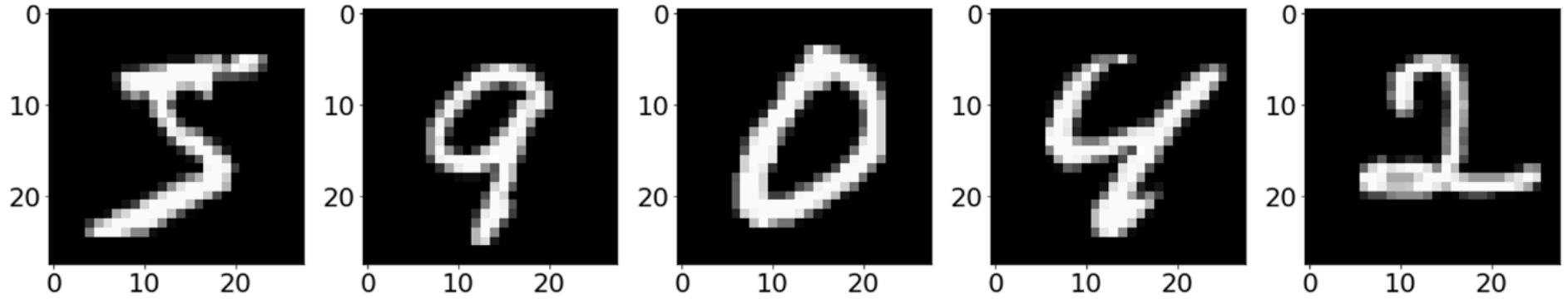
Machine Learning and Data Mining

Lecture 6.1: Neural Networks

CS 434

Homework 3 Out

- Some proofs about Naïve Bayes
- Finishing implementing a simple neural network library
- Classifying hand-written digits with neural networks and seeing the effect of hyperparameter choices



RECAP

From Last Lecture

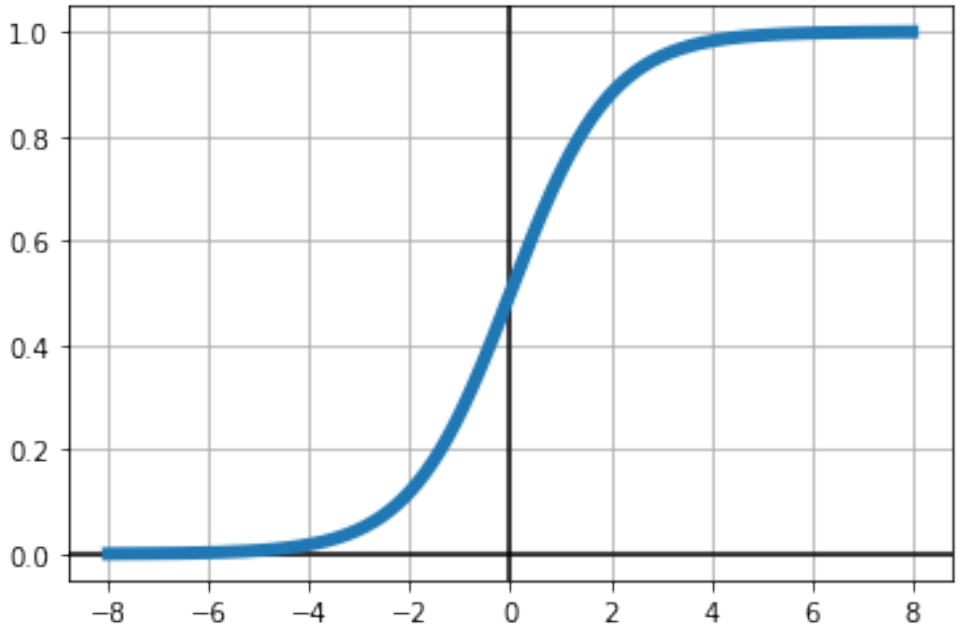
Introducing the logistic function:

- May also see it referred to as a sigmoid function or "logit" function

Logistic Function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Maps $(-\infty, \infty)$ to $(0,1)$

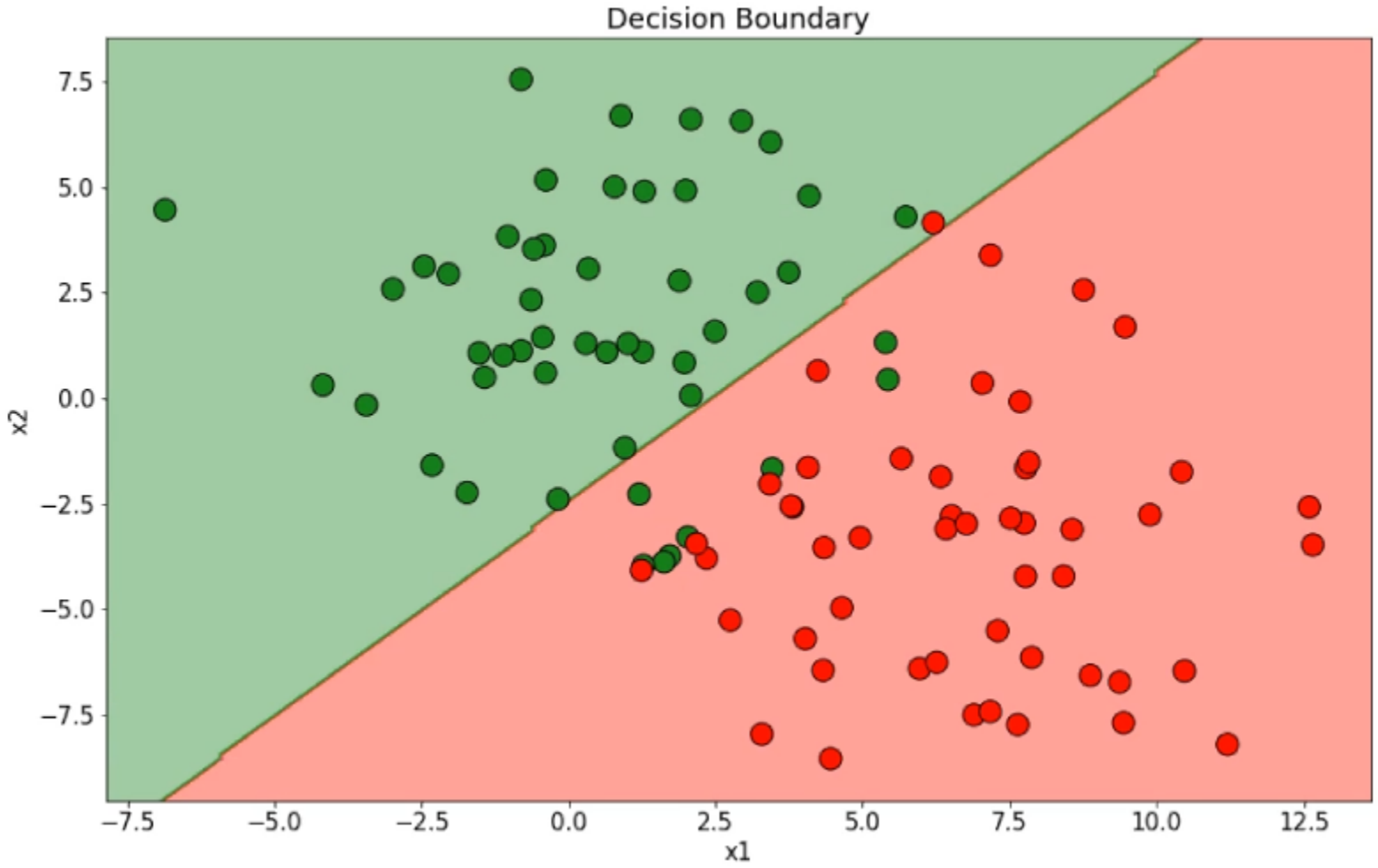


Logistic Regression Model Assumption:

$$P(y_i = 1|\boldsymbol{x_i}; \boldsymbol{w}) = \sigma(\boldsymbol{w^T x_i}) = \frac{1}{1 + e^{-\boldsymbol{w^T x_i}}}$$

$$P(y_i = 0|\boldsymbol{x_i}; \boldsymbol{w}) = 1 - \sigma(\boldsymbol{w^T x_i}) = \frac{e^{-\boldsymbol{w^T x_i}}}{1 + e^{-\boldsymbol{w^T x_i}}}$$

Example logistic regression decision boundary – linear in the features.



An Aside to Understand Gradient Descent

Different starting points may lead to different local minima if the function being optimized is non-convex.

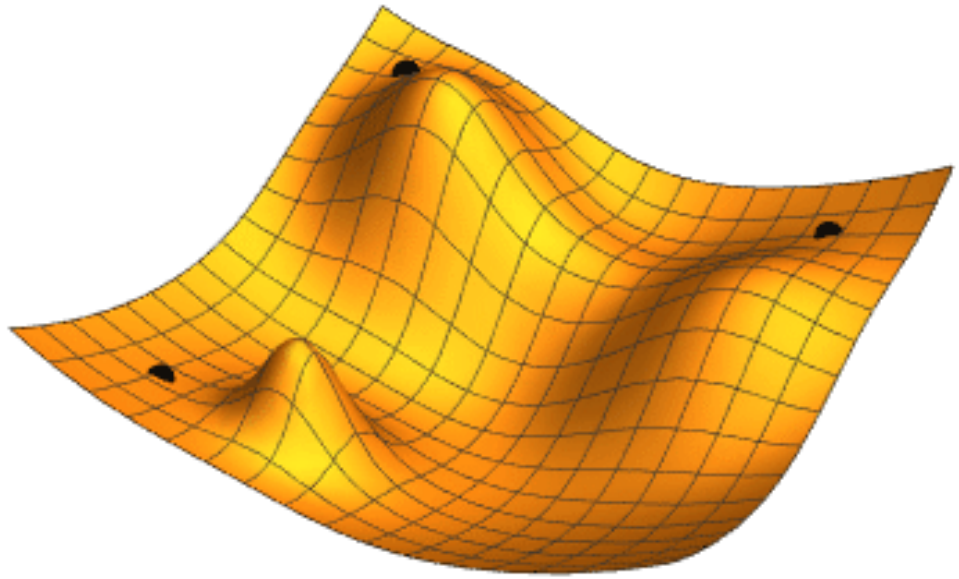
Gradient Descent Algorithm

$w \leftarrow random(\quad)$

while$|\nabla_w \mathcal{L}| > \epsilon$ *or iters remain*

$w = w - \alpha \nabla_w \mathcal{L}(w)$

Learning Rate / Step Size



So we have an expression for the gradient but can't find the optimal analytically…

$$\nabla - logP(D \mid w) = \sum_{i=1}^{N} \nabla l_i = \sum_{i=1}^{N} \left(\sigma\left(\boldsymbol{w^T x_i}\right) - y_i\right)\mathbf{x_i}$$

Gradient Descent Algorithm for Logistic Regression

w = random(d) // randomly initialize weight vector

Repeat:

for each example $x_i, y_i \in D$: // compute gradient

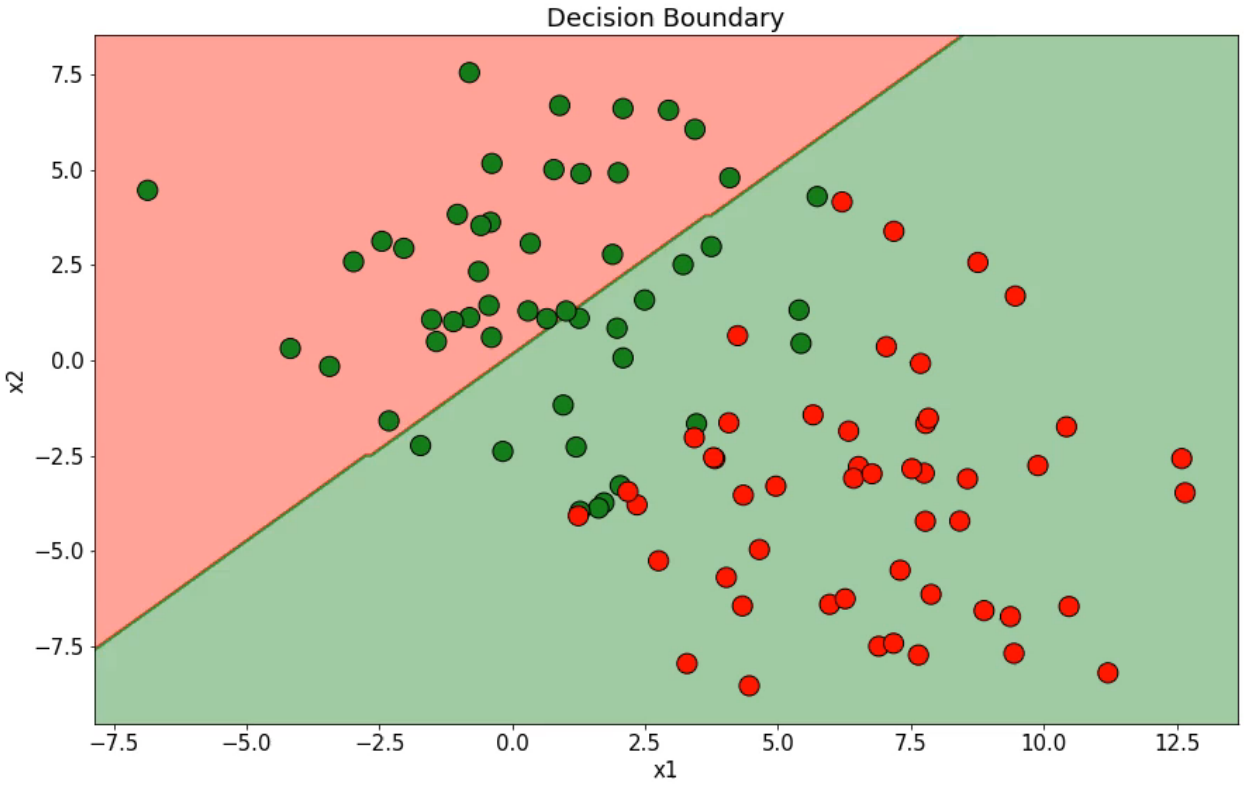
$\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \frac{1}{1+e^{-\boldsymbol{w^T x_i}}}$ // compute the prediction for this point

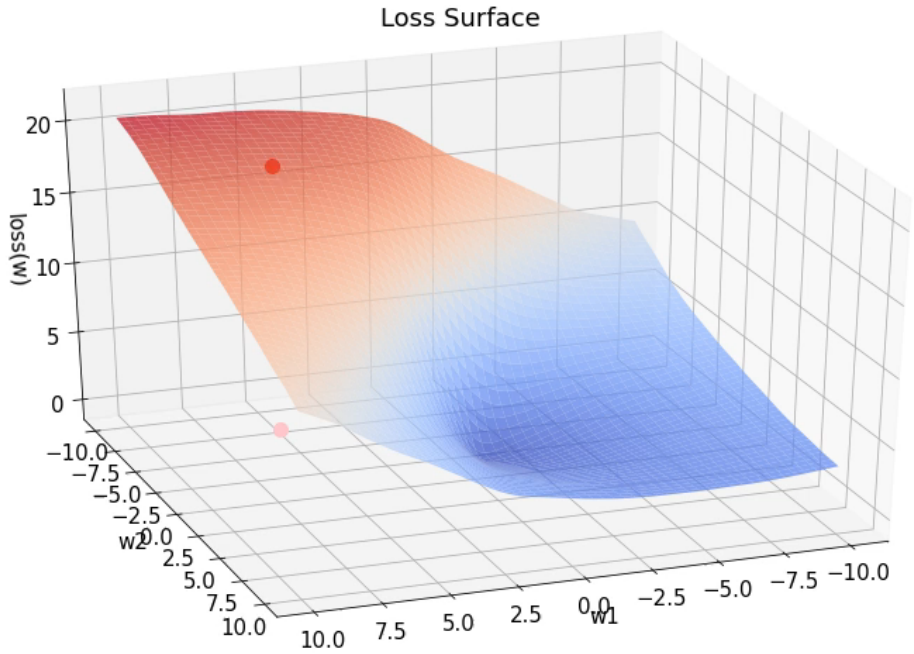
$\boldsymbol{\nabla} \mathrel{+}= (\hat{y}_i - y_i)\boldsymbol{x_i}$ // compute gradient of loss of point and sum

$\mathbf{w} = \mathbf{w} - \alpha\boldsymbol{\nabla}$ // take a step in the opposite direction of gradient

Until $|\nabla| \leq \epsilon$ // keep taking steps until convergence

An example of Logistic Regression with Gradient Descent





Today's Learning Objectives

Be able to answer:

- What is a neuron?
  - How does it relate to logistic regression?
  - What are common activation functions?
- What is fully-connected neural network?
- Build some intuition for training

New Topic: Neural Networks

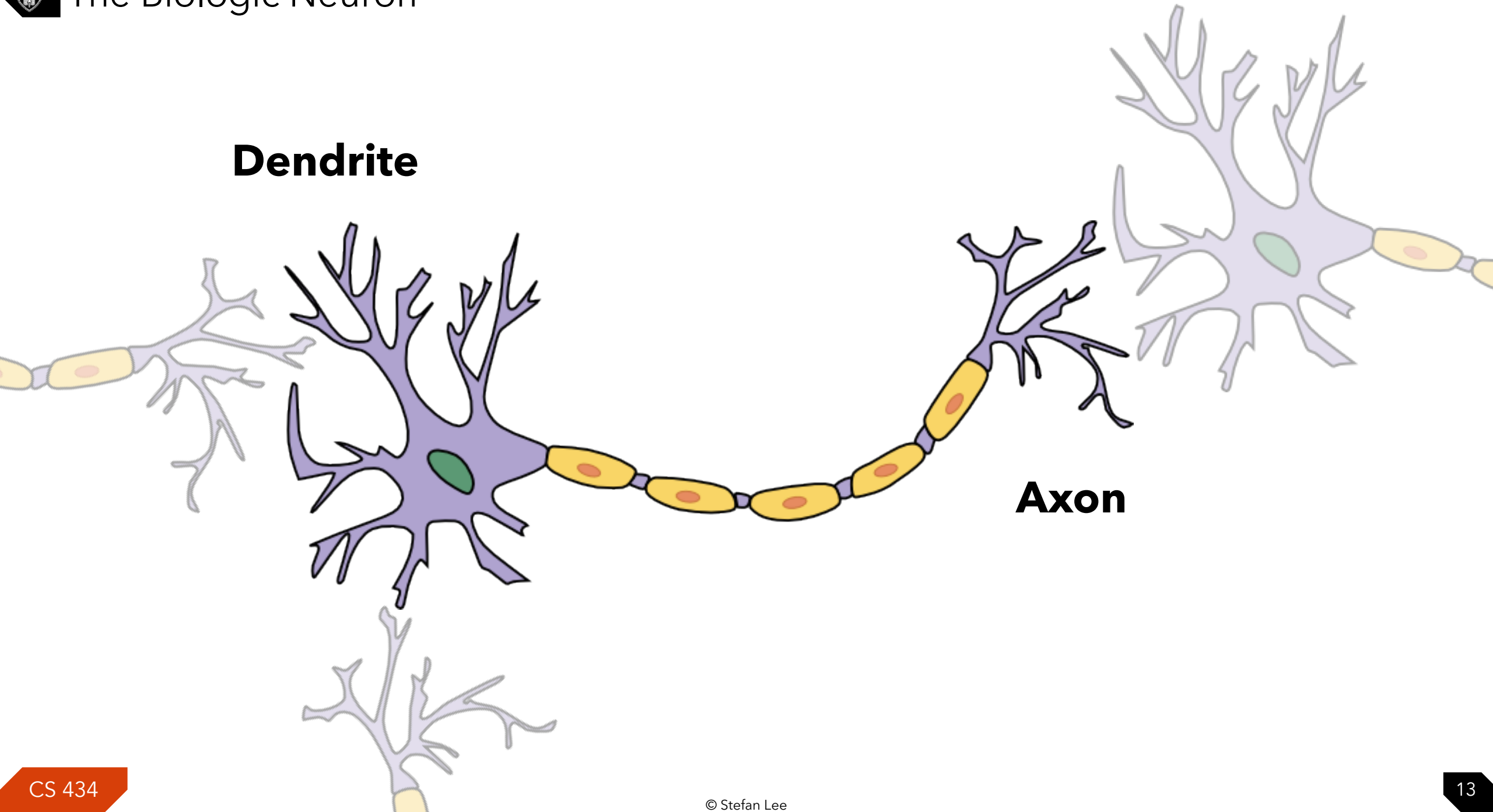
Neural networks are a class of biologically-inspired models for classification and regression

