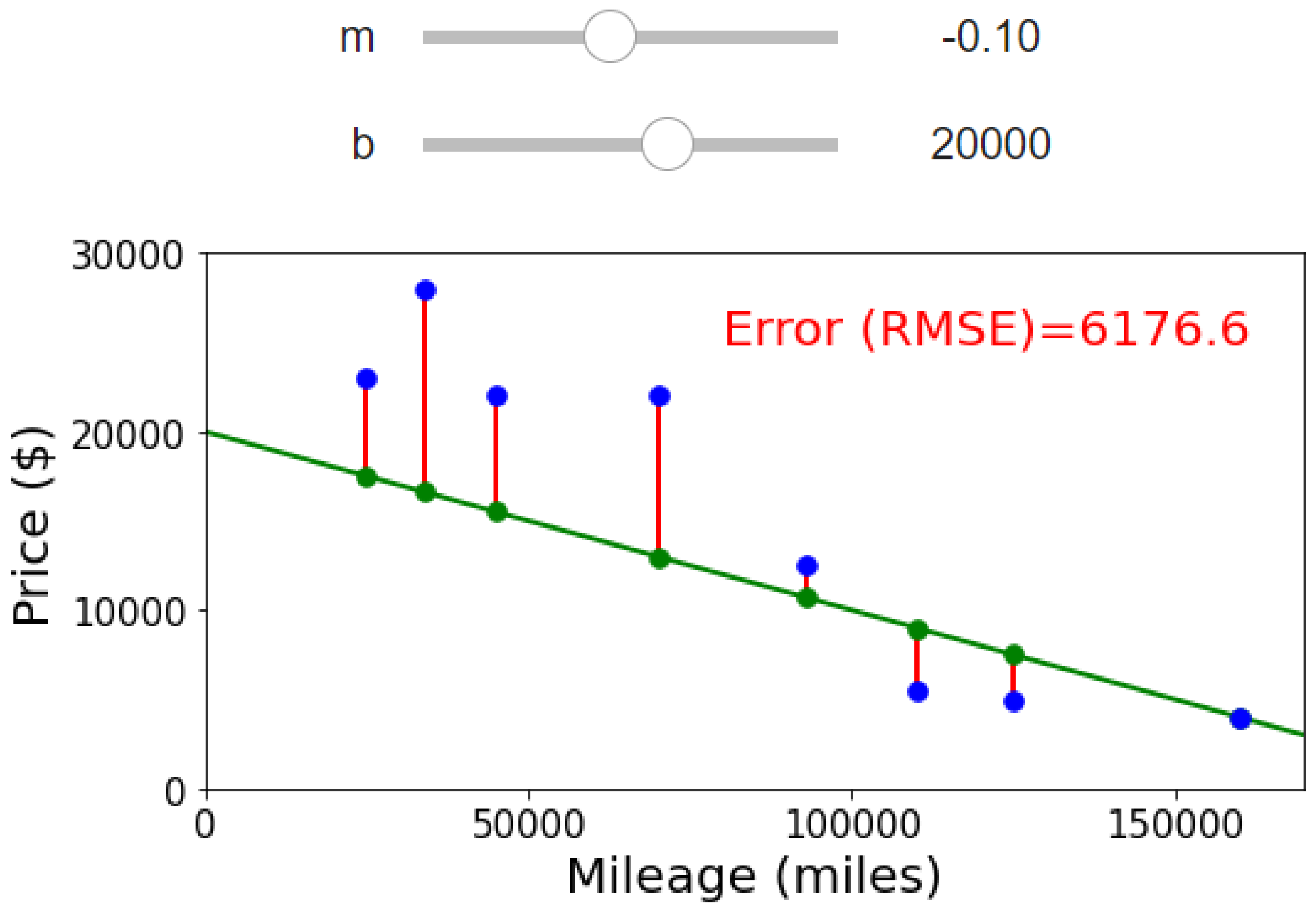


Image: https://en.wikipedia.org/wiki/Neuron

Neural Networks

Building on optimization for linear and logistic regression

- Selling my car

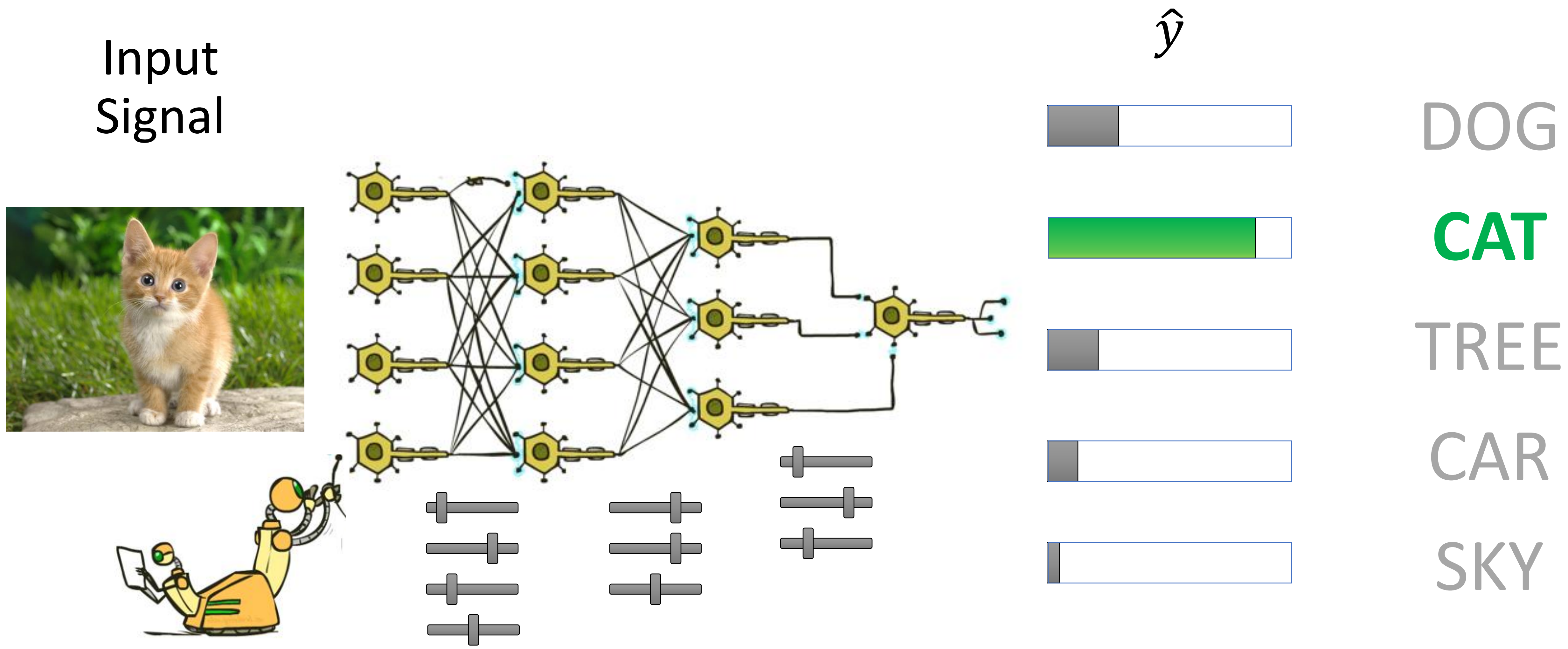


Neural Networks

Many layers of neurons, millions of parameters

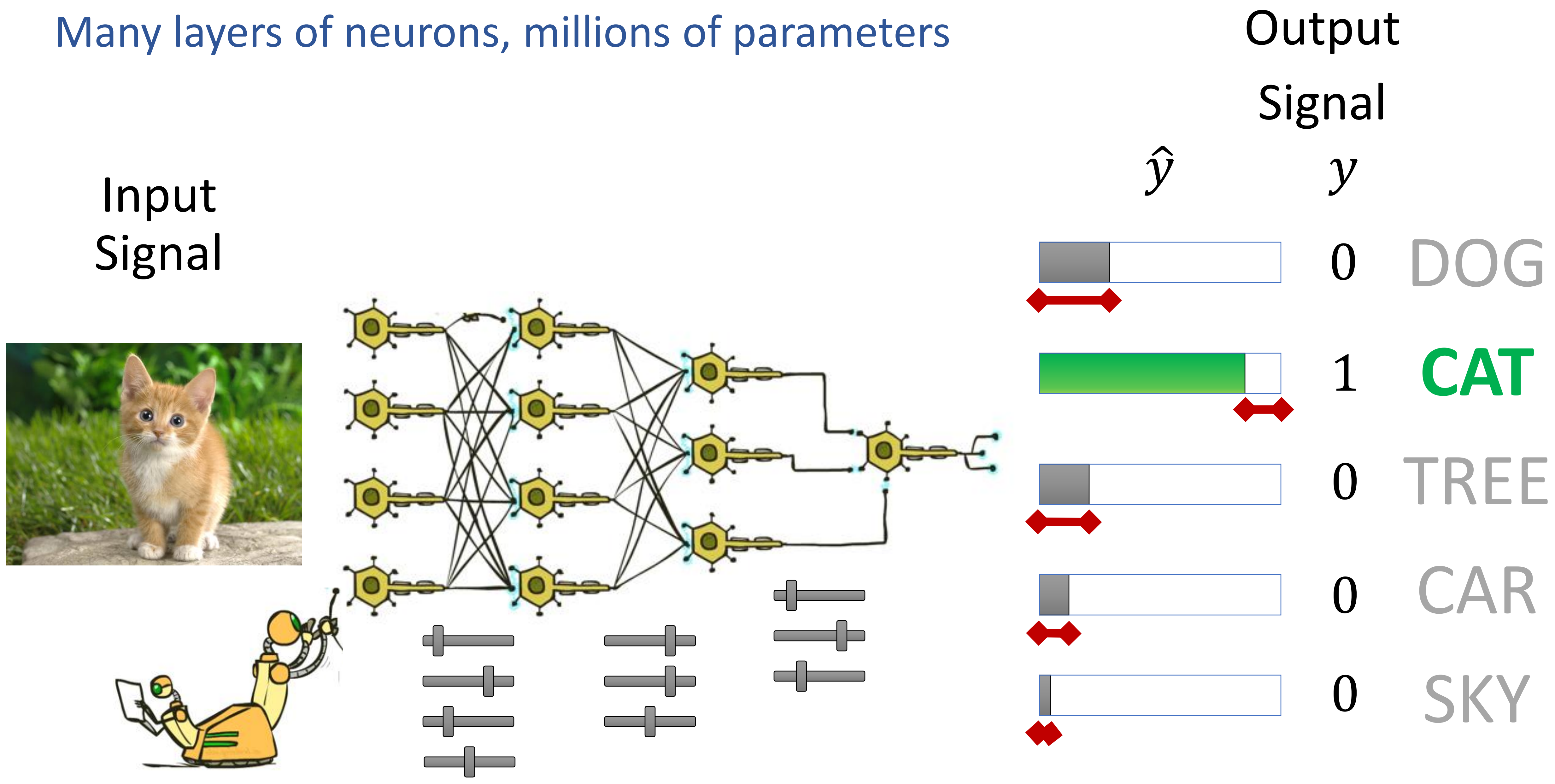
Output Signal

Input Signal



Neural Networks

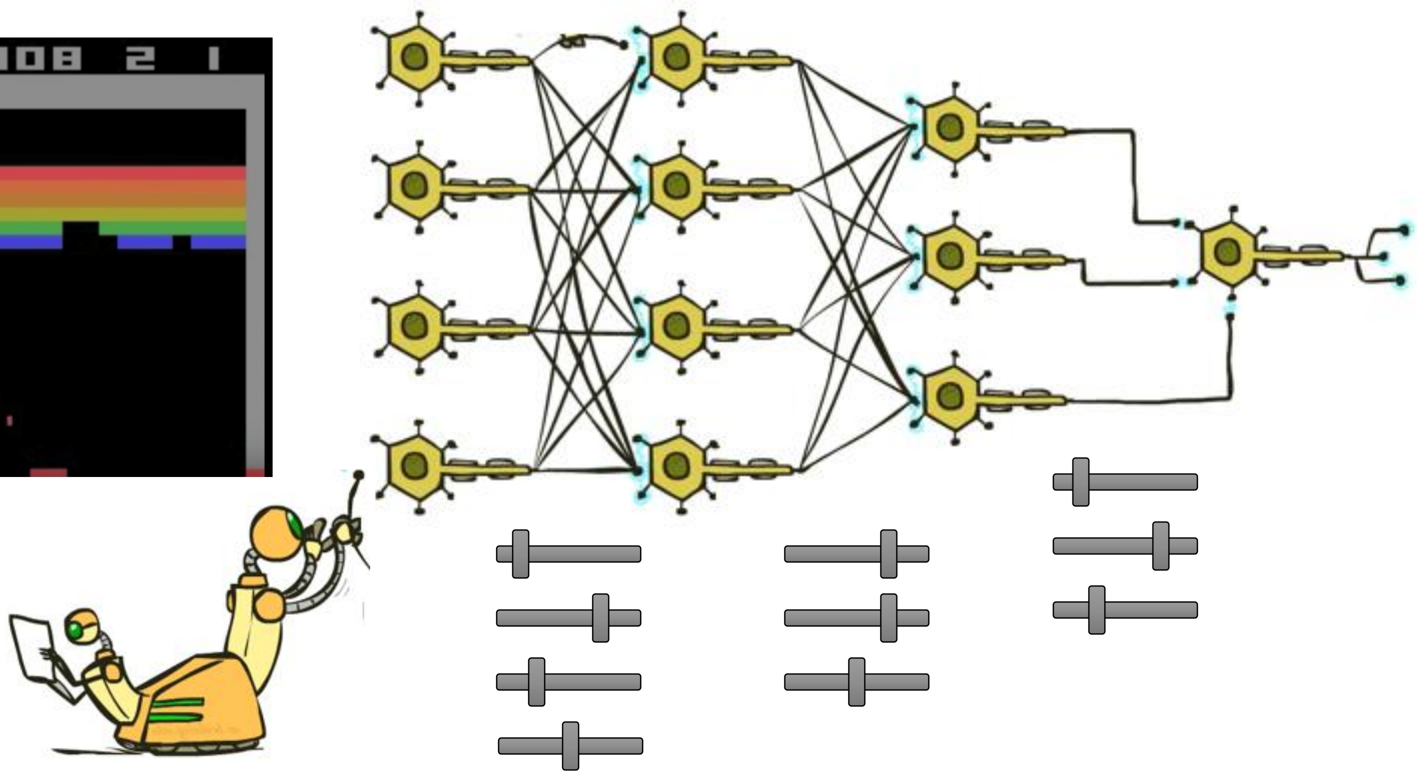
Many layers of neurons, millions of parameters



Neural Networks

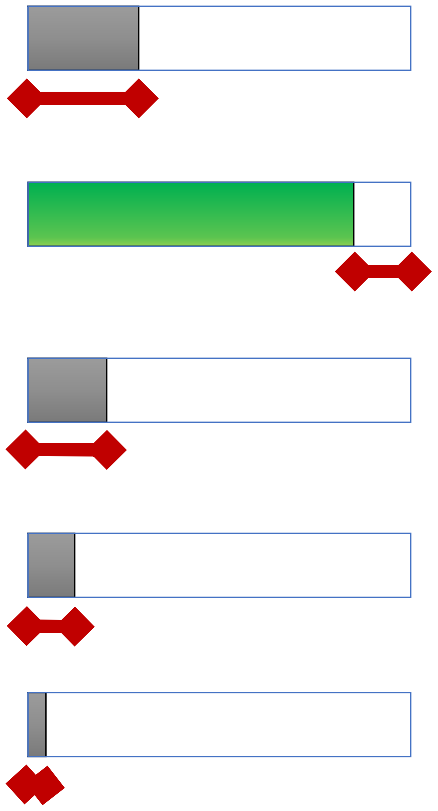
Many layers of neurons, millions of parameters

Input Signal



Output Signal

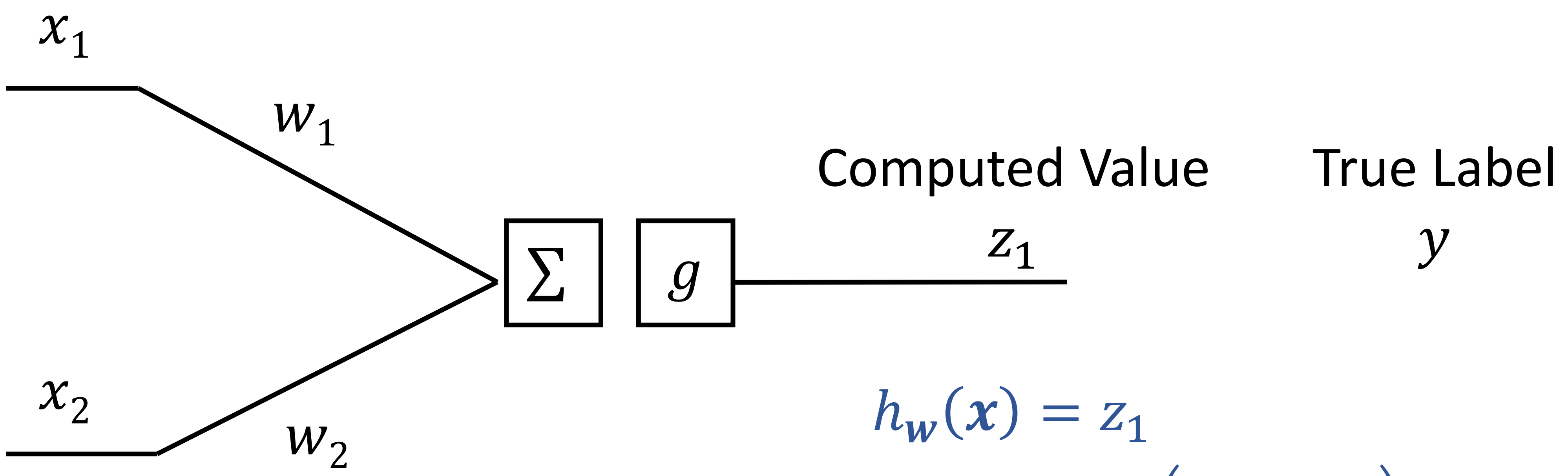
$\hat{y}$



Single Neuron

Single neuron system

- Perceptron (if g is step function)
- Logistic regression (if g is sigmoid)
