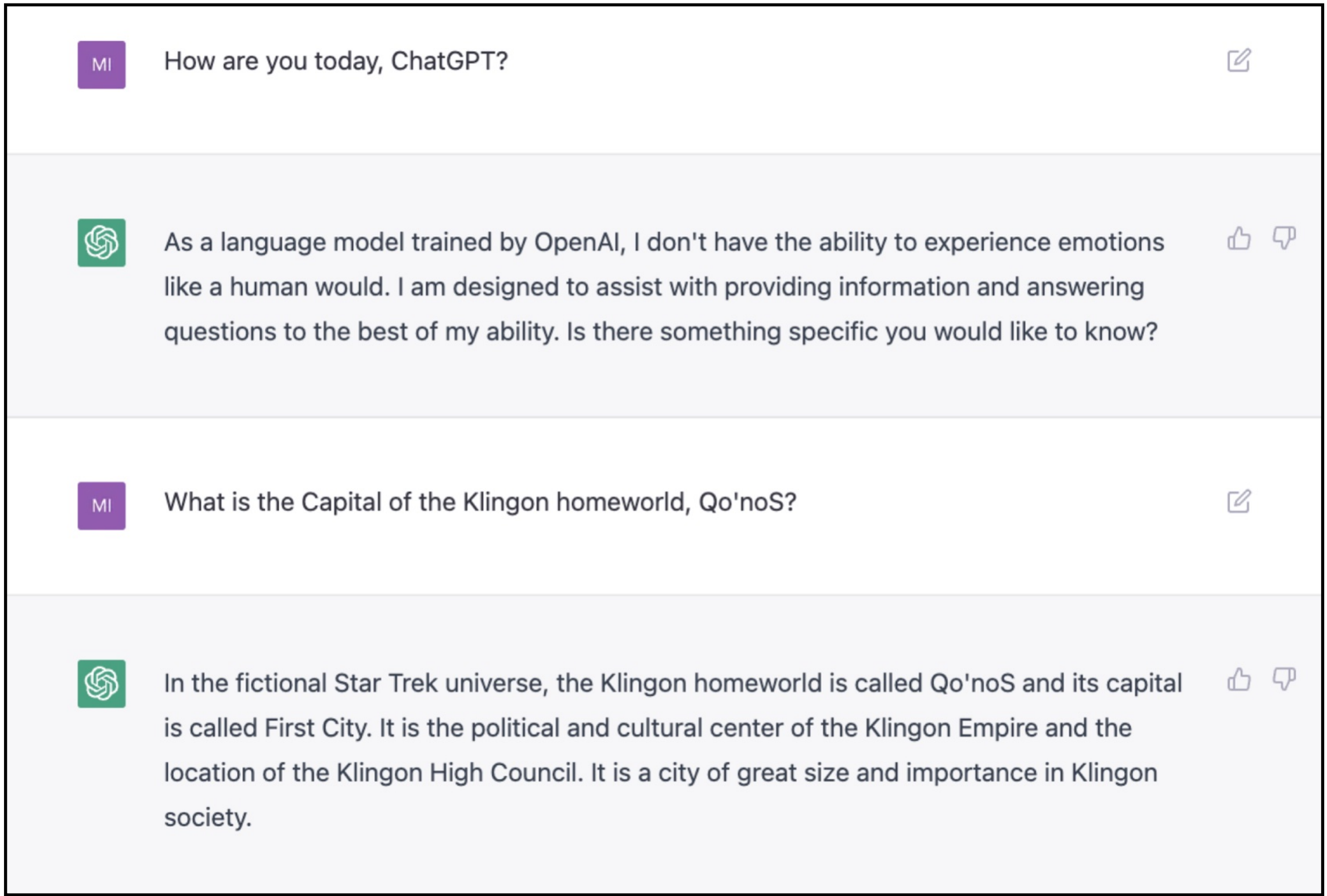
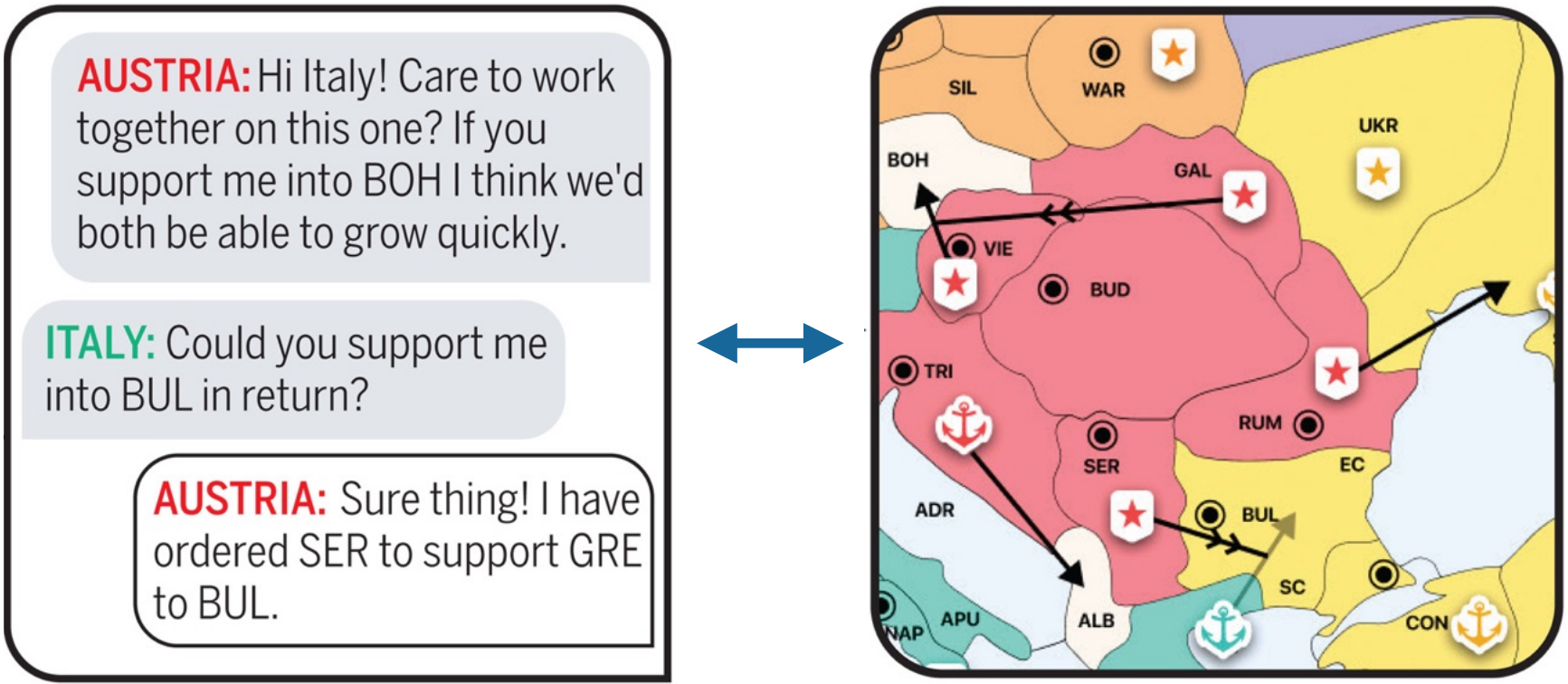


---

## Introduction to machine learning



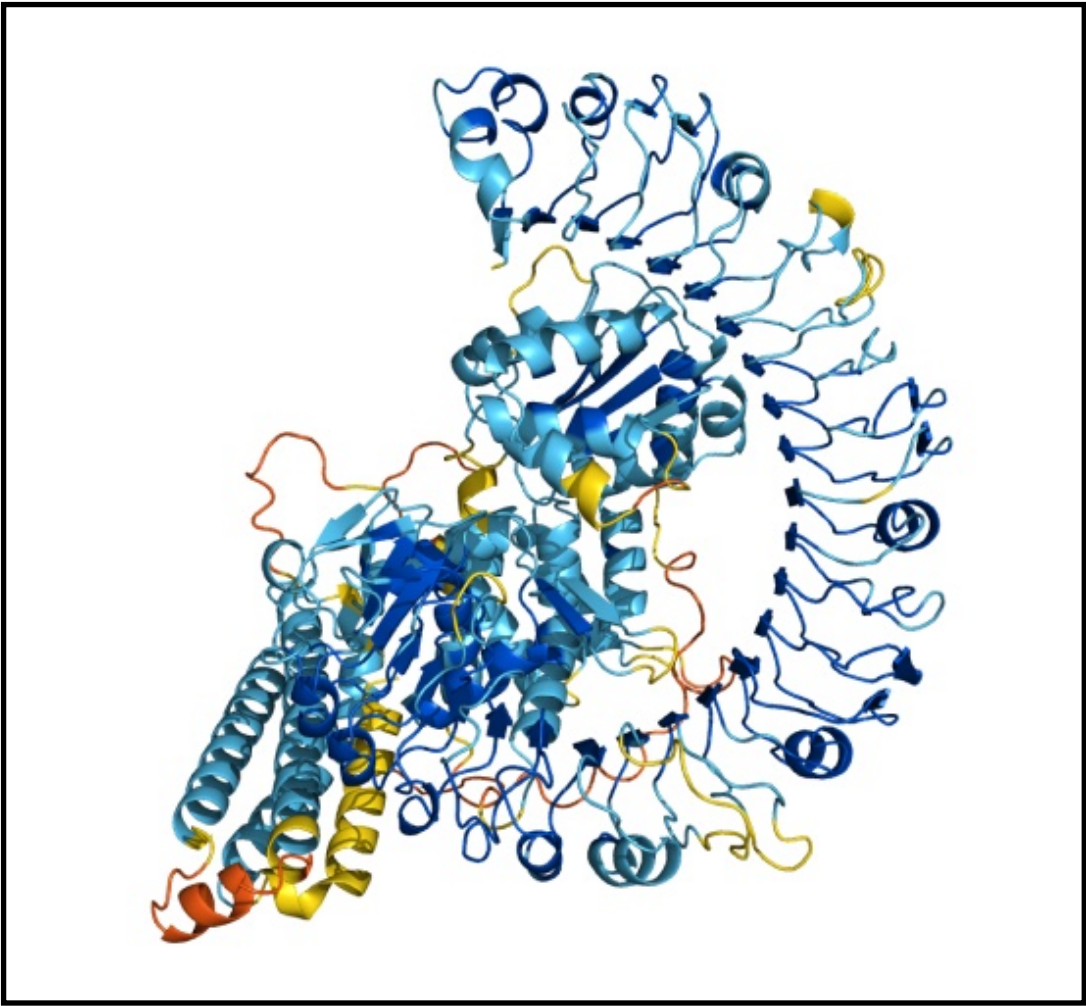
ChatGPT



Cicero



DALLE-2



AlphaFold



MI

How are you today, ChatGPT?

As a language model trained by OpenAI, I don't have the ability to experience emotions like a human would. I am designed to assist with providing information and answering questions to the best of my ability. Is there something specific you would like to know?

👍🗨

MI

What is the Capital of the Klingon homeworld, Qo'noS?

In the fictional Star Trek universe, the Klingon homeworld is called Qo'noS and its capital is called First City. It is the political and cultural center of the Klingon Empire and the location of the Klingon High Council. It is a city of great size and importance in Klingon society.

👍🗨

ChatGPT

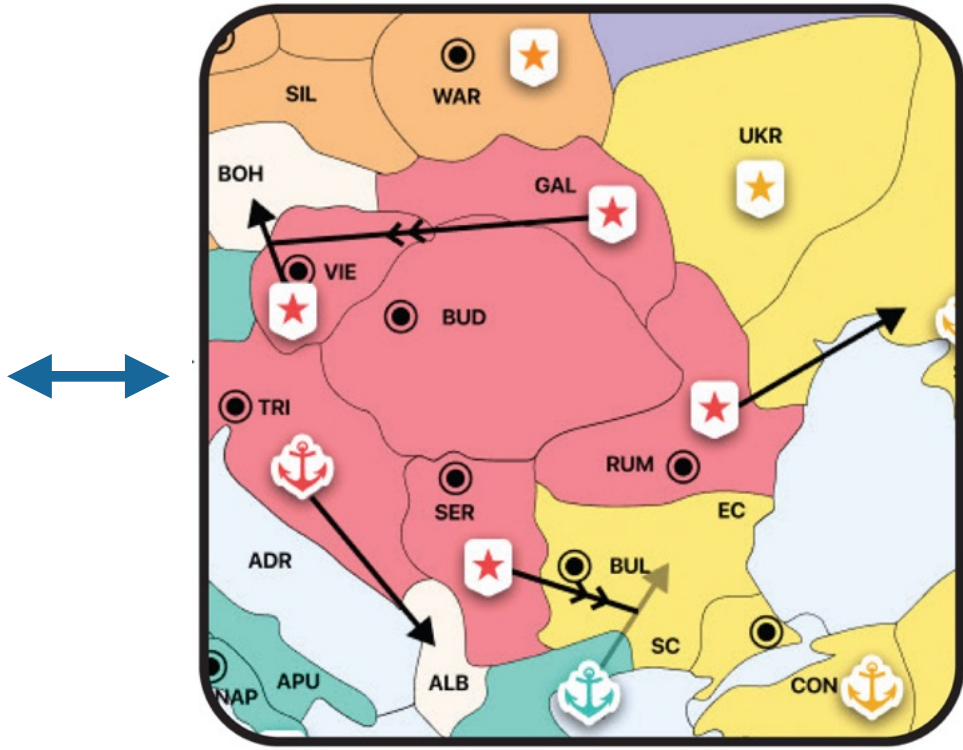


Disney Research

AUSTRIA: Hi Italy! Care to work together on this one? If you support me into BOH I think we'd both be able to grow quickly.

ITALY: Could you support me into BUL in return?

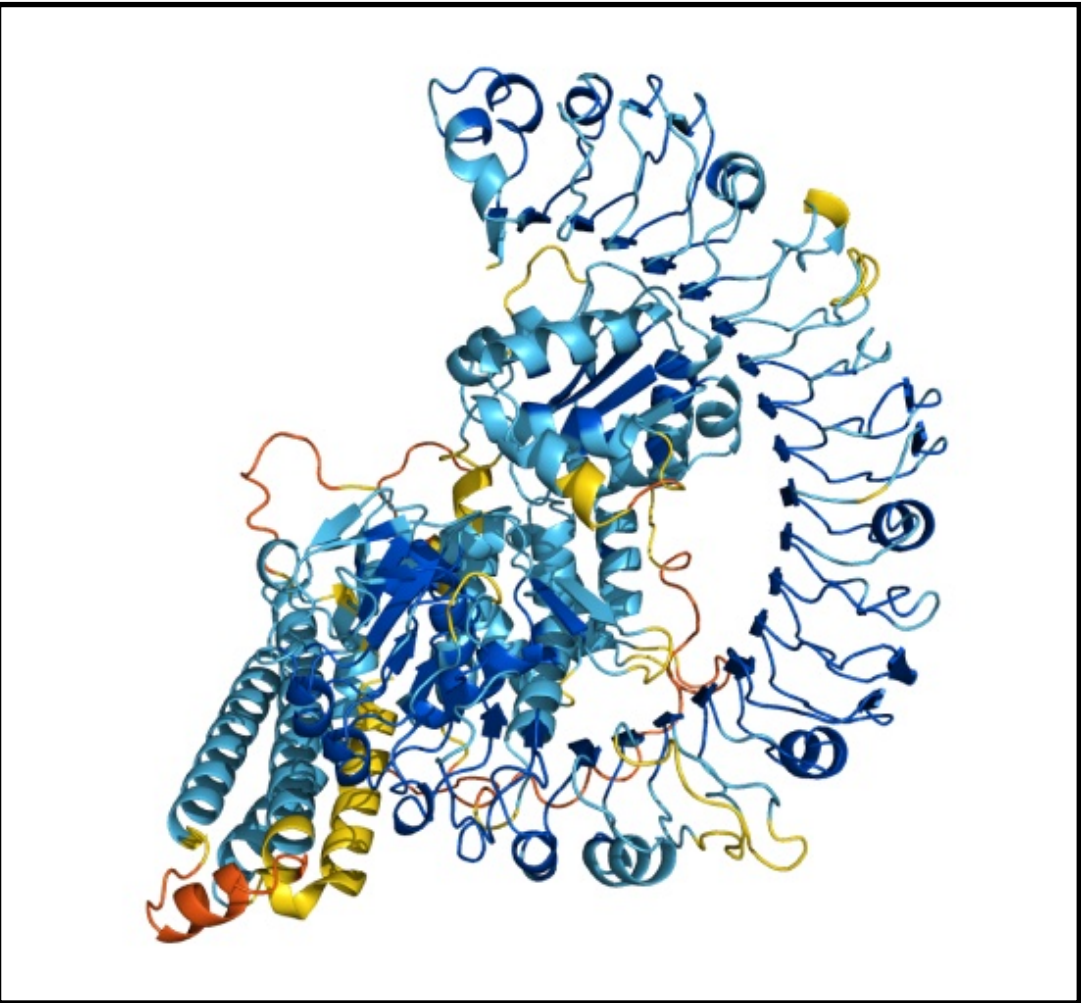
AUSTRIA: Sure thing! I have ordered SER to support GRE to BUL.



Cicero



DALLE-2



AlphaFold



# Roadmap

- What is machine learning?
- Machine learning paradigms
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning
- Perceptron
  - Example: Golden retriever vs. Doberman
  - Perceptron model
  - Training via update rule
  - Testing
- Multi-layer Perceptron (MLP)
- Multi-class Perceptron

# ML data $\Leftrightarrow$ related fields

Player	Minutes	Points	Rebounds	Assists
A	41	20	6	5
B	30	29	7	6
C	22	7	7	2
D	26	3	3	9

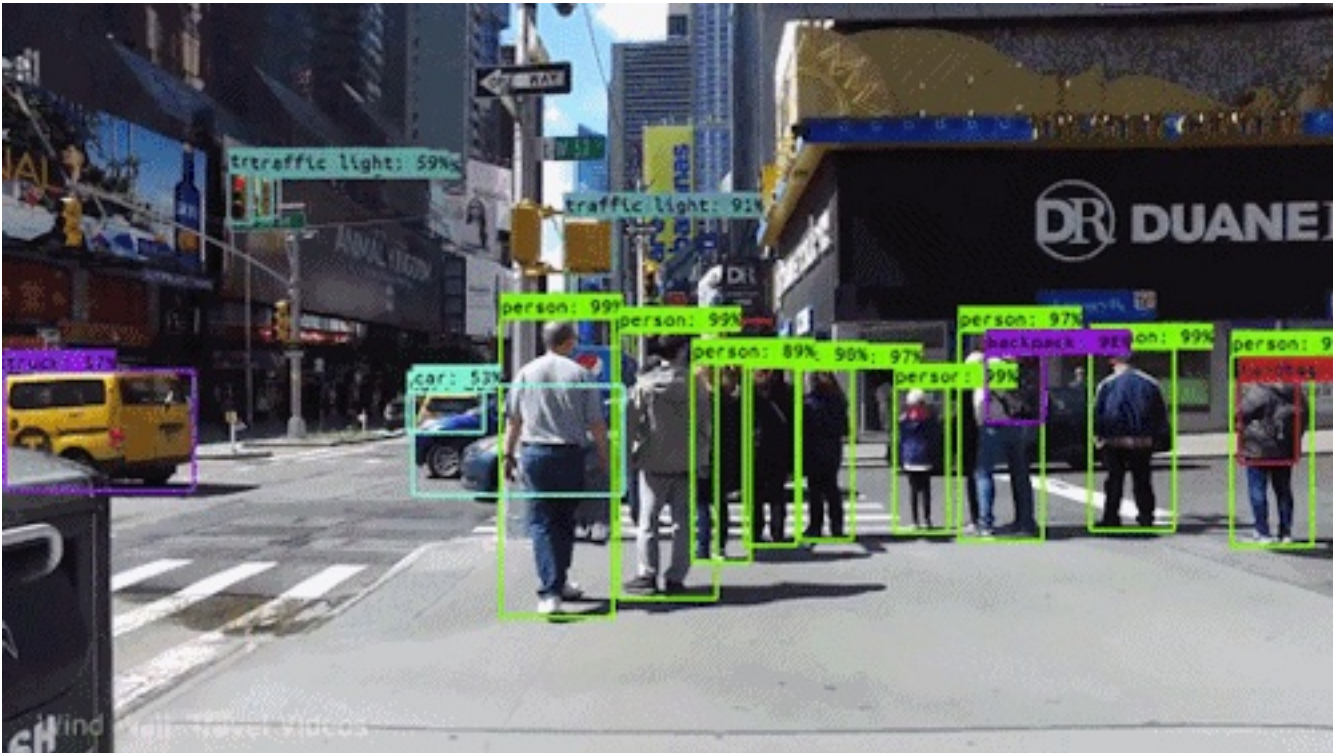
Tabular

**Machine learning**



Real manipulation

**Robotics**



Visual

**Computer vision**



Audio

**Audio processing**

ES

Briefly explain chatGPT using a metaphor that would resonate with an audience of stock-market investors and avoiding AI jargon.



One way to explain chatGPT to stock-market investors is to compare it to a financial advisor who uses past performance data and market trends to make predictions about future investments. Just like a skilled financial advisor, chatGPT uses large amounts of training data to generate intelligent and informed responses to questions, providing valuable insights and suggestions to users.

Text

**Natural language processing (NLP)**

# What is machine learning?

## **Machine learning (ML):**

Algorithms that improve automatically using data or repeated experience.

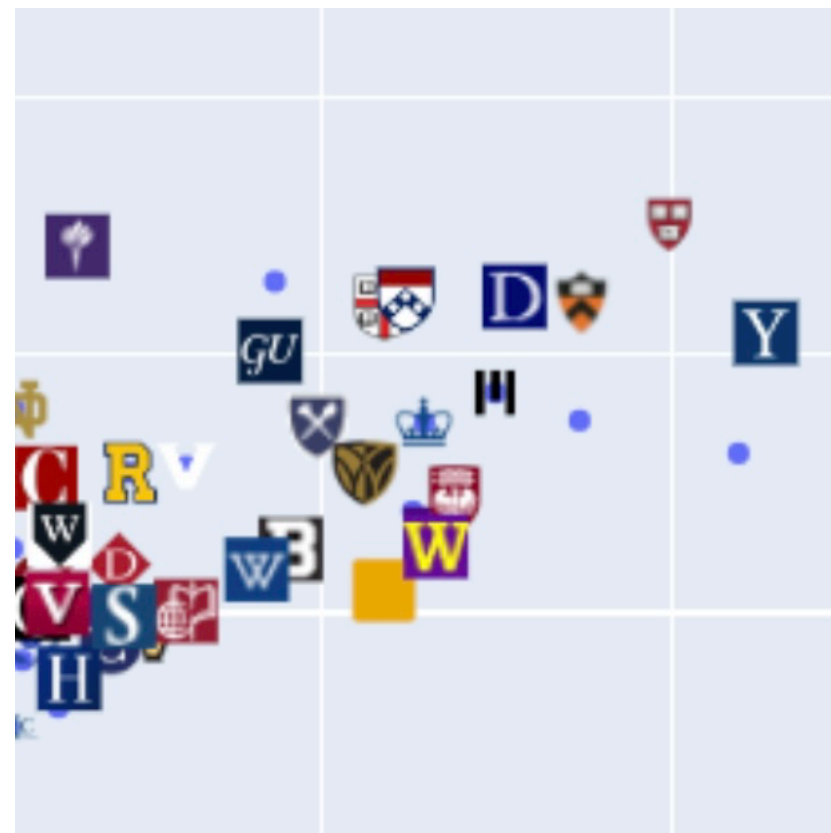
Notes:

- This was **not** true of the programs that you written so far in COS126 (e.g. your programs = explicit instructions for the computer).
- There are many non-ML algorithms in computer vision, NLP, robotics, etc.



# Machine learning paradigms

1. Supervised learning: learn to **predict** an output from an input.
2. Unsupervised learning: learn **patterns** from data.
3. Reinforcement learning: learn by **interacting** with an environment (i.e. **trial-and-error**).



# Supervised learning

Given (input, output) pairs, **predict** an output from its input.

input<sub>1</sub> → output<sub>1</sub>

input<sub>2</sub> → output<sub>2</sub>

input<sub>3</sub> → output<sub>3</sub>





# Supervised learning

## Medicine

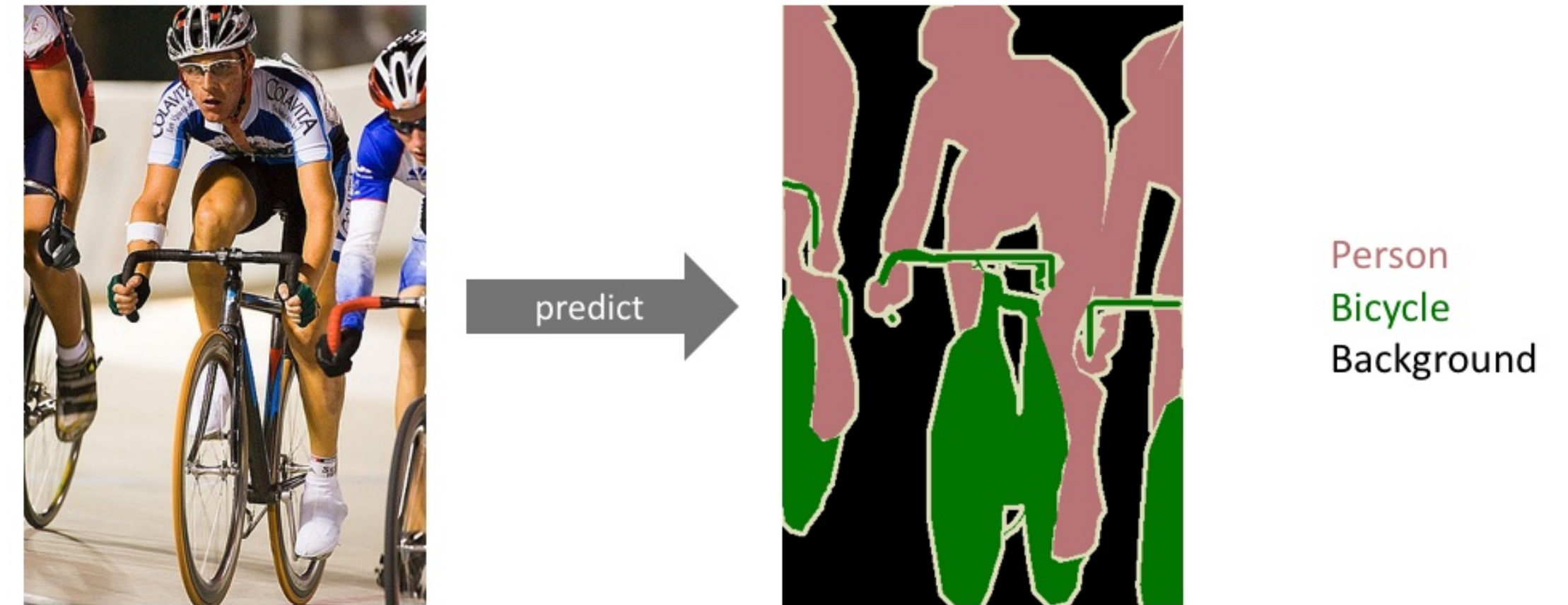
- Input: medical information (i.e. age, body mass index [BMI], blood sugar levels, etc.)
- Output: blood pressure (in mm Hg)



# Supervised learning

## Computer vision: segmentation

- Input: an image
- Output: segmentation map of objects
- **Application: self-driving cars**



Pascal VOC dataset

<http://host.robots.ox.ac.uk/pascal/VOC/voc2012/>



<https://github.com/meetps/pytorch-semseg>



# Supervised learning

## NLP: Machine translation

- Input: a sentence in English
- Output: its translation in Spanish
- **Application: Google Translate**



Gender Bias in Machine Translation:  
<https://towardsdatascience.com/gender-bias-in-machine-translation-819ddce2c452>

“That river is dangerous to swim in.” → “Es peligroso nadar en ese río.”