- **Key Concepts:**
  - Artificial Neurons
    - Activation Functions
  - Neural Networks
    - Linear Layers
    - Universal Approximators (in the limit)
  - Training Neural Networks
    - Backpropagation
    - Stochastic Gradient Descent

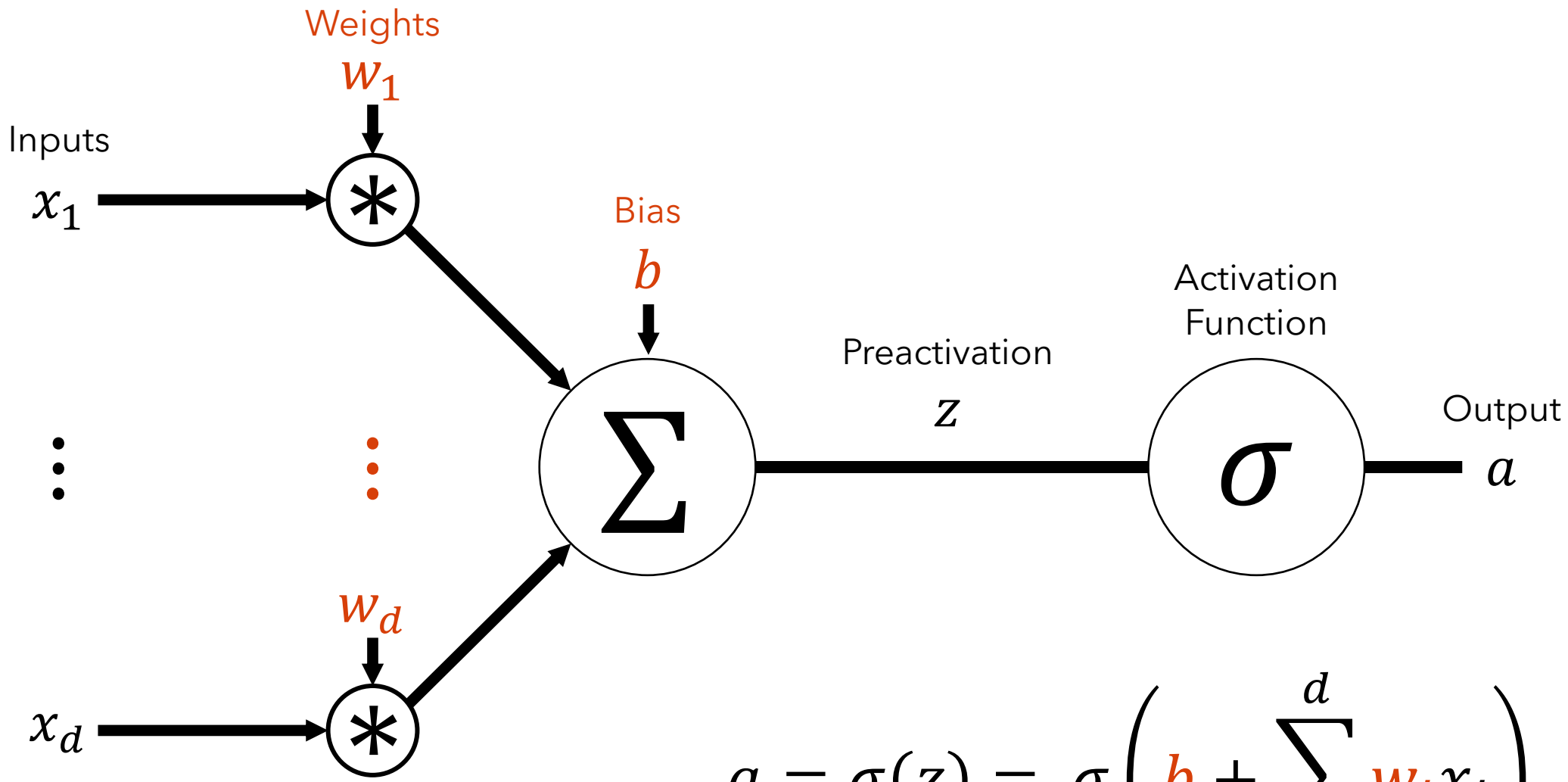
Demo

https://playground.tensorflow.org/

The Biologic Neuron

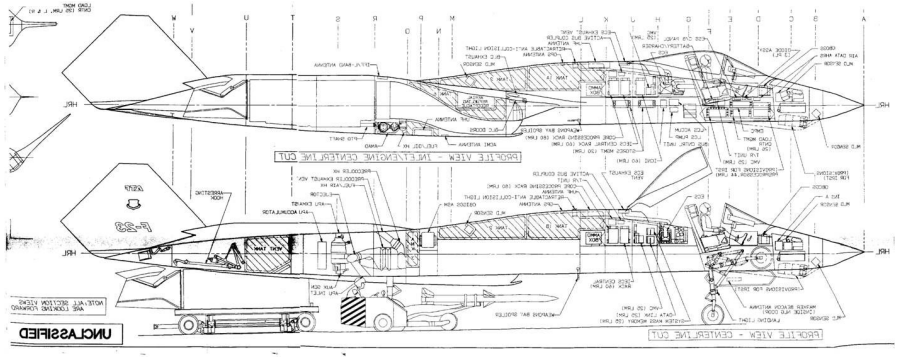


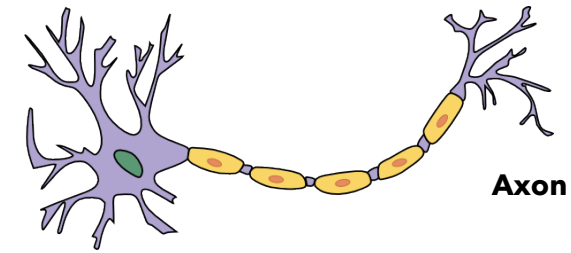
The Artificial Neuron

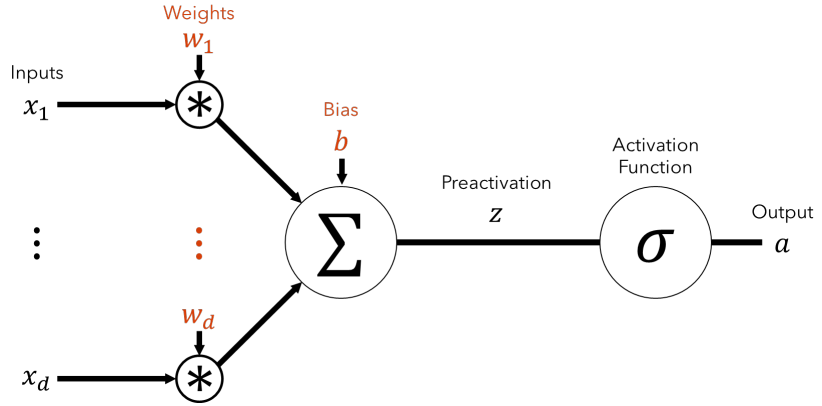


$$a = \sigma(z) = \sigma\left(b + \sum_{i=1}^{d} w_i x_i\right)$$

This is a *very* weak analogy. Artificial Neurons are not like Neurons.







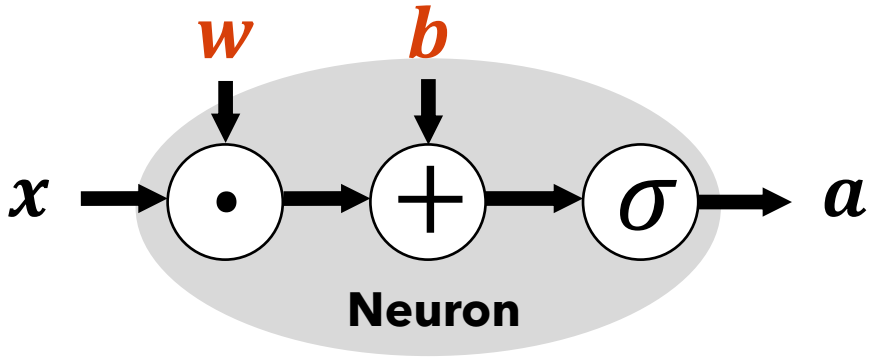
The Humble Neuron Vectorized

$$a = \sigma\left(b + \sum_{i=1}^{d} w_i x_i\right) = \sigma(b + \boldsymbol{w}^T \boldsymbol{x})$$

$$\underset{1\times1}{\boldsymbol{a}} \quad \begin{bmatrix} a \end{bmatrix} = \sigma\left(\begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}^{\boldsymbol{T}} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} + \underset{1\times1}{\boldsymbol{b}} \begin{bmatrix} b \end{bmatrix} \right)$$

$\boldsymbol{w}$ $d\times1$ $\quad$ $\boldsymbol{x}$ $d\times1$

Neuron



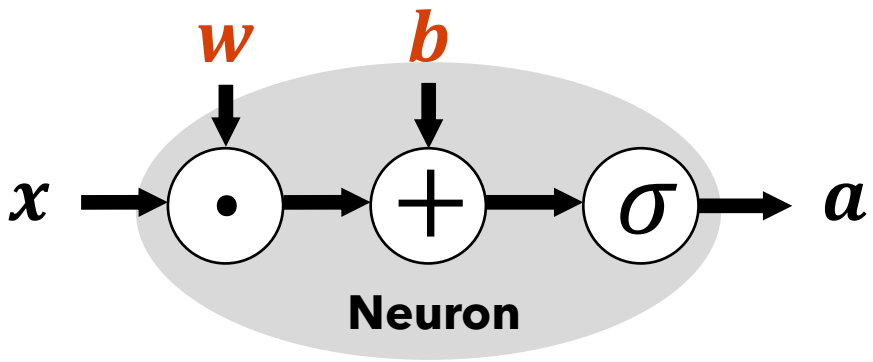
Neuron is a linear function of its input followed by a (typically) non-linear activation function to produce output.

Hyperparameters : activation function (σ)

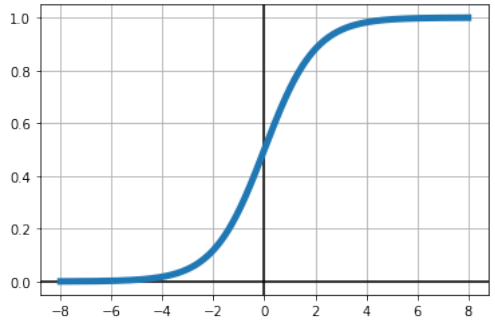
Learnable Parameters: $\mathbf{w} \in \mathbb{R}^d$ and $\mathbf{b} \in \mathbb{R}$

Question Break!

What is another name for a neuron with a sigmoid activation function?



$$a = \sigma(b + \boldsymbol{w}^T \boldsymbol{x})$$



Sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$

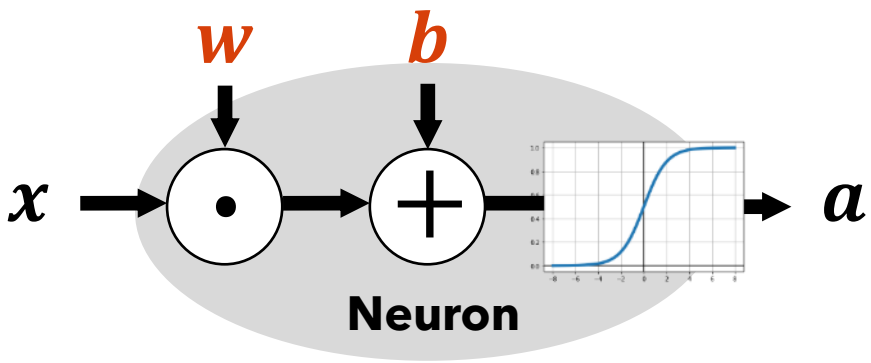
A Support Vector Machine

B Perceptron

C Logistic Regression

D Neural Network

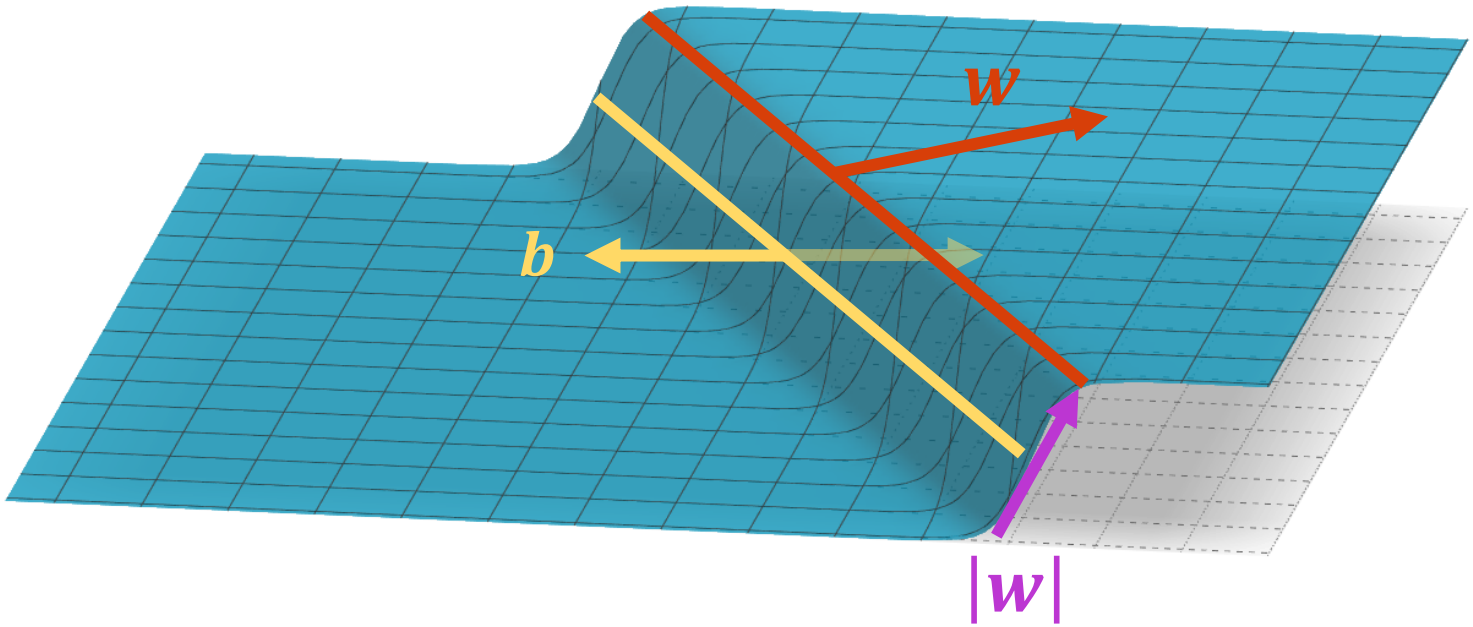
What can one neuron do?



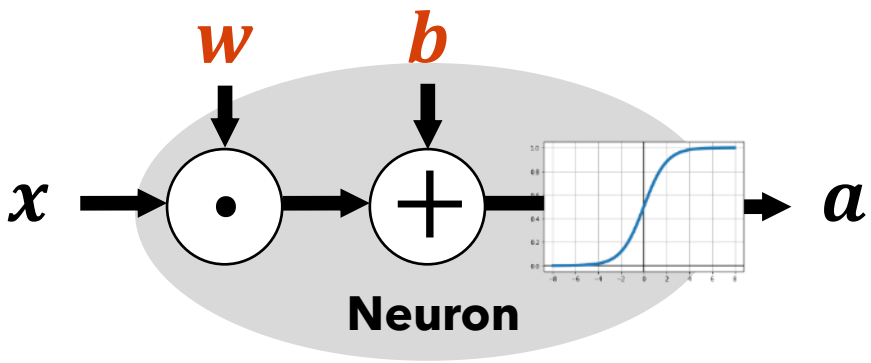
$$\sigma(b + \boldsymbol{w}^T \boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{w}^T \boldsymbol{x} + b)}}$$

$$\boldsymbol{w} = [10, 5]^T$$

$$b = -1$$



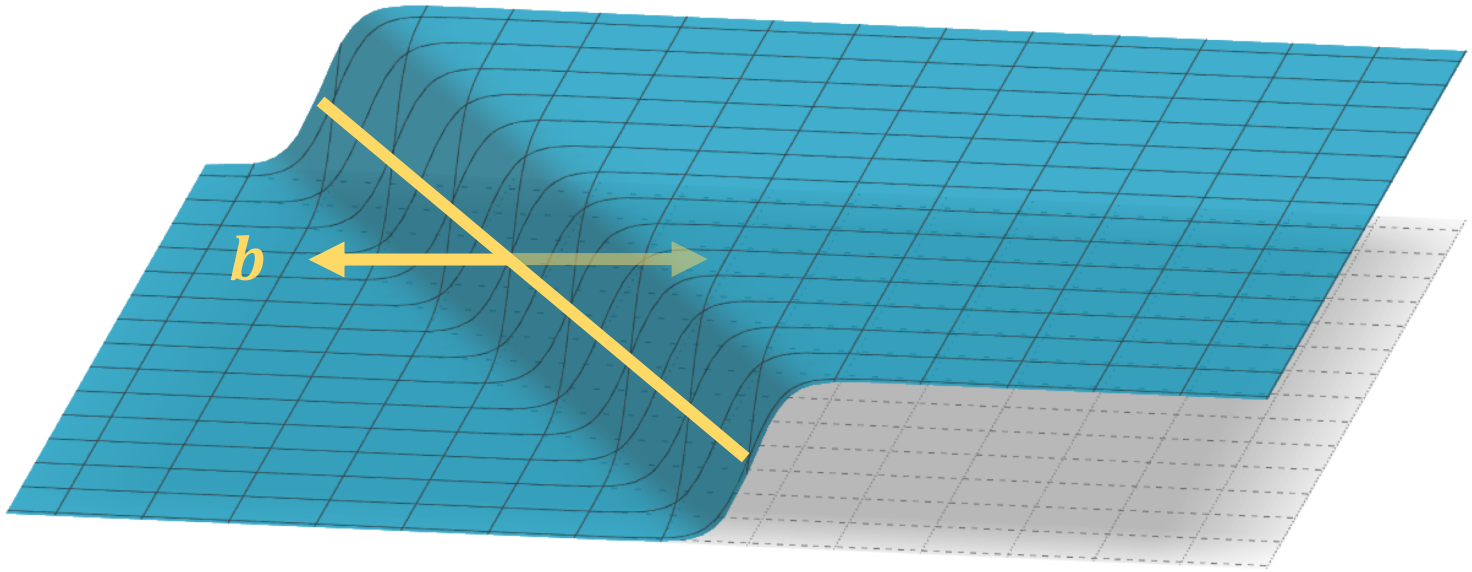
What can one neuron do?