- Linear regression (if g is nothing)



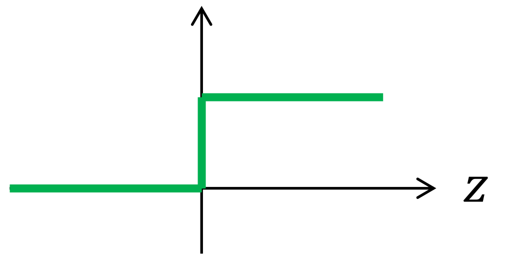
$$h_{\mathbf{w}}(\boldsymbol{x}) = z_1$$

$$h_{\mathbf{w}}(\boldsymbol{x}) = g\left(\sum_i w_i x_i\right)$$

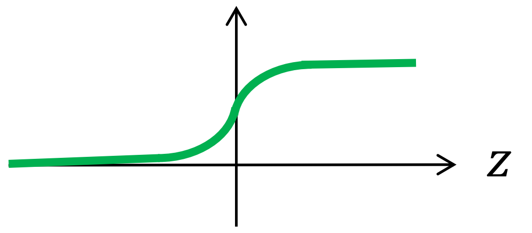
Activation Functions

It would be really helpful to have a g(z) that was nicely differentiable

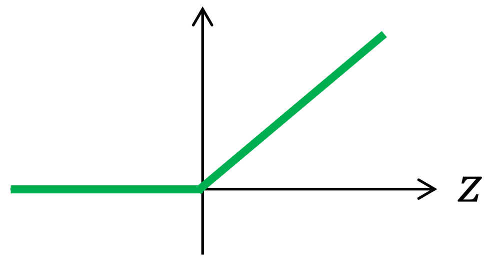
- Hard threshold: $g(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$ $\quad \frac{dg}{dz} = \begin{cases} 0 & z \geq 0 \\ 0 & z < 0 \end{cases}$



- Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$ $\quad \frac{dg}{dz} = g(z)\big(1 - g(z)\big)$
- (Softmax)



- ReLU: $g(z) = \ max(0, z)$ $\quad \frac{dg}{dz} = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$



Optimizing

How do we find the “best” set of weights?