“The baby is playing with some toys.” → “El bebé juega con algunos juguetes.”

Anki Tab-Delimited Bilingual Sentence Pairs  
<https://www.manythings.org/anki/>

# Supervised learning

## Other applications

- Spam filtering: email text → {spam, not spam}
- Speech recognition: audio clip → text transcription
- Medical imaging: CT scan of COVID patients → severity of COVID symptoms
- ...

If you can create a dataset of (input, output) pairs, you can use supervised learning.



# Unsupervised learning

Learn **patterns** from unlabelled data (i.e. no input-output pairs).

Examples:

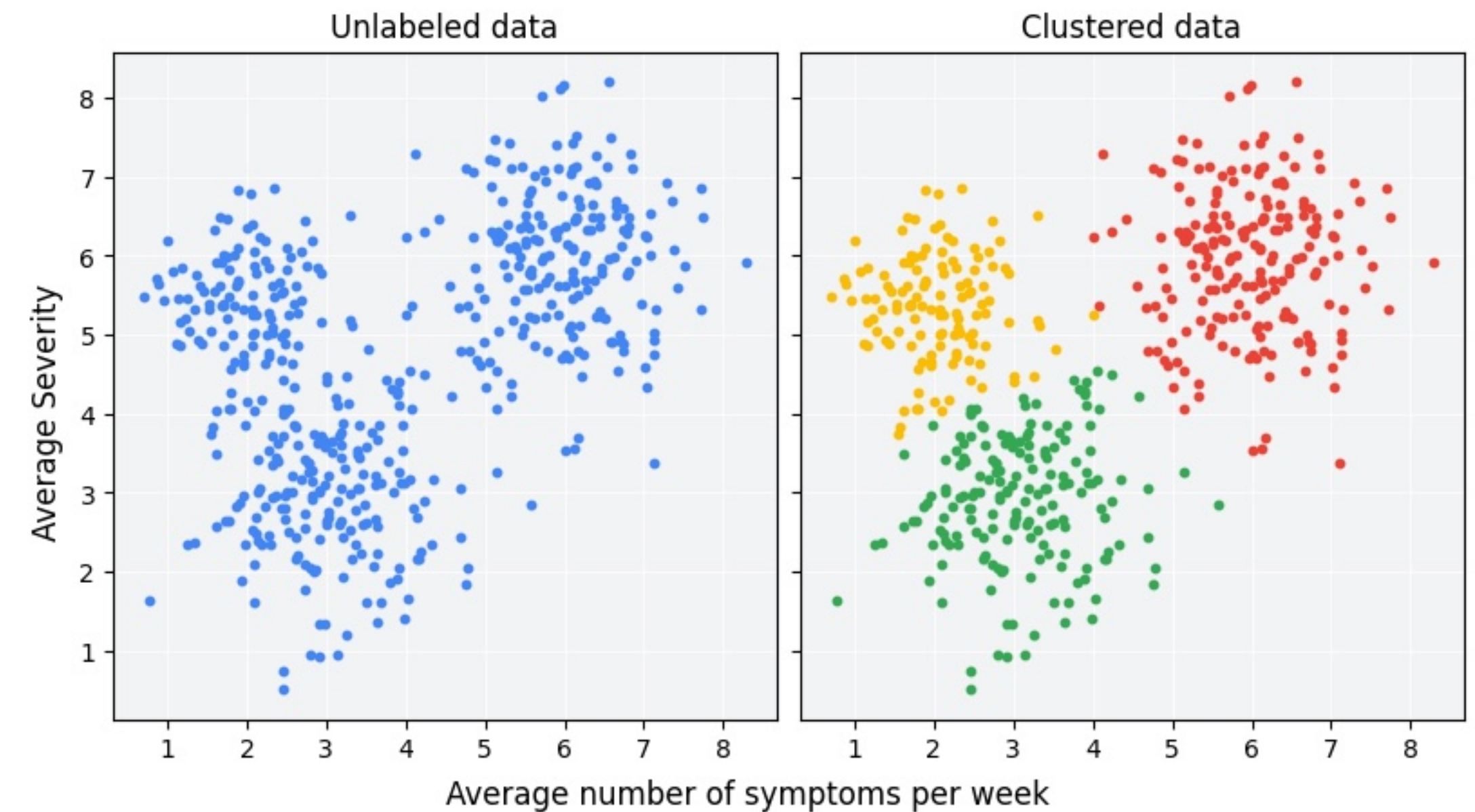
- Clustering
- Dimensionality reduction
- Anomaly detection
- Generative modeling
- ...



# Unsupervised learning

## Clustering

- Goal: Given a data point, categorize it into one of  $k$  clusters.
- Unlike supervised learning, there are no output labels! i.e. no (data point, cluster) pairs
- Application in retail marketing: What kinds of people have similar shopping habits?

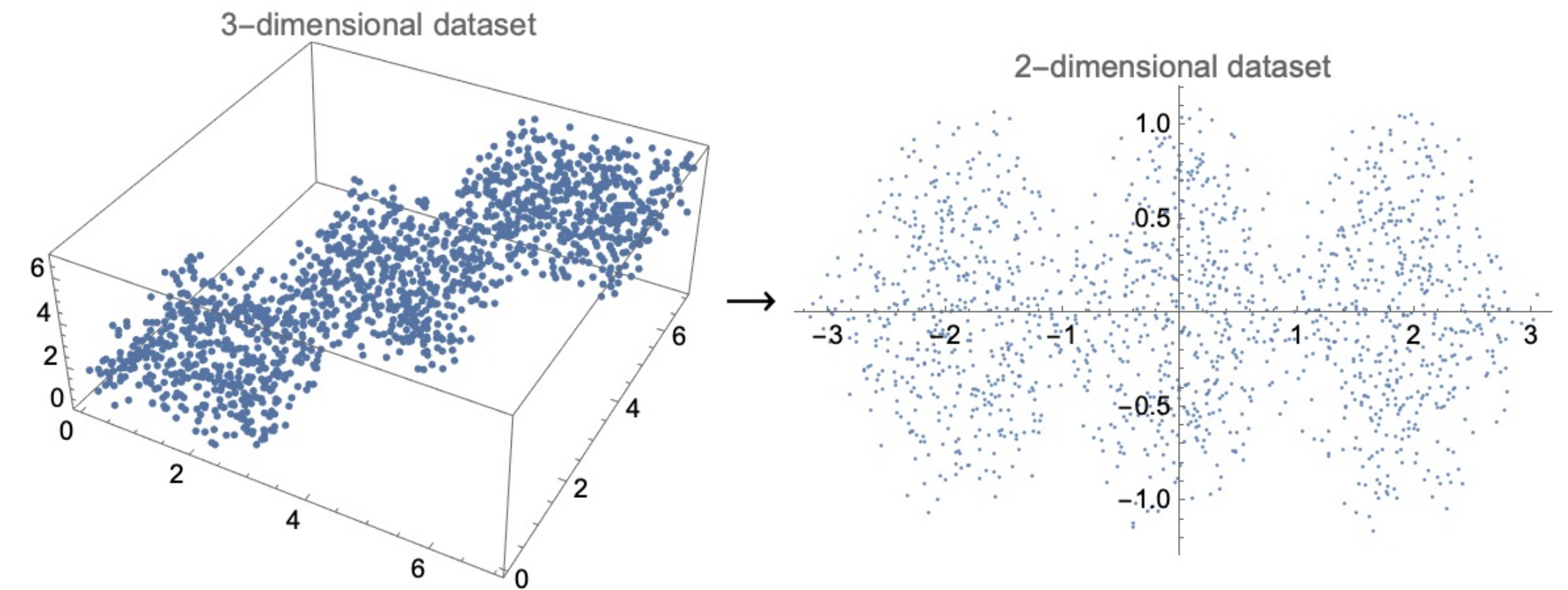




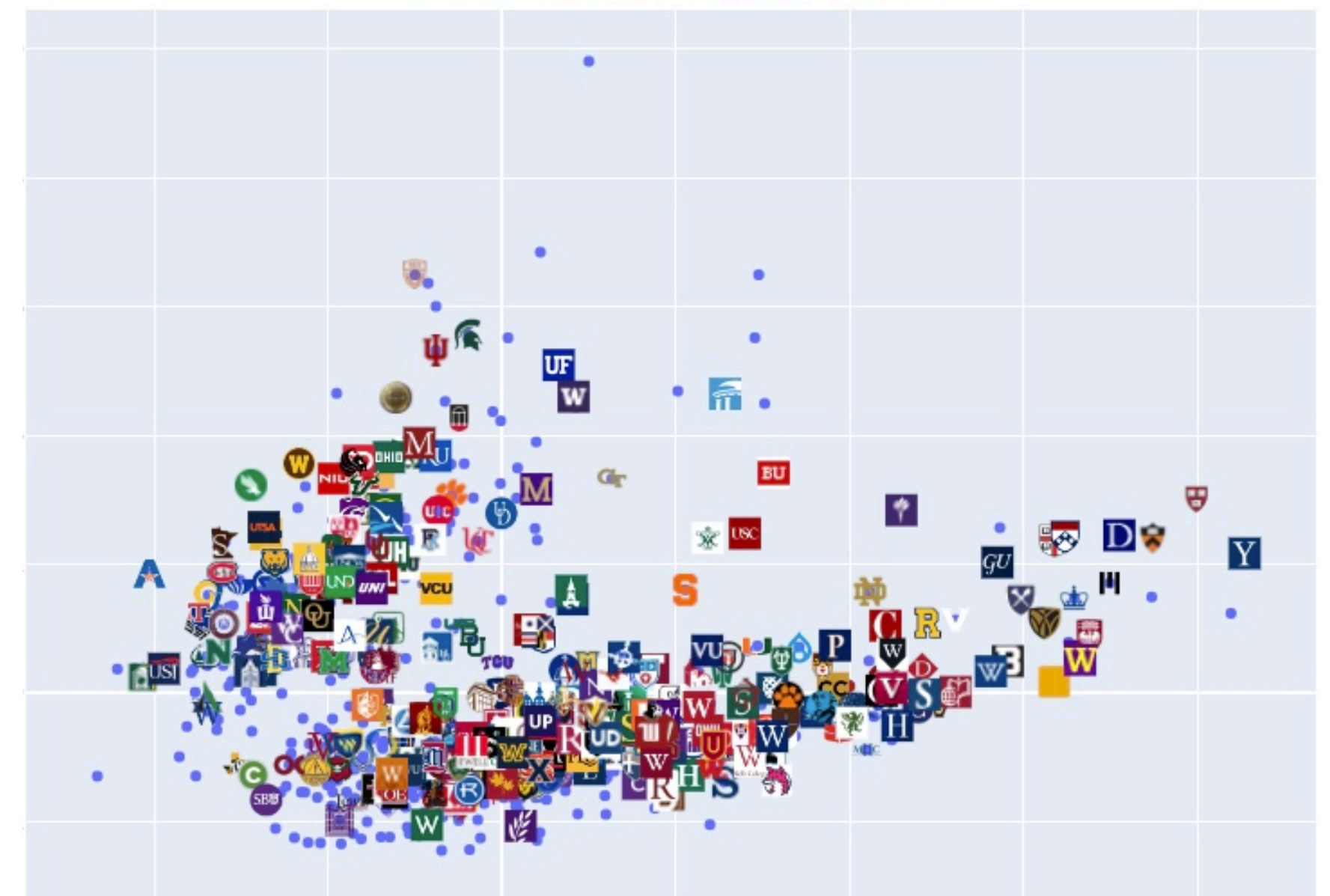
# Unsupervised learning

## Dimensionality reduction

- Goal: Reduce number of variables while preserving patterns (e.g. distances between points).
- Uses:
  - Compression
  - Visualization
  - ...



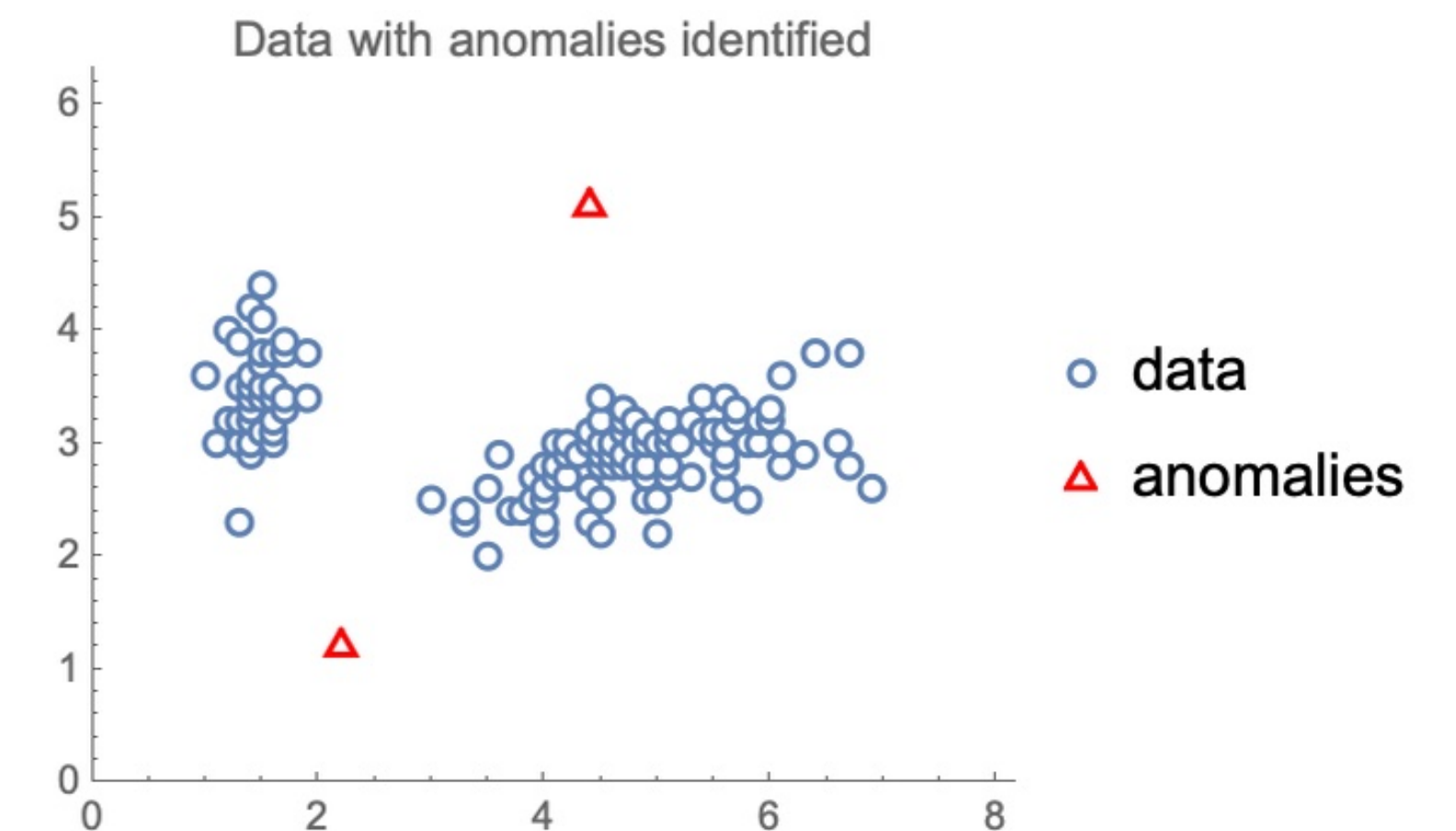
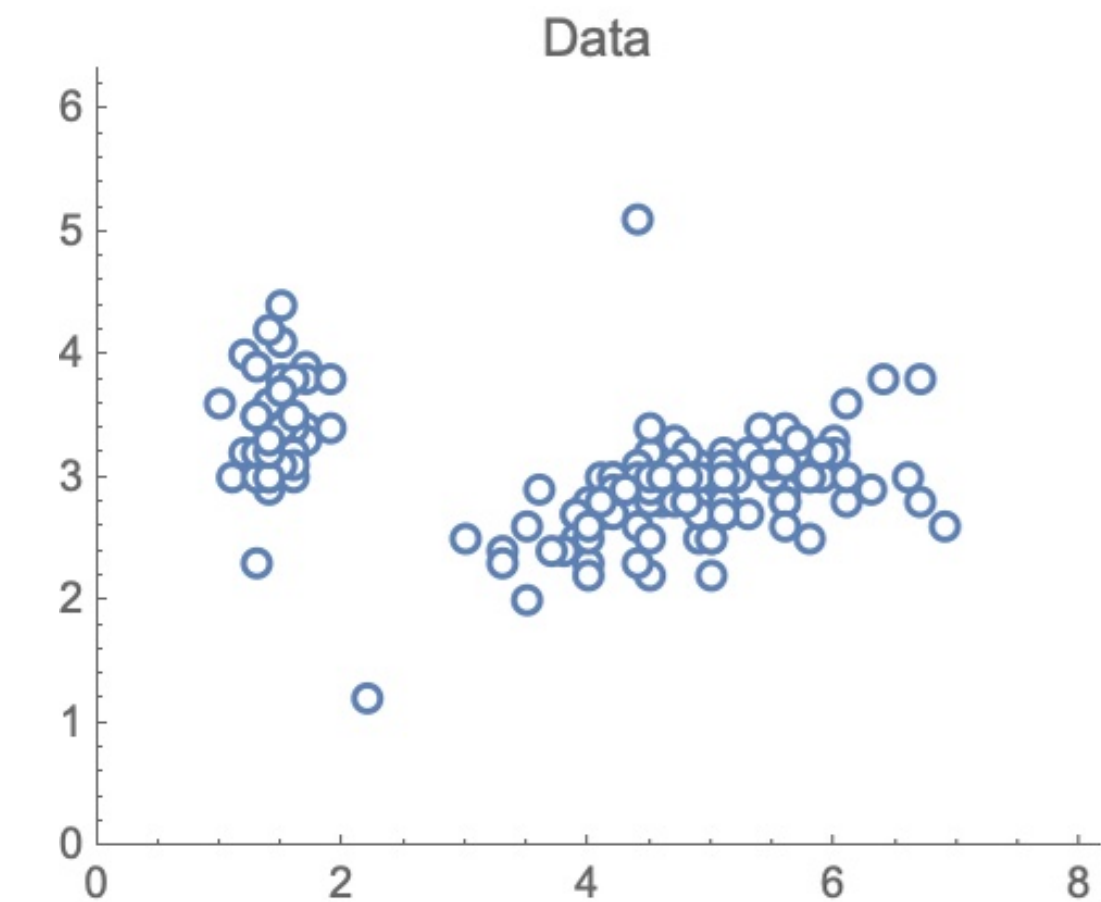
PCA example of colleges dataset



# Unsupervised learning

## Anomaly detection

- Goal: Identify anomalies (outliers)
- Applications:
  - Credit card fraud
  - Medical diagnosis
  - ...





# Unsupervised learning

## Generative model

- Goal: Create synthetic examples that look realistic.
- Applications:
  - Computer vision:  
Given a dataset portraits, generate realistic looking faces.
  - NLP:  
Given Shakespeare's sonnets, generate a realistic sounding, Shakespearean sonnet.



Generated by StyleGAN

(Karras, et al., CVPR 2019)

*“am of my faults thy sweet self dost deceive:  
they look into the beauty of thy deeds  
nativity, once in the chronicle of wasted time  
not from the thing they see”*

Generated by Markov Chains

<https://rpubs.com/malcolmbarrett/shakespeare>



# Reinforcement learning



Learn by **interacting** (i.e. **trial-and-error**) with an *environment*.

- An *agent* interacts with an *environment* by performing *actions* and then gathers *observations* (and sometimes *rewards*) that occur in response to its actions
- Interactions generate data (i.e. no fixed dataset).
- Goal: Learn *agent* to maximize *reward*.

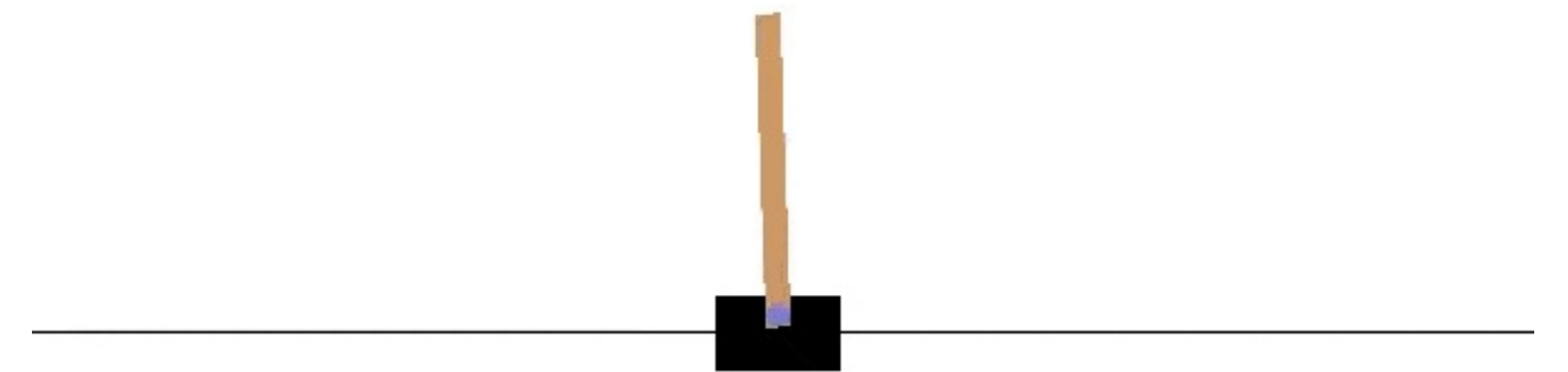




# Reinforcement learning

## CartPole

- Goal: Balance a pole on a cart by moving it left and right.
- Actions: {left, right}
- Reward: +1, for each step the pole is upright.
- Learn over many attempts (i.e. episodes).

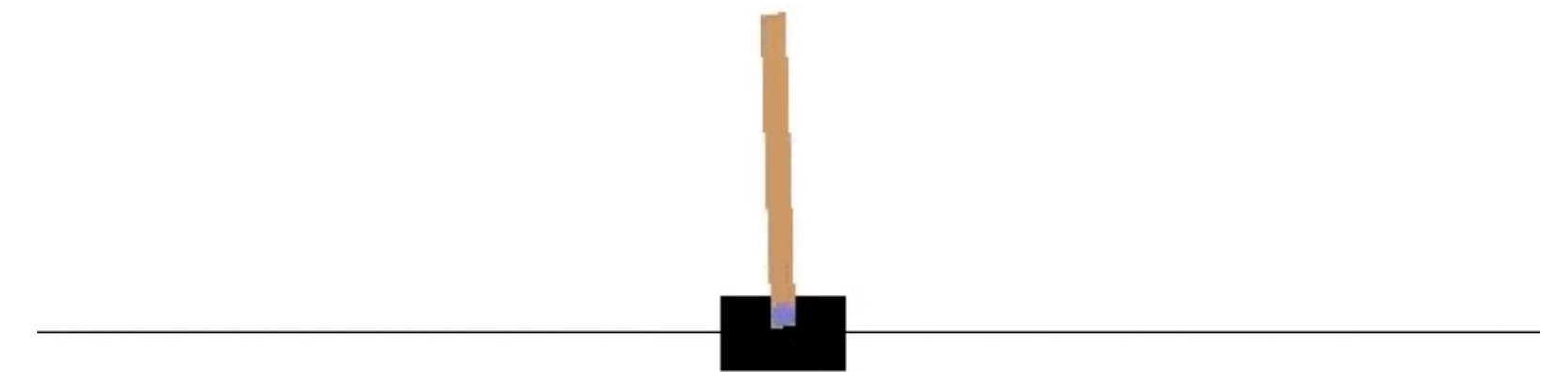


[https://gymnasium.farama.org/environments/classic\\_control/cart\\_pole/](https://gymnasium.farama.org/environments/classic_control/cart_pole/)

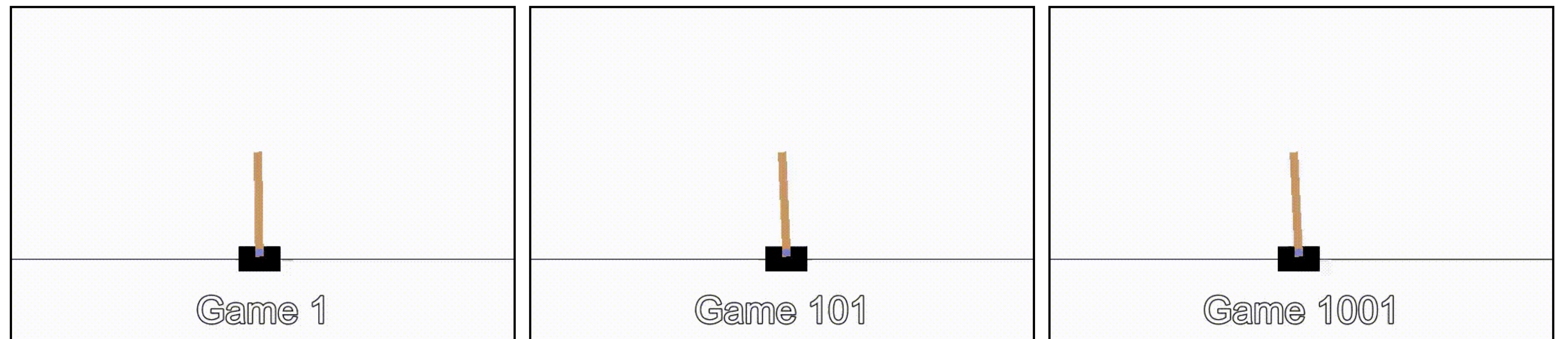
# Reinforcement learning

## CartPole

- **Goal:** Balance a pole on a cart by moving it left and right.
- **Actions:** {left, right}
- **Reward:** +1, for each step the pole is upright.
- Learn over many attempts (i.e. episodes).