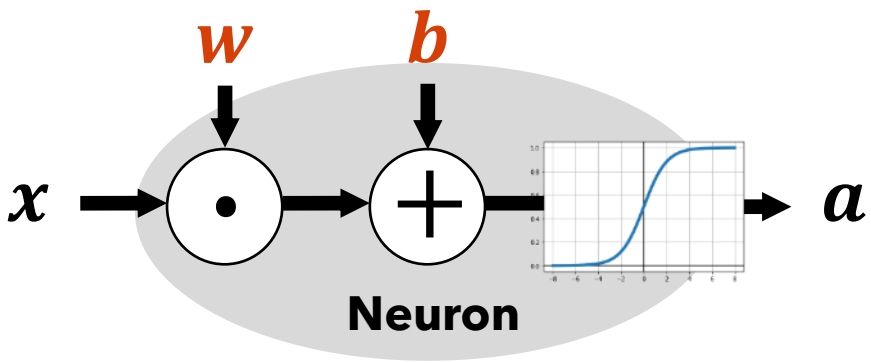
$$\sigma(b + \boldsymbol{w}^T \boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{w}^T \boldsymbol{x} + b)}}$$

$$\boldsymbol{w} = [10, 5]^T$$

$$b = 15$$



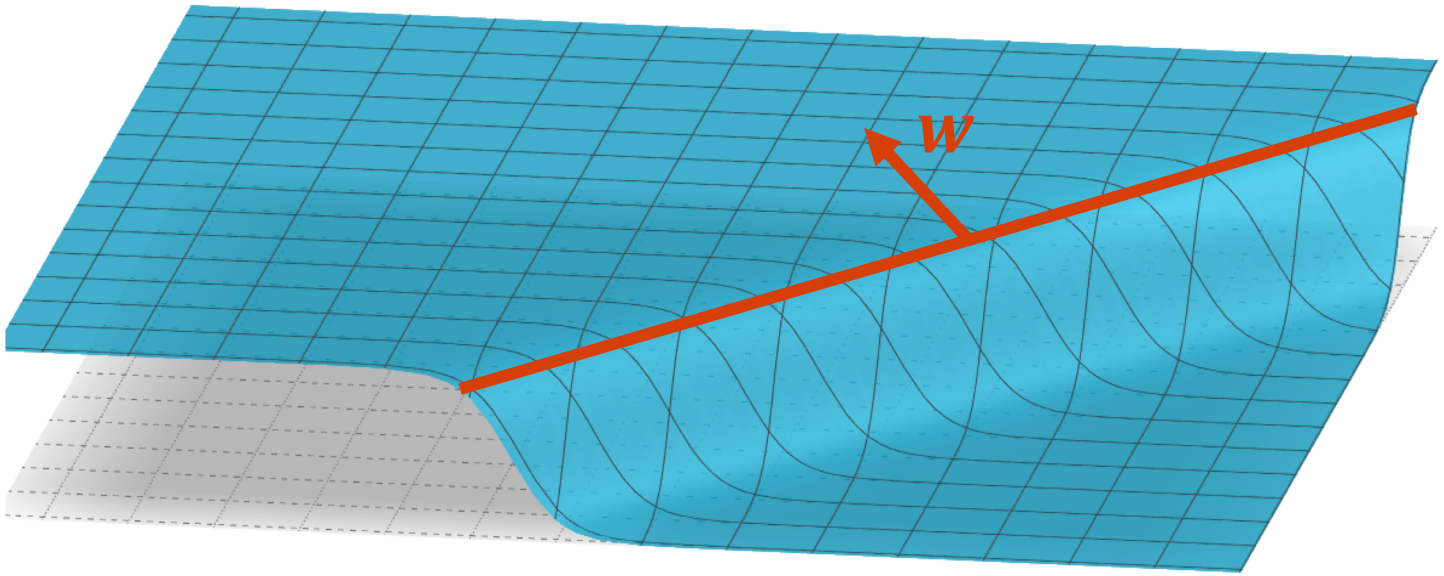
What can one neuron do?



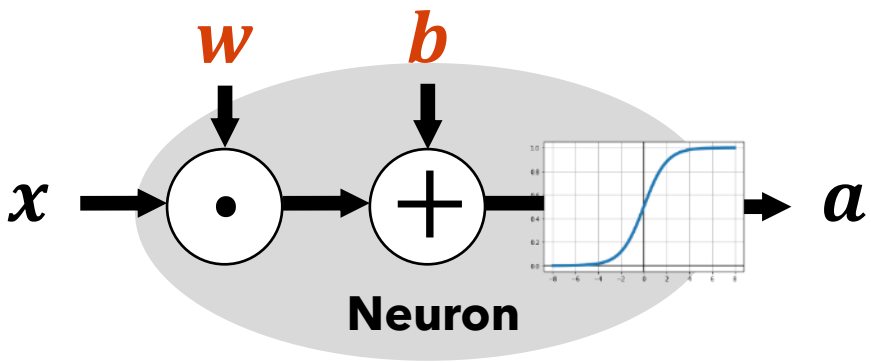
$$\sigma(b + \boldsymbol{w}^T \boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{w}^T \boldsymbol{x} + b)}}$$

$$\boldsymbol{w} = [-7, 5]^T$$

$$b = 15$$



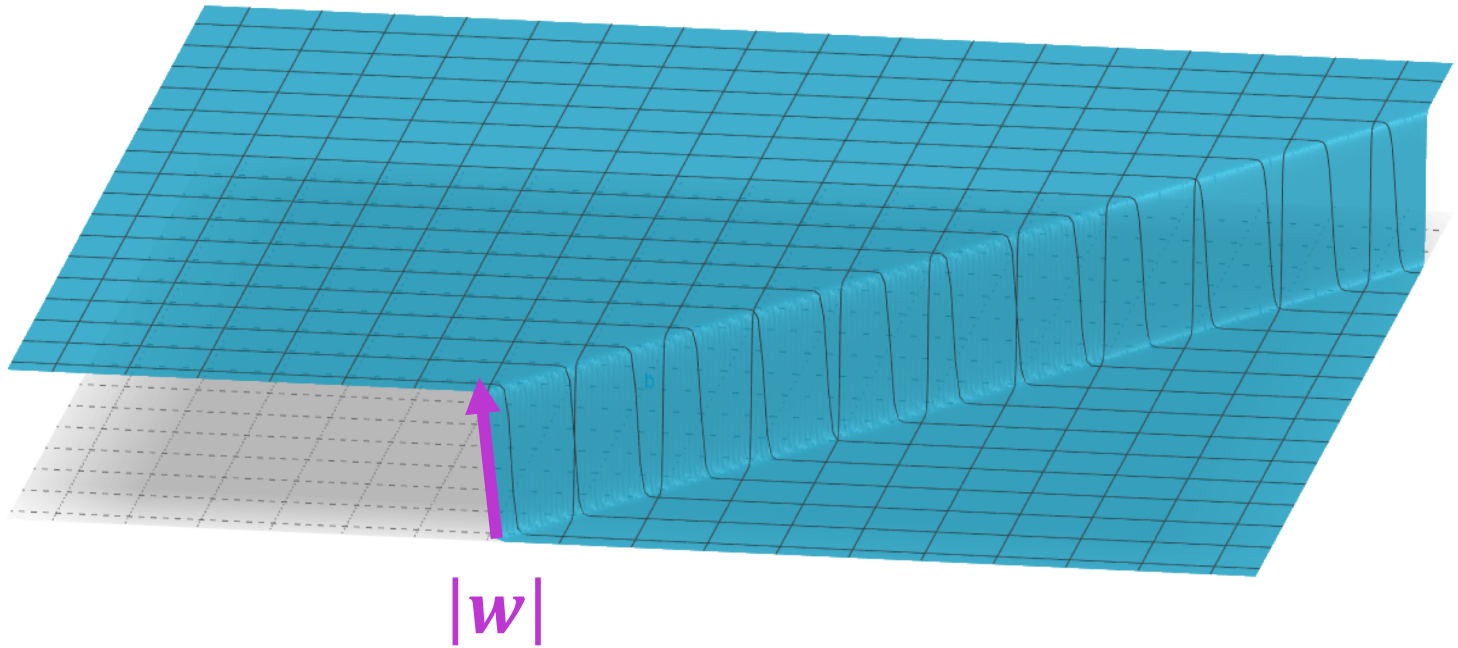
What can one neuron do?



$$\sigma(b + \boldsymbol{w}^T \boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{w}^T \boldsymbol{x} + b)}}$$

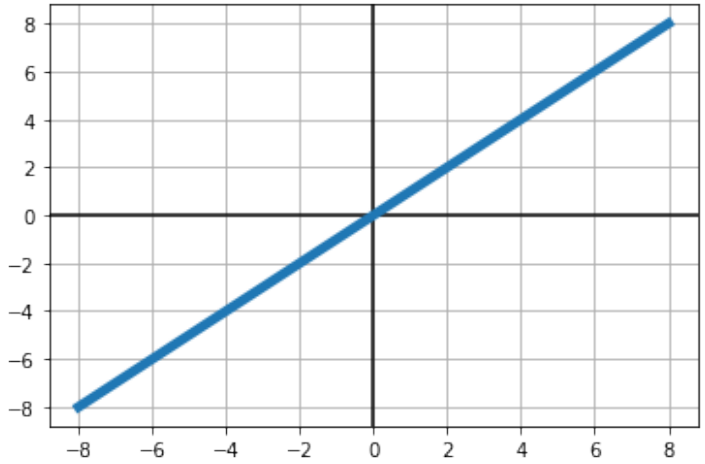
$$\boldsymbol{w} = [-70, 50]^T$$

$$b = 150$$

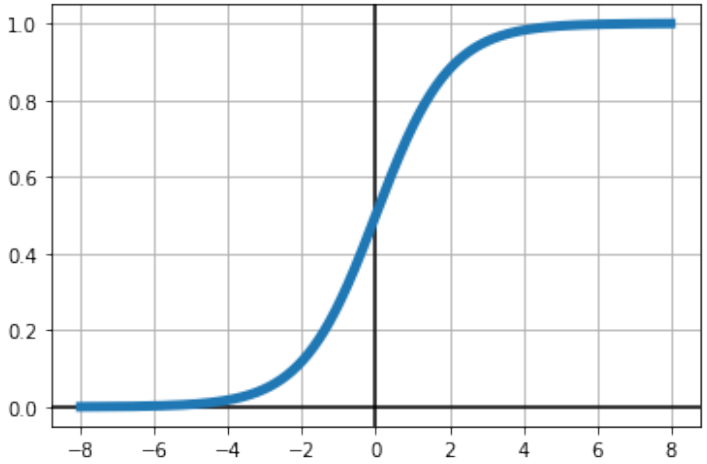


Activation Functions σ

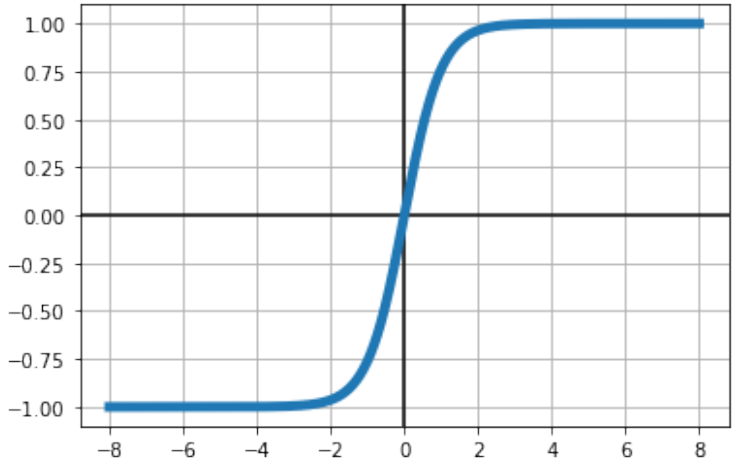
None $\sigma(z) = z$



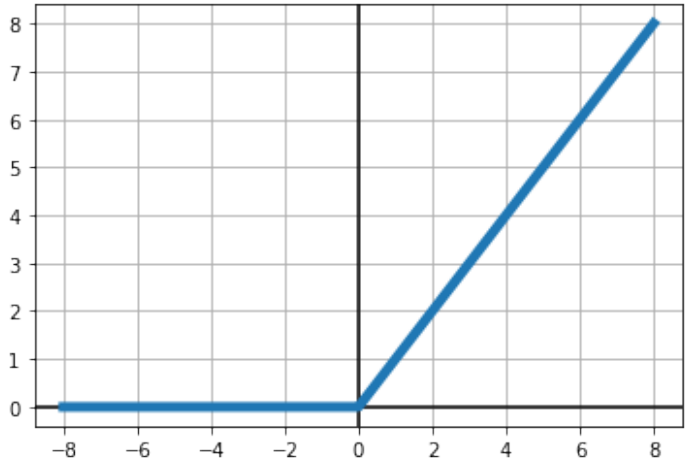
Sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$



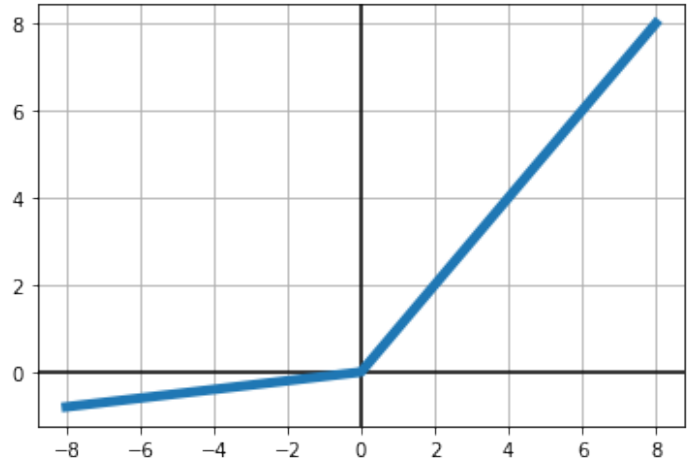
tanh $\sigma(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$



ReLU $\sigma(z) = \max(0, z)$

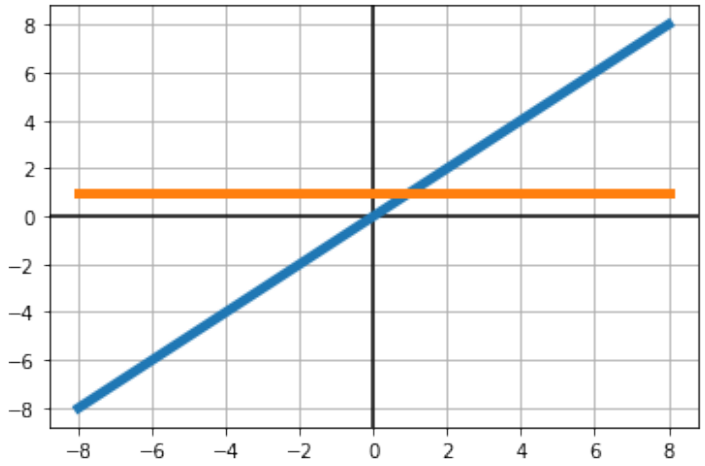


Leaky ReLU $\sigma(z) = \begin{cases} \alpha z & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$

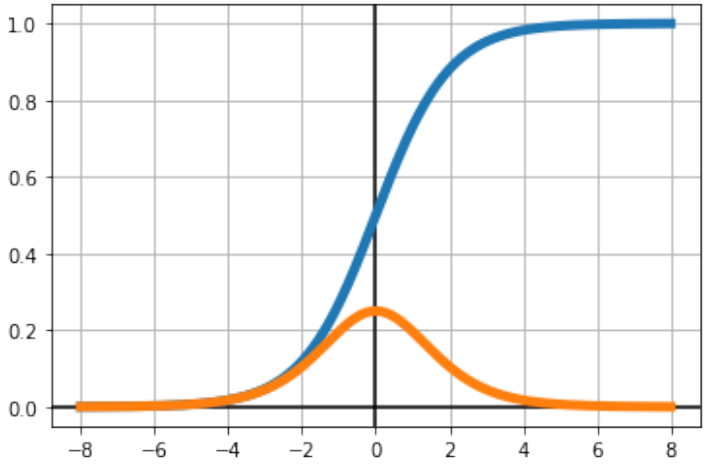


Activation Functions σ

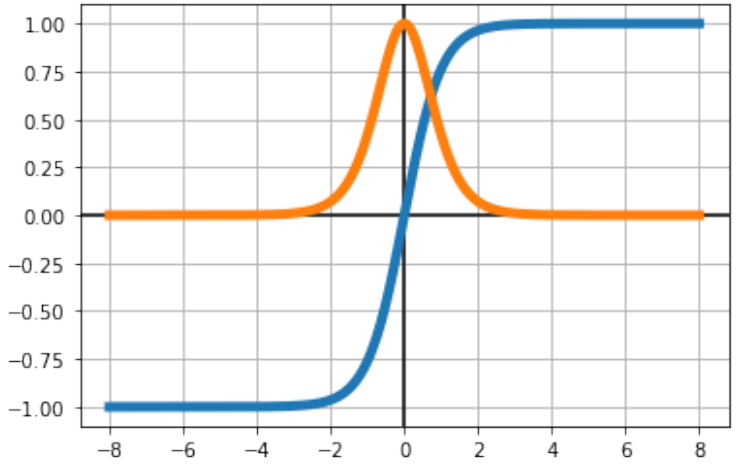
None $\sigma(z) = z$



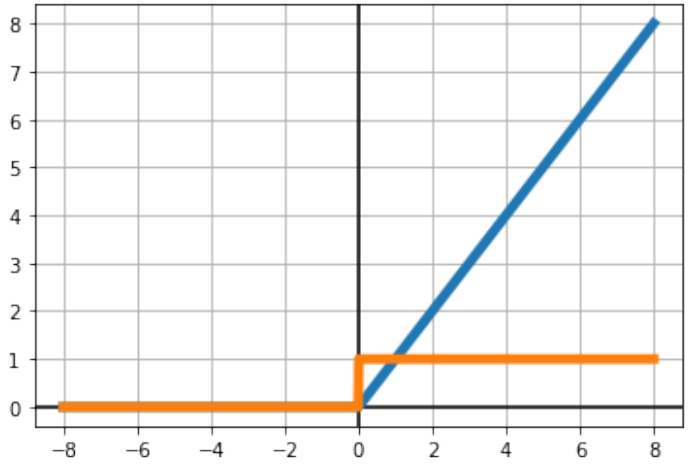
Sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$



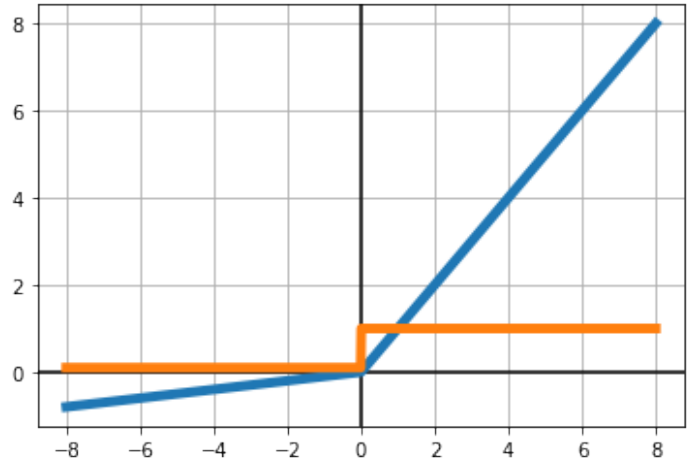
tanh $\sigma(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$



ReLU $\sigma(z) = \max(0, z)$



Leaky ReLU $\sigma(z) = \begin{cases} \alpha z & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$



Questions Break!

Consider the following activations, at what value(s) do they output 0?

None $\sigma(z) = z$

Sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$

tanh $\sigma(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$

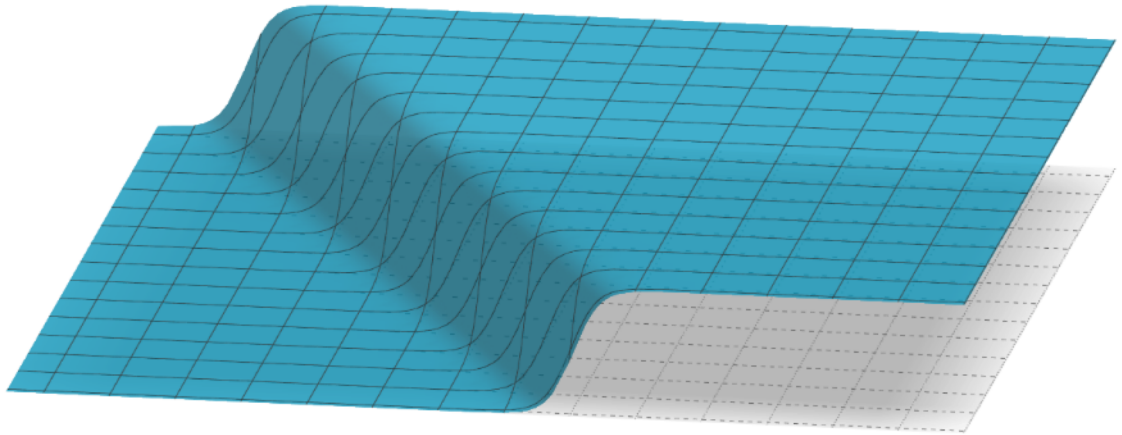
ReLU $\sigma(z) = \max(0, z)$

Leaky ReLU $\sigma(z) = \begin{cases} \alpha z & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$

Questions Break!