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = g\left(\sum_j w_j x_j\right)$$

Loss Functions

Regression

- Squared error: $\ell(y, \hat{y}) = (y - \hat{y})^2$

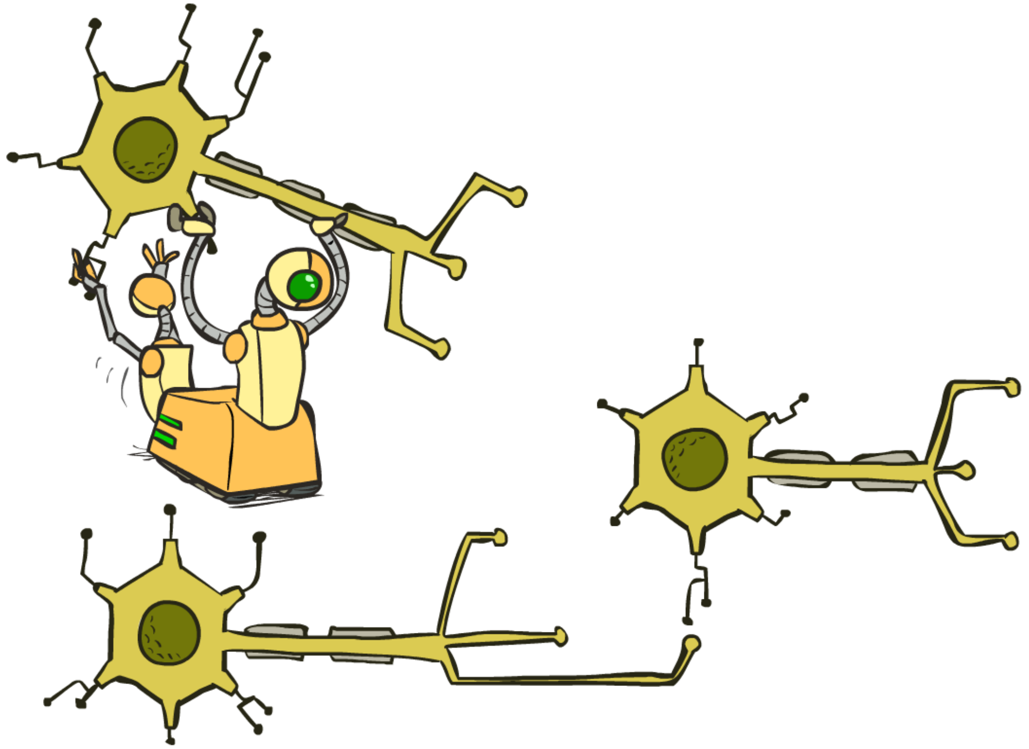
Classification

- Cross entropy: $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_k y_k \log \hat{y}_k$

Multilayer Perceptrons

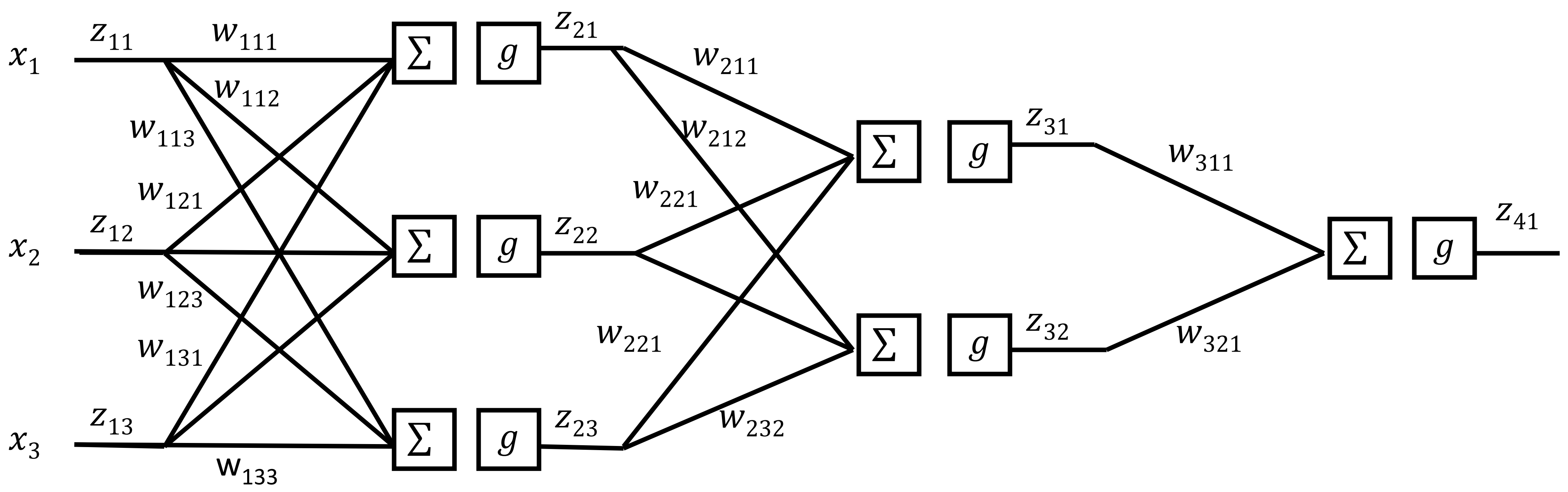
A ***multilayer perceptron*** is a feedforward neural network with at least one ***hidden layer*** (nodes that are neither inputs nor outputs)

MLPs with enough hidden nodes can represent any function



Slide from UC Berkeley AI

Neural Network Equations



$$h_w(\boldsymbol{x}) = z_{4,1}$$

$$z_{1,1} = x_1$$

$$z_{4,1} = g\left(\sum_i w_{3,i,1}\, z_{3,i}\right)$$

$$z_{3,1} = g\left(\sum_i w_{2,i,1}\, z_{2,i}\right)$$

$$z_{d,1} = g\left(\sum_i w_{d-1,i,1}\, z_{d-1,i}\right)$$

$$h_w(x) = g\left(\sum_k w_{3,k,1}\; g\left(\sum_j w_{2,j,k}\; g\left(\sum_i w_{1,i,j}\; x_i\right)\right)\right)$$

Optimizing

How do we find the “best” set of weights?

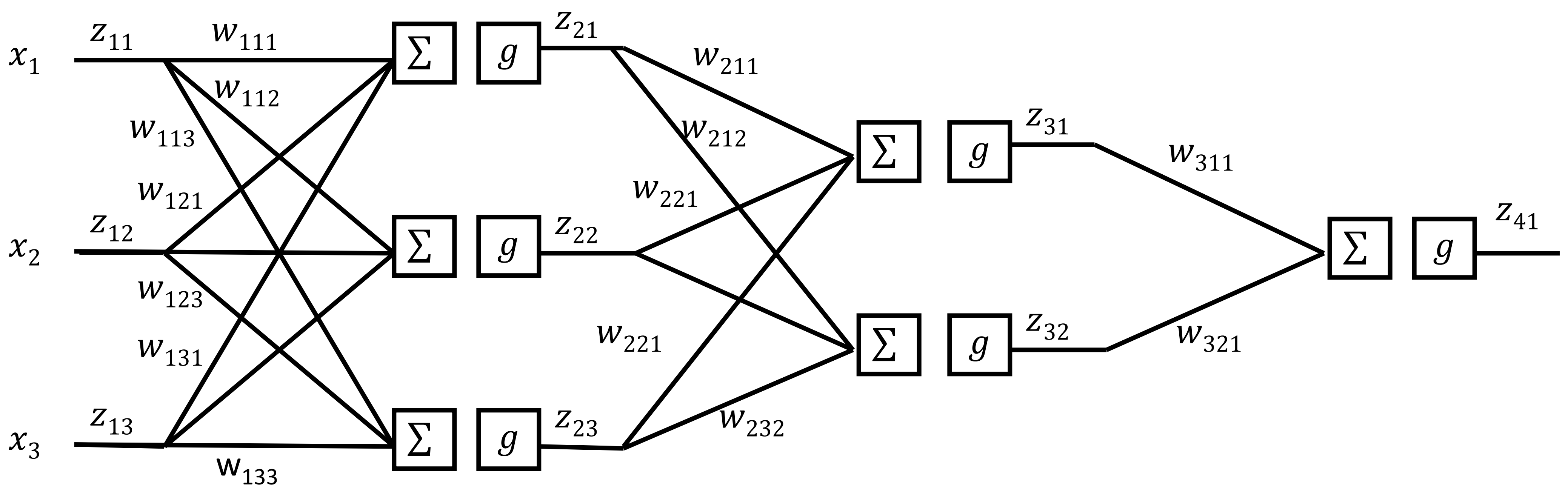
$$h_w(x) = g\left(\sum_k w_{3,k,1}\ g\left(\sum_j w_{2,j,k}\ g\left(\sum_i w_{1,i,j}\ x_i\right)\right)\right)$$

Optimizing

How do we find the “best” set of weights?

$$h_w(x) = g\left(\sum_k w_{3,k,1}\ g\left(\sum_j w_{2,j,k}\ g\left(\sum_i w_{1,i,j}\ x_i\right)\right)\right)$$

Neural Network Equations



How do we describe this network?

input

neuron (node)

hidden layer

output layer

loss

Network Optimization Details

Reminder: Calculus Chain Rule (scalar version)

$$y = f(z)$$
$$z = g(x)$$

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

Network Optimization

$$J(\mathbf{w}) = z_3$$

$$z_3 = f_3(w_3, z_2)$$

$$z_2 = f_2(w_2, z_1)$$

$$z_1 = f_1(w_1, x)$$

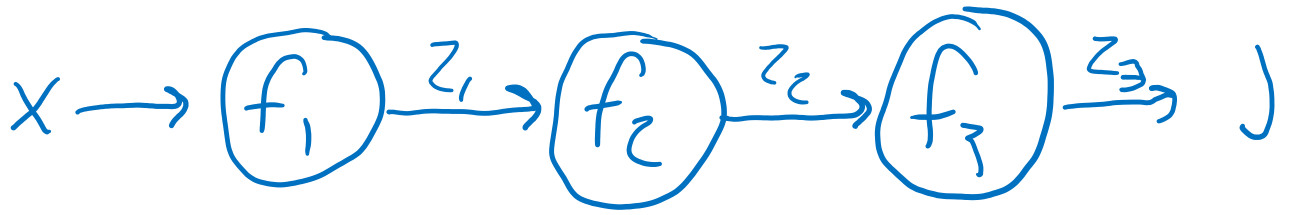
Network Optimization: Forward then Backwards

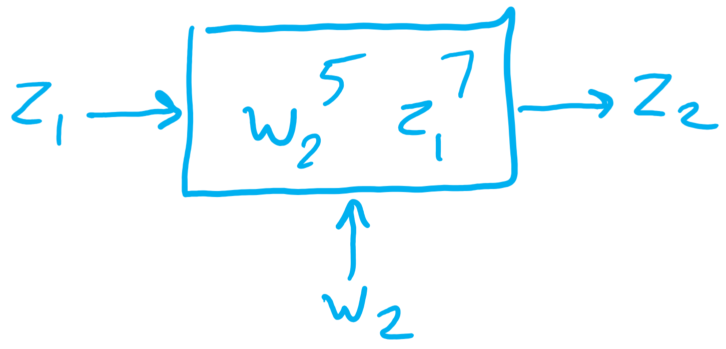
$J(\mathbf{w}) = z_3$

$z_3 = f_3(w_3, z_2)$

$z_2 = f_2(w_2, z_1) = w_2^5 z_1^7$

$z_1 = f_1(w_1, x)$


$$\begin{tikzcd} x \arrow[r] & \boxed{f_1} \arrow[r, "z_1"] & \boxed{f_2} \arrow[r, "z_2"] & \boxed{f_3} \arrow[r, "z_3"] & J \end{tikzcd}$$


$$\begin{tikzcd} z_1 \arrow[r] & \boxed{w_2^5 \; z_1^7} \arrow[r] & z_2 \\ & w_2 \arrow[u] & \end{tikzcd}$$

$$\frac{\partial J}{\partial w_3} = \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$

$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$

Lots of repeated calculations

Network Optimization: Layer Implementation

$J(\mathbf{w}) = z_3$

$z_3 = f_3(w_3, z_2)$

$z_2 = f_2(w_2, z_1)$

$z_1 = f_1(w_1, x)$

$$\frac{\partial J}{\partial w_3} = \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$
$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$
$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$