[https://gymnasium.farama.org/environments/classic\\_control/cart\\_pole/](https://gymnasium.farama.org/environments/classic_control/cart_pole/)



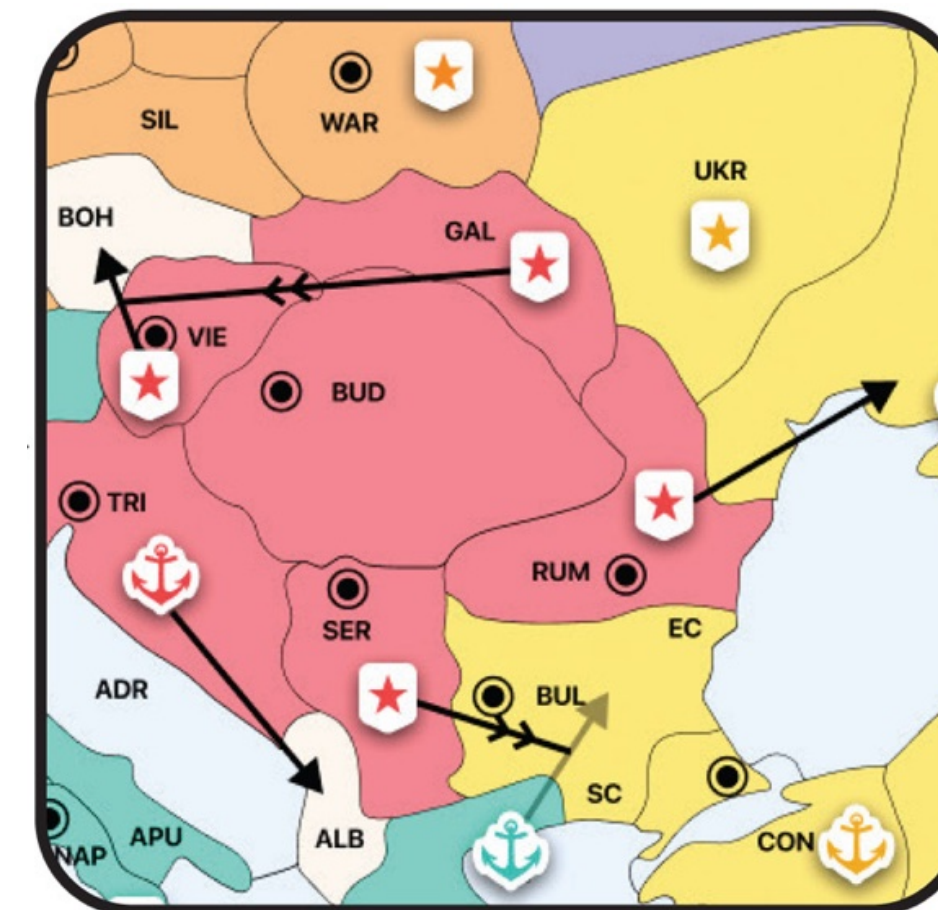
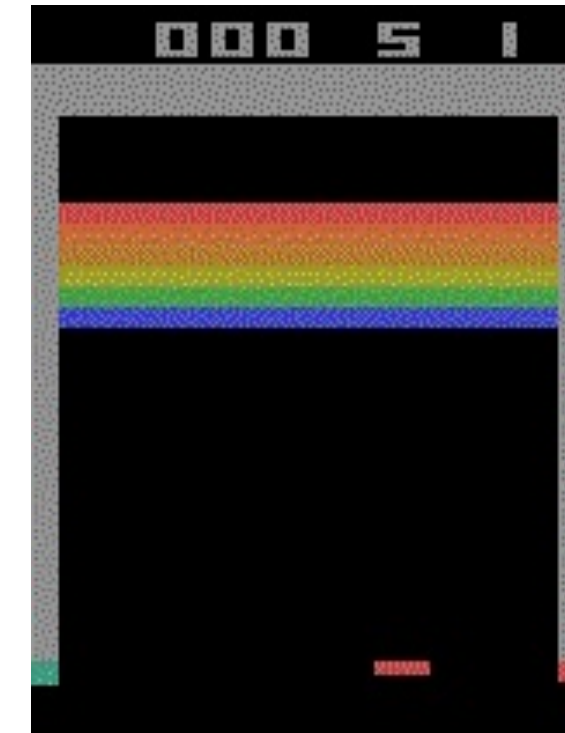
<https://blog.ml6.eu/importance-of-data-visualization-for-ml-936239c355d0>



# Reinforcement learning

## Application: Games

- Atari (2015): DeepMind
- Go (2016): AlphaGo from DeepMind
- Poker (2017, 2019):  
Libratus and Pluribus from CMU
- Diplomacy (2022): Cicero from Meta AI



**AUSTRIA:** Hi Italy! Care to work together on this one? If you support me into BOH I think we'd both be able to grow quickly.

**ITALY:** Could you support me into BUL in return?

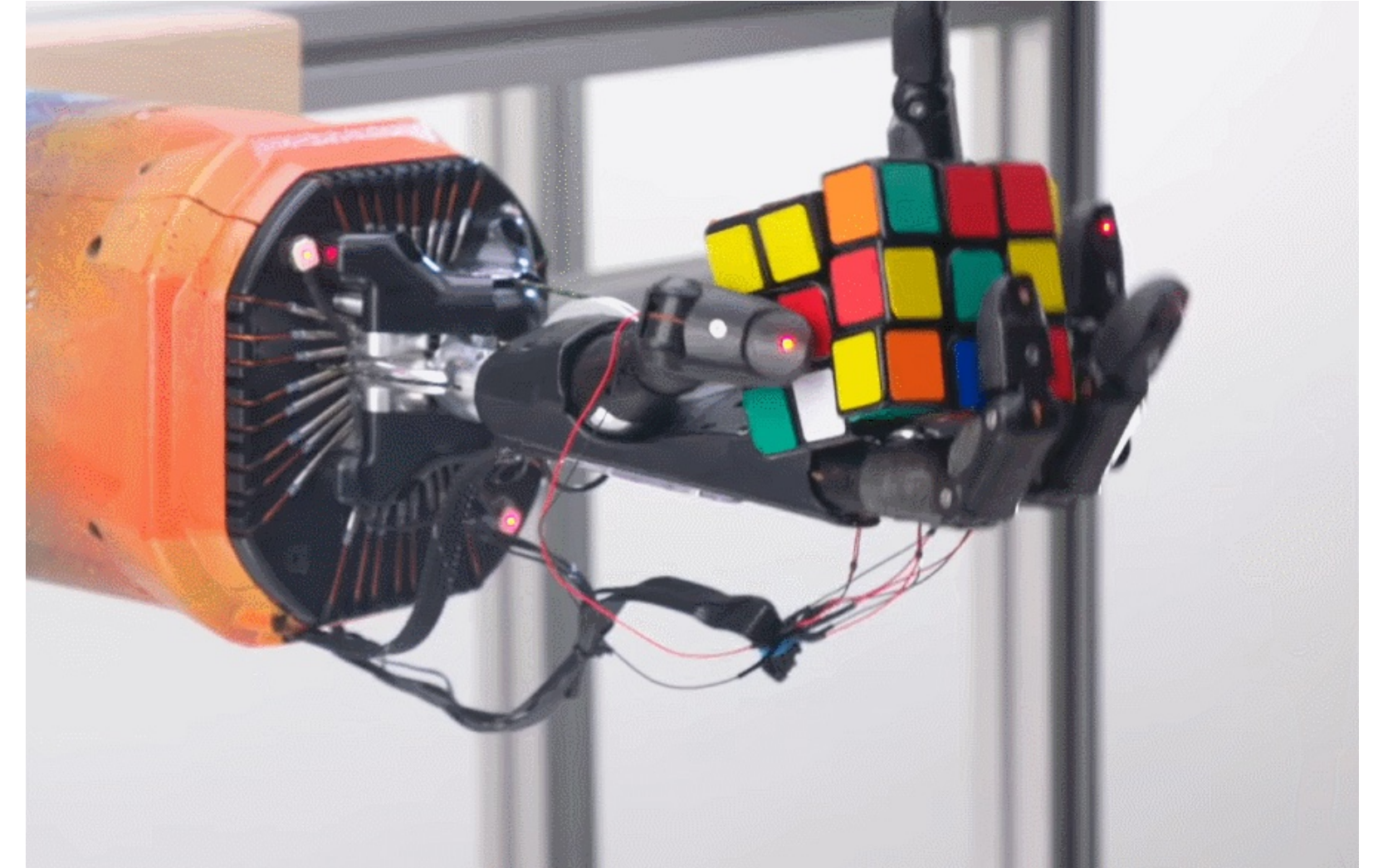
**AUSTRIA:** Sure thing! I have ordered SER to support GRE to BUL.



# Reinforcement learning

## **Application: Robotics**

- Solving a Rubik's cube



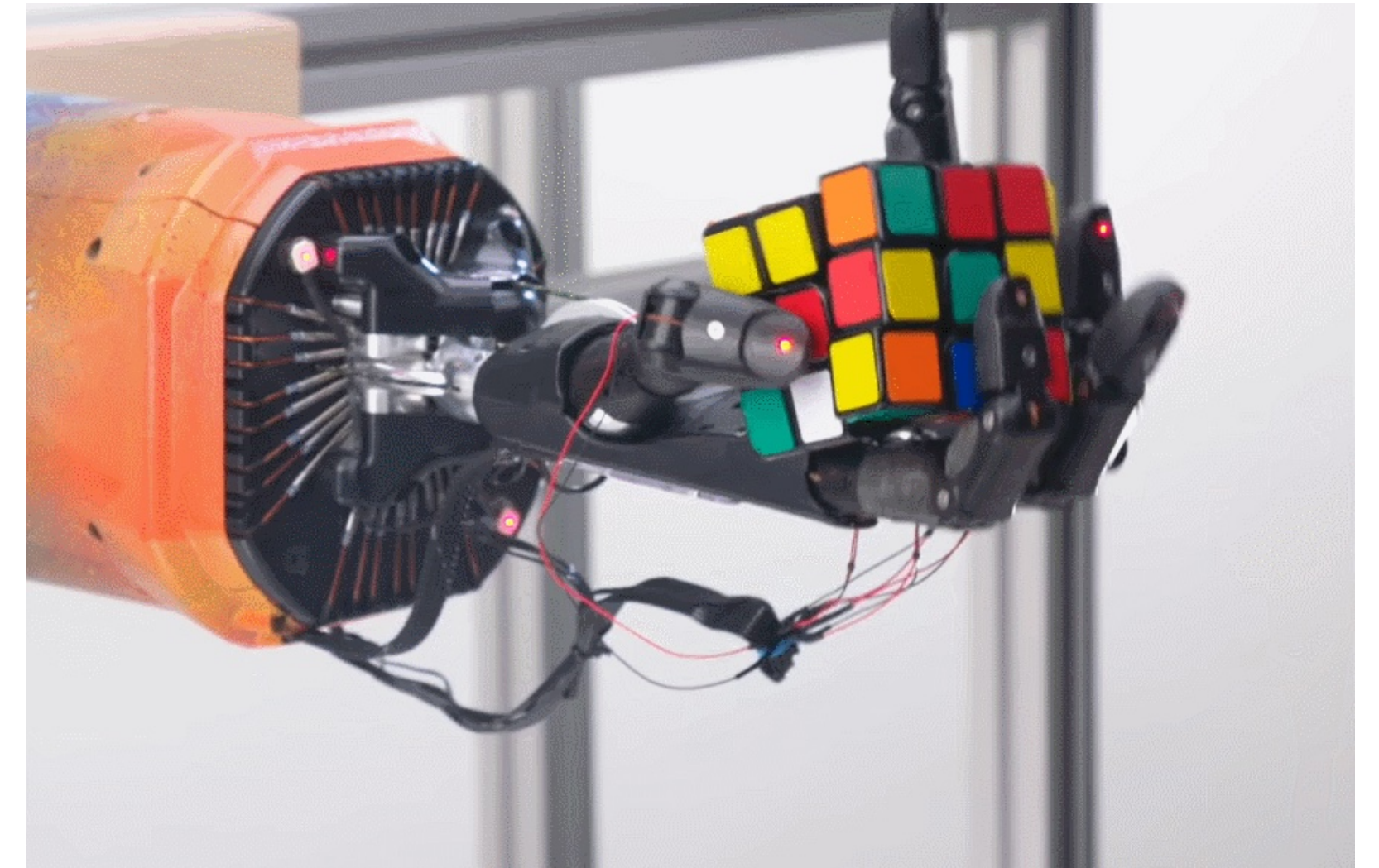
OpenAI's ADR (2019)



# Reinforcement learning

## Application: Robotics

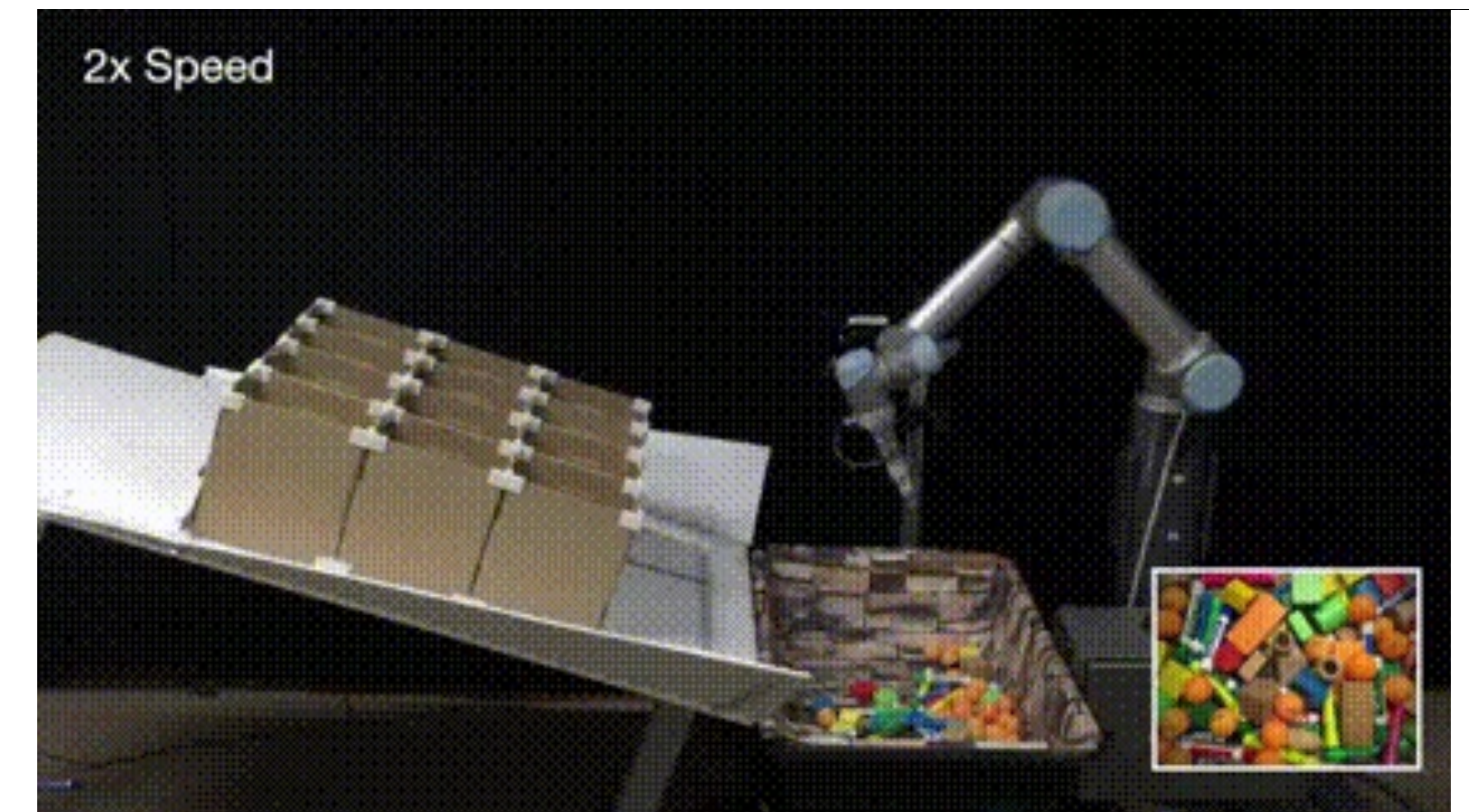
- Solving a Rubik's cube
- Manipulating objects (e.g. grasp, throw, etc.)
- Laundry folding



OpenAI's ADR (2019)

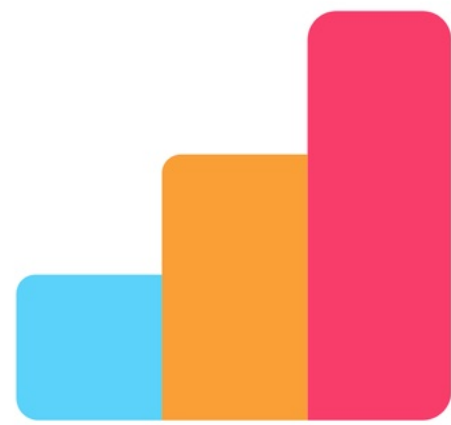


Berkeley's BRETT (2015)



Princeton+Google's TossingBot (2019)



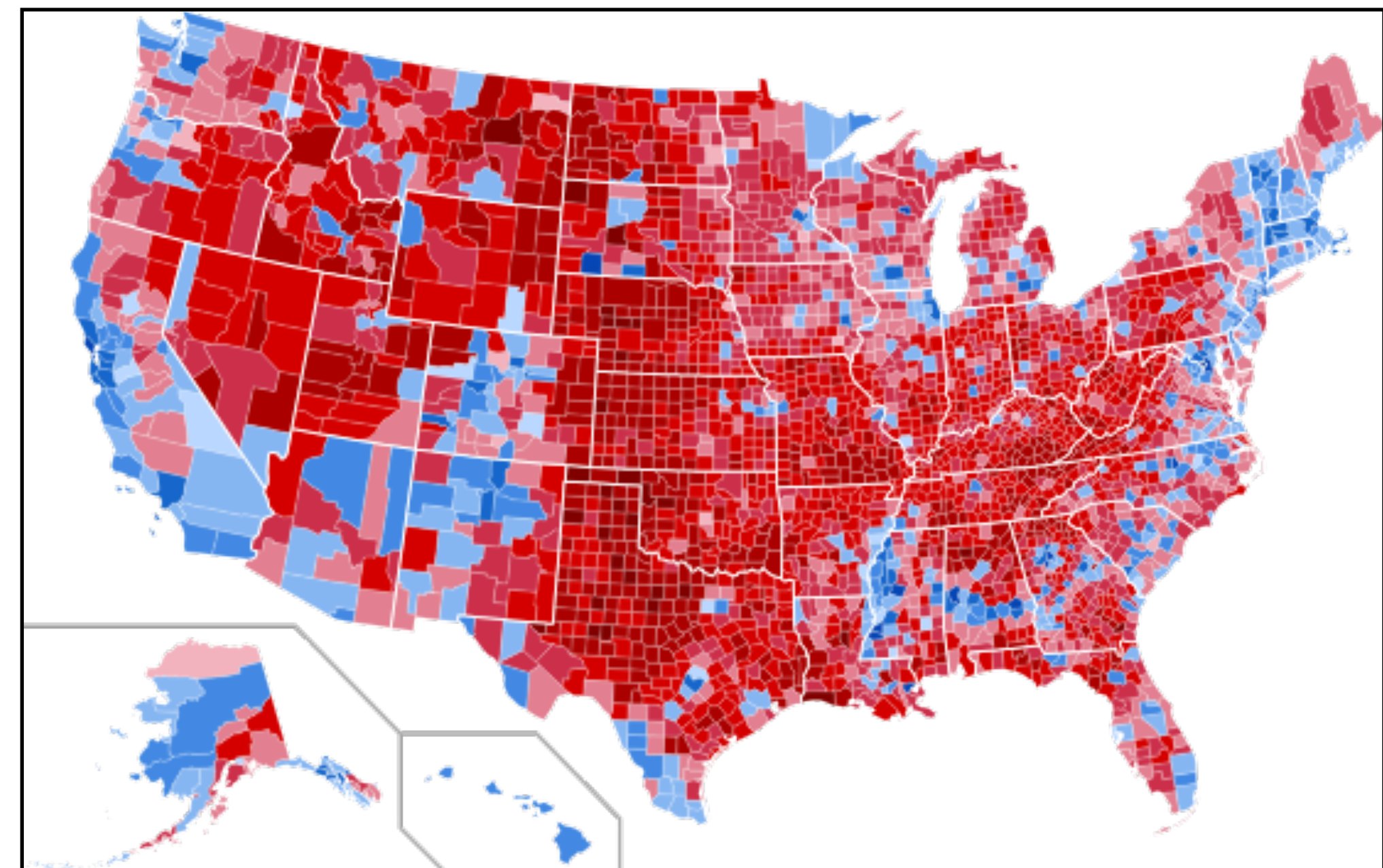


# Poll

*Predict the outcome of the next U.S. presidential election based on polling data and data from prior presidential elections.*

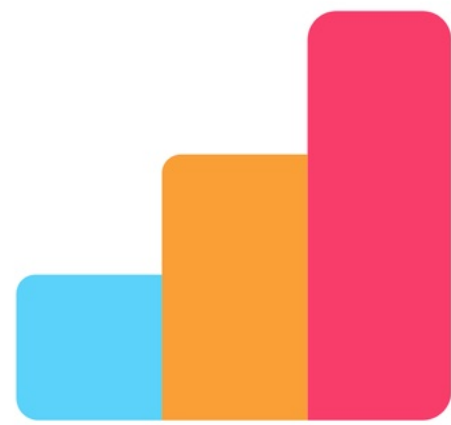
*What kind of machine learning problem is this?*

- a. Supervised learning
- b. Unsupervised learning
- c. Reinforcement learning



[https://commons.wikimedia.org/wiki/File:2020\\_United\\_States\\_presidential\\_election\\_results\\_map\\_by\\_county.svg](https://commons.wikimedia.org/wiki/File:2020_United_States_presidential_election_results_map_by_county.svg)



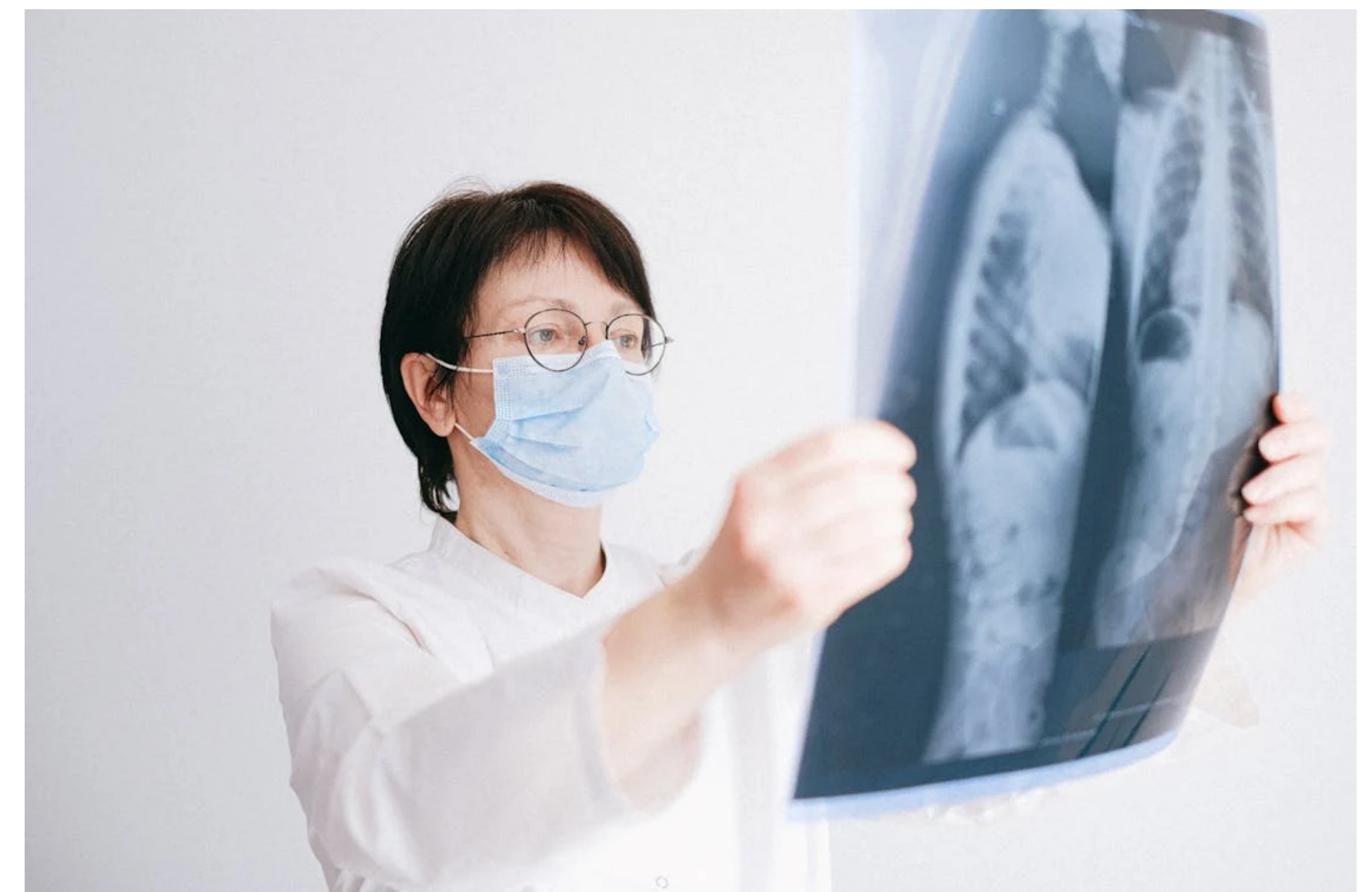


# Poll

*Predict if a patient's lung scan is abnormal based on a huge dataset of lung scans.*

*What kind of machine learning problem is this?*

- a. Supervised learning
- b. Unsupervised learning
- c. Reinforcement learning



<https://www.pexels.com/photo/doctor-looking-at-lung-scans-she-is-holding-4225926/>

# Machine learning paradigms

1. Supervised learning: learn to **predict** an output from an input.
2. Unsupervised learning: learn **patterns** from data.
3. Reinforcement learning: learn by **interacting** with an environment (i.e. **trial-and-error**).