Sigmoid Neuron

$$\sigma(x; w, b) = \frac{1}{1 + e^{-w^T x + b}}$$



ReLU Neuron

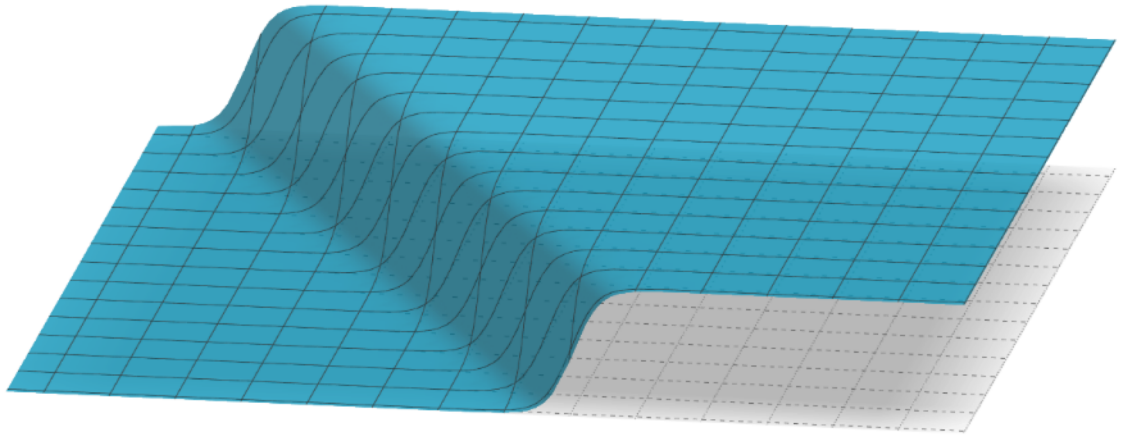
$$\sigma(x; w), b = \max(0, w^T x + b)$$

?

Questions Break!

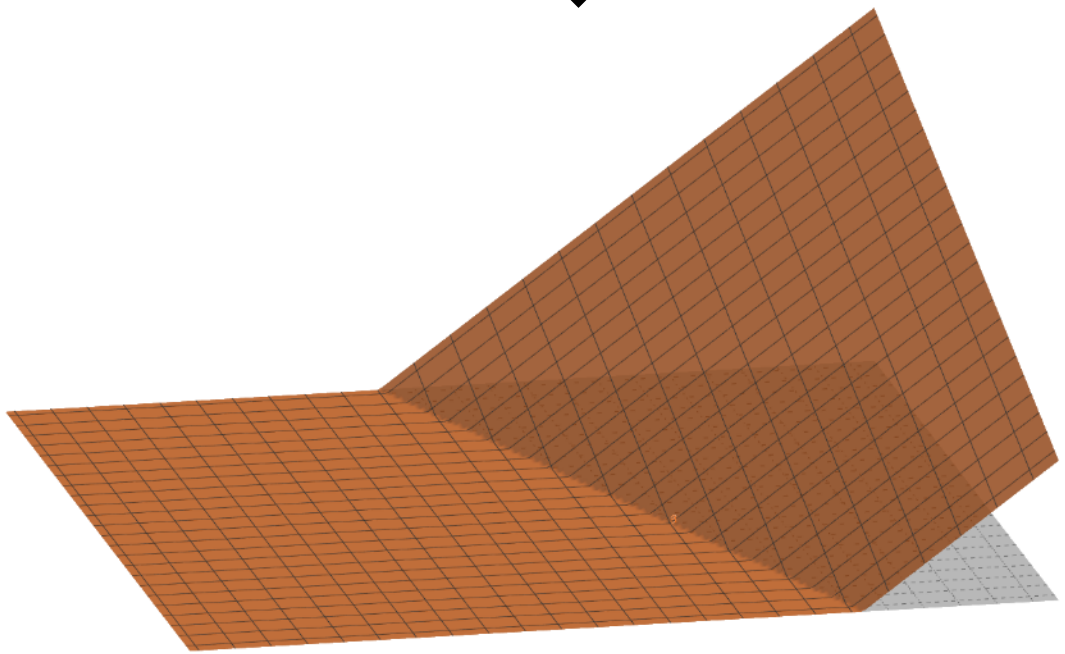
Sigmoid Neuron

$$\sigma(x; w, b) = \frac{1}{1 + e^{-w^T x + b}}$$

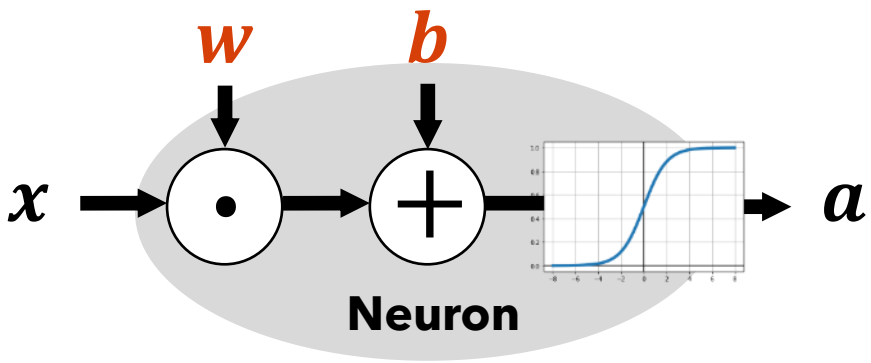


ReLU Neuron

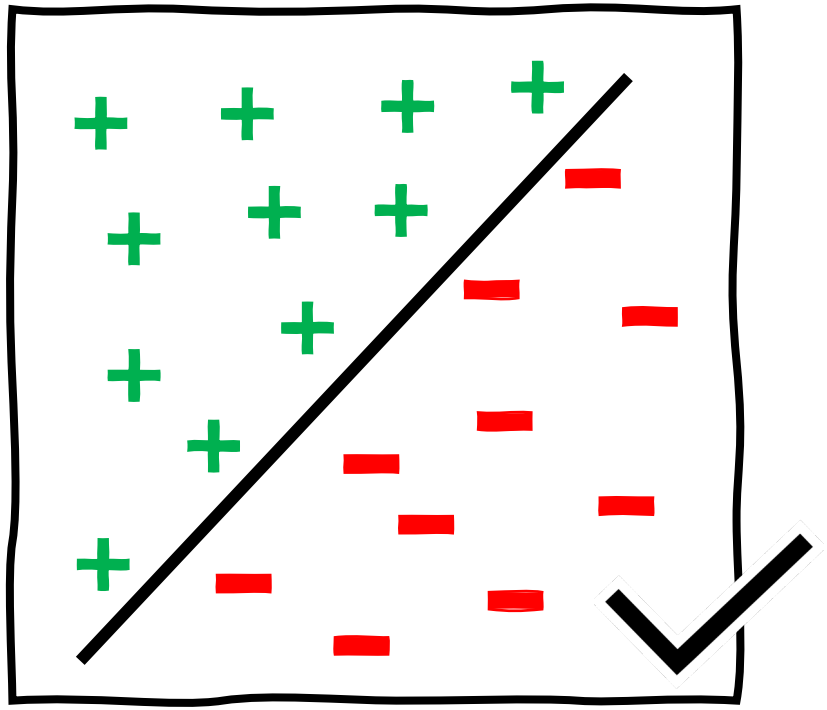
$$\sigma(x; w), b = \max(0, w^T x + b)$$



What can one neuron do?

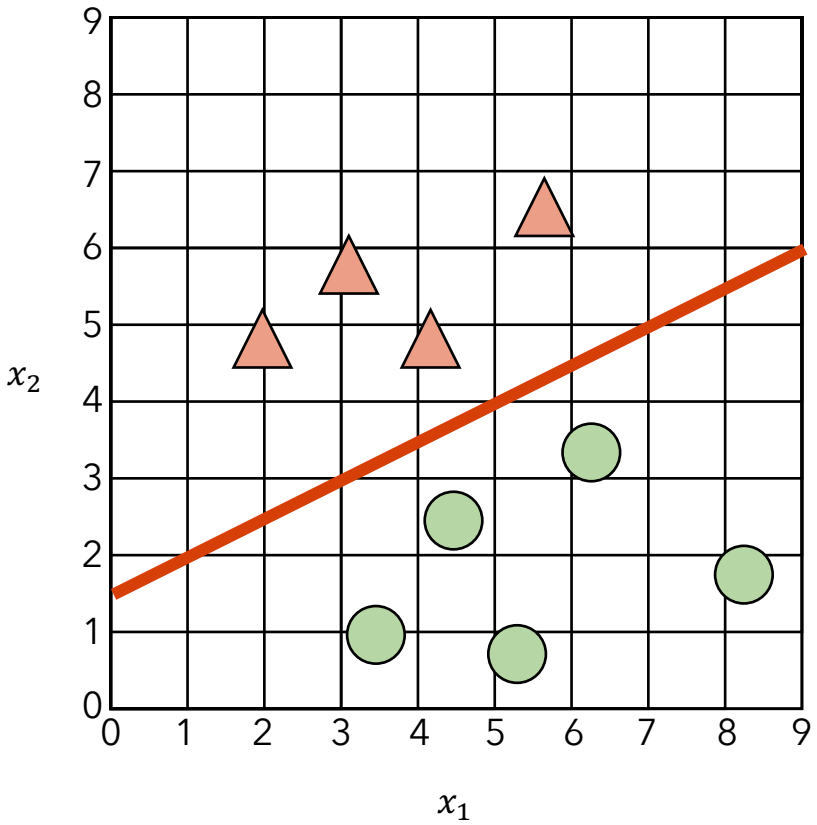


$$\sigma(b + \boldsymbol{w}^T \boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{w}^T \boldsymbol{x} + b)}}$$



Hands On With Linear Classifiers

What set of weights would make it so $w^T x + b > 0$ for all the green points and <0 for red points?



Step 1) Draw a line separating the classes

Step 2) Figure out the equation of that line

$$1x_2 = w_1 x_1 + b$$

$$1x_2 = 0.5x_1 + 1.5$$

A ↙ ↘ **B**

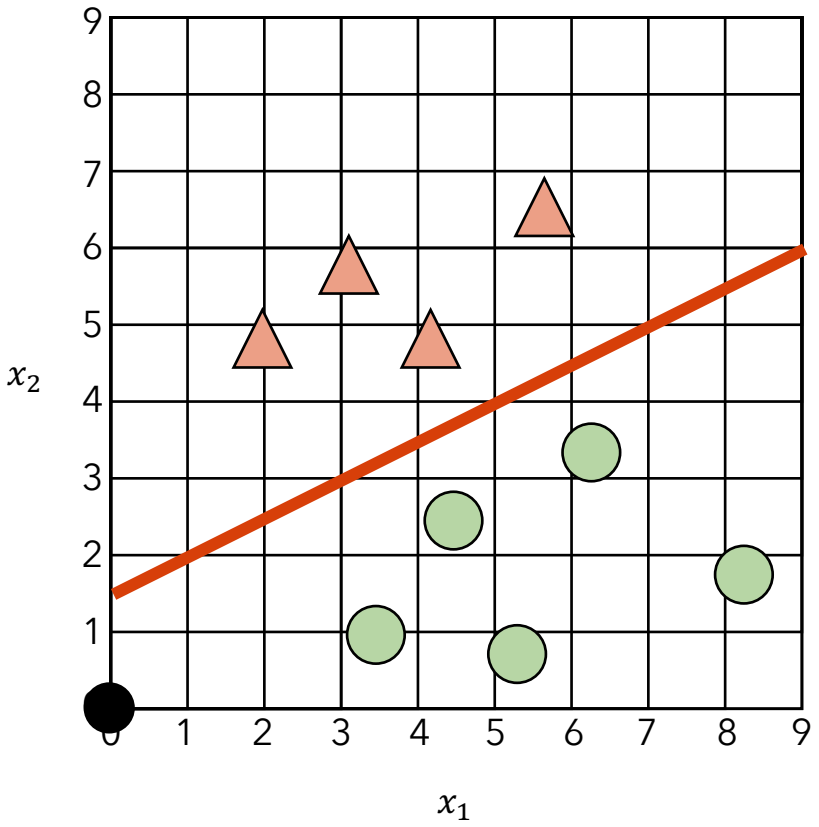
$$-0.5x_1 + 1x_2 - 1.5 = 0 \qquad 0.5x_1 - 1x_2 + 1.5 = 0$$

$\boldsymbol{w = [-0.5, 1]\ \ b = -1.5}$ $\boldsymbol{w = [0.5, -1]\ \ b = 1.5}$

Which to choose?

Hands On With Linear Classifiers

What set of weights would make it so $w^T x + b > 0$ for all the green points and <0 for red points?



Step 1) Draw a line separating the classes

Step 2) Figure out the equation of that line

A $-0.5x_1 + 1x_2 - 1.5 = 0$

B $0.5x_1 - 1x_2 + 1.5 = 0$

Step 3) Check the sign on each side of the line
(I usually use (0,0) such that value is just -b).

A $-0.5 * 0 + 1 * 0 - 1.5 = -1.5$

B $0.5 * 0 - 1 * 0 + 1.5 = 1.5$

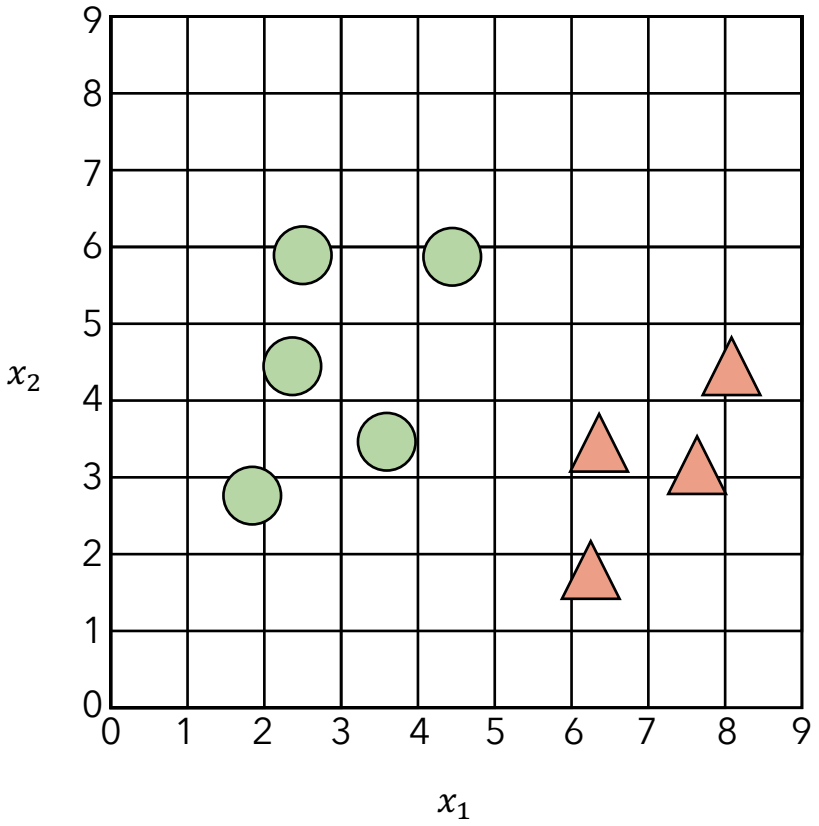
Step 4) Choose weights that match pos/neg classes

$$\boldsymbol{w = [0.5, -1] \;\; b = 1.5}$$

Lines Demo: https://colab.research.google.com/drive/1DELXo_sC5Ff22_N_PB2P7bBwolJe_pH6?usp=sharing

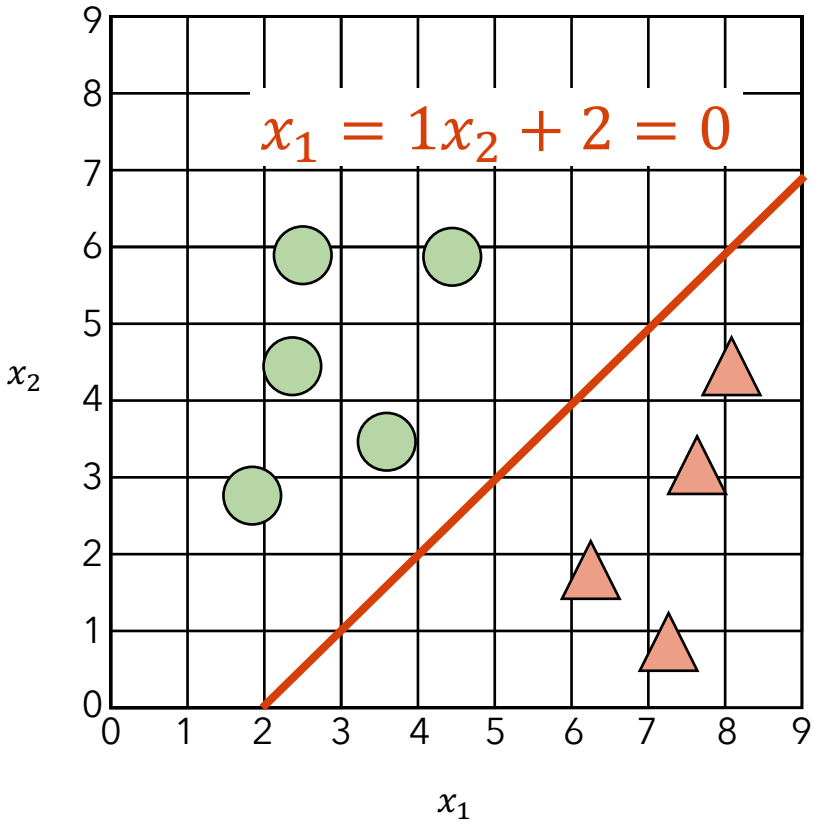
Questions Break!

What set of weights would make it so $w^T x + b > 0$ for all the green points and <0 for red points?



Questions Break!

What set of weights would make it so $w^T x + b > 0$ for all the green points and <0 for red points?



Step 1) Draw a line separating the classes

Step 2) Figure out the equation of that line

A $-1x_1 + 1x_2 + 2 = 0$

B $1x_1 - 1x_2 - 2 = 0$

Step 3) Check the sign on each side of the line
(I usually use (0,0) such that value is just -b).