Lots of repeated calculations

Backpropagation (so-far)

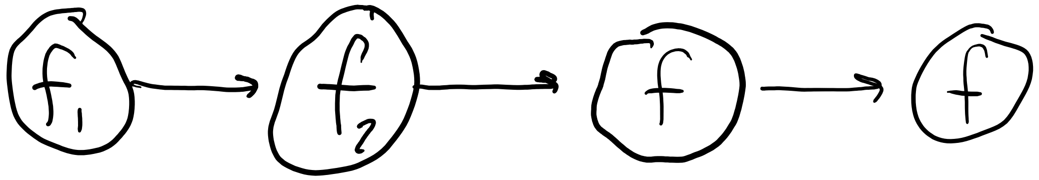
Compute derivatives per layer, utilizing previous derivatives

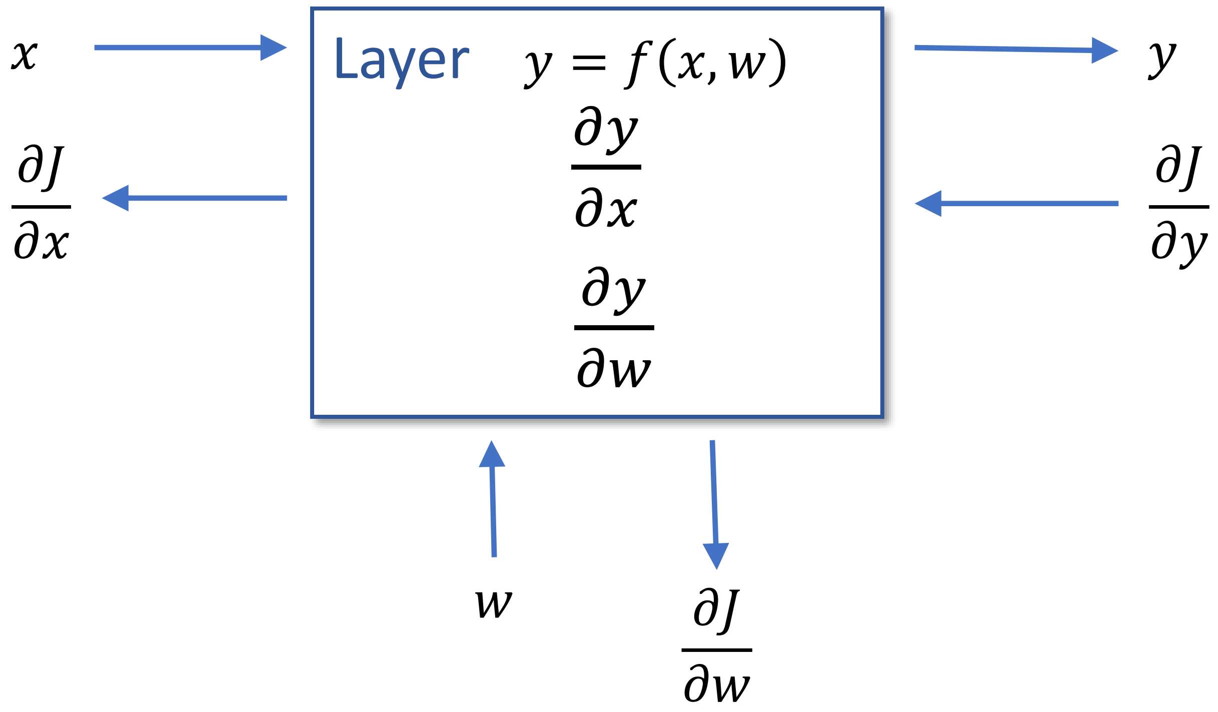
Objective: $J(\boldsymbol{w})$

Arbitrary layer: $y = f(x, w)$

Need:

- $\frac{\partial J}{\partial x} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial x}$
- $\frac{\partial J}{\partial w} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w}$





x
$\frac{\partial J}{\partial x}$
Layer $y = f(x,w)$
$\frac{\partial y}{\partial x}$
$\frac{\partial y}{\partial w}$
y
$\frac{\partial J}{\partial y}$
w
$\frac{\partial J}{\partial w}$

Previous Exercise

Suppose we have a function that takes in a vector and squares each element individually, returning another vector, $\boldsymbol{y} = f(\boldsymbol{x})$.

$$f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) \rightarrow \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \end{bmatrix} \quad y_1 \qquad \text{Example: } f\left(\begin{bmatrix} 7 \\ 3 \\ 5 \end{bmatrix}\right) \rightarrow \begin{bmatrix} 49 \\ 9 \\ 25 \end{bmatrix}$$

What is $\partial \boldsymbol{y} / \partial \boldsymbol{x}$? (use numerator layout)

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 2x_1 & 0 & 0 \\ 0 & 2x_2 & 0 \\ 0 & 0 & 2x_3 \end{bmatrix}$$

Exercise

Prove $\frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T A \mathbf{v} = \left(A^T + A\right)\mathbf{v}$ for $\mathbf{v} \in \mathbb{R}^2$ and $A \in \mathbb{R}^{2\times 2}$

$f(\mathbf{v}) = \mathbf{v}^T A \mathbf{v} \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \qquad A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$

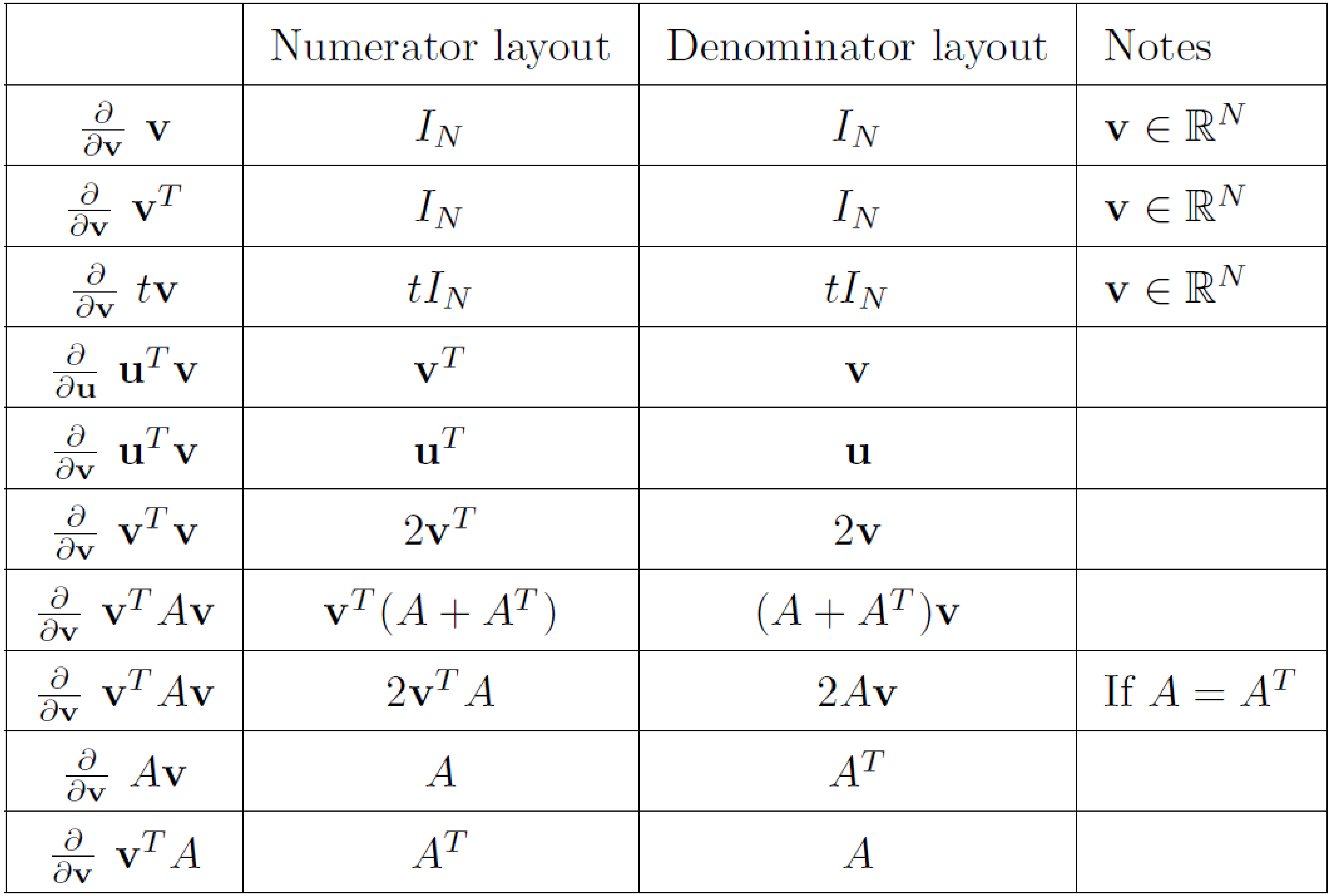
Prove $\frac{\partial f}{\partial \mathbf{v}} = \left(A^T + A\right)\mathbf{v}$

Hint: Start by expanding $A\mathbf{v}$ and then expanding $\mathbf{v}^T A \mathbf{v}$, i.e., write $f(v_1, v_2)$ in terms of scalars $v_1, v_2, A_{1,1}, A_{1,2}, A_{2,1}, A_{2,2}$

Hint: Write out scalar partial derivatives, $\frac{\partial f}{\partial v_1}$ and $\frac{\partial f}{\partial v_2}$

Hint: Work both top-down and bottom-up, i.e., also expand $\left(A^T + A\right)\mathbf{v}$, so you can see where you are going.