Other paradigms:

- *Self-supervised learning*
- *Semi-supervised learning*
- *Transfer learning*
- *Online learning*
- ...



# Ingredients of ML

1. **Data.** Can be labelled, unlabelled, or from experience.
2. **Model.** A model maps from datapoint to a desired answer or output.
3. **Model parameters.** Internal parameters of model that are learnable.
4. **Training.** Given datapoints, find good model parameters.
5. **Testing.** Evaluate the performance of learned model  
... on new, previously unseen data (i.e. test set).

---

# Perceptron



# Supervised learning

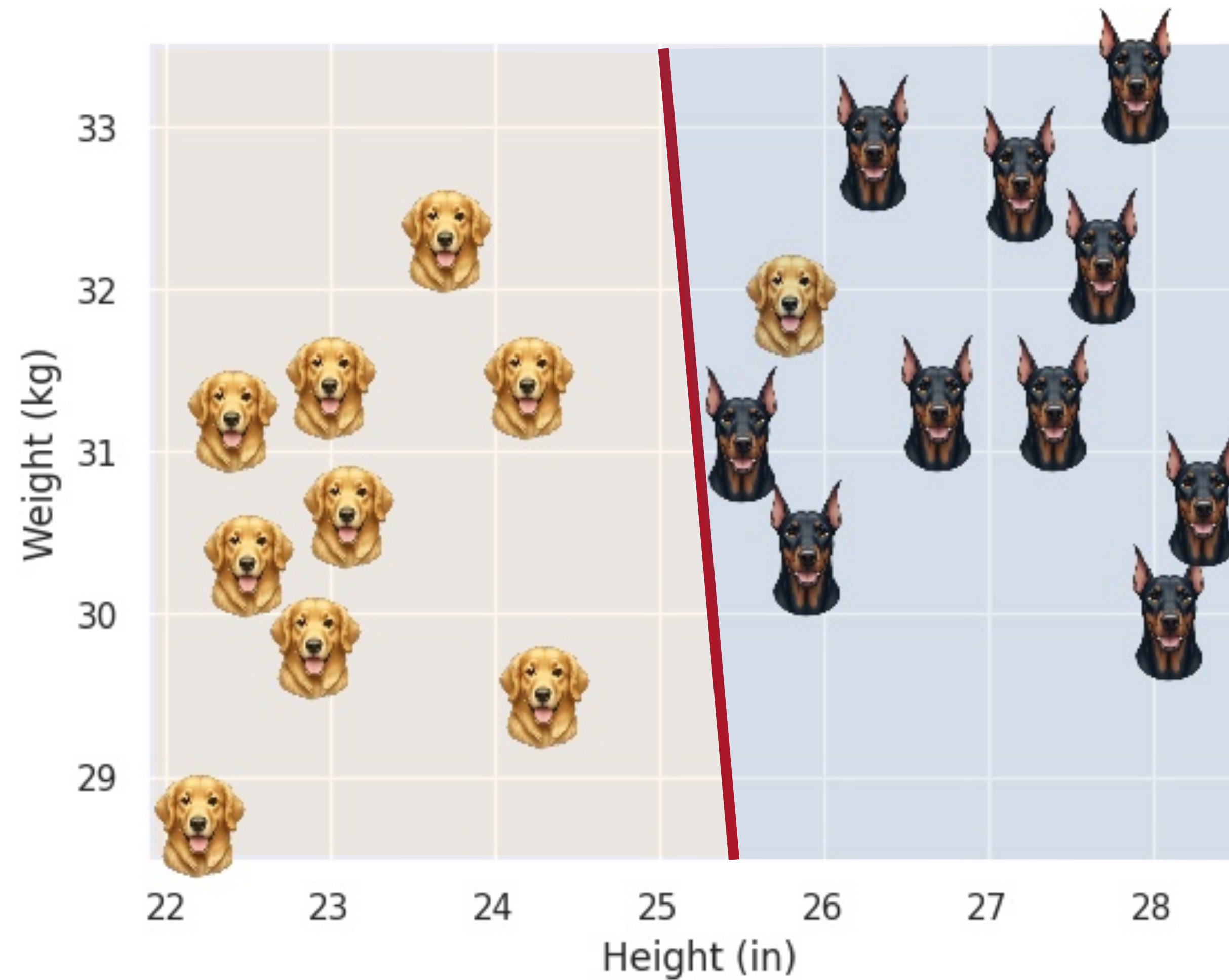
Given (input, output) pairs, **predict** an output from its input.

input<sub>1</sub> → output<sub>1</sub>  
input<sub>2</sub> → output<sub>2</sub>  
input<sub>3</sub> → output<sub>3</sub>

## Basic problems described by output type:

- **Regression**: Predict a continuous value (e.g. number).
- **Classification**: Predict a discrete category (i.e. classes).
  - **Binary classification**: Predict one of **two** classes.
  - **Multi-class classification**: Predict one of **k** classes ( $k > 2$ ).

# Binary classification



Data is nearly **linearly separable**.

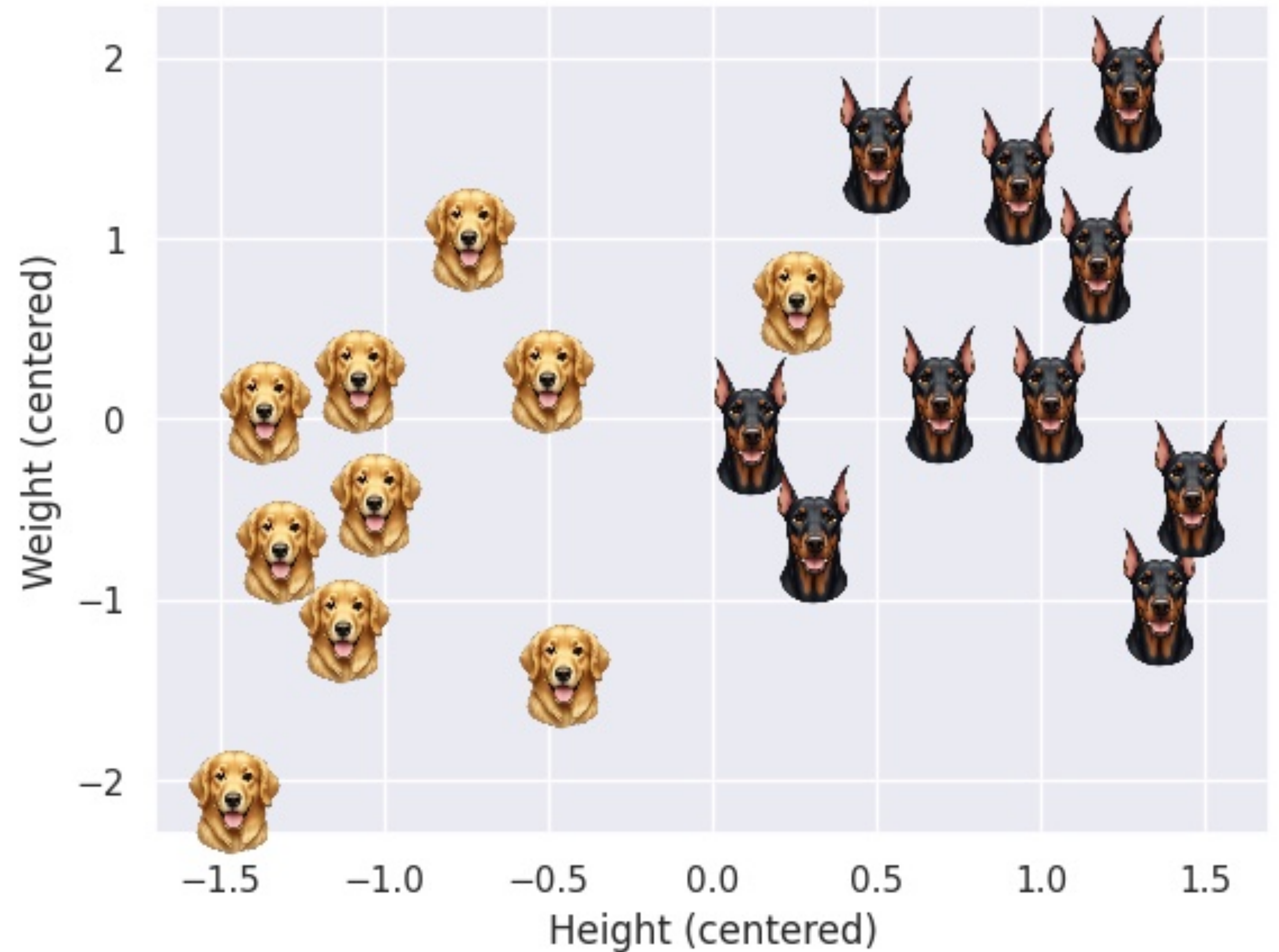


# Binary classification

## Predict dog species

- Input (centered\*)
    - Height
    - Weight
  - Output
    - Golden Retriever:  $+1$
    - Doberman Pinscher:  $-1$
- = "Golden Retriever detector"

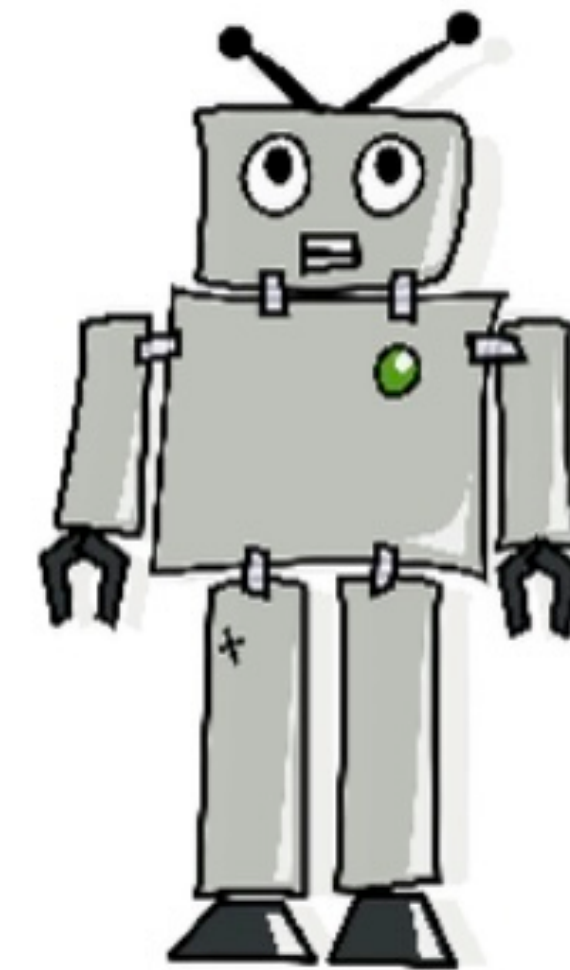
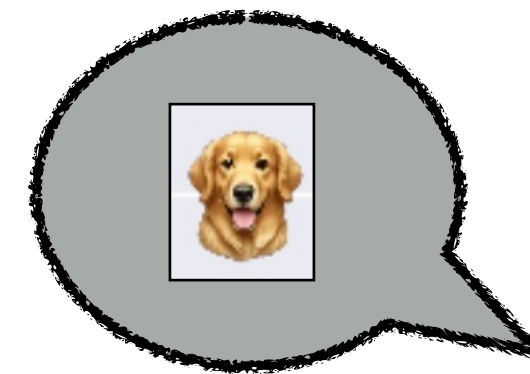
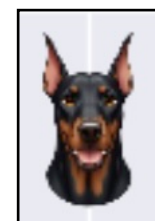
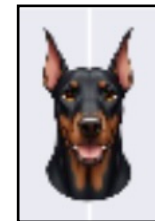
\* Input centered to have mean = 0 and standard deviation = 1.



# True/False Positive/Negative

Ways to get the answer right or wrong:

- True Positive
- False Positive
- True Negative
- False Negative

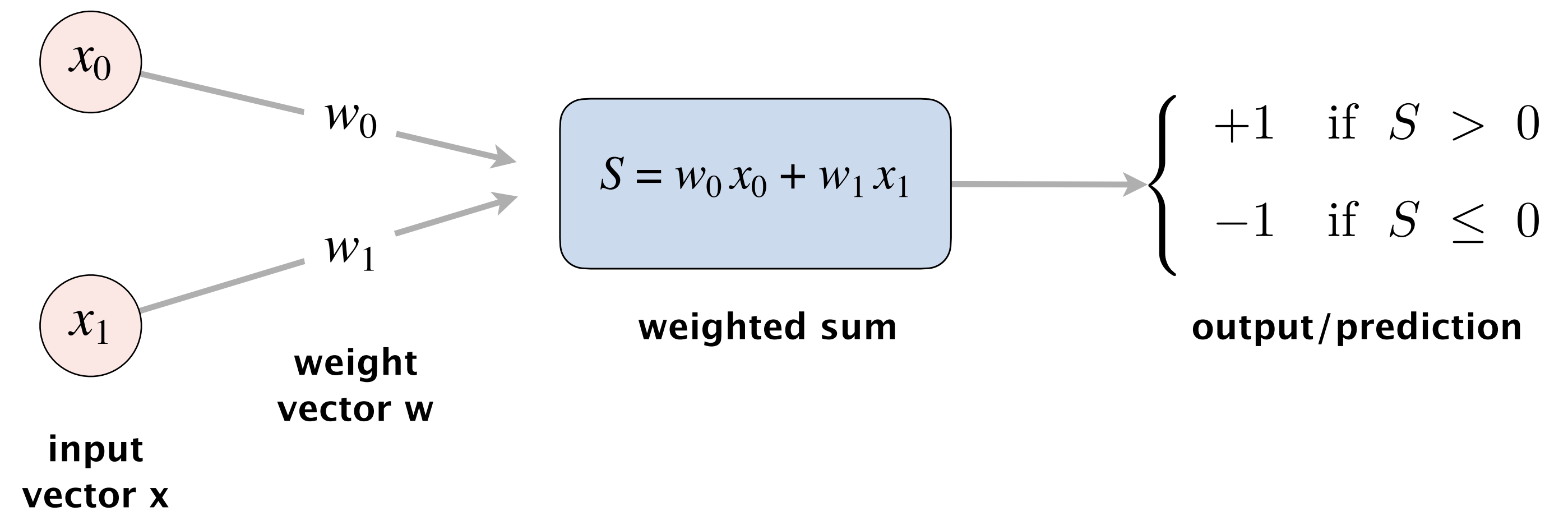


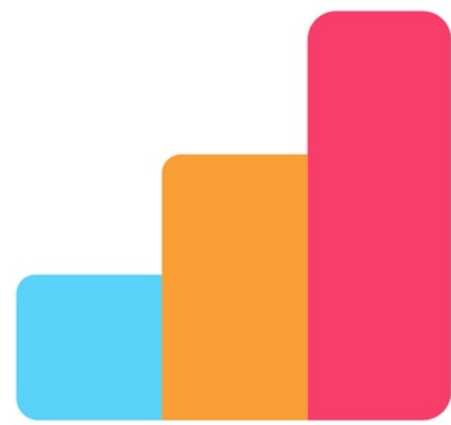


# Perceptron

Algorithm that takes several inputs and produces a single, binary output.

- Inputs:  $x_0, x_1$  (i.e. height, weight)
- Compute weighted sum  
$$S = w_0 x_0 + w_1 x_1$$
- Output  
$$\text{sign}(S) = +1 \text{ if } S > 0$$
$$= -1 \text{ if } S \leq 0$$

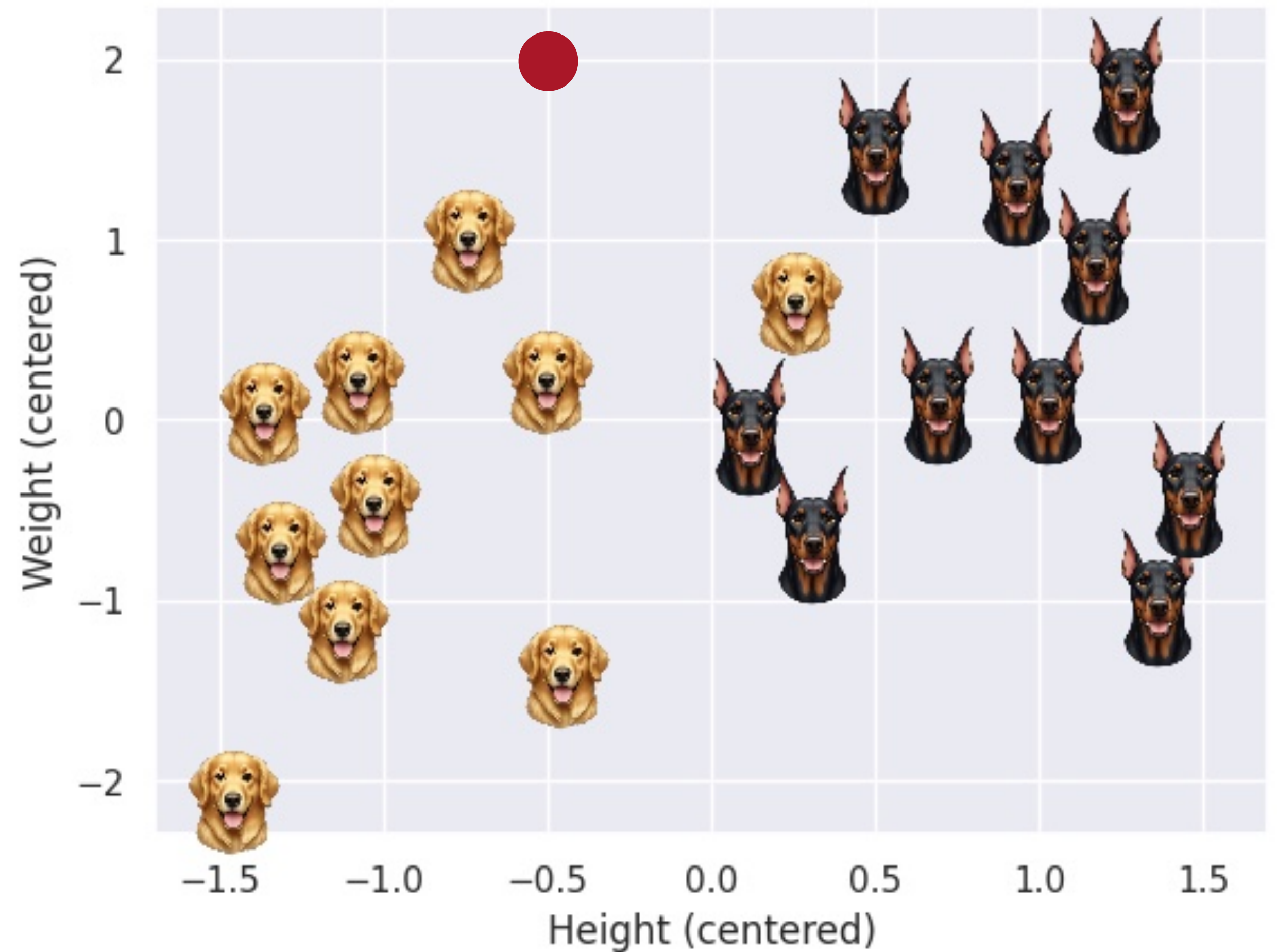




# Poll

- Inputs:  $x_0 = -0.5$ ,  $x_1 = 2$
- Weights:  $w_0 = -2$ ,  $w_1 = -0.1$
- Compute weighted sum  
 $S = w_0x_0 + w_1x_1$
- Output  
 $\text{sign}(S) = +1$  if  $S > 0$   
 $= -1$  if  $S \leq 0$

**What is the weighted sum  $S$ ?**



Golden retriever: +1, Doberman -1

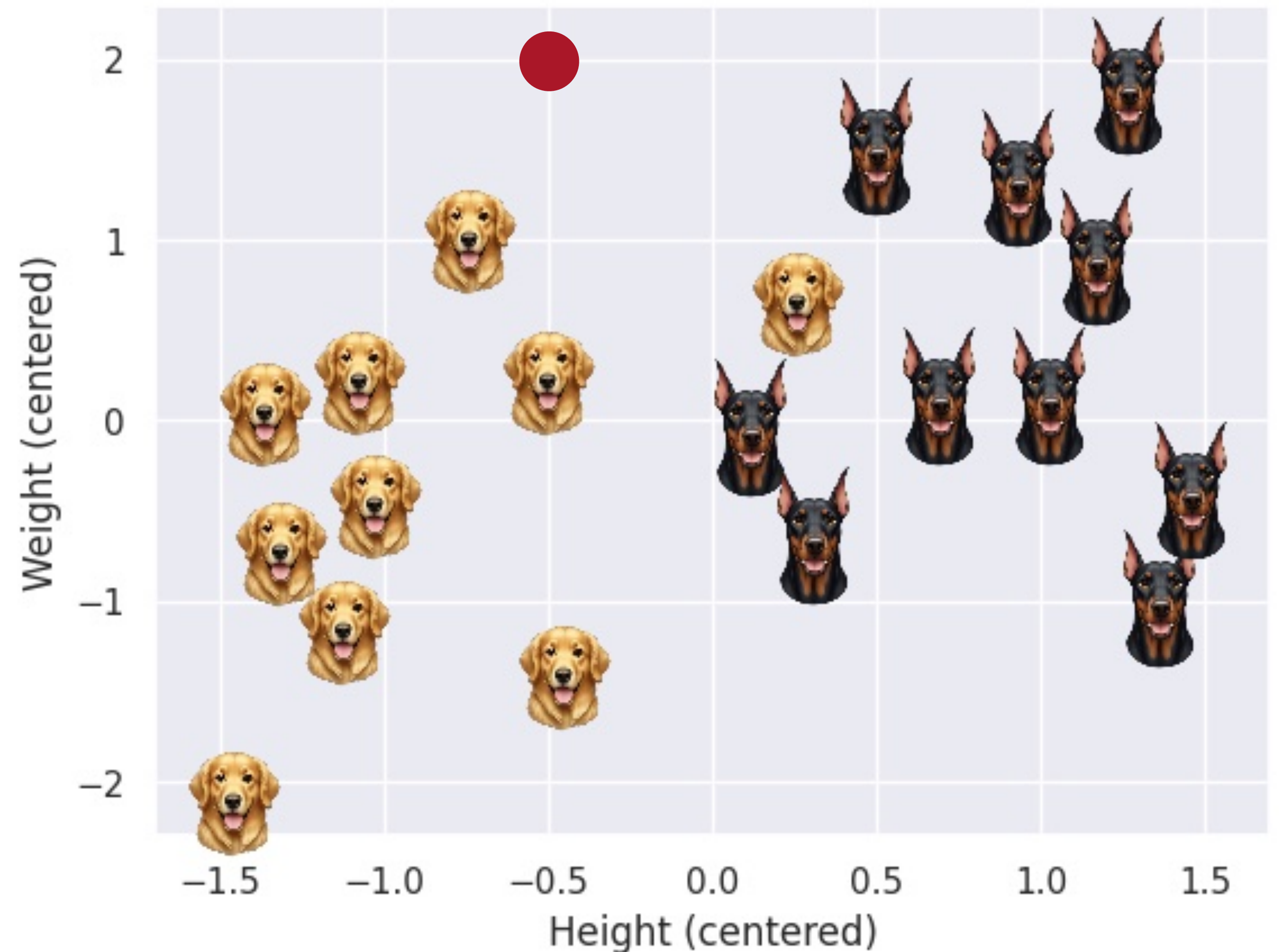


# Perceptron

- Inputs:  $x_0 = -0.5$ ,  $x_1 = 2$
- Weights:  $w_0 = -2$ ,  $w_1 = -0.1$
- Compute weighted sum  
 $S = w_0x_0 + w_1x_1$
- Output  
 $\text{sign}(S) = +1$  if  $S > 0$   
 $= -1$  if  $S \leq 0$

$$\begin{aligned} S &= w_0x_0 + w_1x_1 \\ &= (-2) \cdot (-0.5) + (-0.1) \cdot 2 \\ &= 1 + (-0.2) = 0.8 \end{aligned}$$