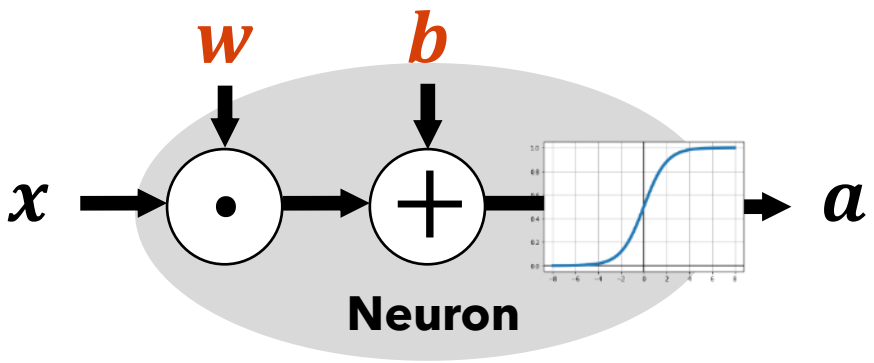
A $-1 * 0 + 1 * 0 + 2 = 2$

B $1 * 0 - 1 * 0 - 2 = -2$

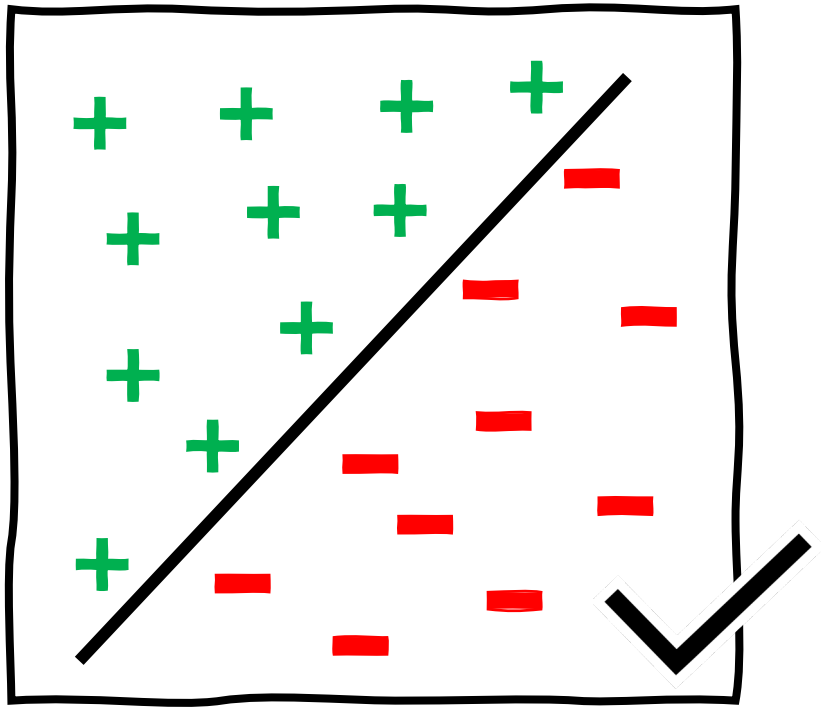
Step 4) Choose weights that match pos/neg classes

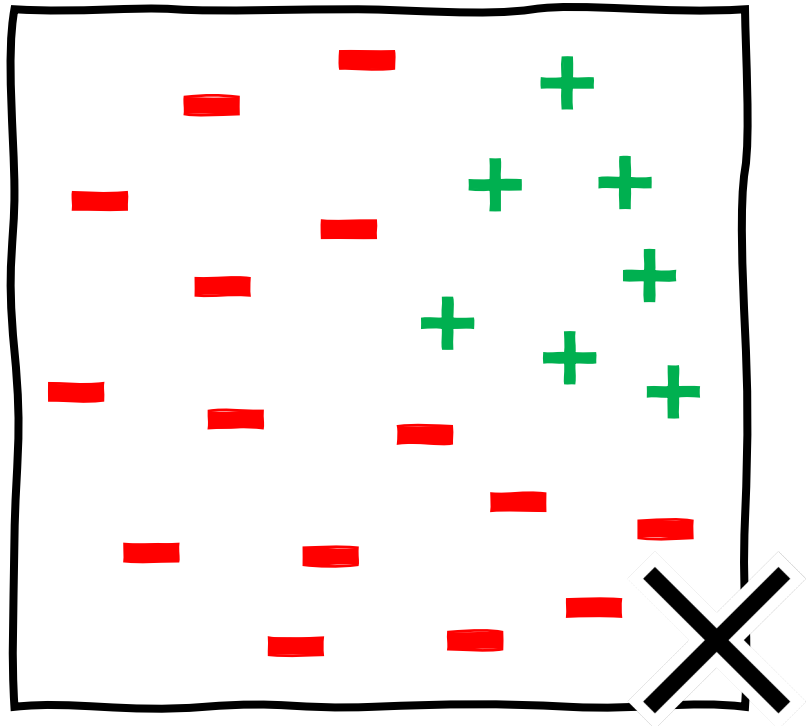
$$\boldsymbol{w = [-1, 1] \;\; b = 2}$$

What can one neuron do?

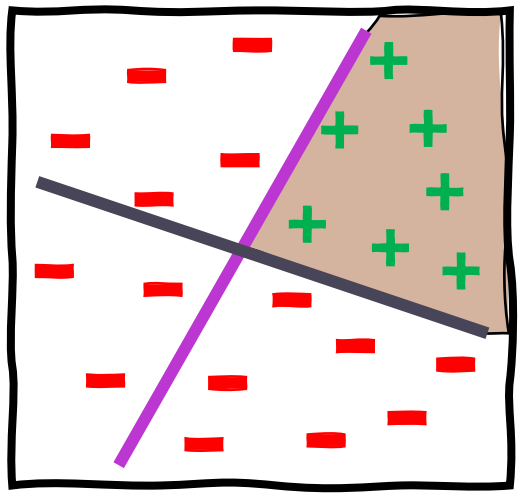


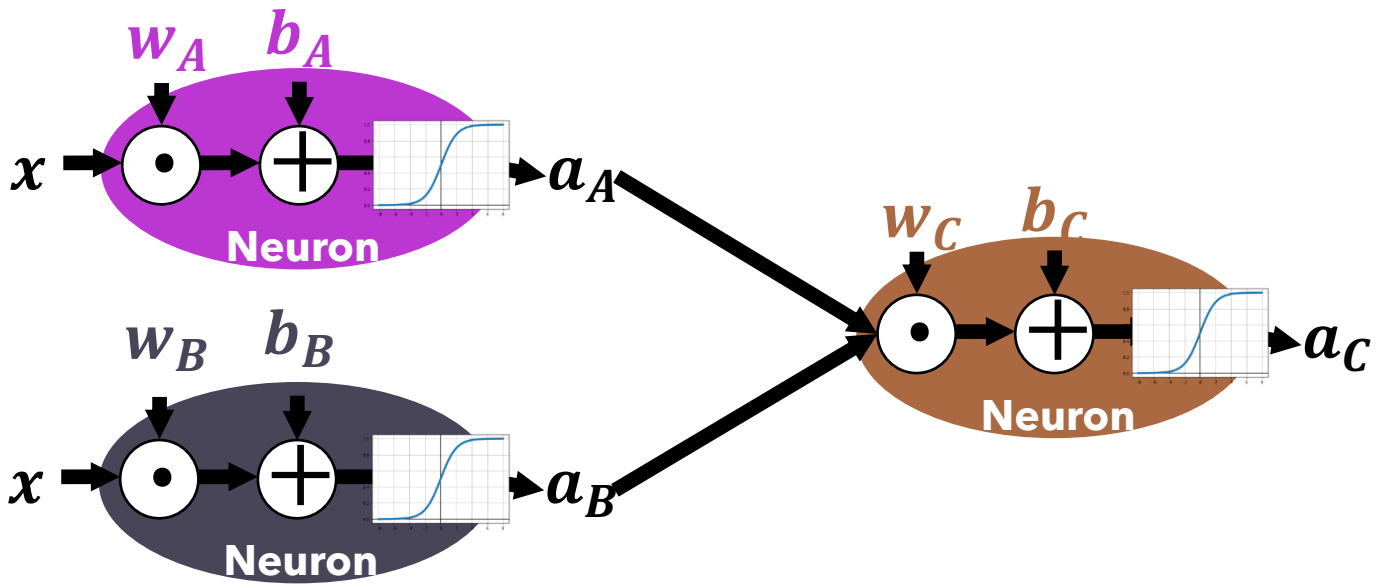
$$\sigma(b + w^T x) = \frac{1}{1 + e^{-(w^T x + b)}}$$



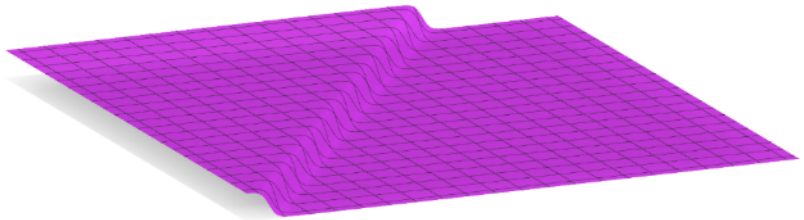


Idea: Stack Them Together to orm Neural Networks



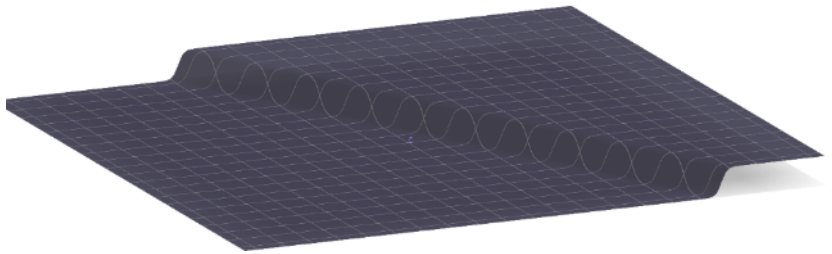


\- and +



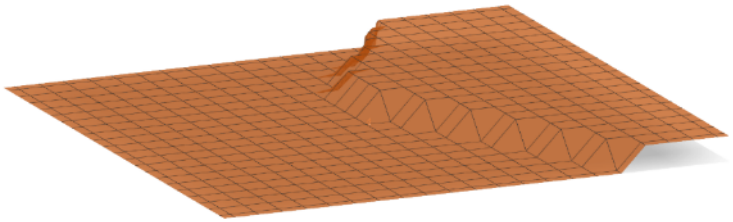
$w = [-7, 5]^T$

$b = -15$



$w = [10, 5]^T$

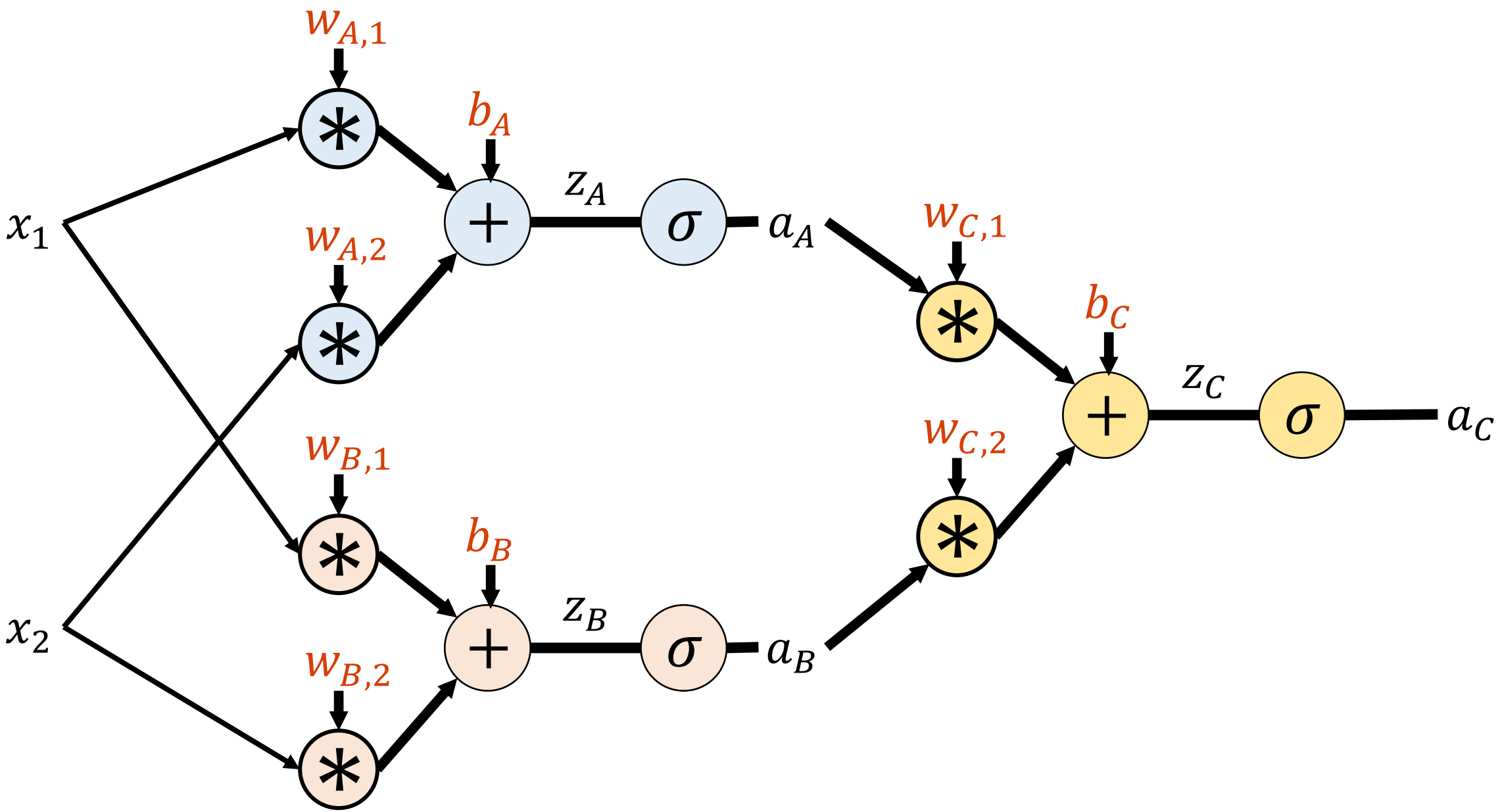
$b = -15$



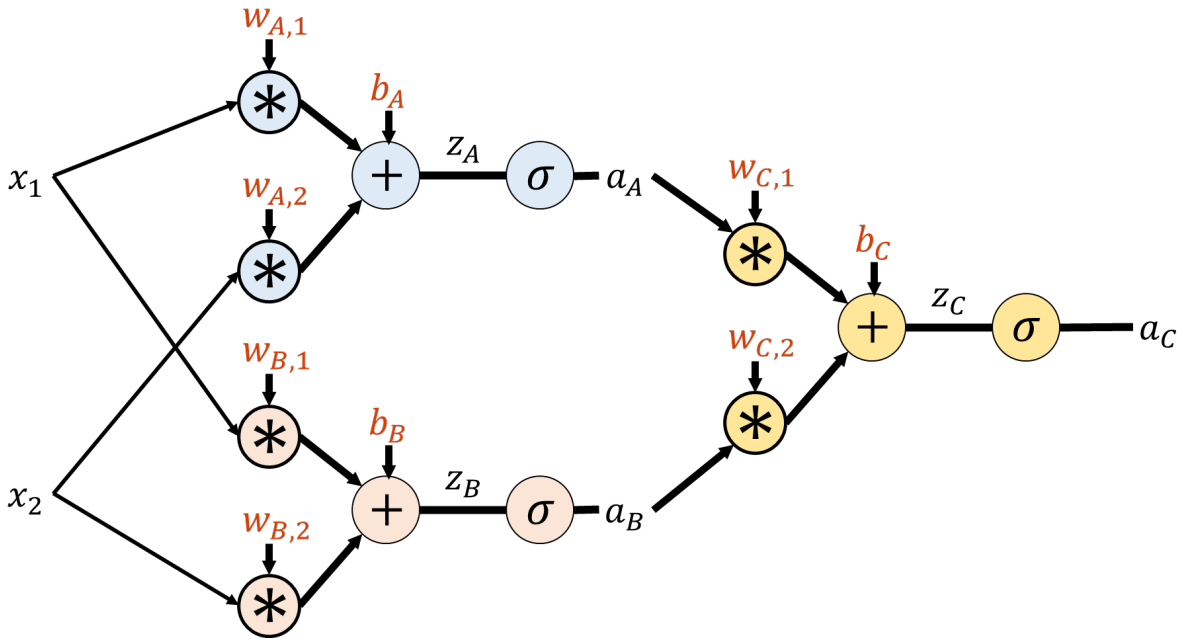
$w = [-100, 100]^T$

$b = -10$

2-layer Feed-Forward Neural Network



Stack Them Together to form Neural Nets



$$z_A = w_{A,1}x_1 + w_{A,2}x_2 + b_A$$

$$a_A = \sigma(z_A)$$

$$z_B = w_{B,1}x_1 + w_{B,2}x_2 + b_B$$

$$a_B = \sigma(z_B)$$

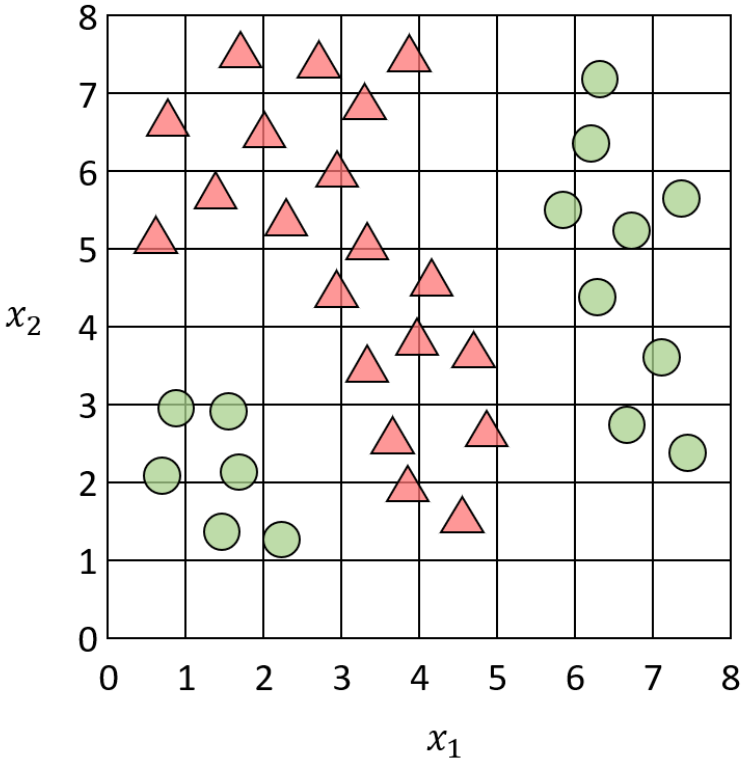
$$z_C = w_{C,1}a_A + w_{C,2}a_B + b_C$$

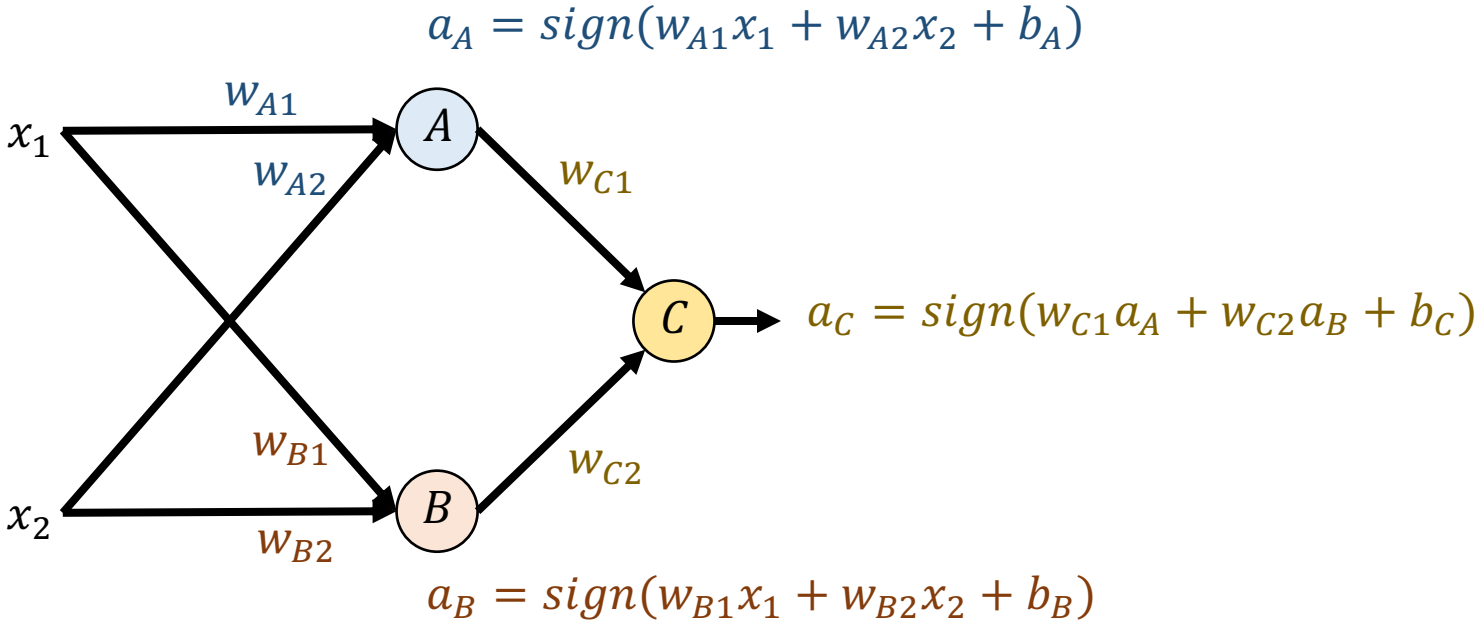
$$a_C = \sigma(z_C)$$

$$a_C = \sigma\left(w_{C,1}\sigma\left(w_{A,1}x_1 + w_{A,2}x_2 + b_A\right) + w_{C,2}\sigma\left(w_{B,1}x_1 + w_{B,2}x_2 + b_B\right) + b_C\right)$$

What does this buy us?