Derivatives of Functions with Respect to Vectors



	Numerator layout	Denominator layout	Notes
$\frac{\partial}{\partial \mathbf{v}} \mathbf{v}$	I_N	I_N	$\mathbf{v} \in \mathbb{R}^N$
$\frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T$	I_N	I_N	$\mathbf{v} \in \mathbb{R}^N$
$\frac{\partial}{\partial \mathbf{v}} t\mathbf{v}$	tI_N	tI_N	$\mathbf{v} \in \mathbb{R}^N$
$\frac{\partial}{\partial \mathbf{u}} \mathbf{u}^T\mathbf{v}$	$\mathbf{v}^T$	$\mathbf{v}$	
$\frac{\partial}{\partial \mathbf{v}} \mathbf{u}^T\mathbf{v}$	$\mathbf{u}^T$	$\mathbf{u}$	
$\frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T\mathbf{v}$	$2\mathbf{v}^T$	$2\mathbf{v}$	
$\frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T A\mathbf{v}$	$\mathbf{v}^T(A + A^T)$	$(A + A^T)\mathbf{v}$	
$\frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T A\mathbf{v}$	$2\mathbf{v}^T A$	$2A\mathbf{v}$	If $A = A^T$
$\frac{\partial}{\partial \mathbf{v}} A\mathbf{v}$	A	A^T	
$\frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T A$	A^T	A	

Calculus

Multivariable Chain Rule

Multivariable Chain Rule

$g_1(x) = 3x$

$g_2(x) = 5x$

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

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

Exercise: Multivariable Chain Rule

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

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

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

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

Multivariable Chain Rule

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

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

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

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

Calculus Chain Rule

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

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

Poll 4

Assume numerator layout:
Num outputs $\times$ Num inputs

$y = f(\boldsymbol{z}) \qquad y \in \mathbb{R},\ \boldsymbol{z} \in \mathbb{R}^N,\ x \in \mathbb{R}$

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

$$\frac{\partial y}{\partial x} = \cdots$$

A. $\frac{\partial y}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial x}$

B. $\left(\frac{\partial y}{\partial \boldsymbol{z}}\right)^T \frac{\partial \boldsymbol{z}}{\partial x}$

C. $\frac{\partial y}{\partial \boldsymbol{z}} \left(\frac{\partial \boldsymbol{z}}{\partial x}\right)^T$

D. $\left(\frac{\partial y}{\partial \boldsymbol{z}}\right)^T \left(\frac{\partial \boldsymbol{z}}{\partial x}\right)^T$

E. None of the above

Poll 5

Assume numerator layout:
Num outputs × Num inputs

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

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

$$\frac{\partial y}{\partial \boldsymbol{x}} = \cdots$$

A. $\frac{\partial y}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}$

B. $\left(\frac{\partial y}{\partial \boldsymbol{z}}\right)^T \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}$

C. $\frac{\partial y}{\partial \boldsymbol{z}} \left(\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}\right)^T$

D. $\left(\frac{\partial y}{\partial \boldsymbol{z}}\right)^T \left(\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}\right)^T$

E. None of the above

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!

Network Optimization

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

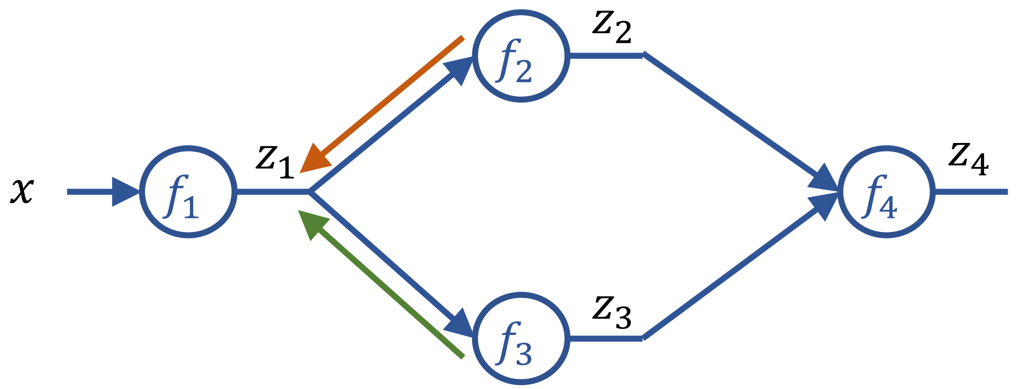
$$z_4 = f_4(w_D, w_E, z_2, z_3)$$

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

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

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

Need multivariate chain rule!



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

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

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

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

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

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

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

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

Backpropagation (updated)

Compute derivatives per layer, utilizing previous derivatives

Objective: $J(\boldsymbol{w})$

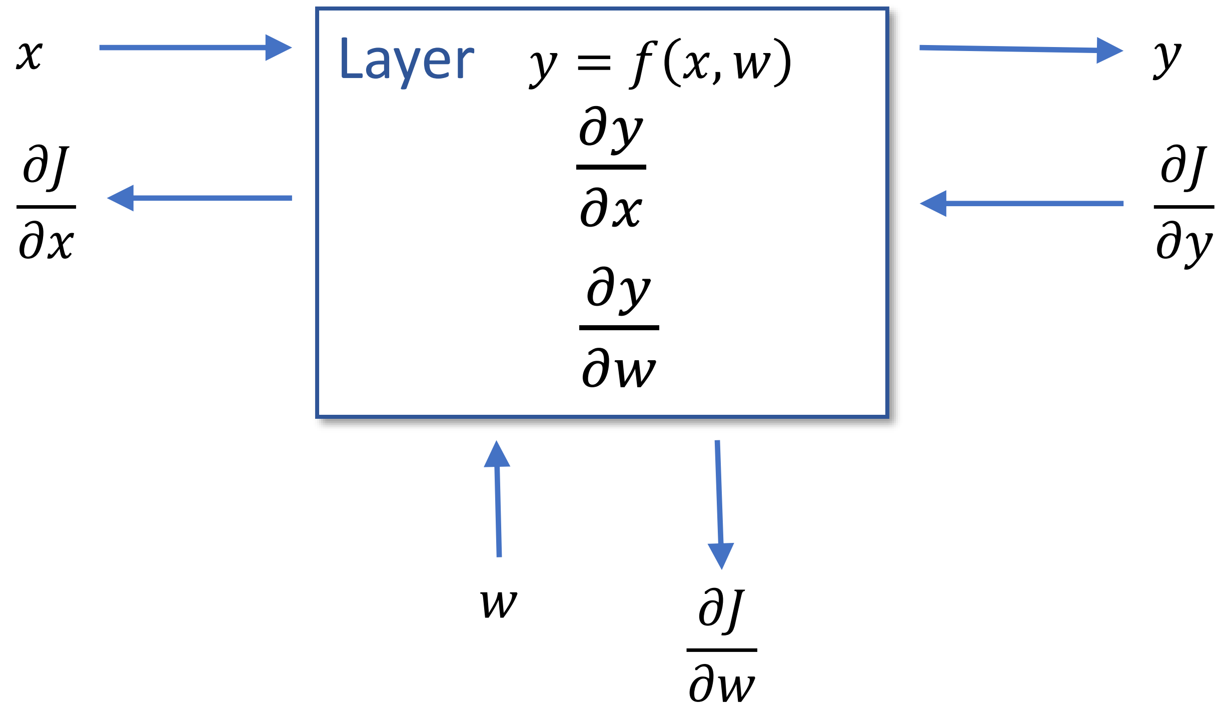
Arbitrary layer: $y = f(x, w)$

Init:

- $\frac{\partial J}{\partial x} = 0$
- $\frac{\partial J}{\partial w} = 0$

Compute:

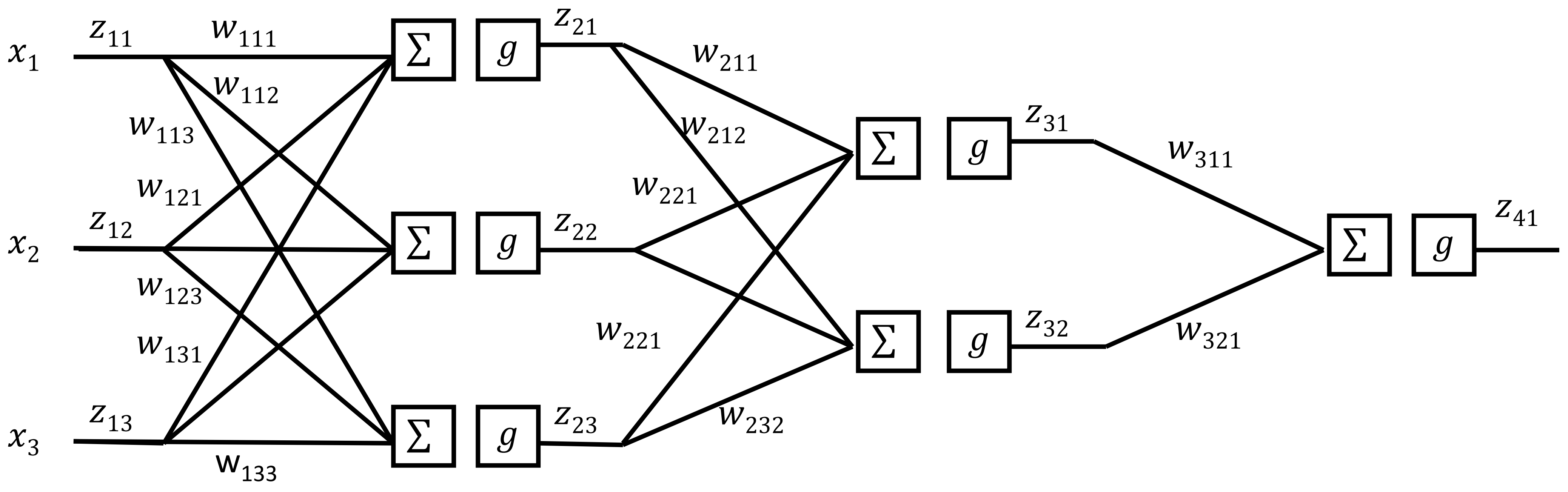
- $\frac{\partial J}{\partial x} \mathrel{+}= \frac{\partial J}{\partial y}\frac{\partial y}{\partial x}$
- $\frac{\partial J}{\partial w} \mathrel{+}= \frac{\partial J}{\partial y}\frac{\partial y}{\partial w}$



x
$\frac{\partial J}{\partial x}$
Layer $y = f(x, w)$
$\frac{\partial y}{\partial x}$
$\frac{\partial y}{\partial w}$
y
$\frac{\partial J}{\partial y}$
w
$\frac{\partial J}{\partial w}$

Neural Network Implementation

Which pieces to we treat as functions?