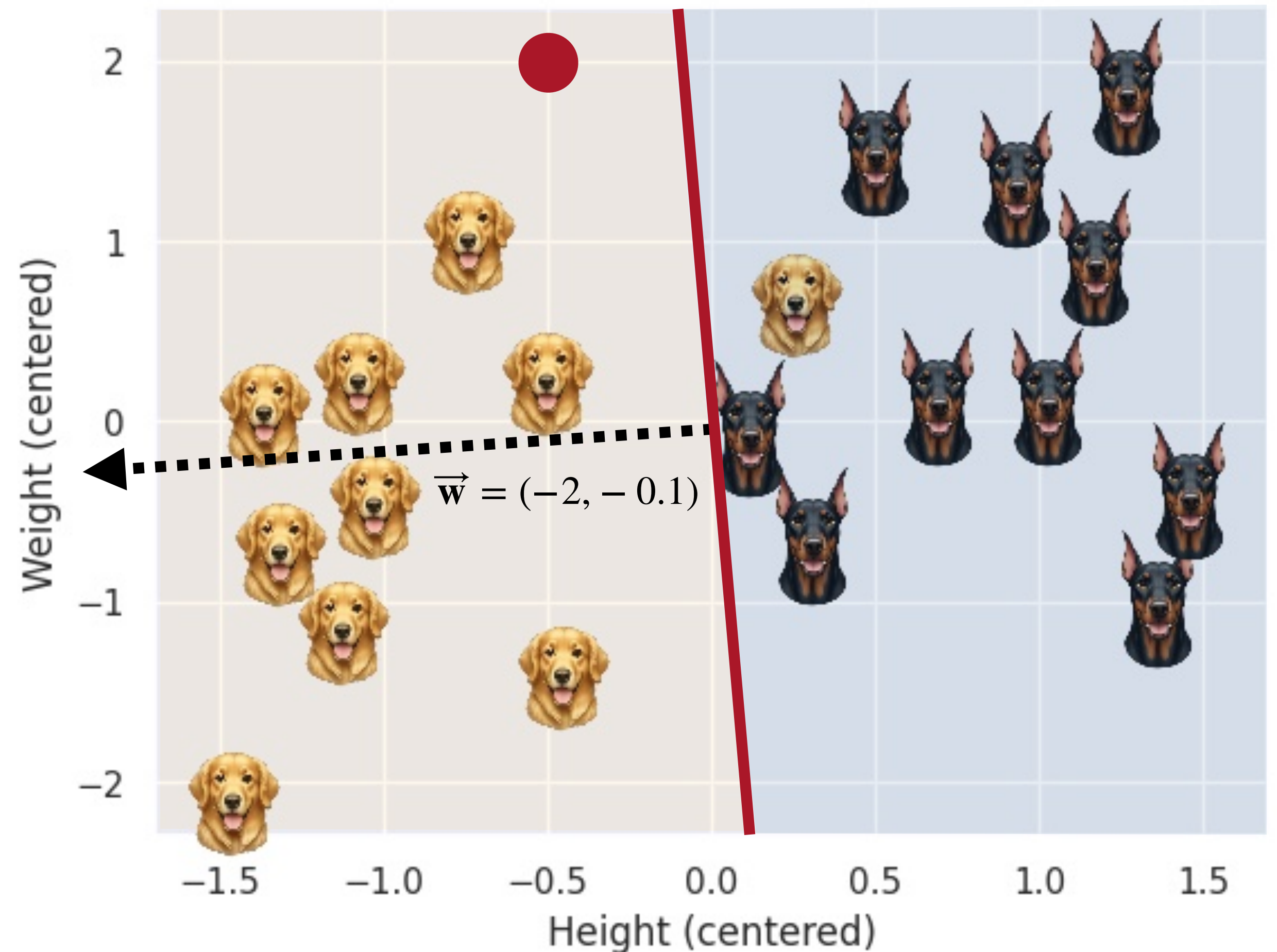
$$\text{sign}(S) = \text{sign}(0.8) = +1$$



Golden retriever: +1, Doberman -1

# Perceptron: Geometric Perspective

- Inputs:  $x_0 = -0.5$ ,  $x_1 = 2$
- Weights:  $w_0 = -2$ ,  $w_1 = -0.1$
- Compute weighted sum  
 $S = w_0x_0 + w_1x_1 = 0.8$
- Output  
 $\text{sign}(S) = +1$
- Weight vector  $\vec{w}$  is **perpendicular** to linear decision boundary.
- Datapoints on the “side” with weight vector  $\vec{w}$  are predicted as  $+1$ ; those on the “opposite side” as  $-1$ .



Golden retriever: +1, Doberman -1



# Ingredients of ML

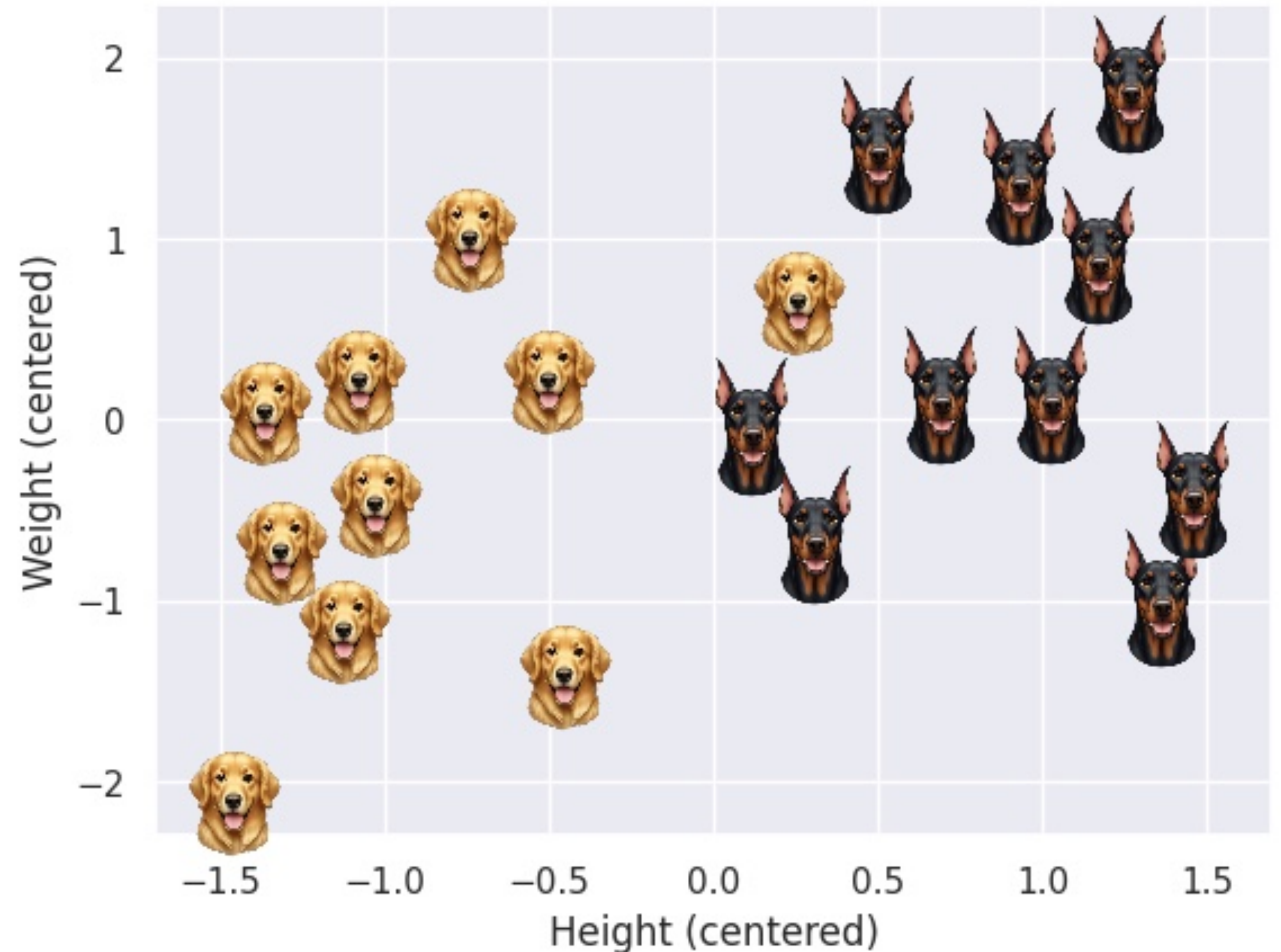
1. **Data.** Height and weight  $(x_0, x_1)$  and label  $y \in \{+1, -1\}$  for Golden Retriever or Doberman.
2. **Model.** Perceptron model.
3. **Model parameters.** Weight vector  $\vec{w} = (w_0, w_1)$ .
4. **Training.** Given datapoints, find good model parameters.
5. **Testing.** Evaluate the performance of learned model on new, previously unseen data (i.e. test set).

# Perceptron: Training

Initialize all weights to 0.

For each training example with label  $y \in \{+1, -1\}$ :

- Compute predicted output  $\hat{y} = \text{sign}(S)$
- Update weights if incorrect (i.e.  $\hat{y} \neq y$ )



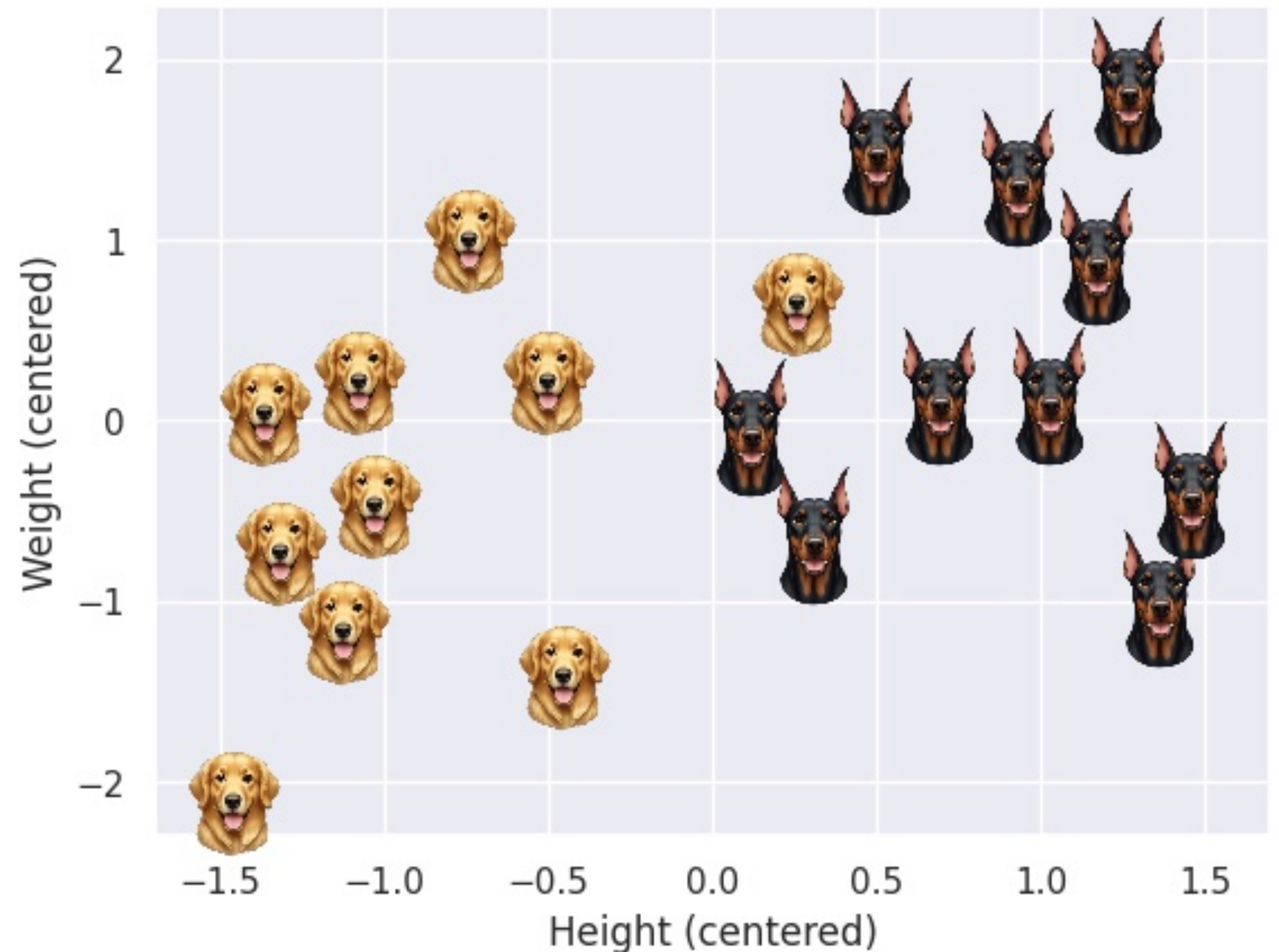
Golden retriever: +1, Doberman -1



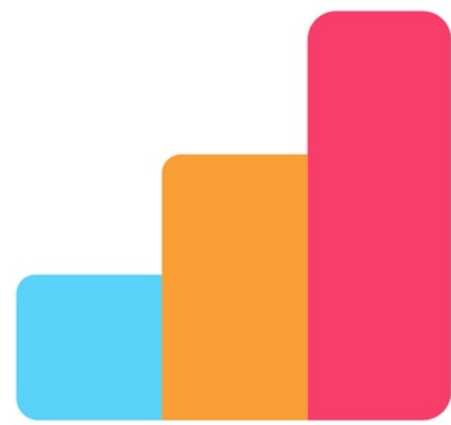
# Perceptron: Update Rule

For each training example with label  $y \in \{+1, -1\}$ :

- Compute predicted output  
 $S = w_0x_0 + w_1x_1$   
 $\hat{y} = \text{sign}(S) = +1$  if  $S > 0$   
 $\quad = -1$  if  $S \leq 0$
- If correct ( $y = \hat{y}$ ), do nothing.
- If false positive ( $y = -1, \hat{y} = +1$ ):  
 $w'_j = w_j - x_j$
- If false negative ( $y = +1, \hat{y} = -1$ ):  
 $w'_j = w_j + x_j$



Golden retriever: +1, Doberman -1

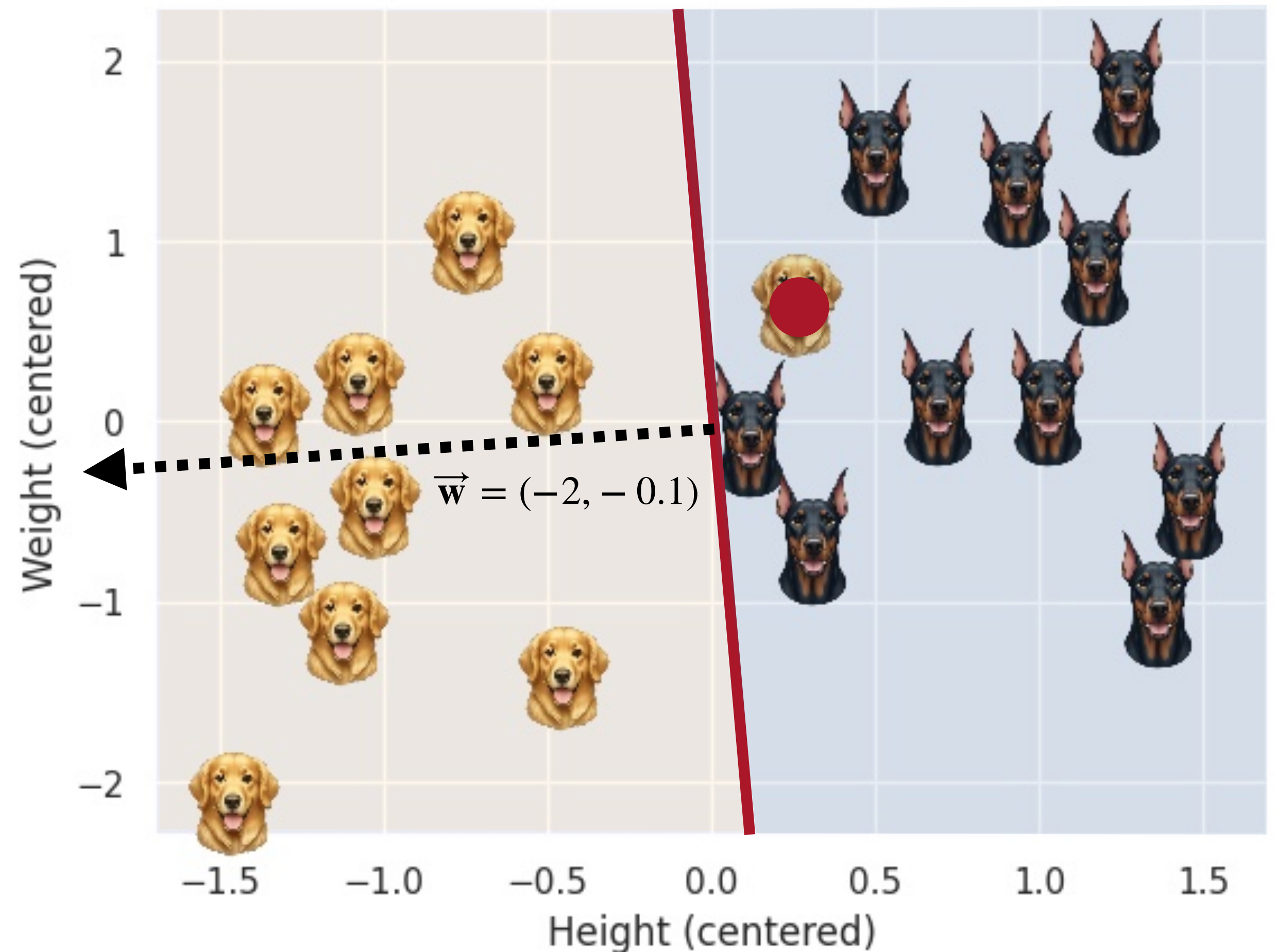


# Poll

- Inputs:  $x_0 = 0.3$ ,  $x_1 = 0.6$ ,  $y = +1$
- Weights:  $w_0 = -2$ ,  $w_1 = -0.1$
- Compute predicted output  
 $S = w_0x_0 + w_1x_1$   
 $\hat{y} = \text{sign}(S) = +1$  if  $S > 0$   
 $= -1$  if  $S \leq 0$

**Which scenario is true?**

- A. Correct prediction ( $y = \hat{y}$ )
- B. False positive ( $y = -1$ ,  $\hat{y} = +1$ )
- C. False negative ( $y = +1$ ,  $\hat{y} = -1$ )



Golden retriever: +1, Doberman -1



# Perceptron: Update Rule

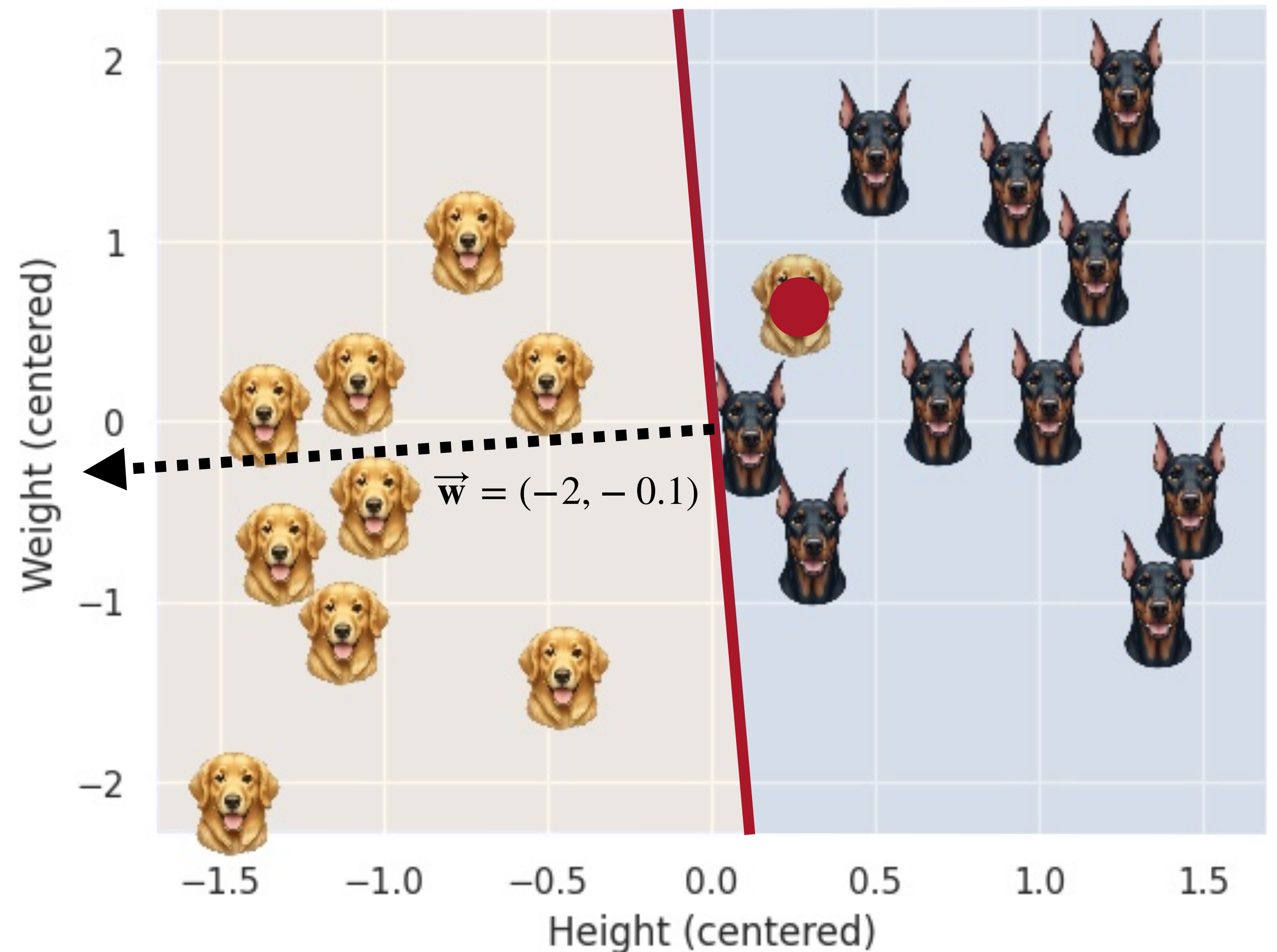
- Inputs:  $x_0 = 0.3$ ,  $x_1 = 0.6$ ,  $y = +1$
- Weights:  $w_0 = -2$ ,  $w_1 = -0.1$

$$S = w_0x_0 + w_1x_1 = -0.66$$

$$\hat{y} = \text{sign}(S) = -1$$

## Update rules:

- If correct ( $y = \hat{y}$ ), do nothing.
- If false positive ( $y = -1$ ,  $\hat{y} = +1$ ):  
 $w'_j = w_j - x_j$
- If false negative ( $y = +1$ ,  $\hat{y} = -1$ ):  
 $w'_j = w_j + x_j$



# Perceptron: Update Rule

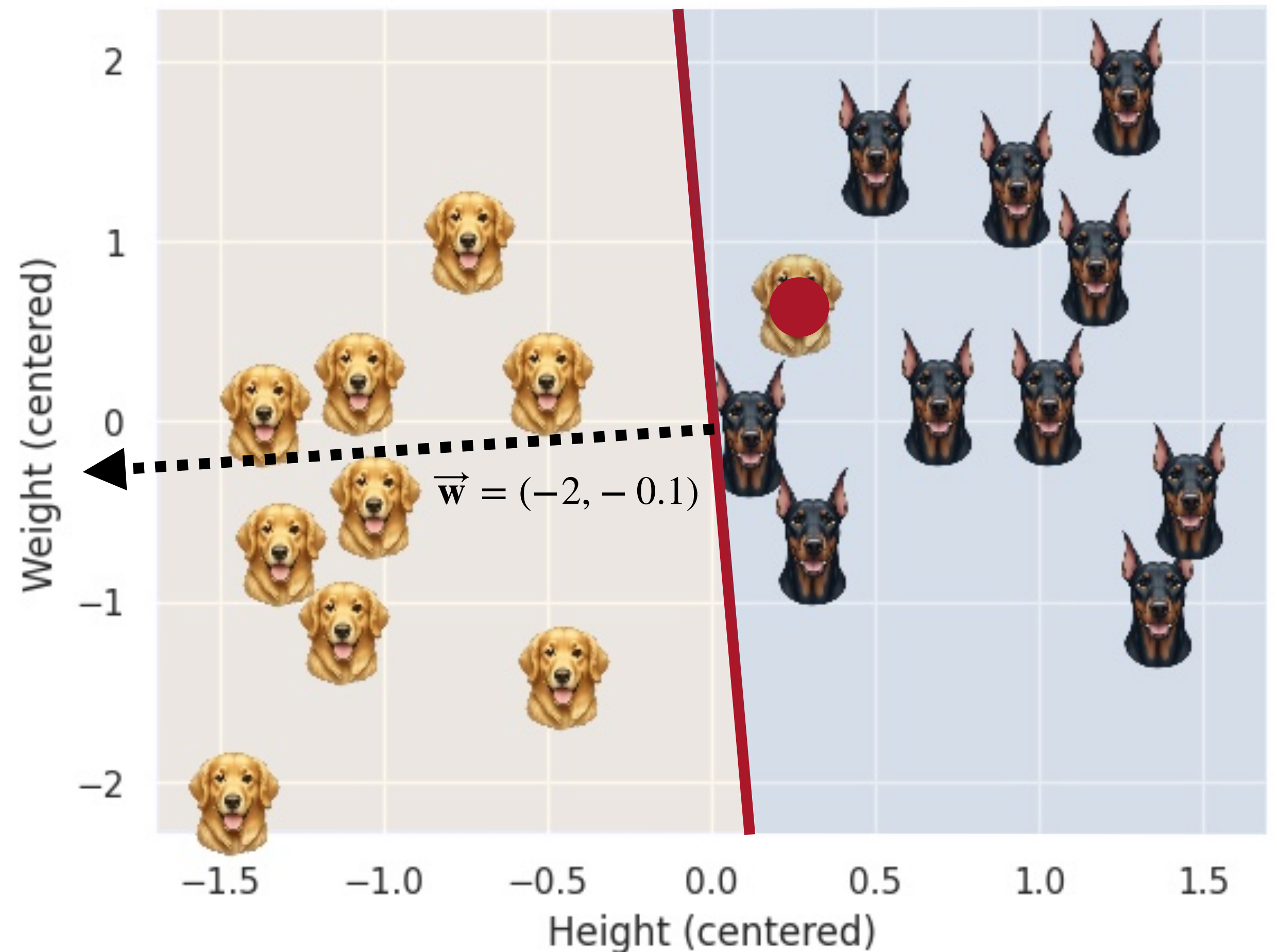
- Inputs:  $x_0 = 0.3$ ,  $x_1 = 0.6$ ,  $y = +1$
- Weights:  $w_0 = -2$ ,  $w_1 = -0.1$

$$S = w_0x_0 + w_1x_1 = -0.66$$

$$\hat{y} = \text{sign}(S) = -1$$

## Update rules:

- If correct ( $y = \hat{y}$ ), do nothing.
- If false positive ( $y = -1$ ,  $\hat{y} = +1$ ):  
 $w'_j = w_j - x_j$
- **If false negative ( $y = +1$ ,  $\hat{y} = -1$ ):**  
 $w'_j = w_j + x_j$