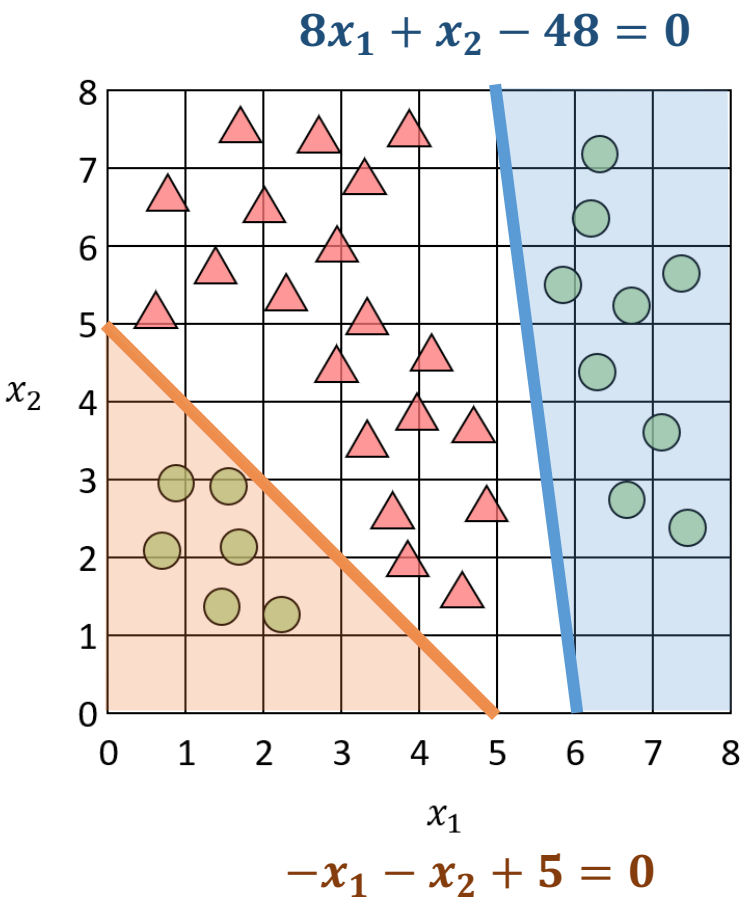
Let's consider neurons with a sign activation:

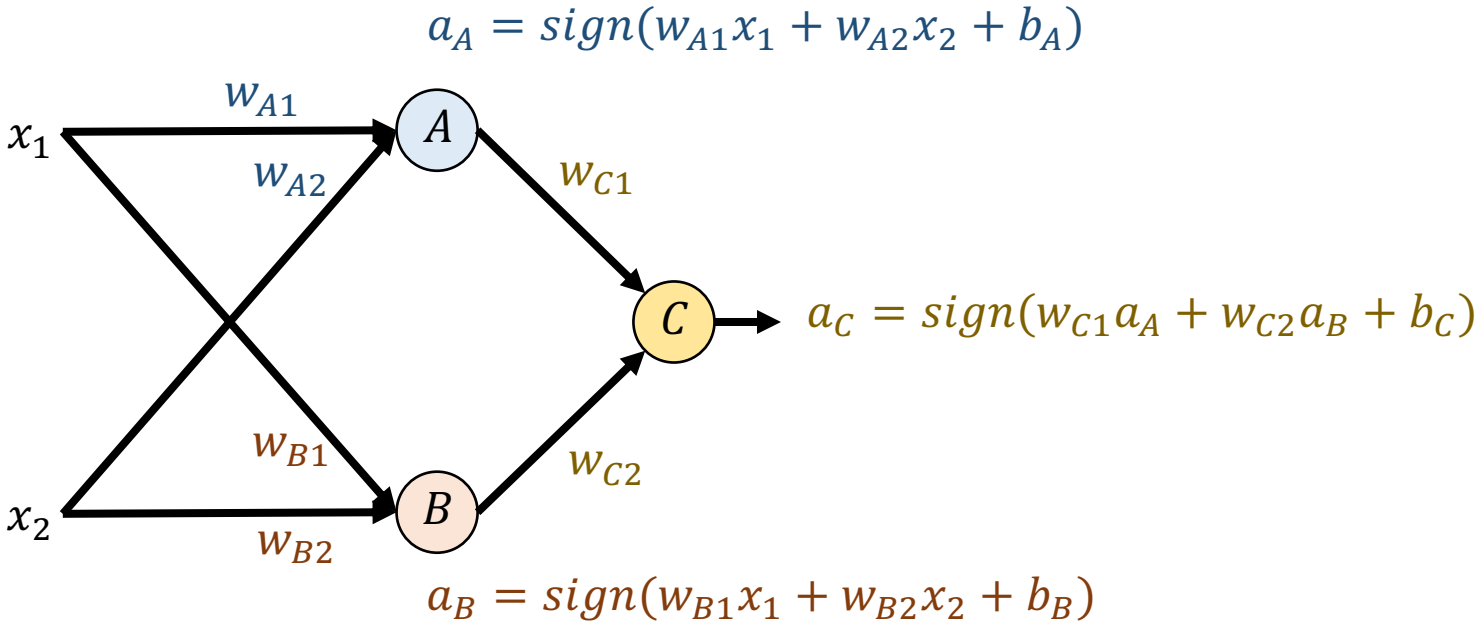




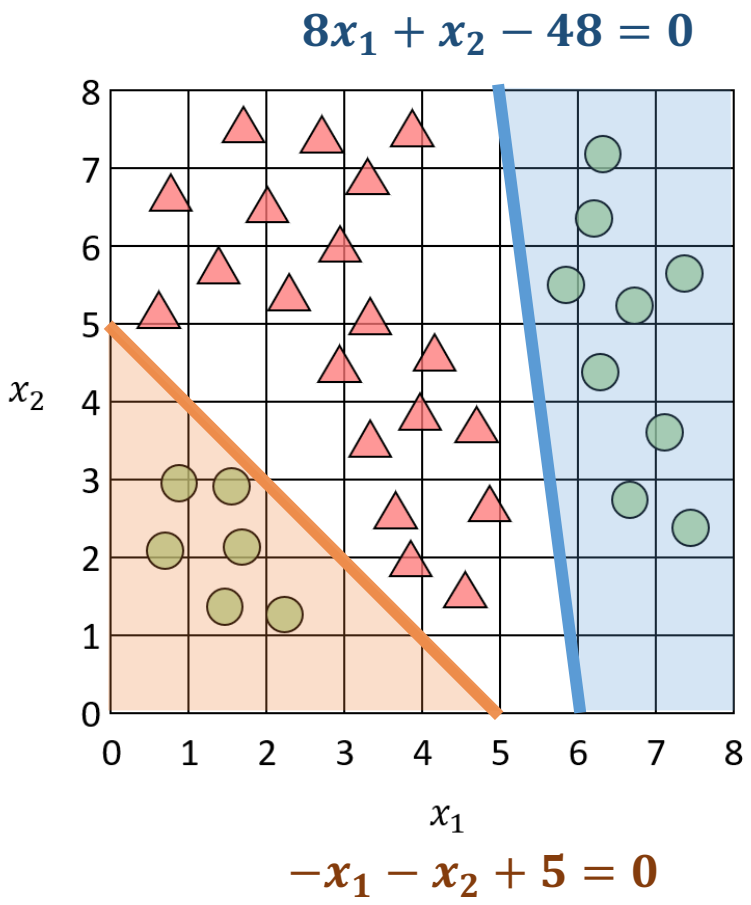
What values can we set the parameters to let the network solve this dataset?

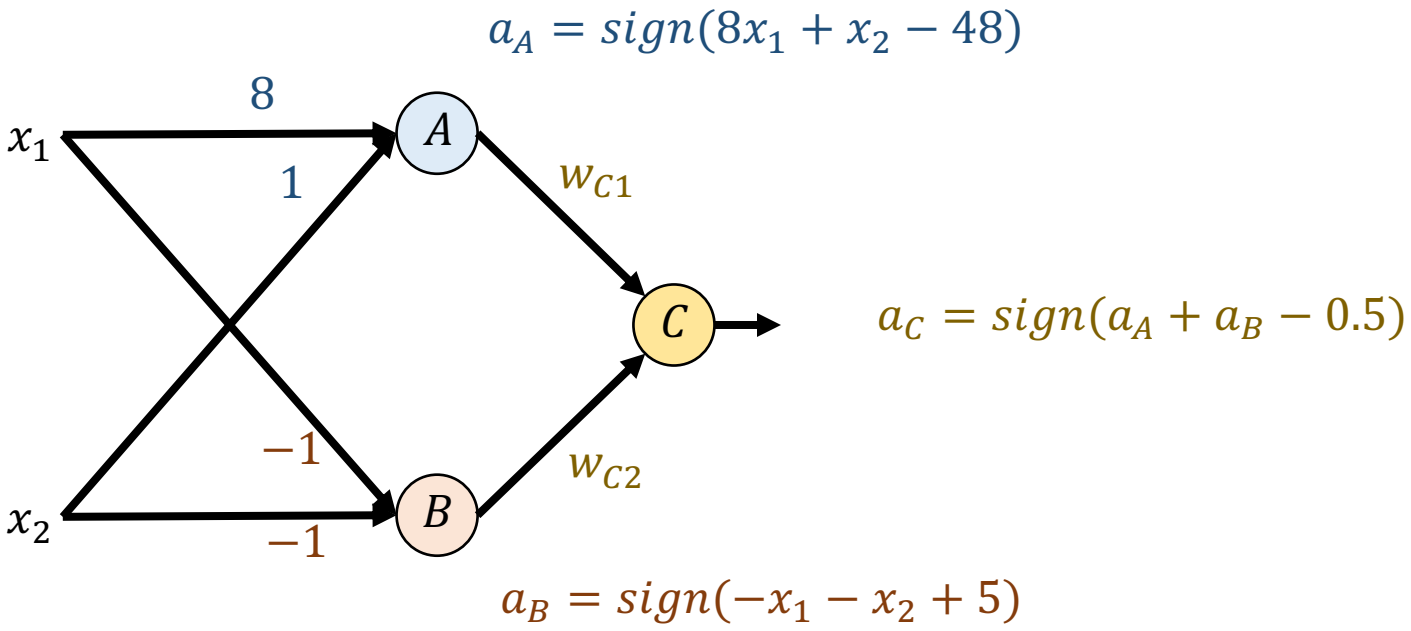
What does this buy us?





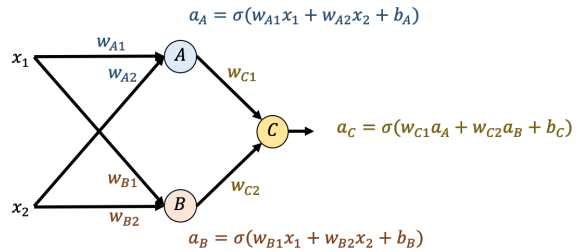
What does this buy us?





What does this buy us? Let's see this a different way.

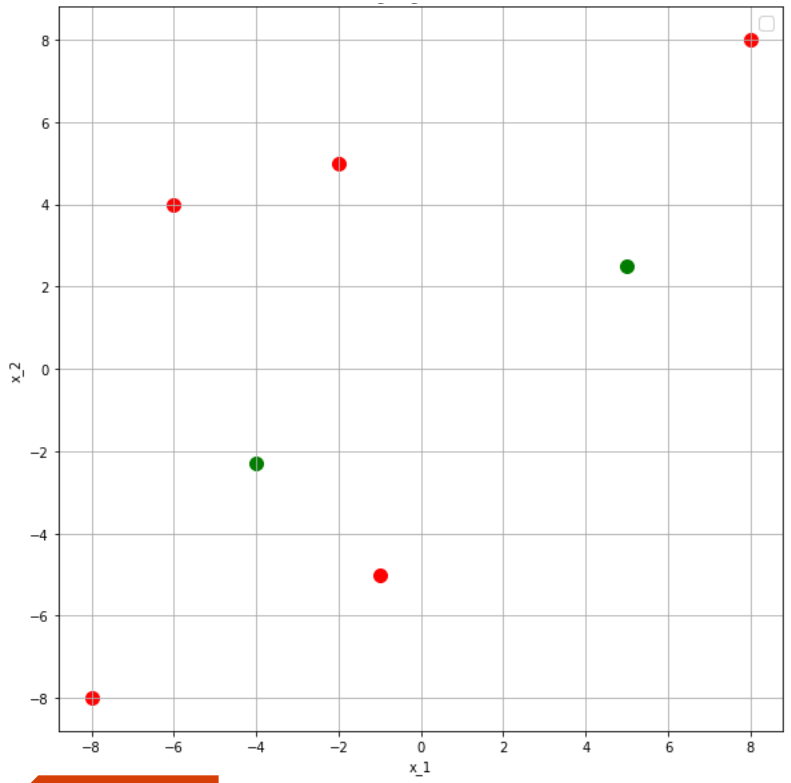
Let's consider neurons with a logistic activation:



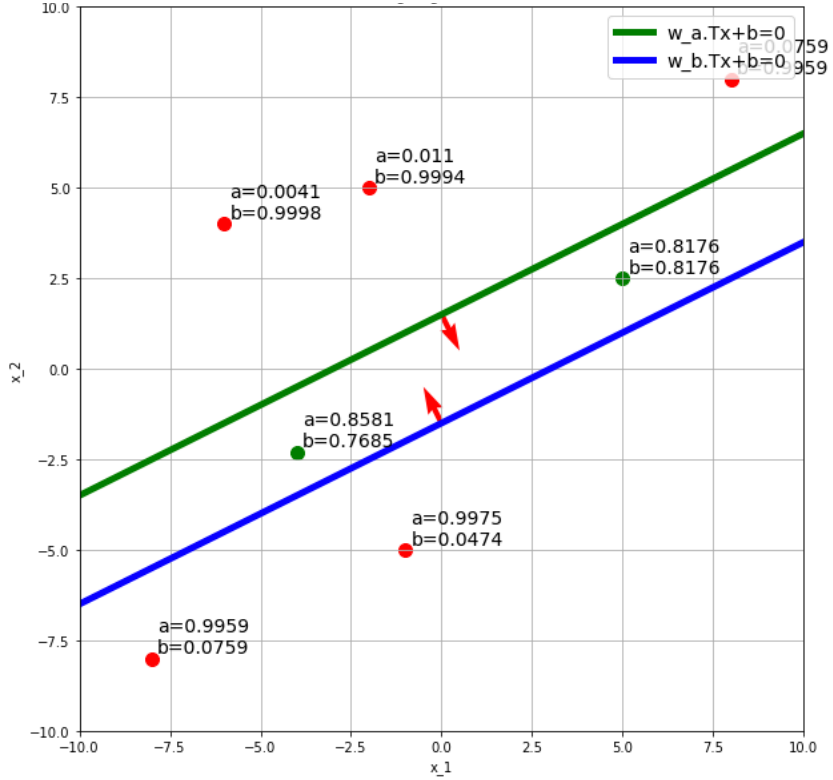
Intuition: Each layer transforms the input space to a new space.
If well trained, the new input space is *easier* to classify in.

Consider the same neural network structure but with a logistic activation.

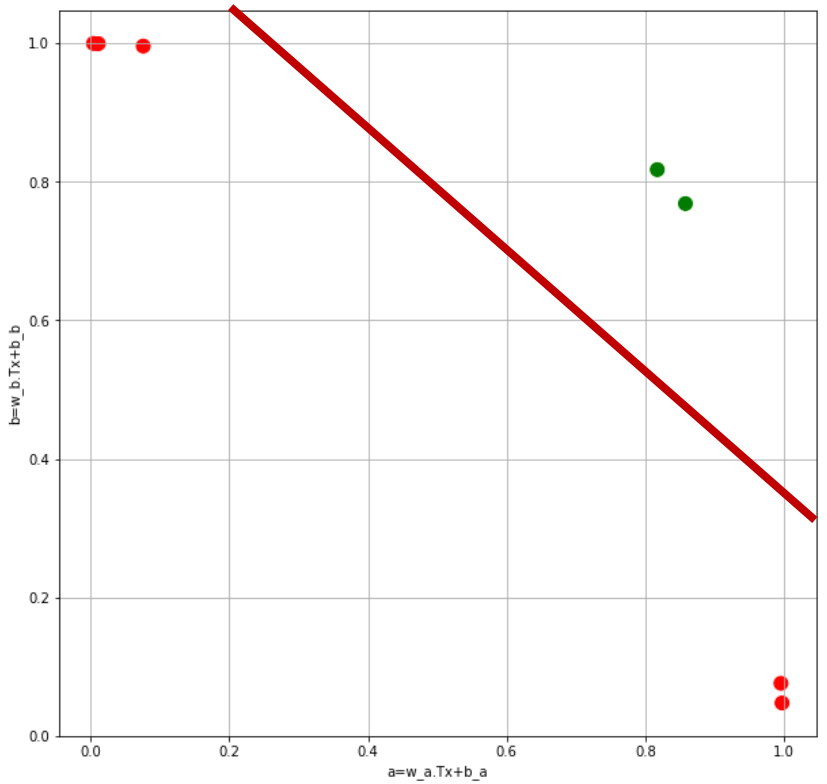
Original Input Space



Lines for neurons A and B

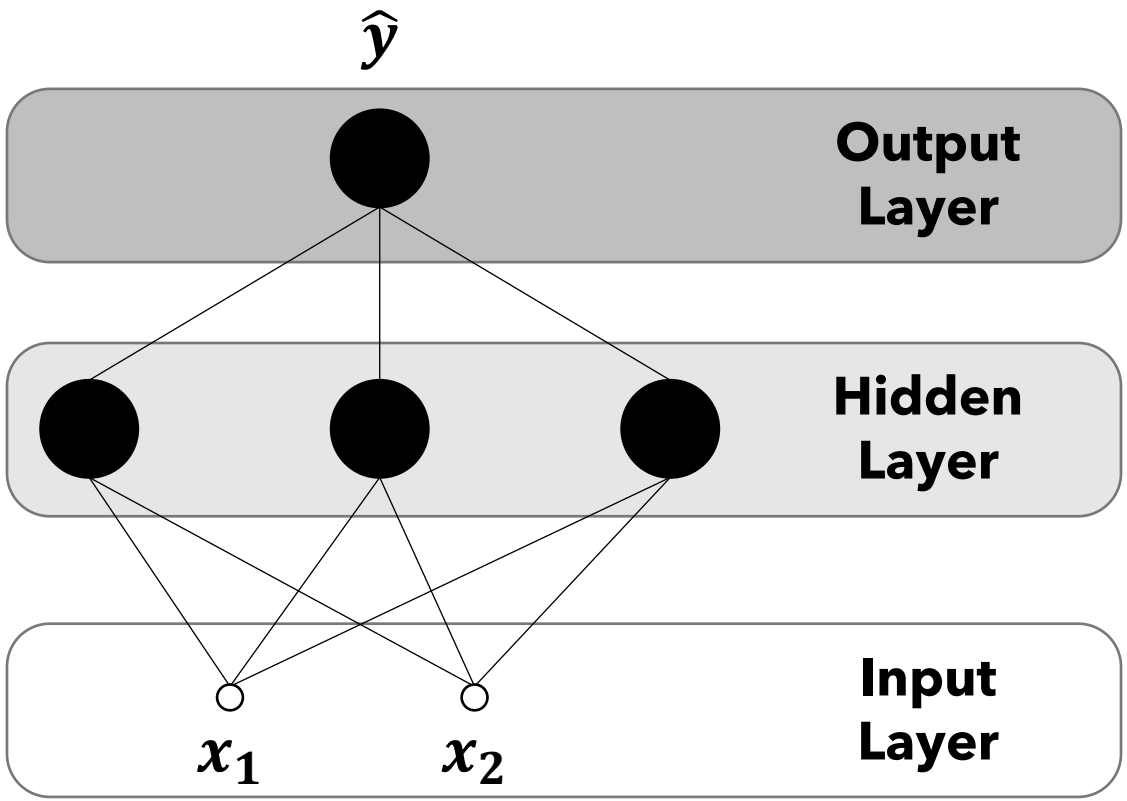


Input Space for neuron C



Basic Multilayer Neural Network

A Neural Network is a set of connected neurons. A very typical arrangement is a feed-forward multilayer neural network like the one shown below.