Neural Network Properties

Neural Networks Properties

Practical considerations

- Large number of neurons
  - Danger for overfitting
- Modelling 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

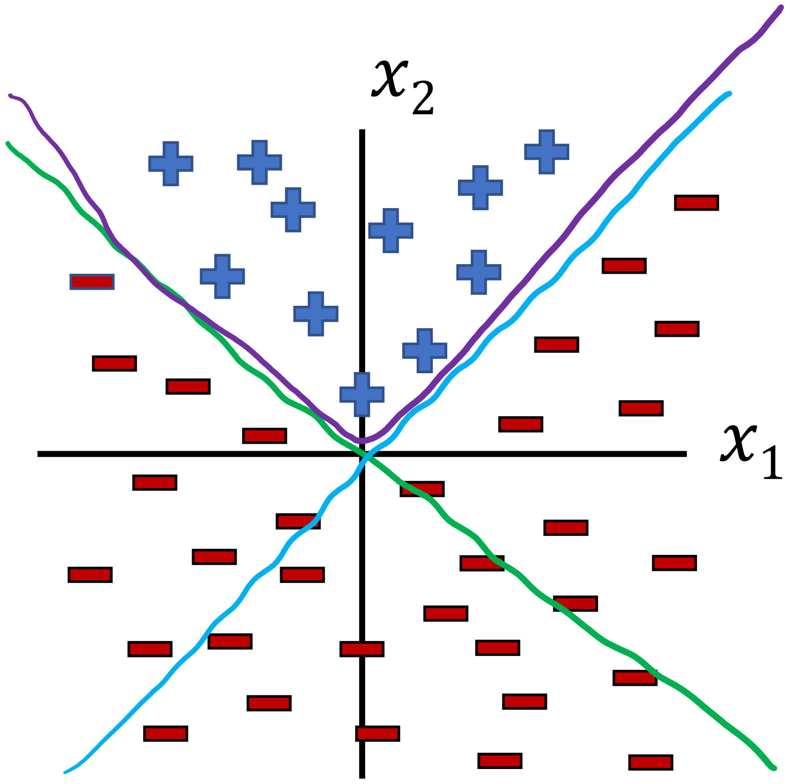
Universal Approximation Theorem:

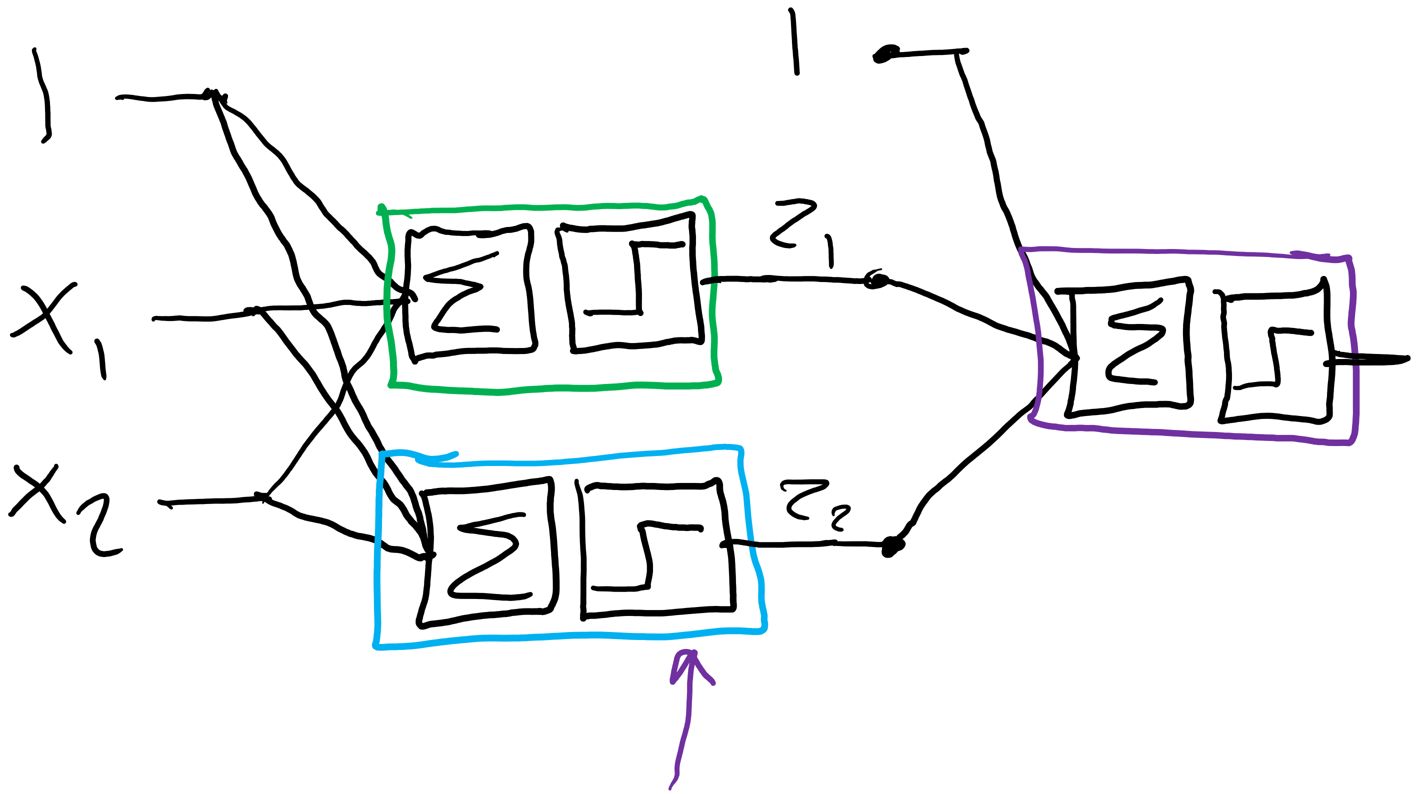
- A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy.

Classification Design Challenge

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

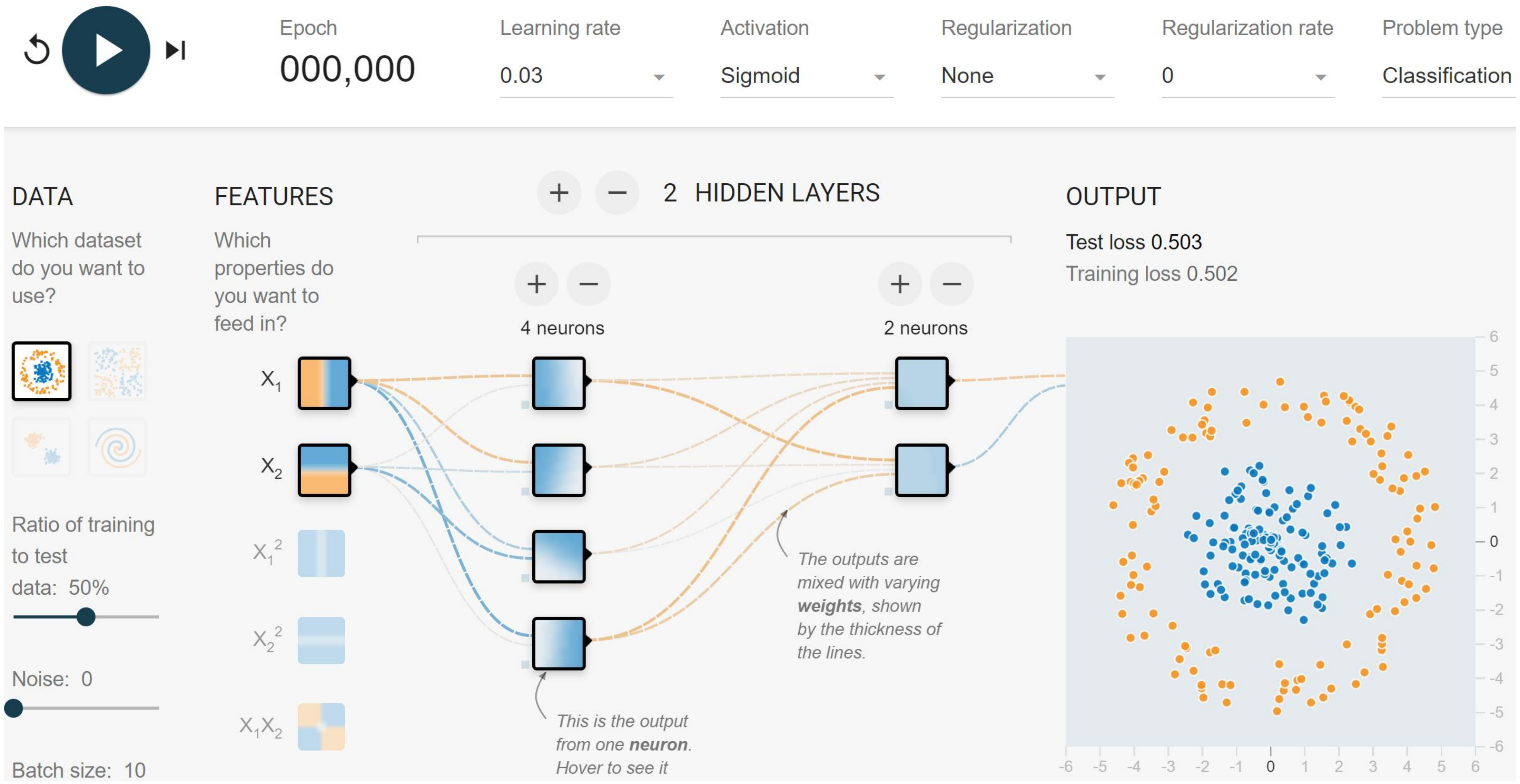
$$h_A(\boldsymbol{x}) = sign(\boldsymbol{w}_A^T\boldsymbol{x} + b_A)$$
$$h_B(\boldsymbol{x}) = sign(\boldsymbol{w}_B^T\boldsymbol{x} + b_B)$$
$$h_C(\underset{z}{\cancel{\boldsymbol{x}}}) = sign(\boldsymbol{w}_C^T\underset{z}{\cancel{\boldsymbol{x}}} + b_C)$$





Network to Approximate Binary Classification

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



Epoch 000,000

Learning rate 0.03

Activation Sigmoid

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

4 neurons

2 neurons

The outputs are mixed with varying ***weights****, shown by the thickness of the lines.*