# Perceptron: Update Rule

- Inputs:  $x_0 = 0.3$ ,  $x_1 = 0.6$ ,  $y = +1$
- Weights:  $w_0 = -2$ ,  $w_1 = -0.1$

$$S = w_0 x_0 + w_1 x_1 = -0.66$$

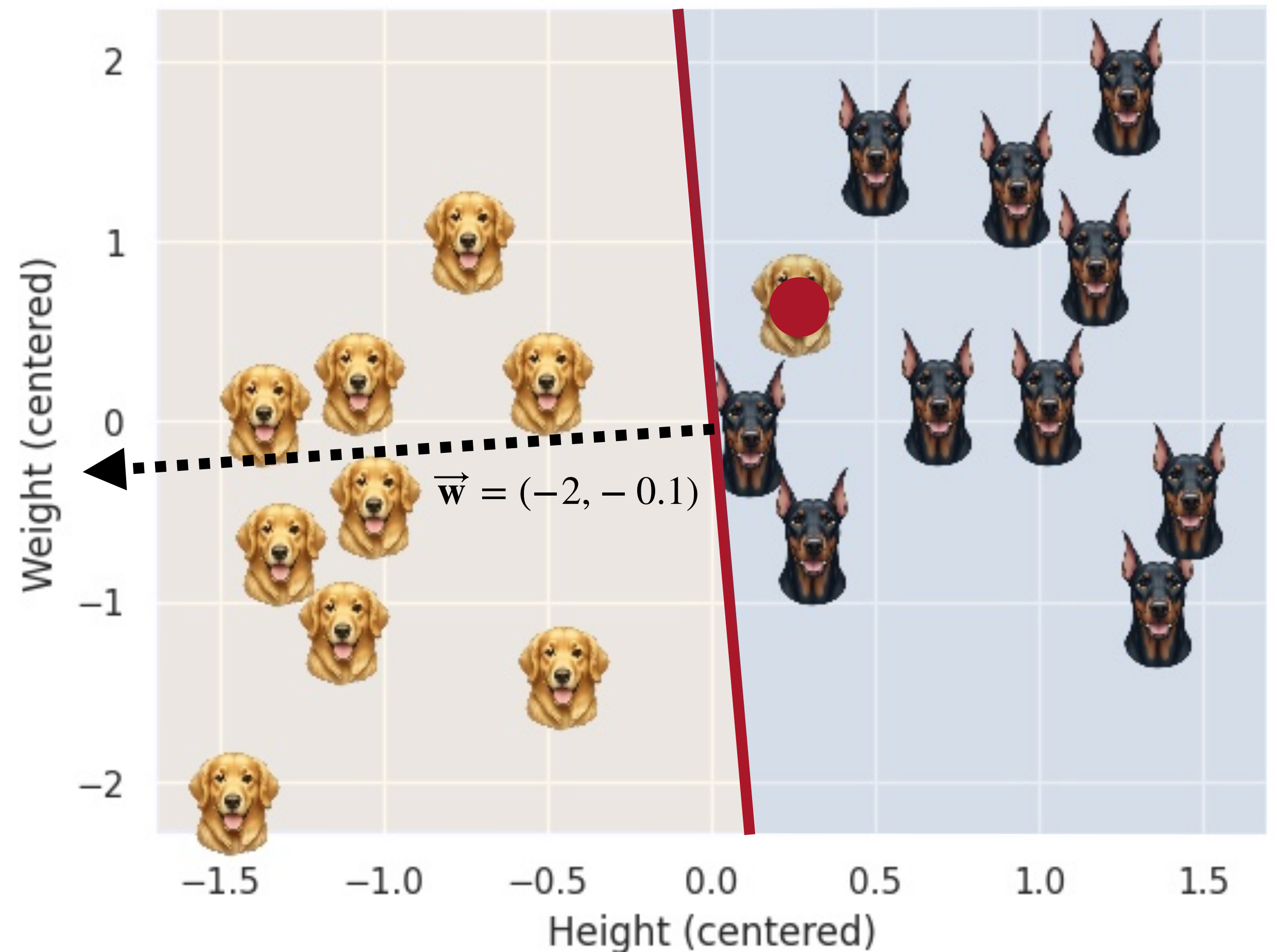
$$\hat{y} = \text{sign}(S) = -1$$

**If false negative ( $y = +1$ ,  $\hat{y} = -1$ ):**

$$w'_j = w_j + x_j$$

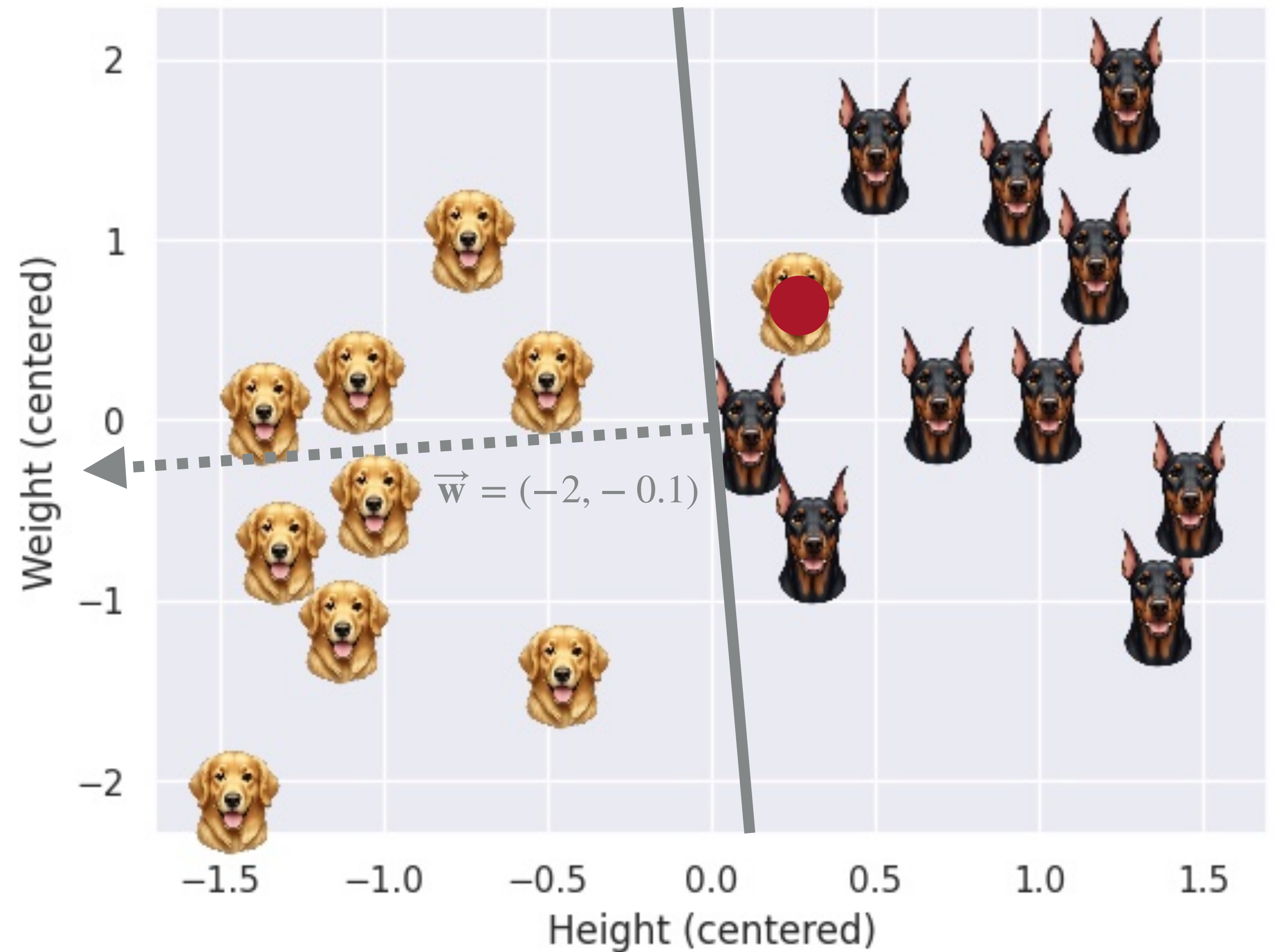
$$w'_0 = w_0 + x_0 = -2 + 0.3 = -1.7$$

$$w'_1 = w_1 + x_1 = -0.1 + 0.6 = 0.5$$



# Perceptron: Update Rule

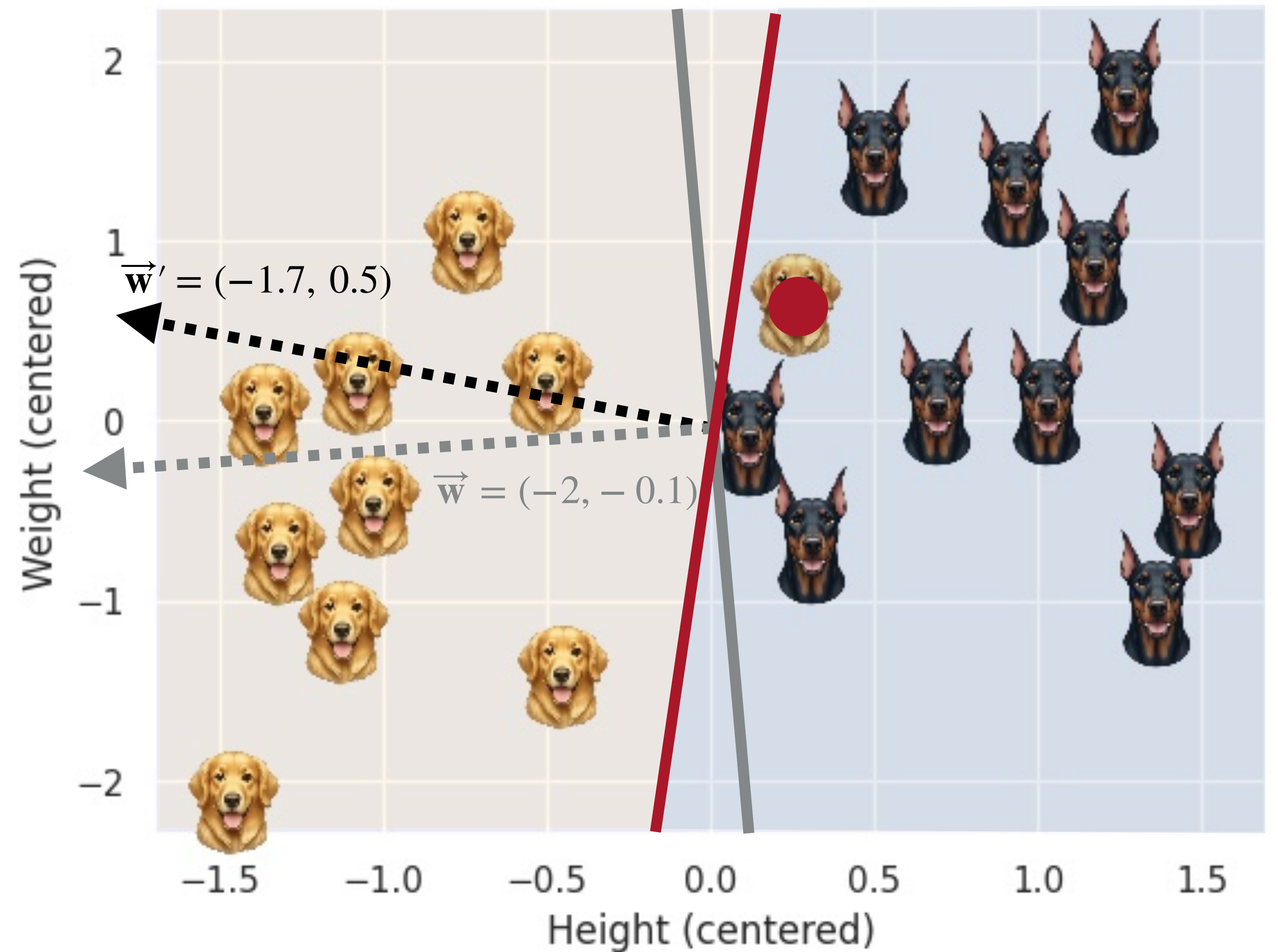
- Inputs:  $x_0 = 0.3$ ,  $x_1 = 0.6$ ,  $y = +1$
- Old weights:  $w_0 = -2$ ,  $w_1 = -0.1$





# Perceptron: Update Rule

- Inputs:  $x_0 = 0.3$ ,  $x_1 = 0.6$ ,  $y = +1$
- Old weights:  $w_0 = -2$ ,  $w_1 = -0.1$
- New weights:  $w'_0 = -1.7$ ,  $w'_1 = 0.5$



# Perceptron: Update Rule

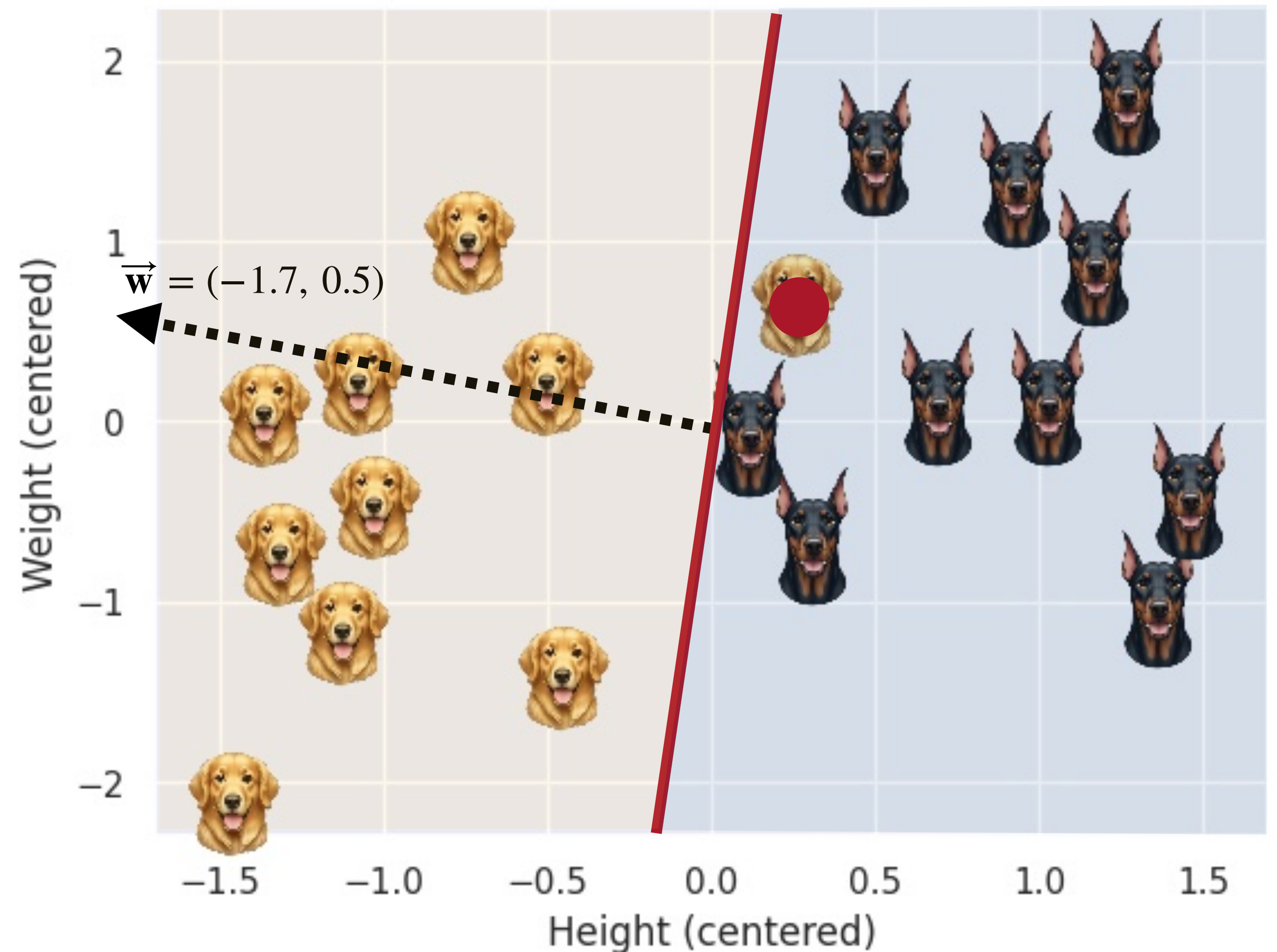
- Inputs:  $x_0 = 0.3$ ,  $x_1 = 0.6$ ,  $y = +1$
- New weights:  $w_0 = -1.7$ ,  $w_1 = 0.5$

$$S = w_0x_0 + w_1x_1 = -0.21$$

$$\hat{y} = \text{sign}(S) = -1$$

Still false negative, but closer to boundary.

**Intuition:** Update rule moves decision boundary to be better for the current training example.

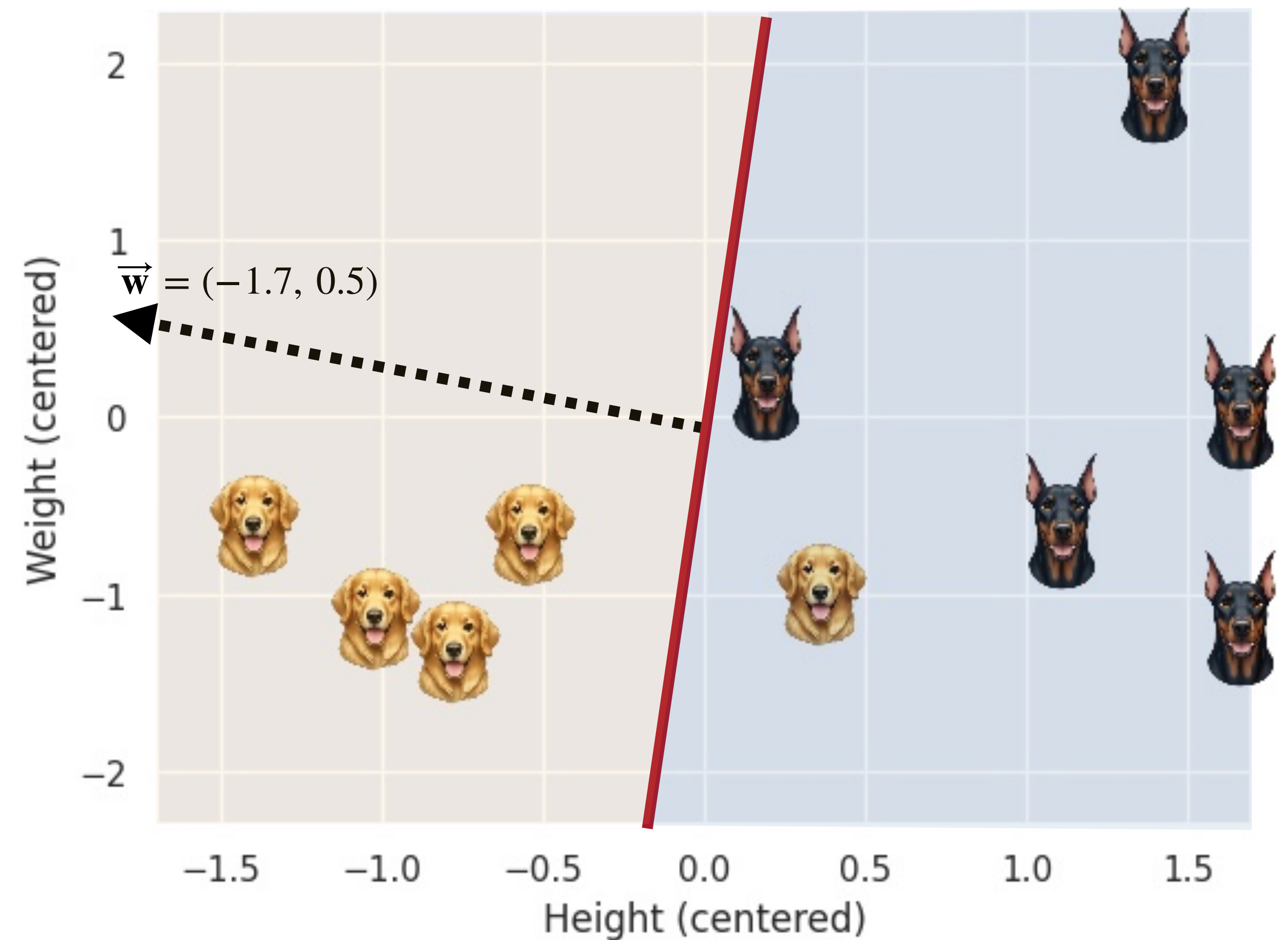


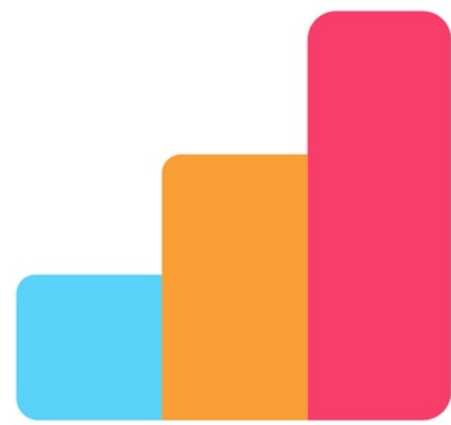


# Perceptron: Testing

Evaluate performance using an evaluation metric on an unseen set of examples (e.g. test examples).

$$\text{Error rate} = \frac{\# \text{ incorrect}}{\# \text{ total}}$$





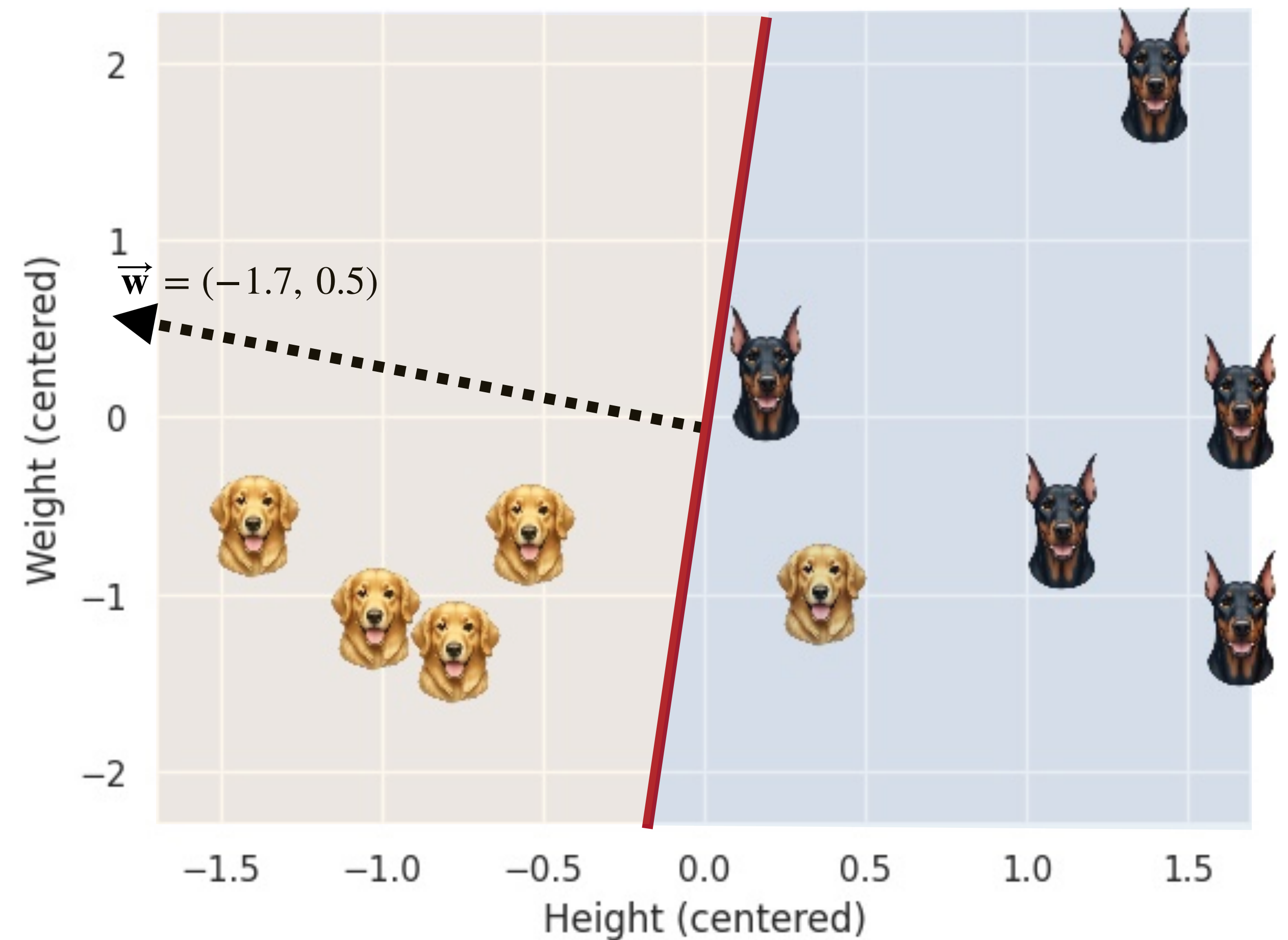
# Poll

Evaluate performance using an evaluation metric on an unseen set of examples (e.g. test examples).

$$\text{Error rate} = \frac{\# \text{ incorrect}}{\# \text{ total}}$$

Consider the following unseen examples (5 Golden Retrievers, 5 Dobermans) and the given Perceptron model.