Each layer receives its input from the previous layer and forwards its output to the next – thus the **feed-forward** description.

The layers of neurons between the input and output are referred to as **hidden layers**.

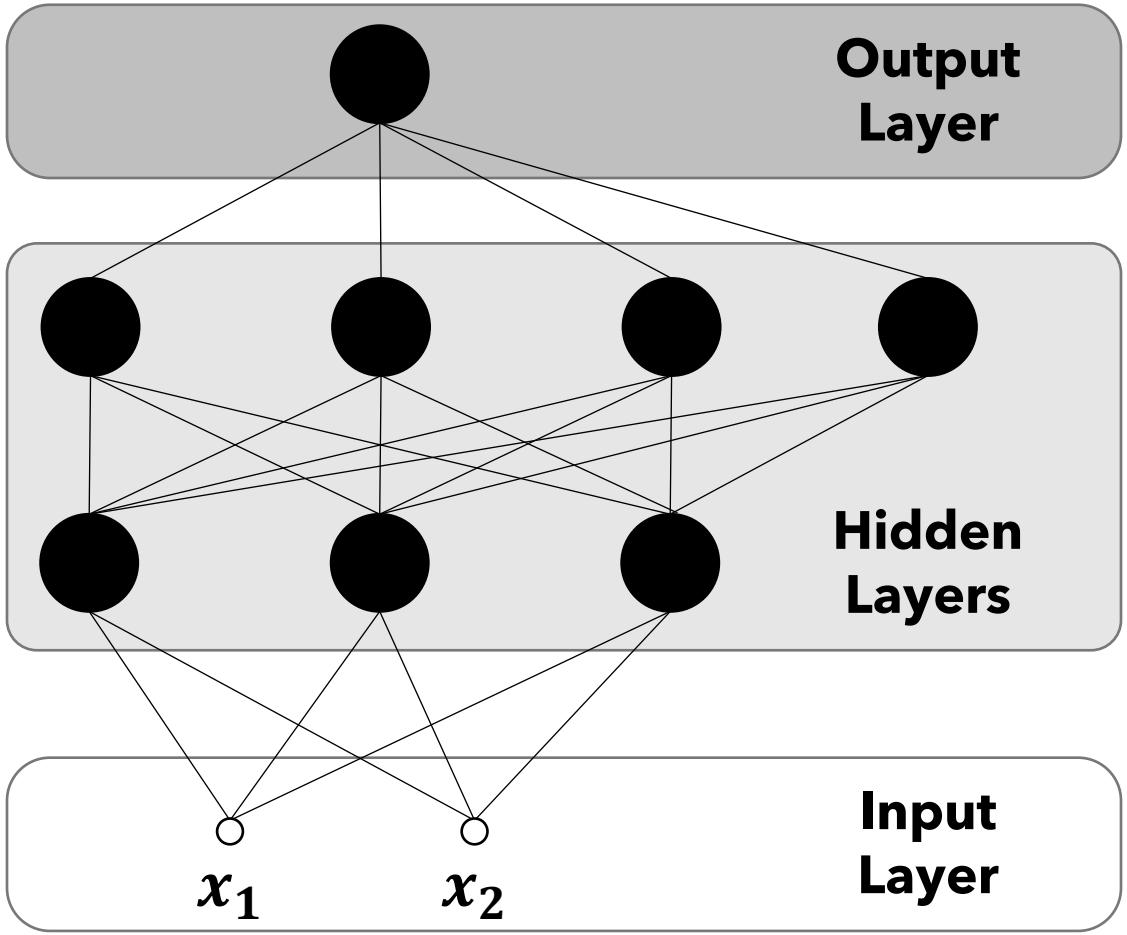
- This network for instance is **a 2-layer neural** network (1 hidden and 1 output)
- Number of neurons in a layer is referred to as its width.

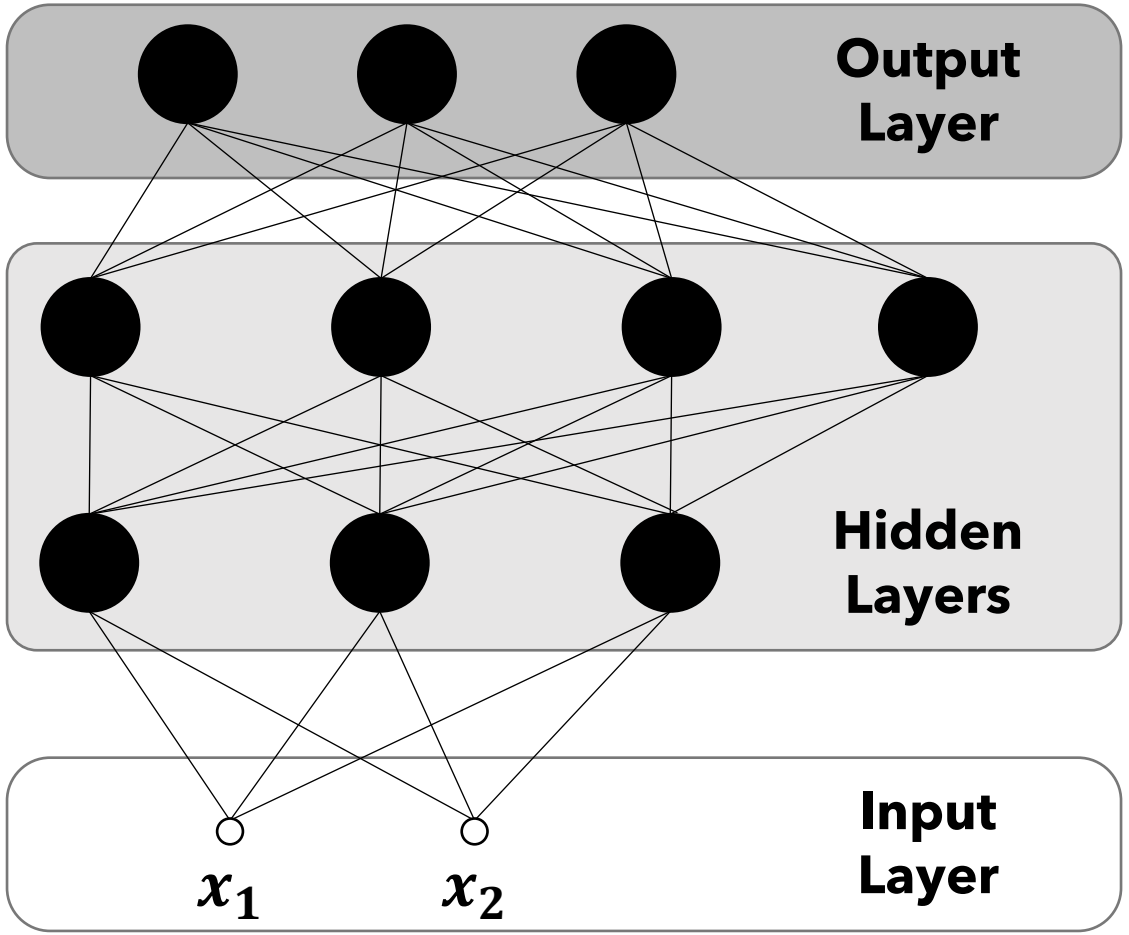
Activation functions in different layers can be heterogenous.

- Output layer's activation is task dependent
  - Linear for regression
  - Sigmoid or softmax for classification

Basic Multilayer Neural Network

Can make arbitrary configurations of # of hidden layers, layer widths, and number of inputs / outputs. Very flexible models!



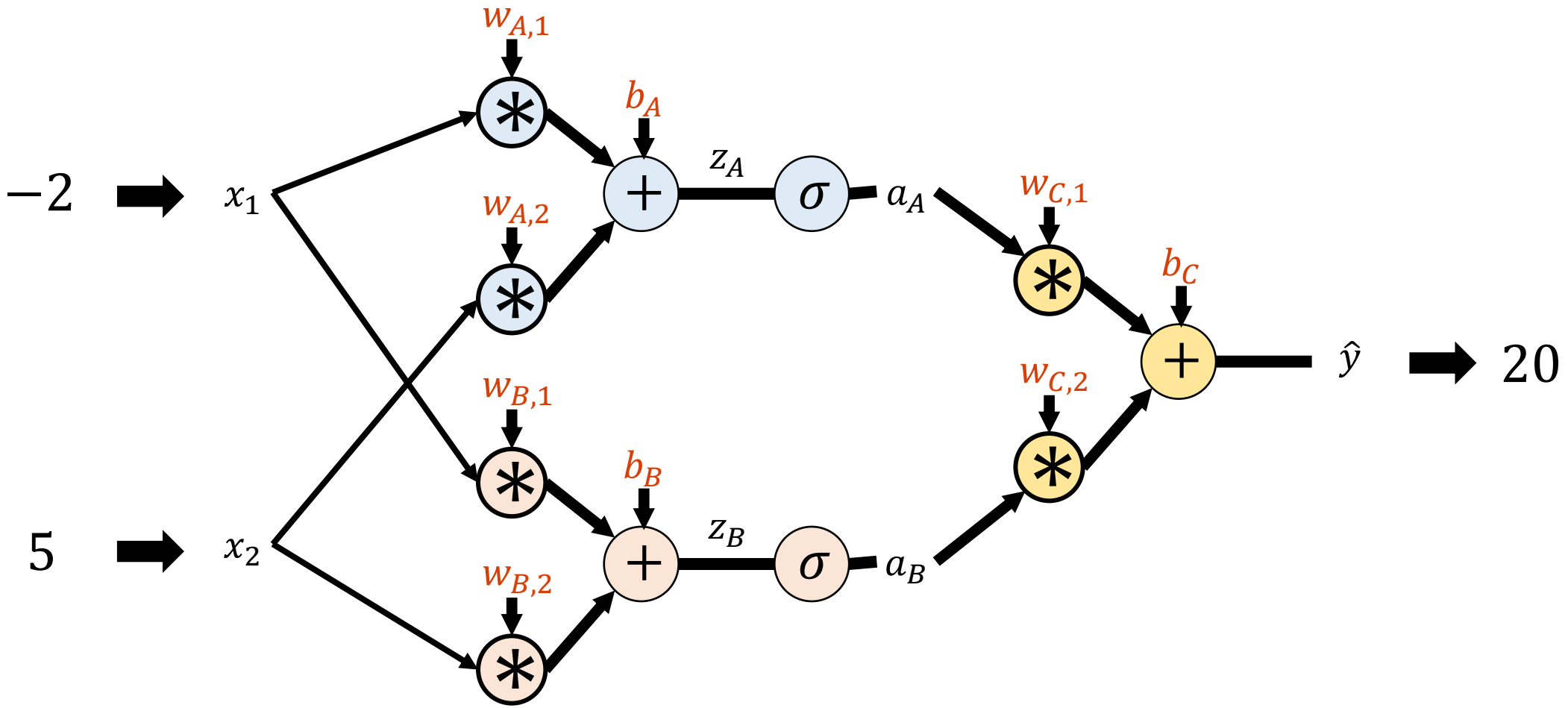


Demo

https://playground.tensorflow.org/

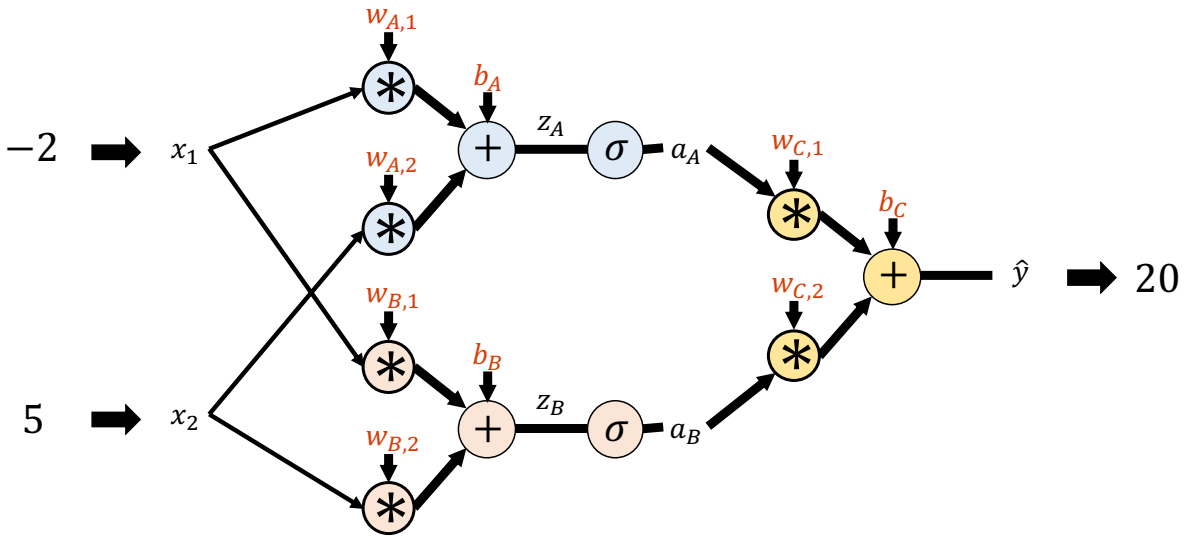
How do I train these things? An Example from Regression

Say our dataset just has one point $\boldsymbol{x} = [-2,5]$ with y value 15.



How can we get it to do better? AKA How can we train it?

Loss Function *L*– A function measuring "how bad" a network's output is, usually relative to some gold-standard for what the output should be.



$\boldsymbol{x} = [-2,5]$ with y value 15

$$\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$$

For example, the L2 loss is commonly used for regression

$$\mathcal{L}(y = 15, \hat{y} = 25) = (15 - 25)^2 = 100$$

How do I train these things? An Example from Regression

How can I update my parameters to reduce the loss $\mathcal{L}$?

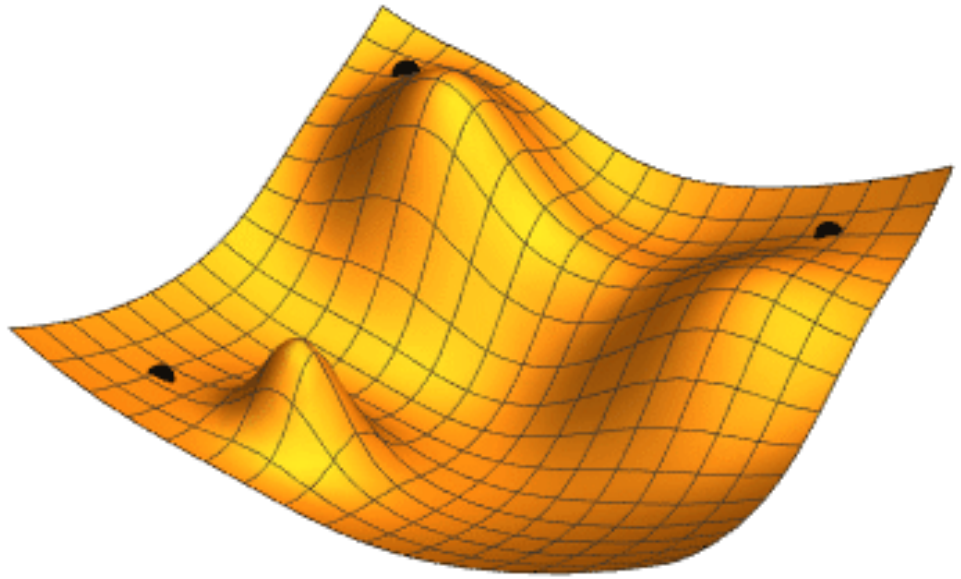
Gradient Descent Algorithm

$w \leftarrow random(\quad)$

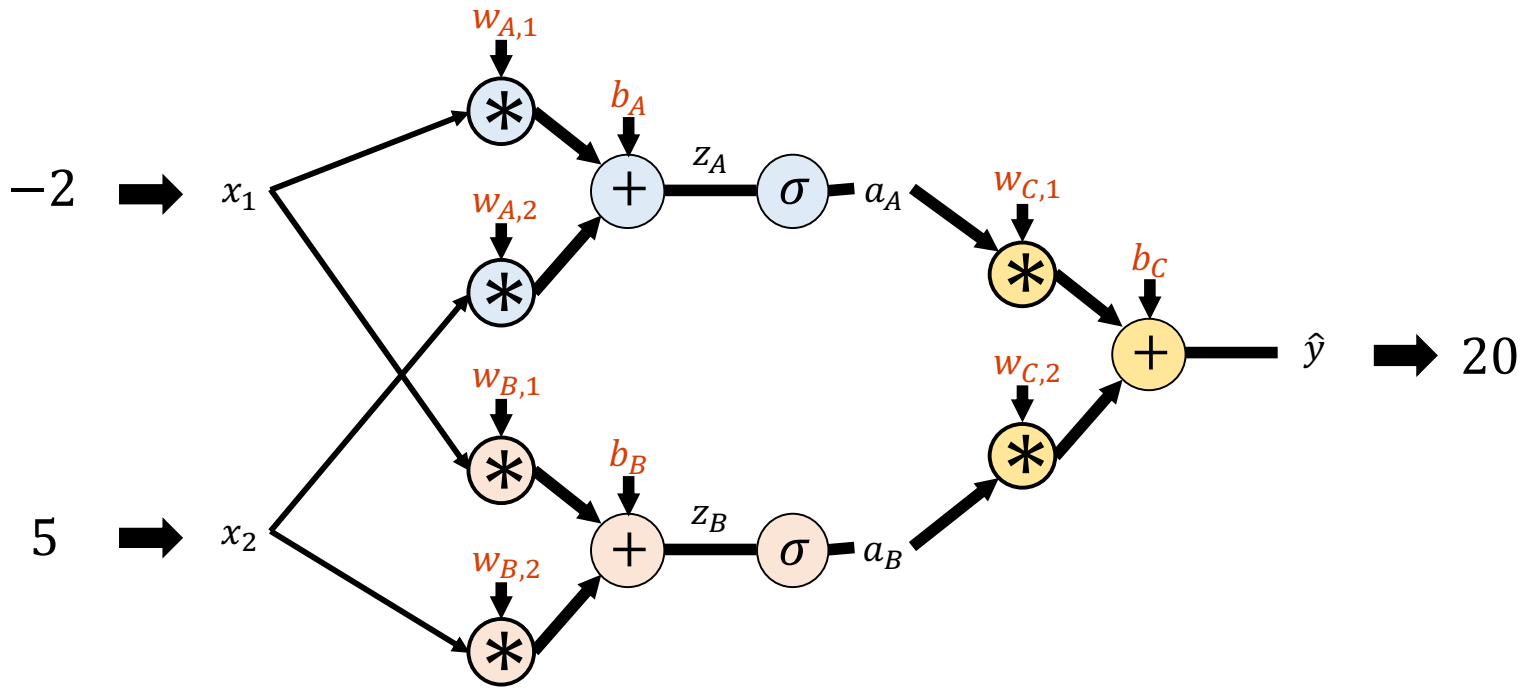
while$|\nabla_w \mathcal{L}| > \epsilon$ *or iters remain*

$w = w - \alpha \nabla_w \mathcal{L}(w)$

Learning Rate / Step Size



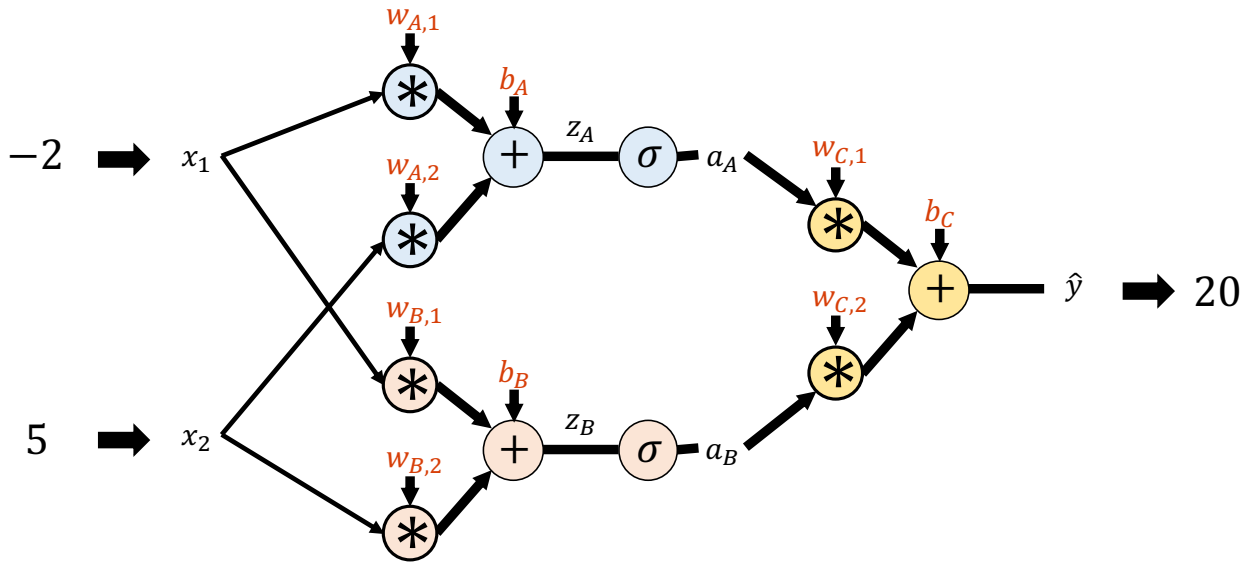
Let's consider the partial derivatives here: $\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$



Remember what we mean here – derivatives talk about how the output of the function would change as you change one of its inputs.