This is the output from one ***neuron****. Hover to see it*

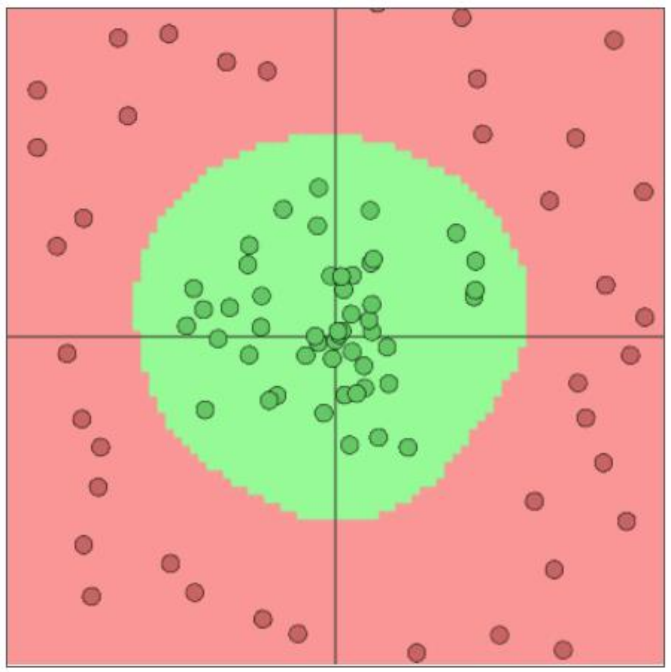
OUTPUT

Test loss 0.503

Training loss 0.502

Network to Approximate Binary Classification

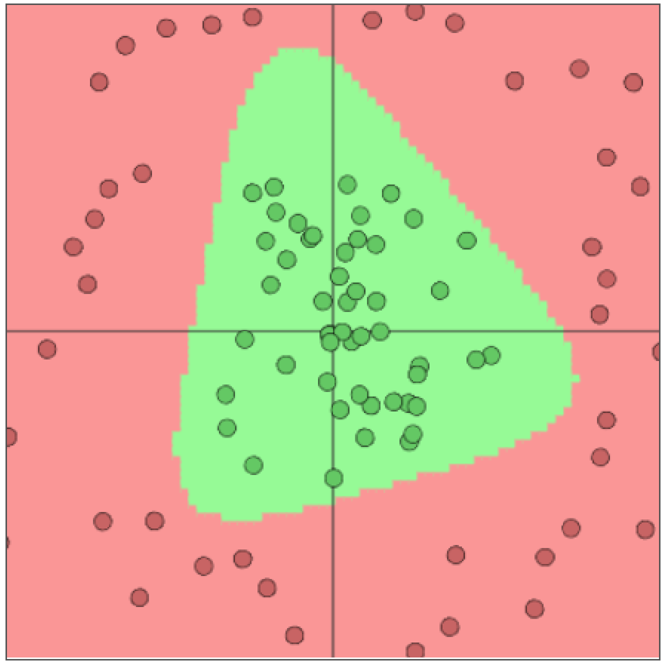
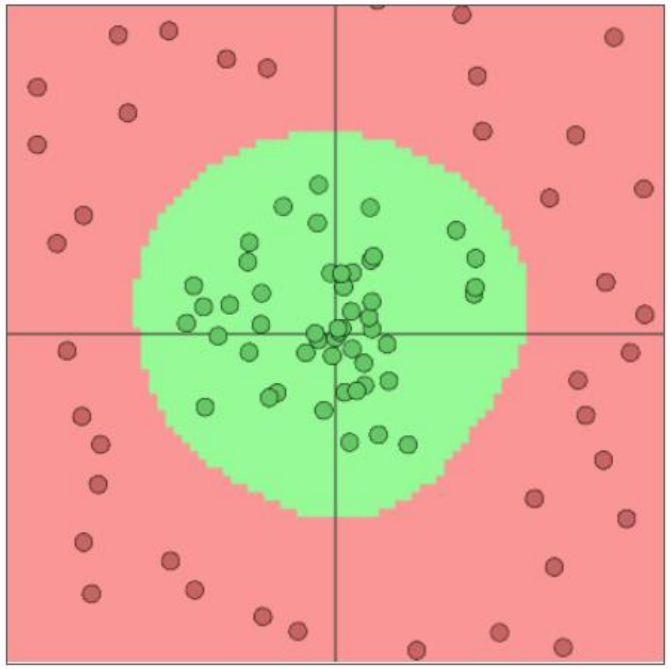
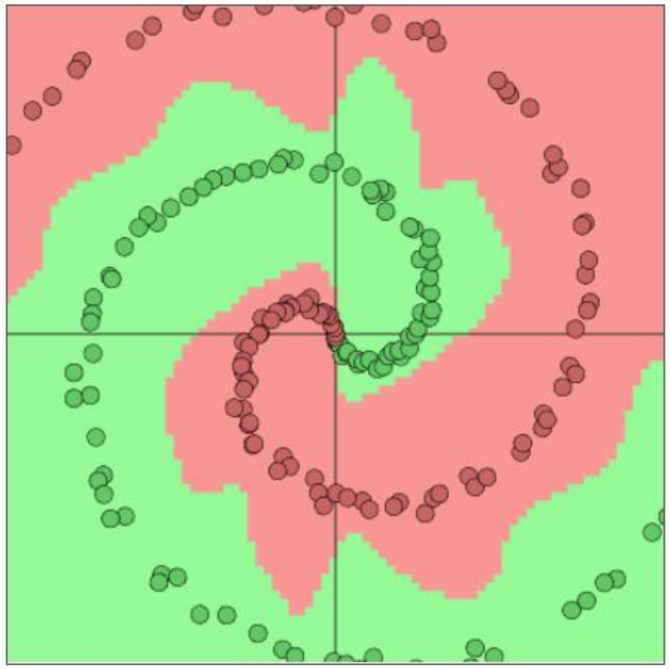
Approximate arbitrary decision boundary



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

Network to Approximate Binary Classification

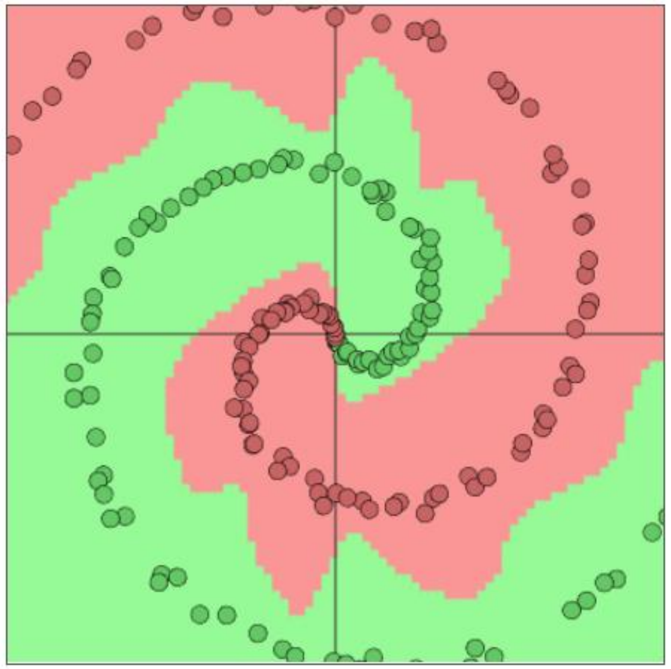
Approximate arbitrary decision boundary

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

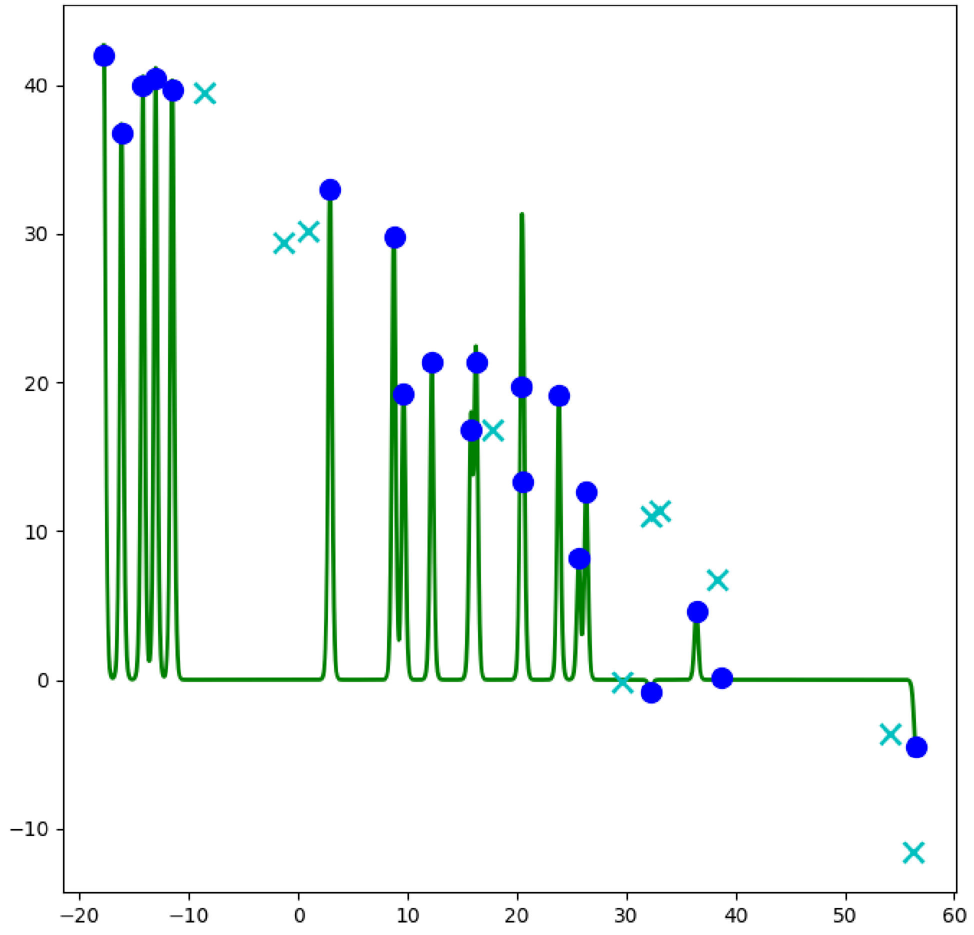
Network to Approximate Binary Classification