**What is the error rate on test set?**





# Ingredients of ML

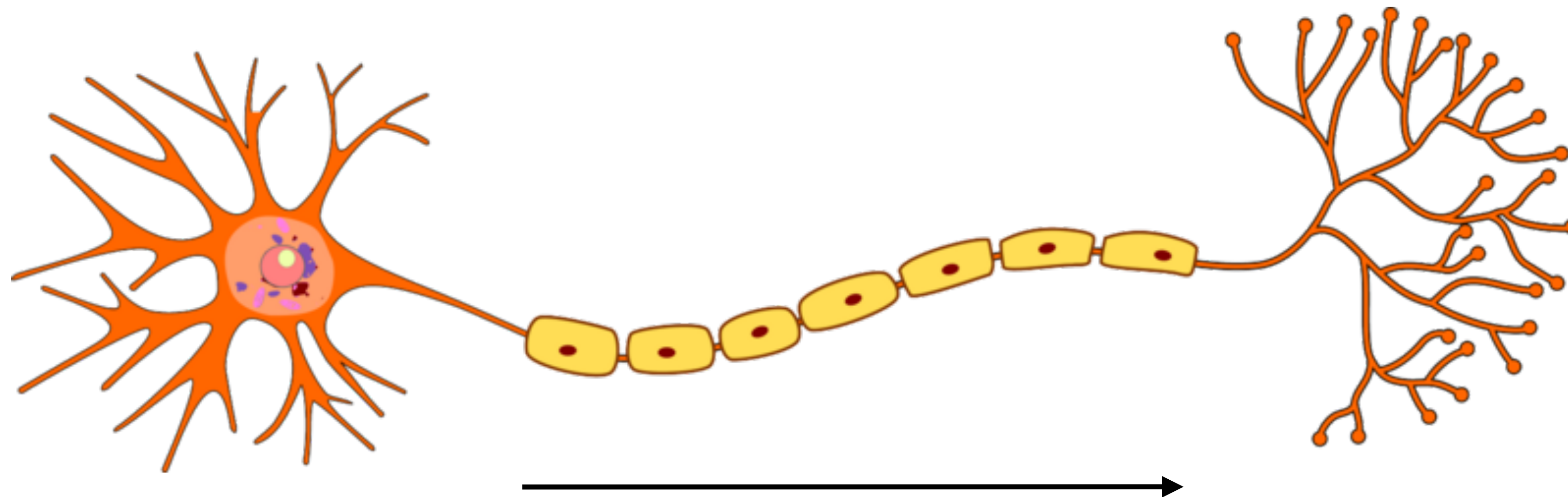
1. **Data.** Height and weight  $(x_0, x_1)$  and label  $y \in \{+1, -1\}$  for Golden Retriever or Doberman.
2. **Model.** Perceptron model.
3. **Model parameters.** Weight vector  $\vec{w} = (w_0, w_1)$ .
4. **Training.** Use the Perceptron update rule to train on training examples.
5. **Testing.** Compute error rate on new, previously unseen data (i.e. test set).

---

# Multi-layer Perceptron



# Biological Neuron



## **Dendrites**

Receive information  
from other neurons

## **Action potential**

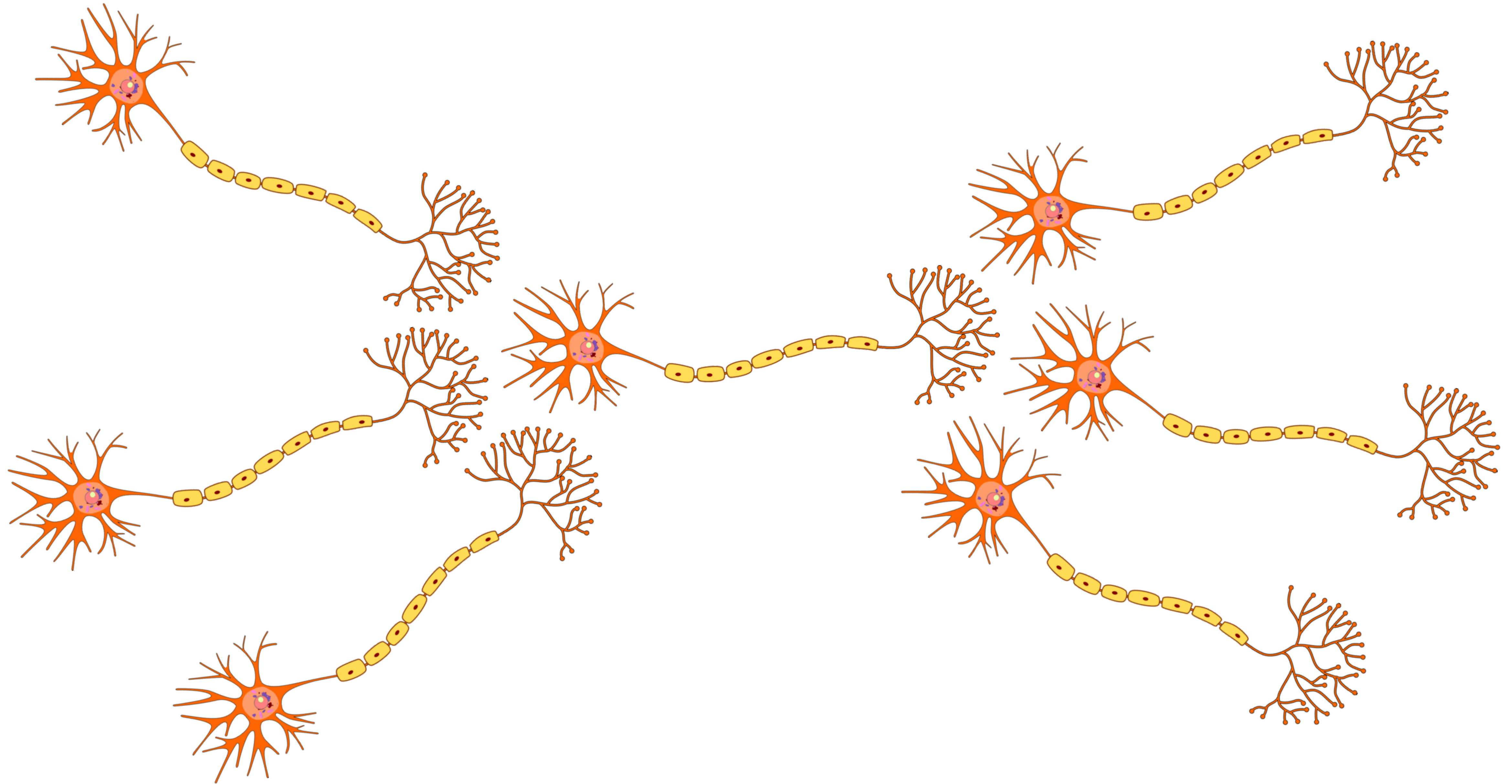
Neuron “fires” based  
on received information

## **Synaptic terminals**

Relay “firing” to  
other neurons

**Neuron firing (or not) is a binary output.**

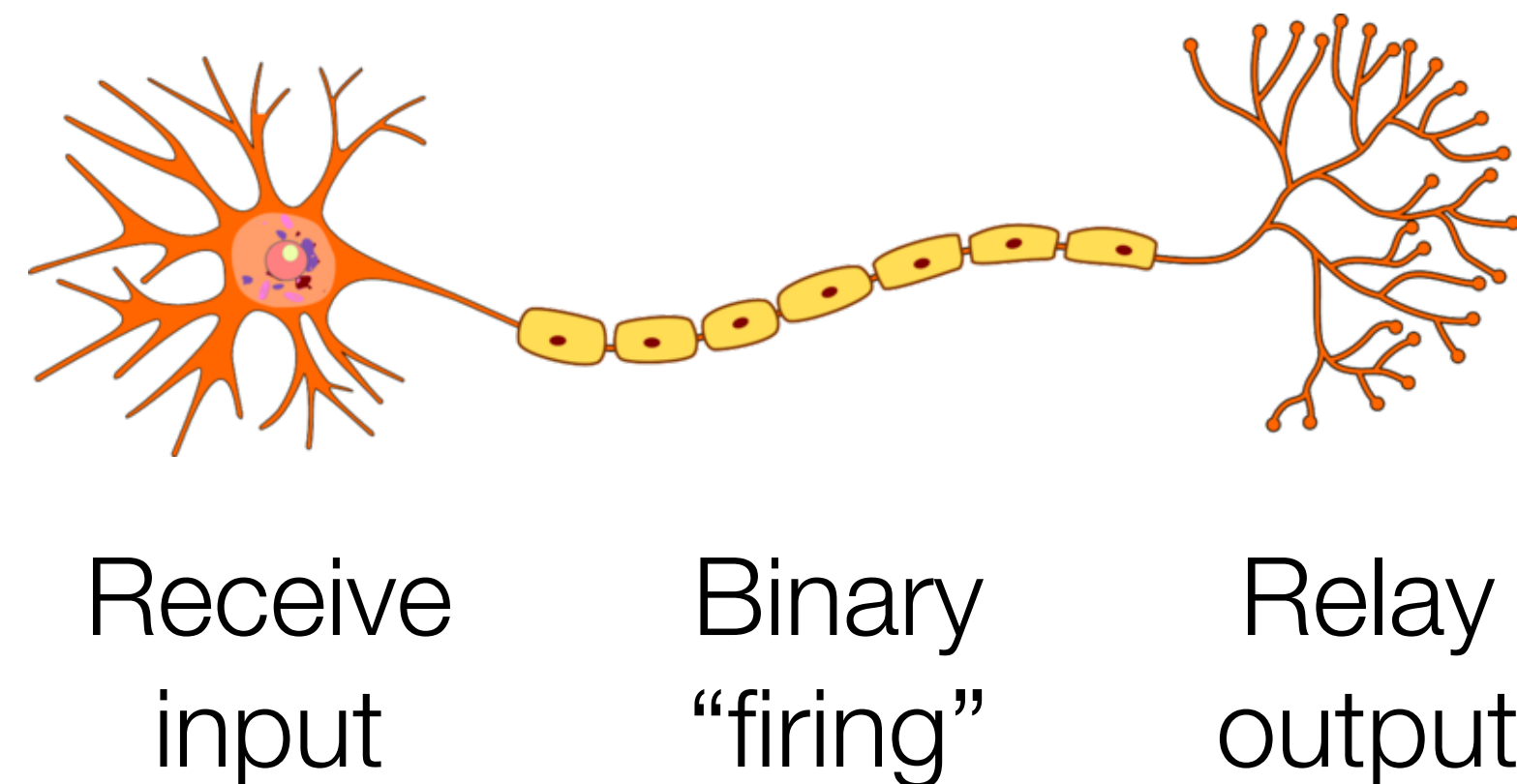
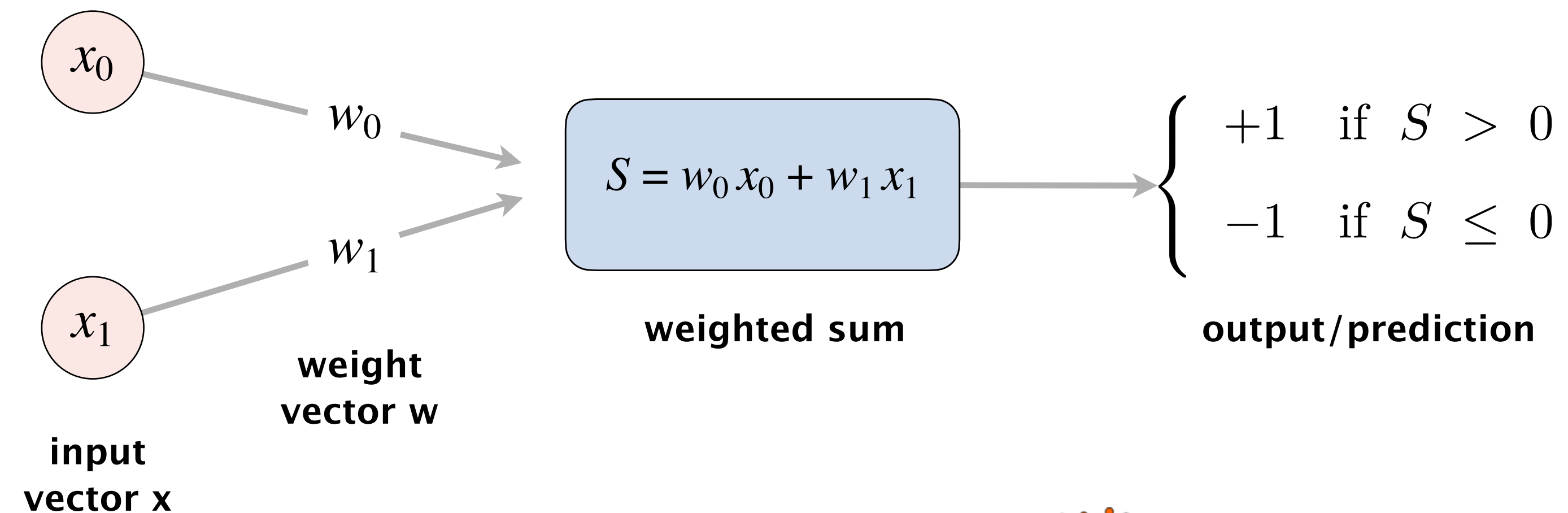
# Biological Neuron





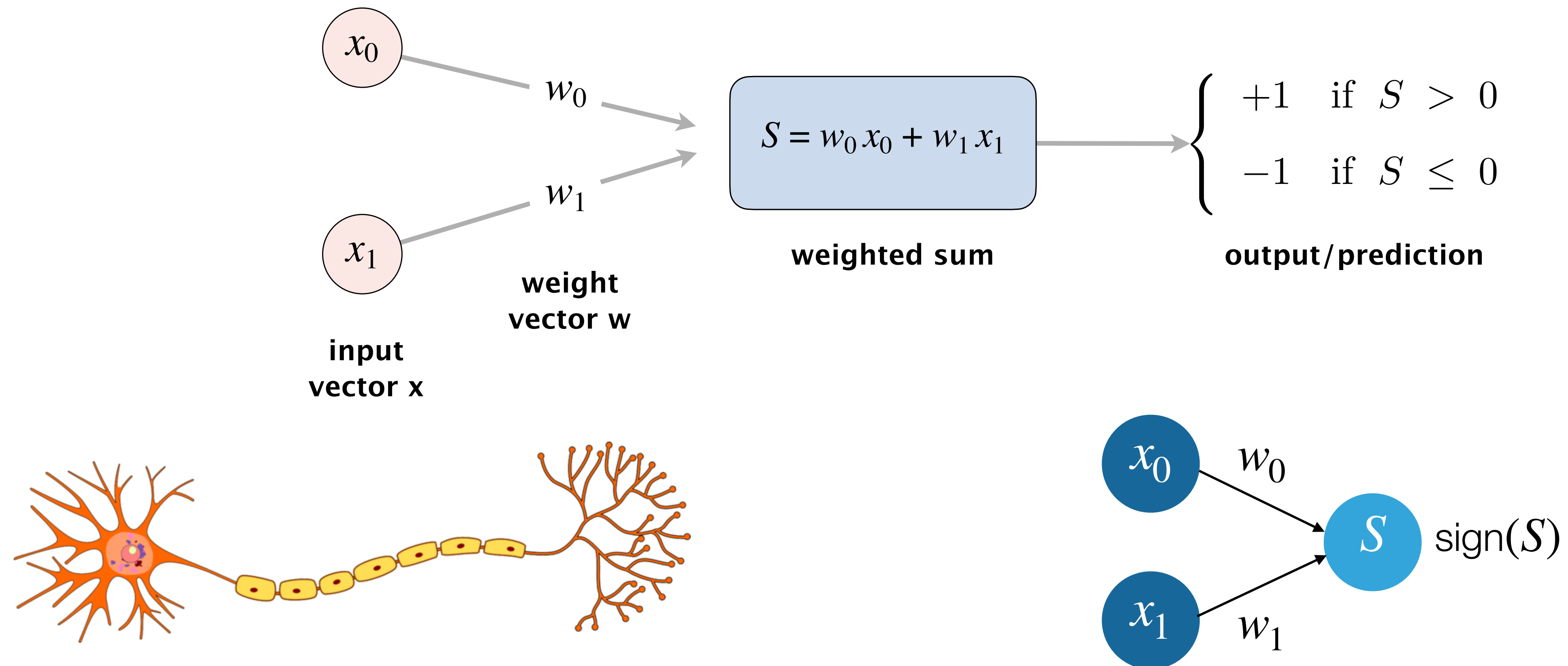
# Perceptron

Takes several inputs and produces a single binary output.



# Perceptron

Takes several inputs and produces a single binary output.

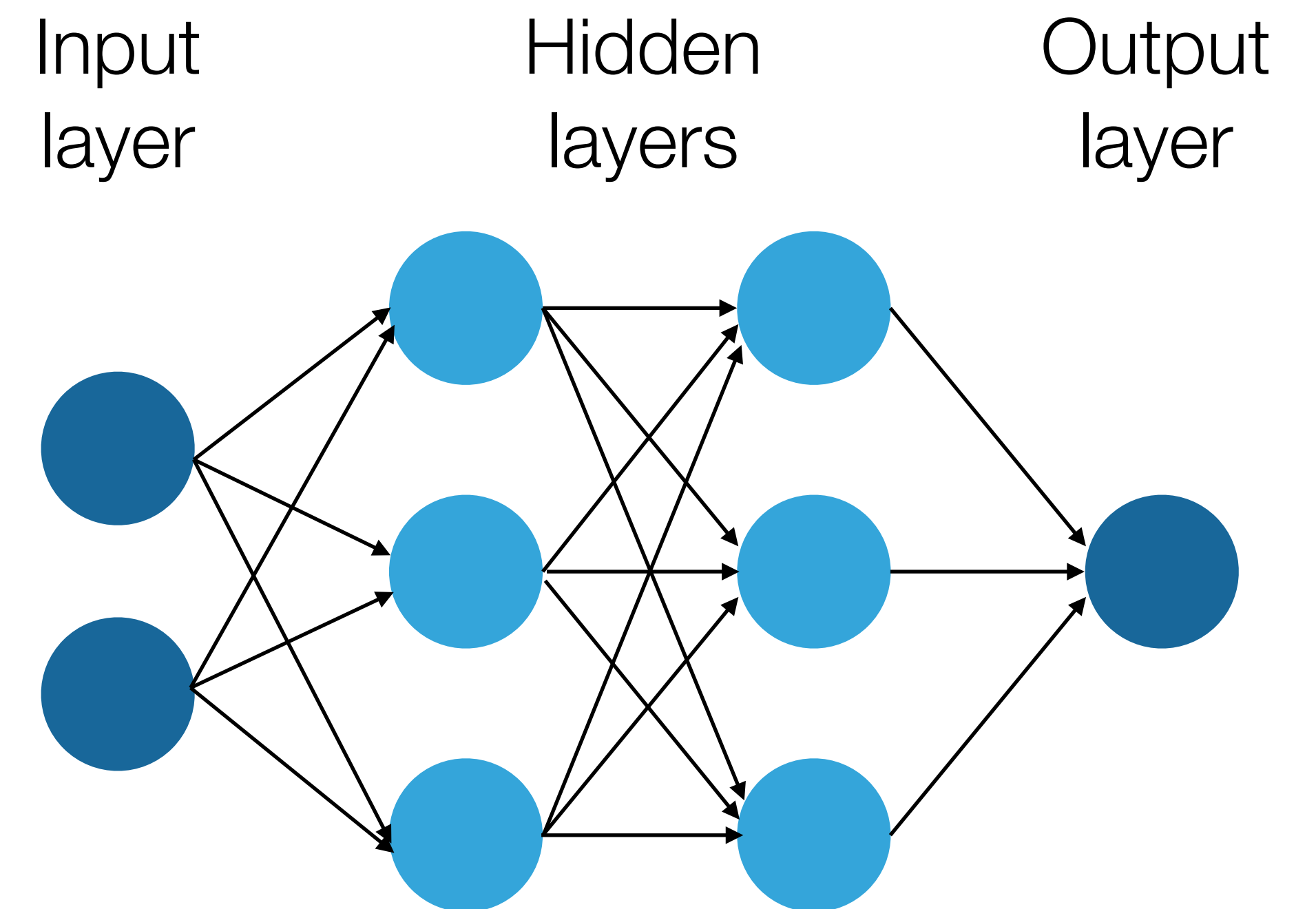
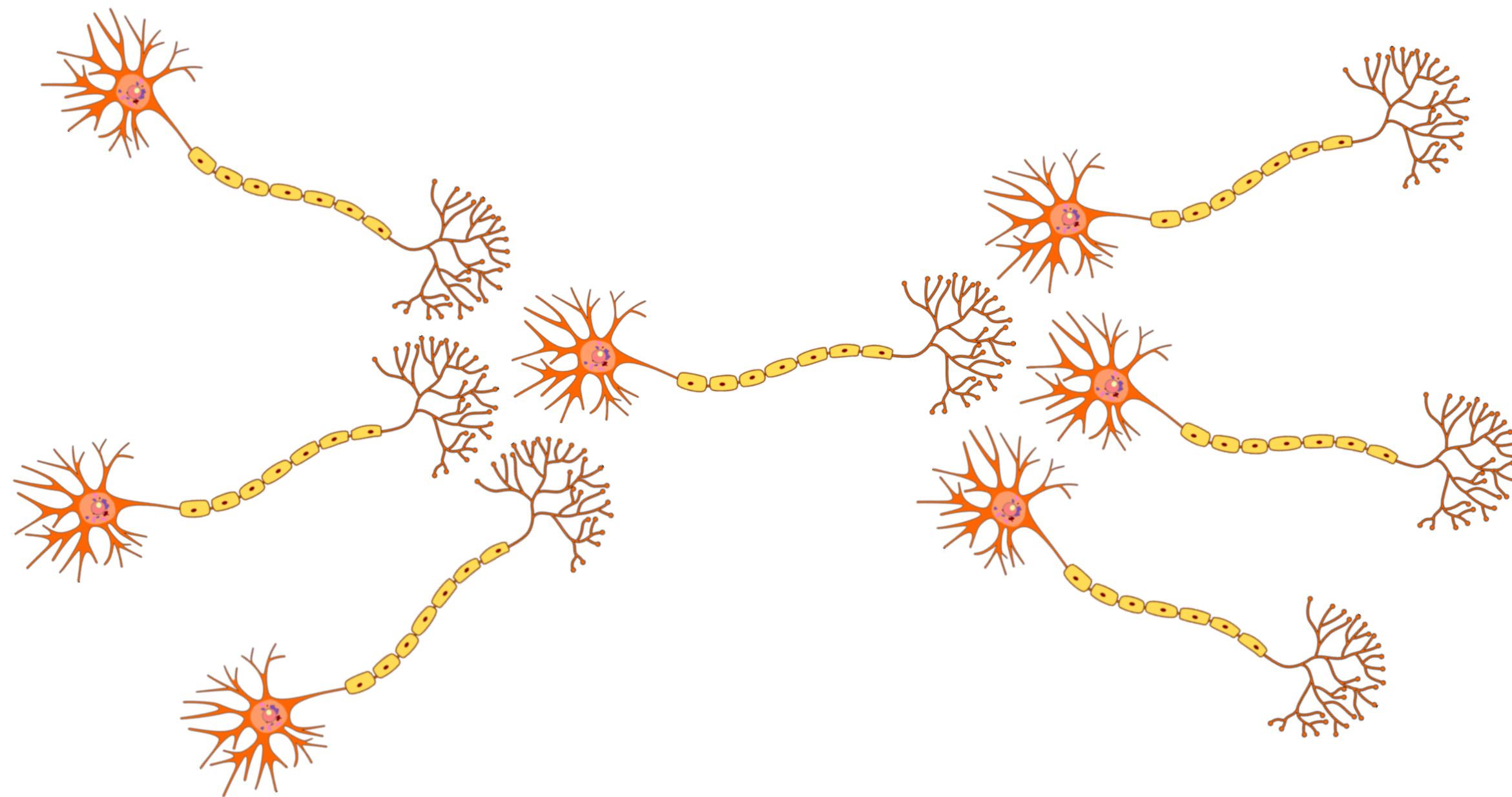




# Multi-Layer Perceptron (MLP)

Composed of multiple layers of perceptron neurons, where each neuron is connected to all neurons in the next layer (i.e. fully-connected).

MLPs are an example of a deep neural network.