Let's consider the partial derivatives here: $\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$

$\boldsymbol{x} = [-2,5]$ with y value 15



$$z_A = w_{A,1}x_1 + w_{A,2}x_2 + b_A \qquad a_A = \sigma(z_A)$$

$$z_B = w_{B,1}x_1 + w_{B,2}x_2 + b_B \qquad a_B = \sigma(z_B)$$

$$\hat{y} = w_{C,1}a_A + w_{C,2}a_B + b_C$$

$$\frac{\delta \mathcal{L}(y, \hat{y})}{\delta b_C} = \frac{\delta \mathcal{L}(y, \hat{y})}{\delta \hat{y}} * \frac{\delta \hat{y}}{b_C}$$

$$\frac{\delta \mathcal{L}(y, \hat{y})}{\delta b_C} = -2(y - \hat{y}) * 1$$

Evaluate at the current point

$$\frac{\delta \mathcal{L}(y, \hat{y})}{\delta b_C} = -2(15 - 20) * 1 = 10$$

Questions Break!

We had an example $\boldsymbol{x} = [-2,5]$ with y=15 and our network predicted $\hat{y} = 20$.

$$\hat{y} = w_{C,1}a_A + w_{C,2}a_B + b_C$$

How should b_c change to make the prediction more correct if we reran it?

Get more negative so $\hat{y}$ reduces.

Questions Break!

We had an example $\boldsymbol{x} = [-2,5]$ with y=15 and our network predicted $\hat{y} = 20$.

$$\hat{y} = w_{C,1}a_A + w_{C,2}a_B + b_C$$

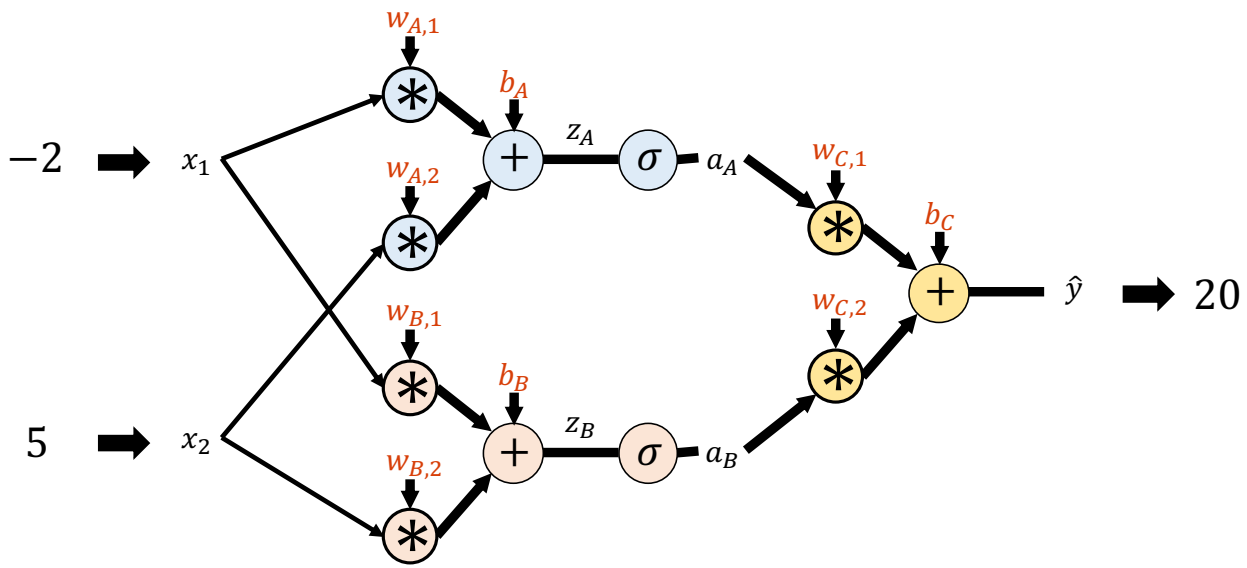
We just computed the partial derivative of the L2 loss with respect to b_c

$$\frac{\delta \mathcal{L}(y, \hat{y})}{\delta b_C} = -2(15 - 20) * 1 = 10$$

If we took a step of gradient descent with this gradient, would that move in the right direction for b_C?

Let's consider the partial derivatives here: $\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$

$\boldsymbol{x} = [-2,5]$ with y value 15



$$\frac{\delta \mathcal{L}(y, \hat{y})}{\delta w_{C,1}} = \frac{\delta \mathcal{L}(y, \hat{y})}{\delta \hat{y}} * \frac{\delta \hat{y}}{w_{c,1}}$$

$$= -2(y - \hat{y}) * a_A$$

$$= -2(15 - 20) * a_A$$

$$= 10 a_A$$

$z_A = w_{A,1}x_1 + w_{A,2}x_2 + b_A$ $\quad a_A = \sigma(z_A)$

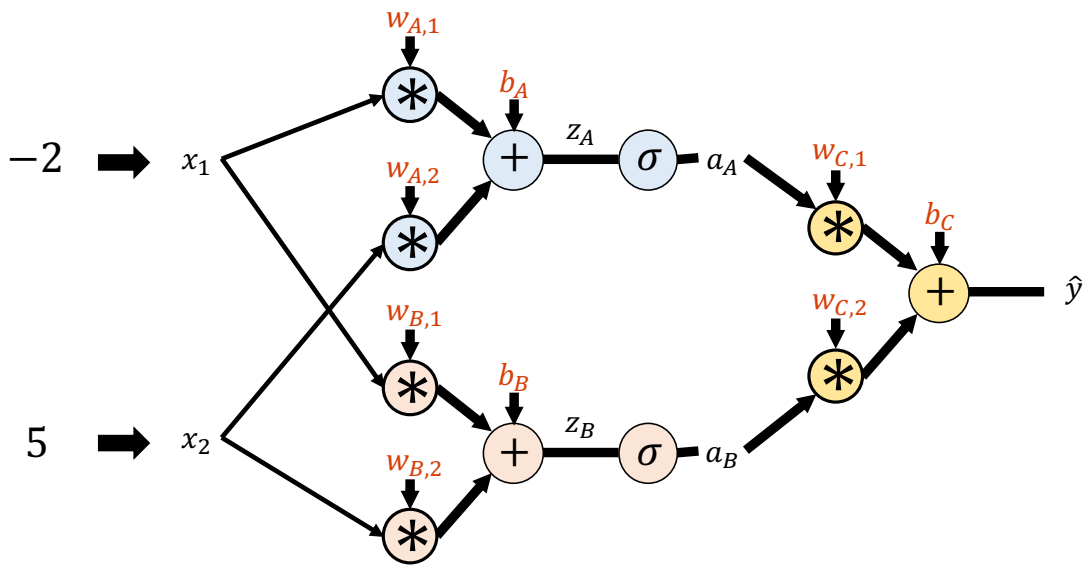
$z_B = w_{B,1}x_1 + w_{B,2}x_2 + b_B$ $\quad a_B = \sigma(z_B)$

$\hat{y} = w_{C,1}a_A + w_{C,2}a_B + b_C$

If a_A is positive, gradient descent will reduce $w_{C,1}$. Opposite if a_A is positive

Let's consider the partial derivatives here: $\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$

$\boldsymbol{x} = [-2,5]$ with y value 15



$$\frac{\delta \mathcal{L}(y, \hat{y})}{\delta b_A} = \frac{\delta \mathcal{L}(y, \hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta a_A} \frac{\delta a_A}{\delta z_A} \frac{\delta z_A}{\delta b_A}$$

$$= -2(y - \hat{y}) * w_{C,1} * a_A(1 - a_A) * 1$$

$z_A = w_{A,1}x_1 + w_{A,2}x_2 + b_A$ $\quad a_A = \sigma(z_A)$

$z_B = w_{B,1}x_1 + w_{B,2}x_2 + b_B$ $\quad a_B = \sigma(z_B)$

$\hat{y} = w_{C,1}a_A + w_{C,2}a_B + b_C$

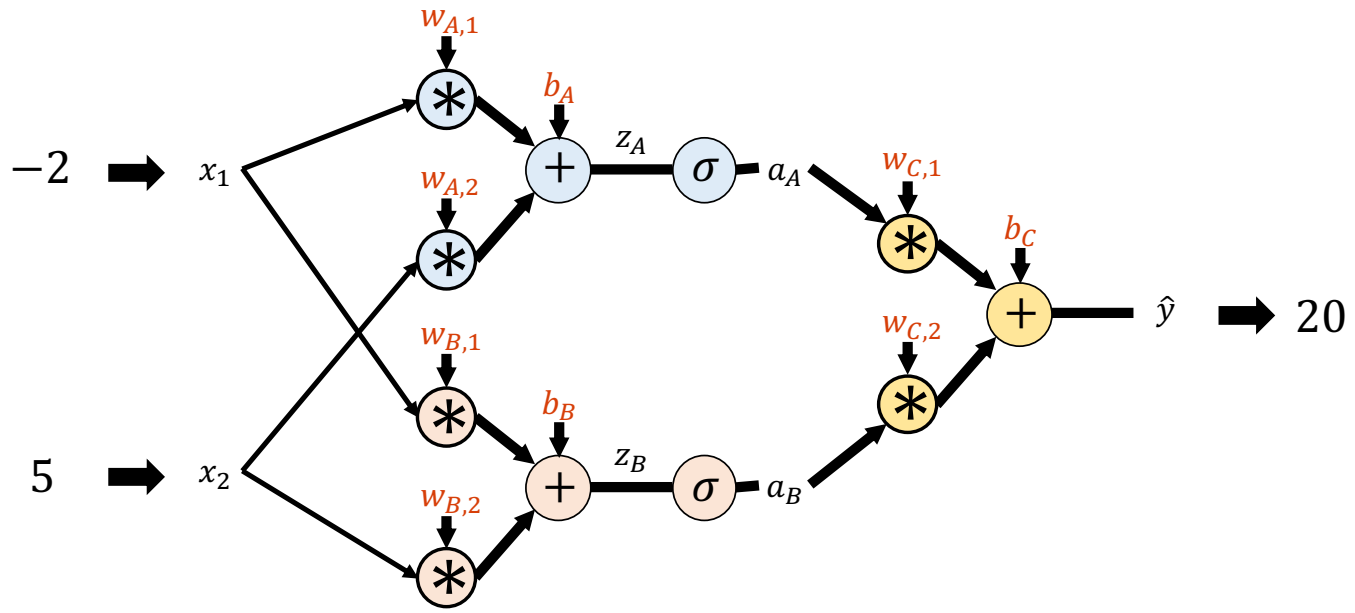
Recall that the derivative of logistic(x) with respect to x is logistic(x)*(1-logistic(x)). Way back in logistic regression slides.

How do I train these things? An Example from Regression

$$\frac{\delta \mathcal{L}(y,\hat{y})}{\delta w_{A,1}} = \frac{\delta \mathcal{L}(y,\hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta a_A} \frac{\delta a_A}{\delta z_A} \frac{\delta z_A}{\delta w_{A,1}}$$

$$\frac{\delta \mathcal{L}(y,\hat{y})}{\delta w_{A,2}} = \frac{\delta \mathcal{L}(y,\hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta a_A} \frac{\delta a_A}{\delta z_A} \frac{\delta z_A}{\delta w_{A,2}}$$

$$\frac{\delta \mathcal{L}(y,\hat{y})}{\delta b_A} = \frac{\delta \mathcal{L}(y,\hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta a_A} \frac{\delta a_A}{\delta z_A} \frac{\delta z_A}{\delta b_A}$$



$$\frac{\delta \mathcal{L}(y,\hat{y})}{\delta w_{B,1}} = \frac{\delta \mathcal{L}(y,\hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta a_B} \frac{\delta a_B}{\delta z_B} \frac{\delta z_B}{\delta w_{B,1}}$$

$$\frac{\delta \mathcal{L}(y,\hat{y})}{\delta w_{B,2}} = \frac{\delta \mathcal{L}(y,\hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta a_B} \frac{\delta a_B}{\delta z_B} \frac{\delta z_B}{\delta w_{B,2}}$$

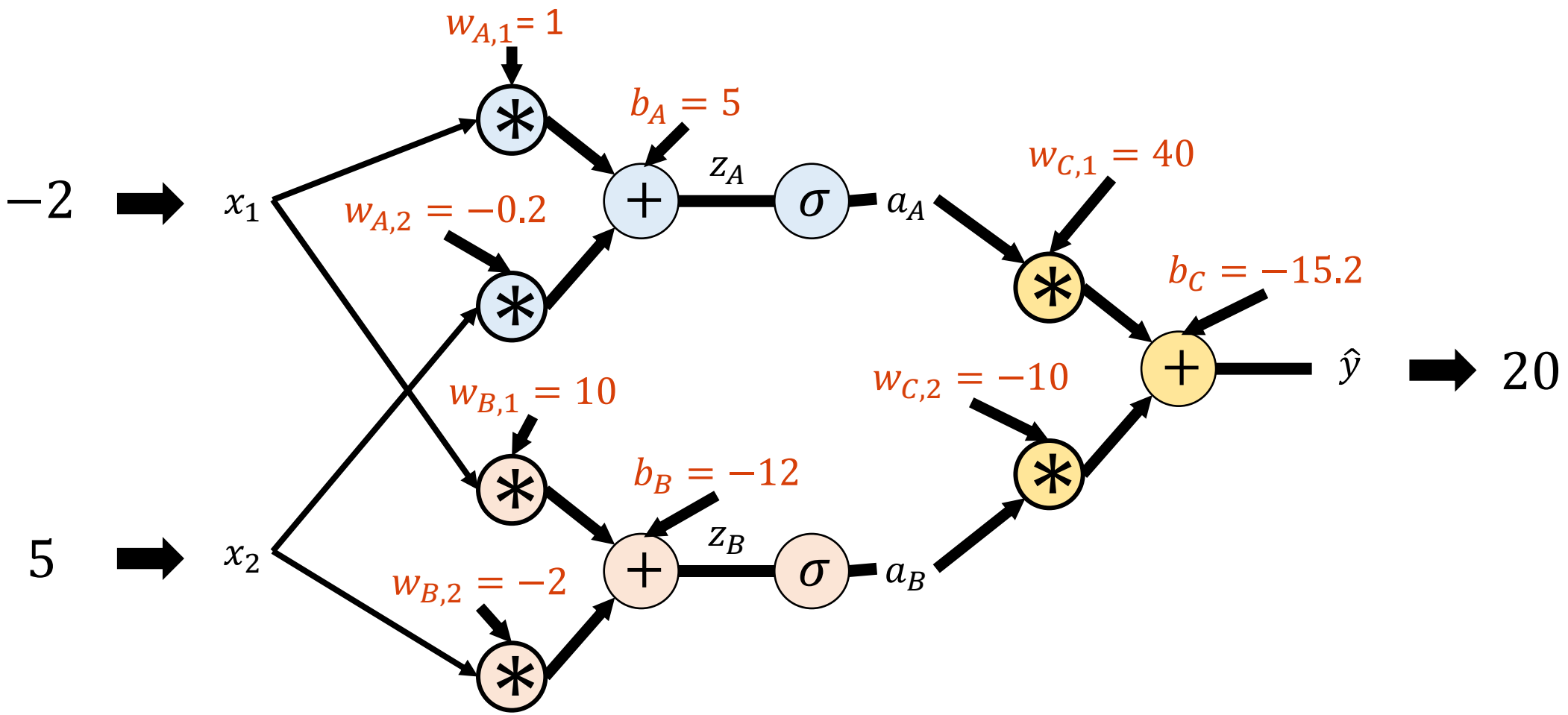
$$\frac{\delta \mathcal{L}(y,\hat{y})}{\delta b_B} = \frac{\delta \mathcal{L}(y,\hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta a_B} \frac{\delta a_B}{\delta z_B} \frac{\delta z_B}{\delta b_B}$$

$$\frac{\delta \mathcal{L}(y,\hat{y})}{\delta w_{C,1}} = \frac{\delta \mathcal{L}(y,\hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{w_{c,1}}$$

$$\frac{\delta \mathcal{L}(y,\hat{y})}{\delta b_C} = \frac{\delta \mathcal{L}(y,\hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{b_c}$$

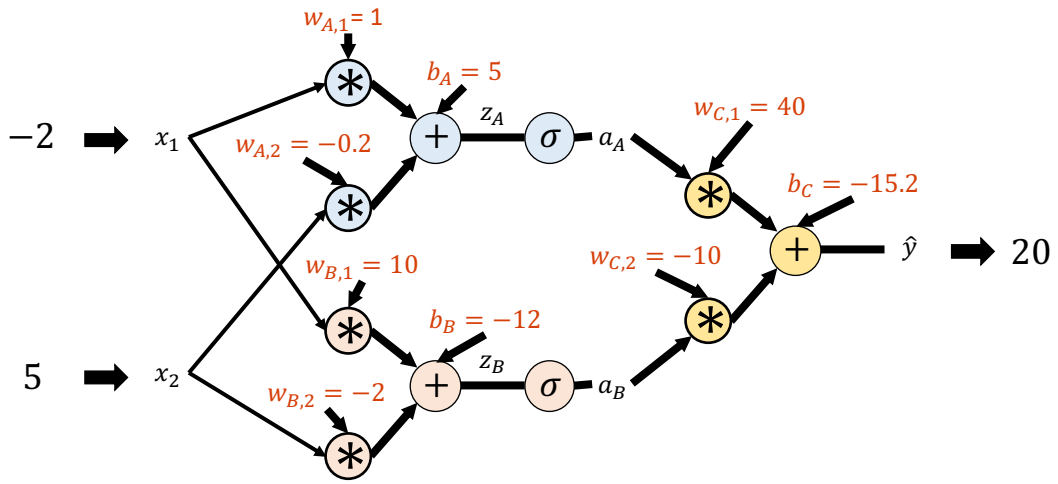
$$\frac{\delta \mathcal{L}(y,\hat{y})}{\delta w_{C,2}} = \frac{\delta \mathcal{L}(y,\hat{y})}{\delta \hat{y}} \frac{\delta \hat{y}}{w_{c,w}}$$

How do I train these things? An Example from Regression



How do I train these things? An Example from Regression

https://colab.research.google.com/drive/1aCyd04JhiLZEmDlTmsty_fmd-Ud2pOda?usp=sharing



$$z_A = w_{A,1}x_1 + w_{A,2}x_2 + b_A$$
$$= 1 * -2 + 5 * -0.2 + 5 = 2$$

$$a_A = \sigma(z_A) = \sigma(2) \approx 0.88$$

$$z_B = w_{B,1}x_1 + w_{B,2}x_2 + b_B$$
$$= 10 * -2 - 2 * 5 - 12 = -42$$

$$a_B = \sigma(z_B) = \sigma(-42) \approx 0$$

$$\hat{y} = w_{C,1}a_A + w_{C,2}a_B + b_C$$
$$= 40 * 0.88 \ - 10 * 0 - 15.2$$
$$\approx 20$$

Next time

Next Time: We'll keep talking about neural networks! Specifically, how to train them.