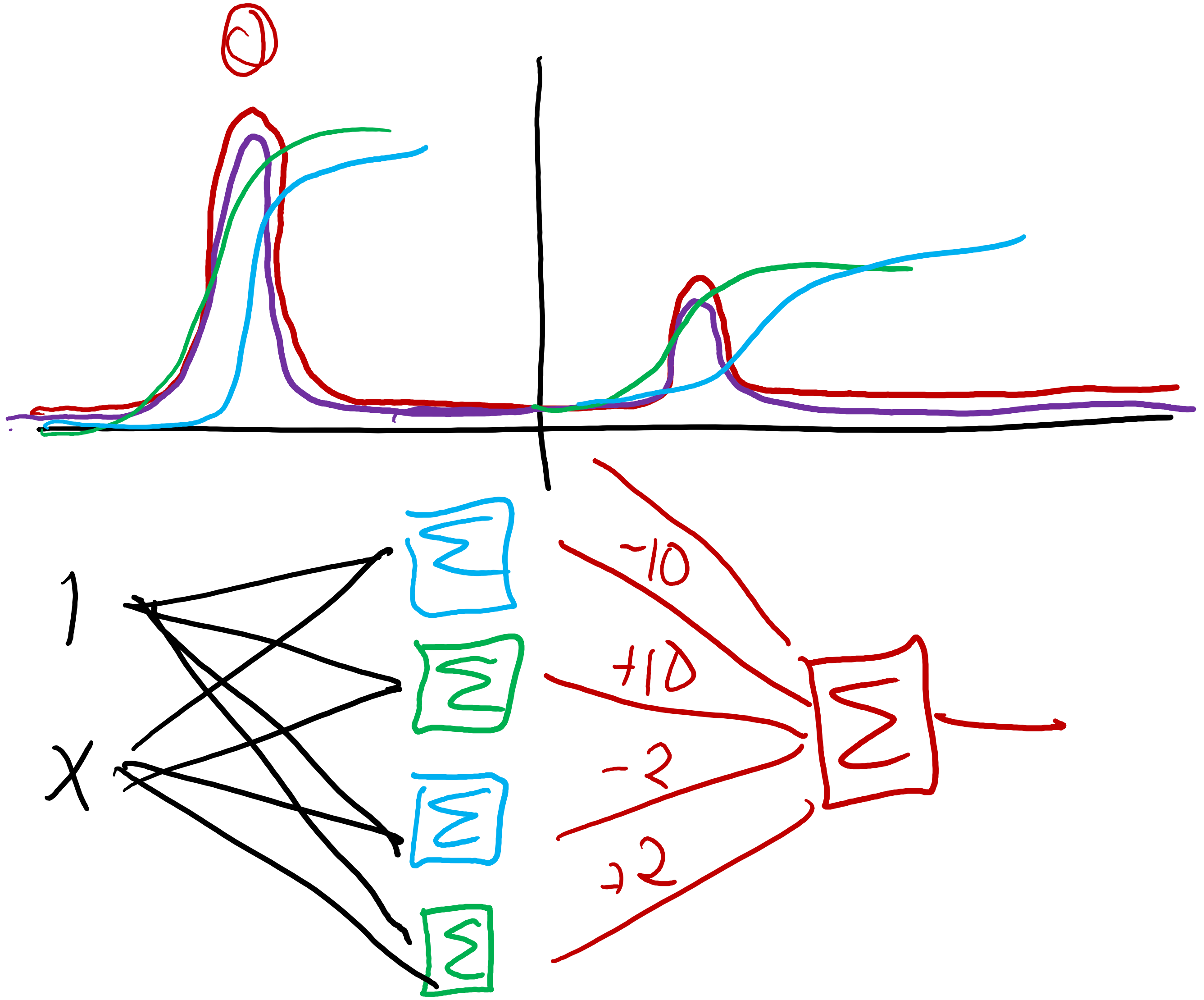
Approximate arbitrary decision boundary



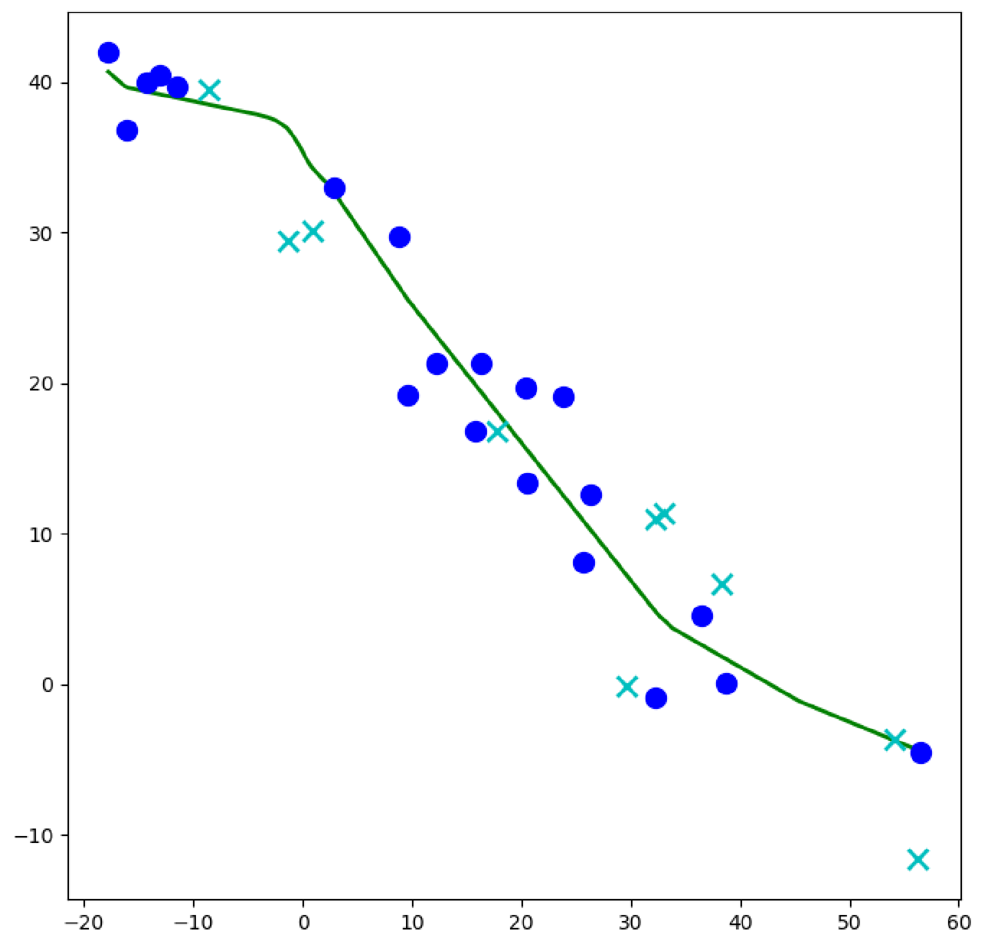
https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

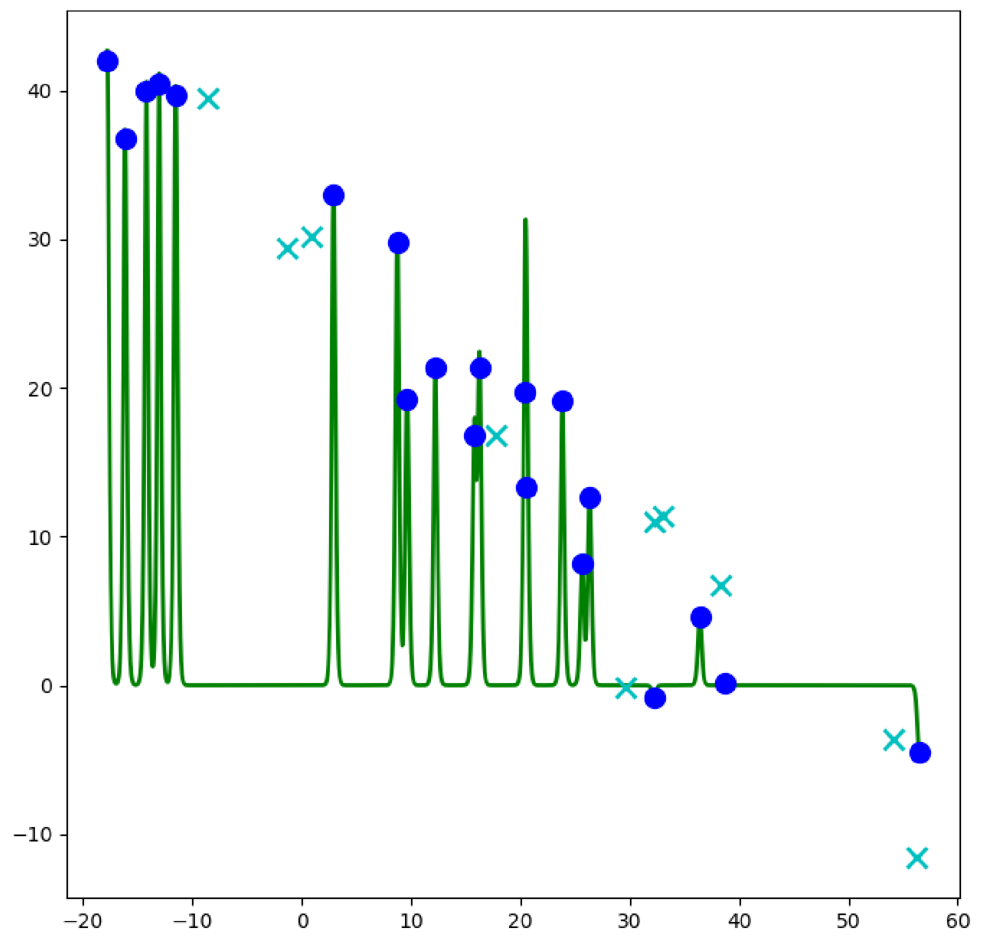
Network to Approximate a 1-D Function





Network to Approximate a 1-D Function





Objective Function is Not Convex

Objective function for…	Convex?	Closed-form solution?
Linear regression		
Logistic regression		
Neural networks		

Optimization

Linear function

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

- $f(\boldsymbol{x}+\boldsymbol{z}) = f(\boldsymbol{x}) + f(\boldsymbol{z})$
- $f(\alpha\boldsymbol{x}) = \alpha f(\boldsymbol{x}) \quad \forall\alpha$
- $f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{z}) = \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{z}) \quad \forall\alpha$

Optimization

Linear function

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

- $f(\boldsymbol{x}+\boldsymbol{z}) = f(\boldsymbol{x}) + f(\boldsymbol{z})$
- $f(\alpha\boldsymbol{x}) = \alpha f(\boldsymbol{x}) \quad \forall\alpha$
- $f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{z}) = \alpha f(\boldsymbol{x}) + (1-\alpha)f(\boldsymbol{z}) \quad \forall\alpha$

Optimization

Convex function

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

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

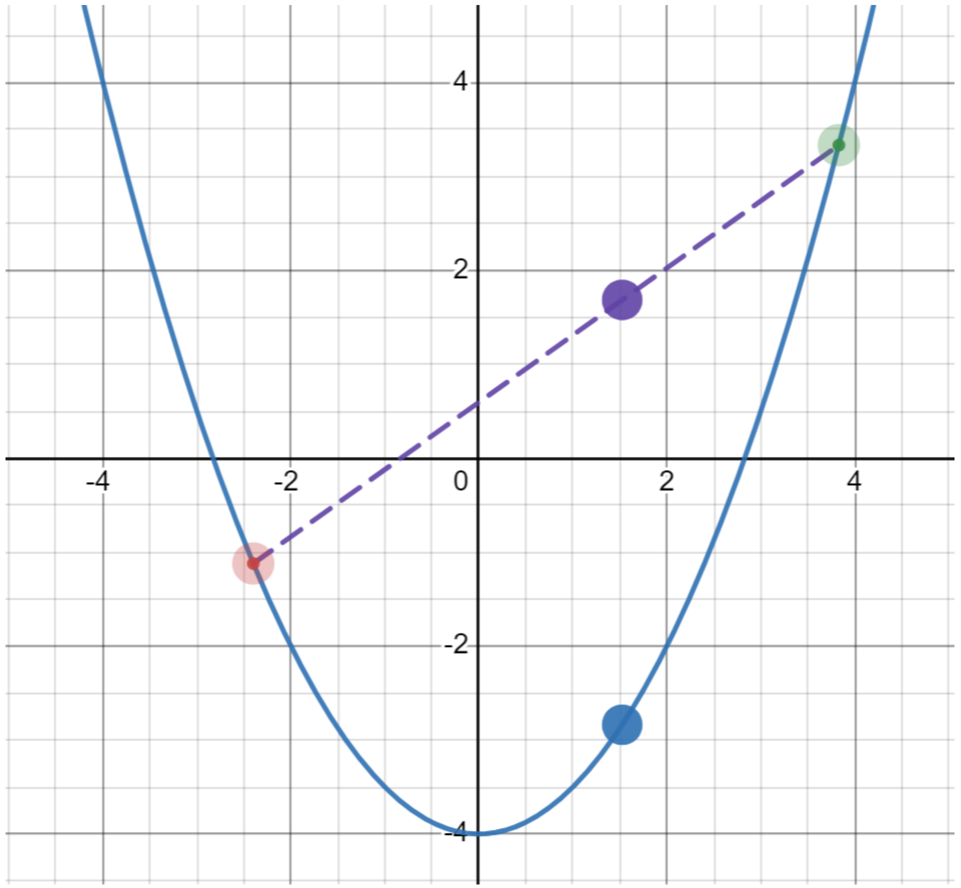
Optimization

Convex function

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

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

Demo on Desmos



Optimization

Convex function

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

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

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 ☺