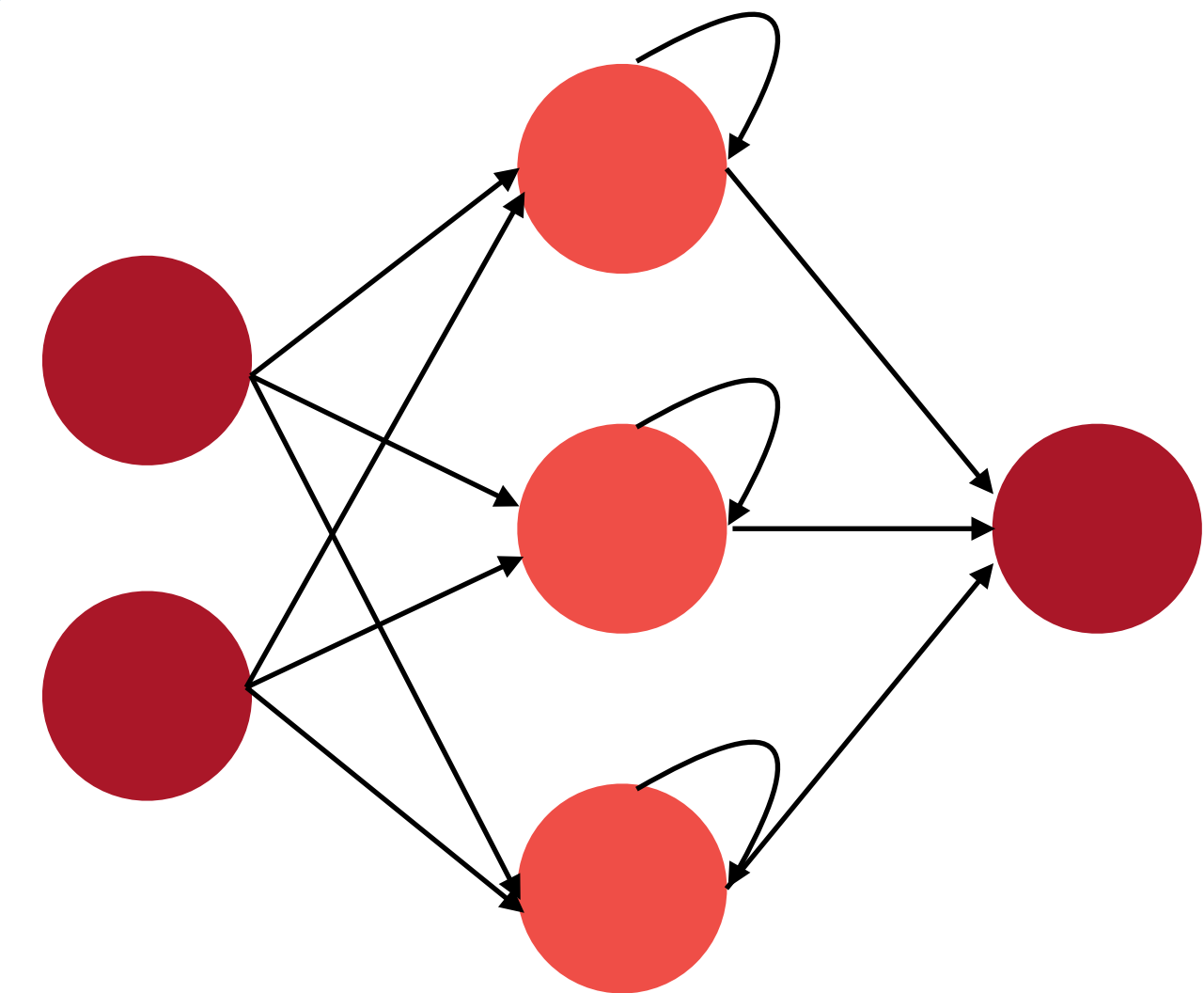


# Deep Learning

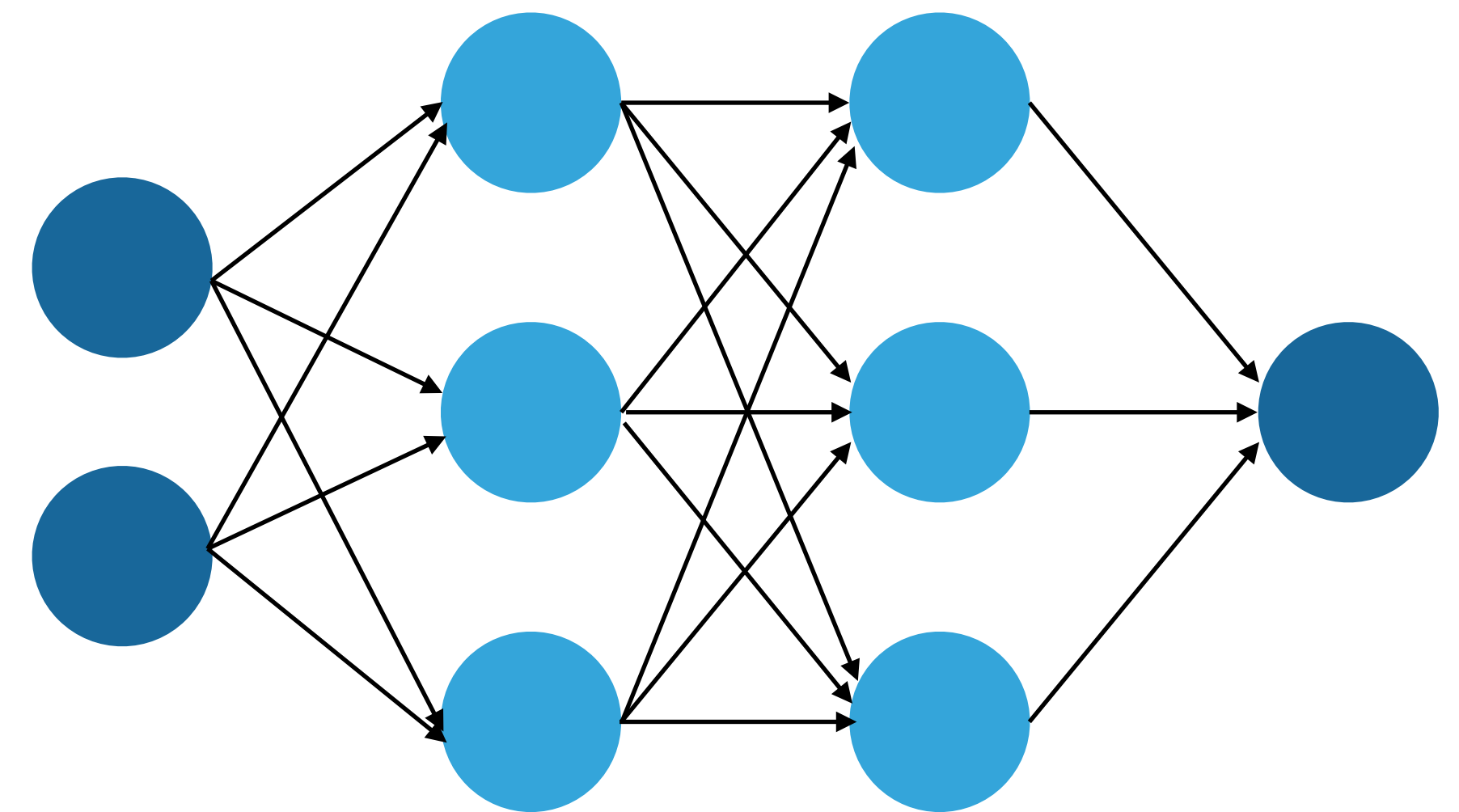
Refers to ML models that use deep neural networks.

Popular neural network architectures:

- Recurrent neural networks
- Convolutional neural networks
- Transformers



Recurrent NN



Fully-connected NN



---

# Multi-class Perceptron

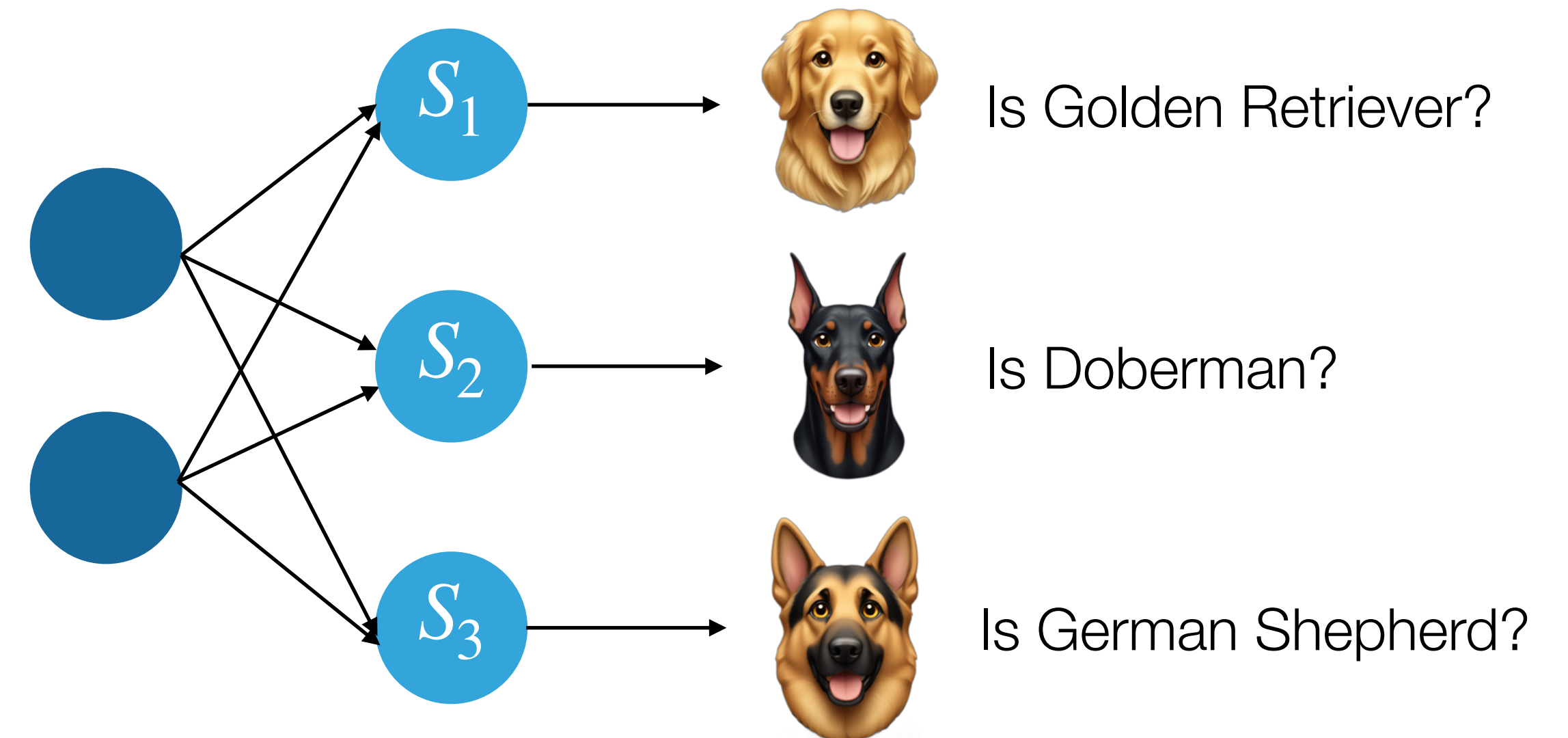
# Multi-class Classification

Predict one of **k** classes ( $k > 2$ ).

Suppose we have **k = 3** dog species we are trying to classify.

Train **k = 3** binary Perceptron models and compute **3** weighted sums:  $S_1, S_2, S_3$ .

Final prediction is the class with the largest weighted sum.





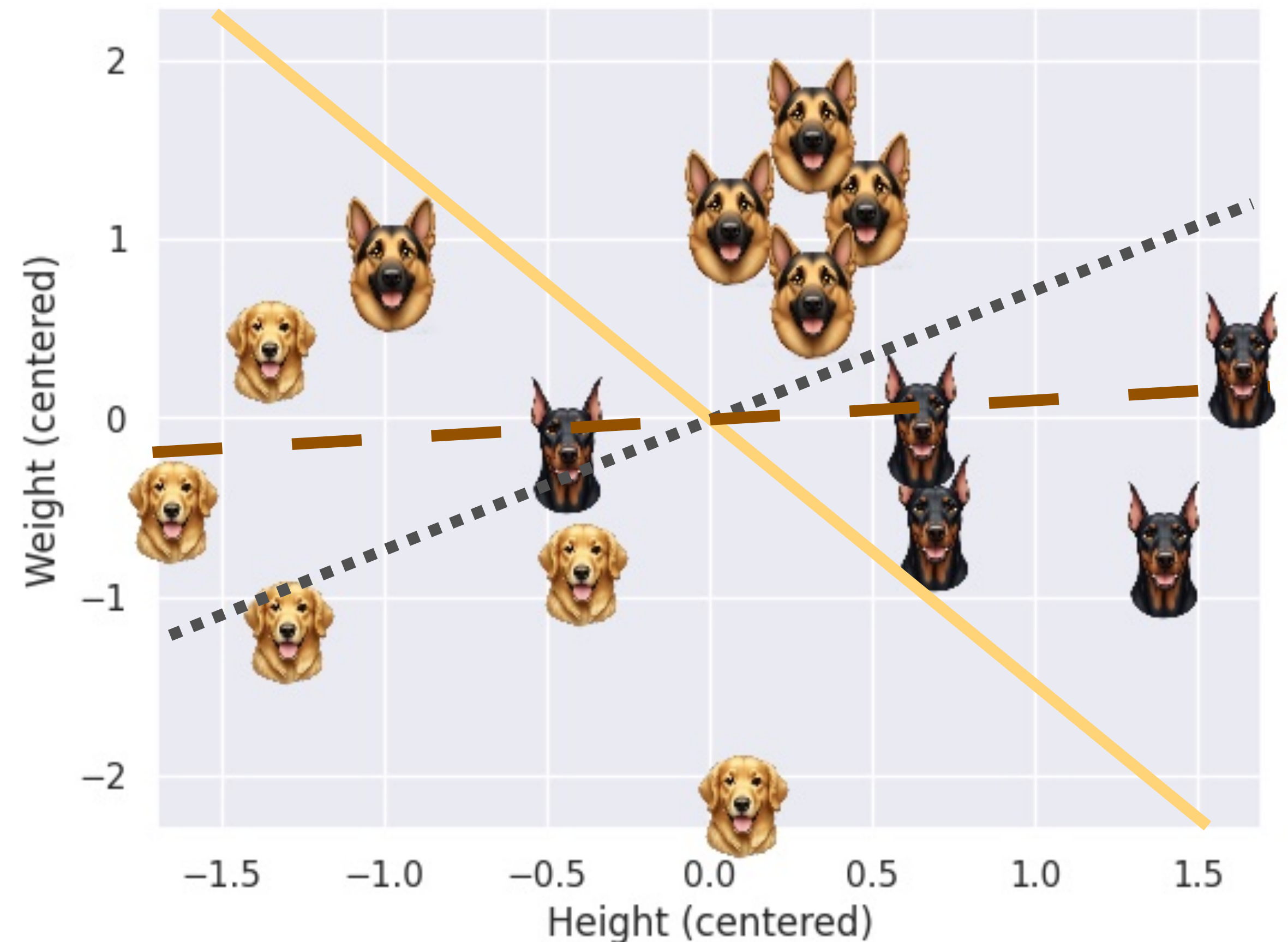
# Multi-class Classification: Geometric Perspective

Predict one of **k** classes ( $k > 2$ ).

Suppose we have **k = 3** dog species we are trying to classify.

Train **k = 3** binary Perceptron models and compute **3** weighted sums:  $S_1, S_2, S_3$ .

Final prediction is the class with the largest weighted sum.



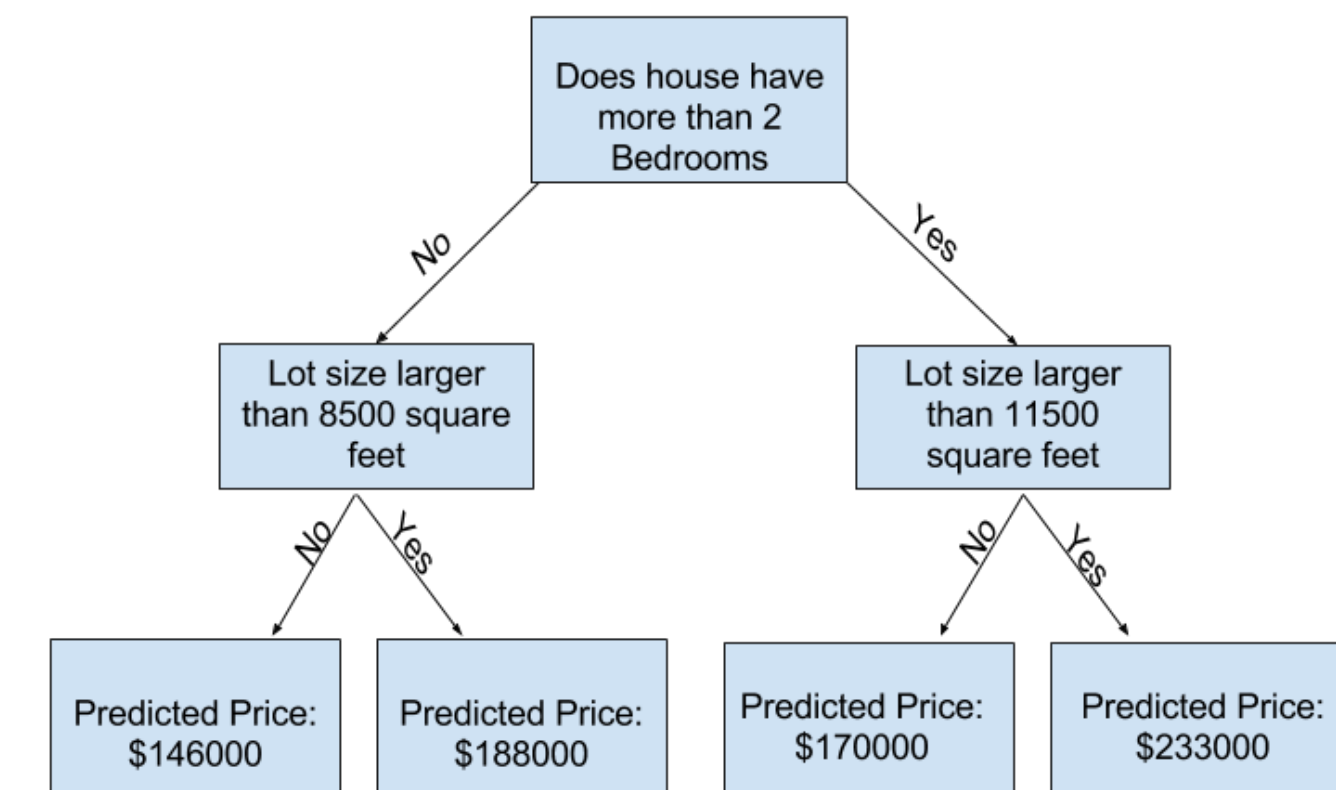
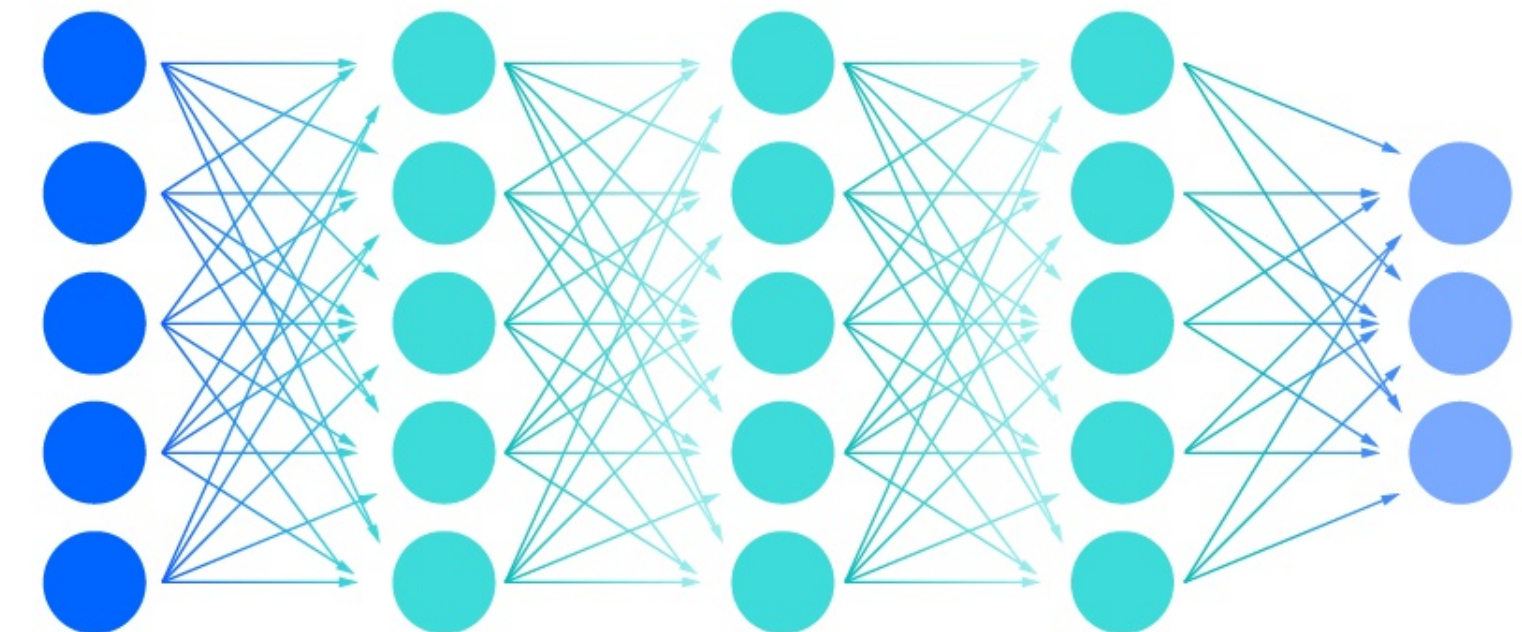
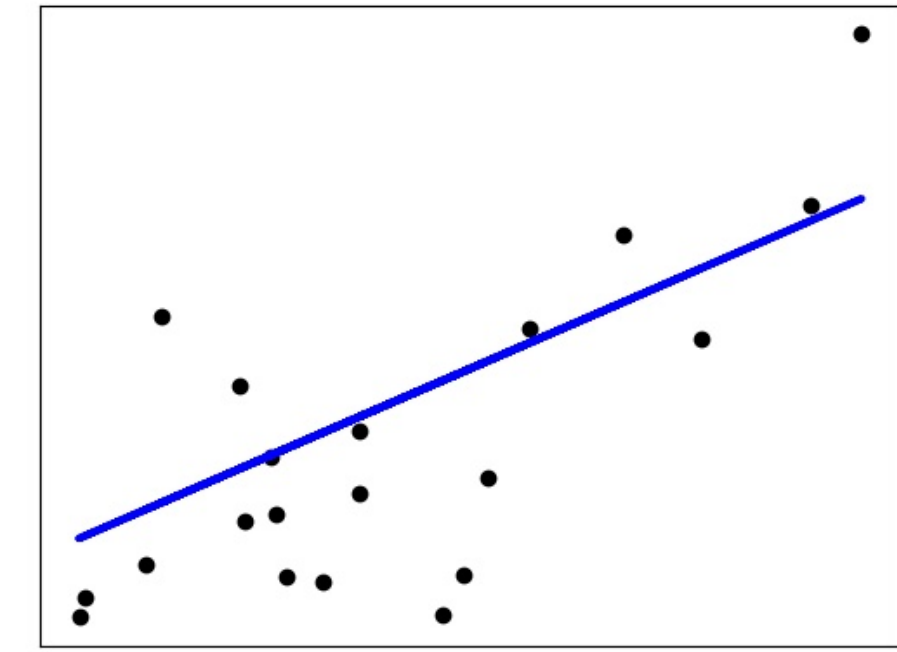
# Machine learning models

ML model: an algorithm used to perform a task.

Supervised ML models:

- Linear models (e.g. perceptron)
- Deep neural networks\* (e.g. multi-layer perceptron)
- Decision trees
- Ensembles (e.g. random forests)
- ...

\*also for unsupervised and reinforcement learning



# Readings

Required:

- Etienne Bernard, Wolfram. Introduction to Machine Learning: Machine Learning Paradigms. <https://www.wolfram.com/language/introduction-machine-learning/machine-learning-paradigms/>



# Credits

Lecture contents based in part on the following:

- COS324 instructors (especially Ruth Fong) and lectures.
- Etienne Bernard, Wolfram. Introduction to Machine Learning: Machine Learning Paradigms. <https://www.wolfram.com/language/introduction-machine-learning/machine-learning-paradigms/>
- Shree Nayar. Perceptron | Neural Networks. [https://youtu.be/OFbnpY\\_k7js?si=wrWOvNXE3UVCq-7Z](https://youtu.be/OFbnpY_k7js?si=wrWOvNXE3UVCq-7Z)

# Credits

Media	Source	License
AlphaFold	<a href="#">AlphaFold</a>	<a href="#">CC BY 4.0</a>
ChatGPT screenshot	<a href="#">Mashable</a>	
Cicero	<a href="#">Science</a>	
DALL-E 2	<a href="#">Open AI</a>	
Dancing Robot	<a href="#">Disney Research</a>	
Basketball Tabular Data	<a href="#">Statology</a>	
Object detection NYC	<a href="#">Ilija Mihajlovic</a>	
Chat GPT market	<a href="#">MarketWatch</a>	
Robotic arms	<a href="#">Facebook Research</a>	
Audio wave	<a href="#">COS 126</a>	
Blood pressure	<a href="#">Wikimedia</a>	<a href="#">CC0 1.0</a>
Object segmentation	<a href="#">PASCAL VOC</a>	
Segmentation for cars	<a href="#">pytorch-semseg</a>	

# Credits

Media	Source	License
Google Translate	<a href="#">Analytics Drift</a>	
Clustering	<a href="#">Google Developers</a>	
Dimensionality reduction	<a href="#">Wolfram</a>	
PCA Colleges	<a href="#">Ruth Fong</a>	
Anomaly detection	<a href="#">Wolfram</a>	
StyleGAN	<a href="#">arXiv</a>	
Reinforcement learning	<a href="#">Wolfram</a>	
CartPole	<a href="#">Gymnasium</a>	
Cart Pole Games	<a href="#">Andreas De Neve</a>	
Atari Breakout	<a href="#">OpenAI Gym</a>	
AlphaGo	<a href="#">New Yorker</a>	
Rubik's Cube robot arm	<a href="#">OpenAI</a>	
TossingBot	<a href="#">Google Research</a>	



# Credits

Media	Source	License
Laundry folding robot	<a href="#">NPR</a>	
US election map	<a href="#">Wikimedia</a>	<a href="#">CC0 1.0</a>
Doctor lung scan	<a href="#">Pexels</a>	
Golden retriever	<a href="#">Emojis.sh</a>	
Doberman	<a href="#">Emojis.sh</a>	
Neuron	<a href="#">Wikimedia</a>	<a href="#">CC BY-SA 4.0</a>
Linear regression	<a href="#">scikit-learn</a>	
Deep neural network	<a href="#">IBM</a>	
Decision tree	<a href="#">Kaggle</a>	