



CS 540 Introduction to Artificial Intelligence

Neural Networks I: Perceptron

University of Wisconsin-Madison
Fall 2026 Sections 1 & 2

Announcements

- **Homework:** HW5 due on **Friday Oct 16 at 11:59 PM**
- Midterm course evaluations are this week.
- **Midterm exam:** Oct 23 (next Friday) at 5.45 pm
Humanities Building Room 2340

- Class roadmap:

Machine Learning: Neural
Networks I (Perceptron)

Machine Learning: Neural
Networks II

Machine Learning: Neural
Networks III

} Supervised Learning

Outline

- Single-layer Neural Network
- Activation Functions
- Multi-layer Neural Network



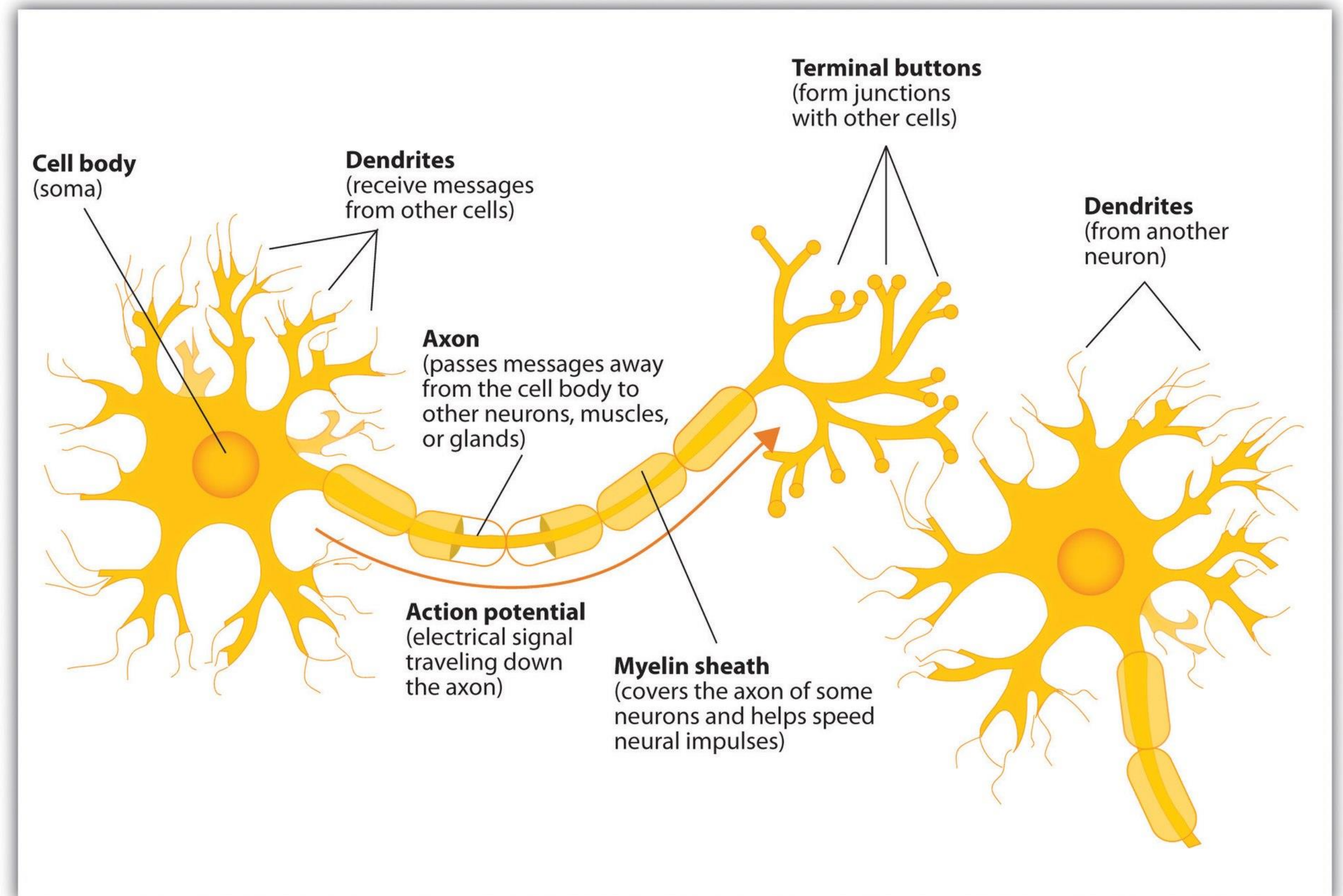
Single-layer Neural Network

Inspiration from neuroscience

- Inspirations from human brains
- Networks of **simple** and **homogenous** units

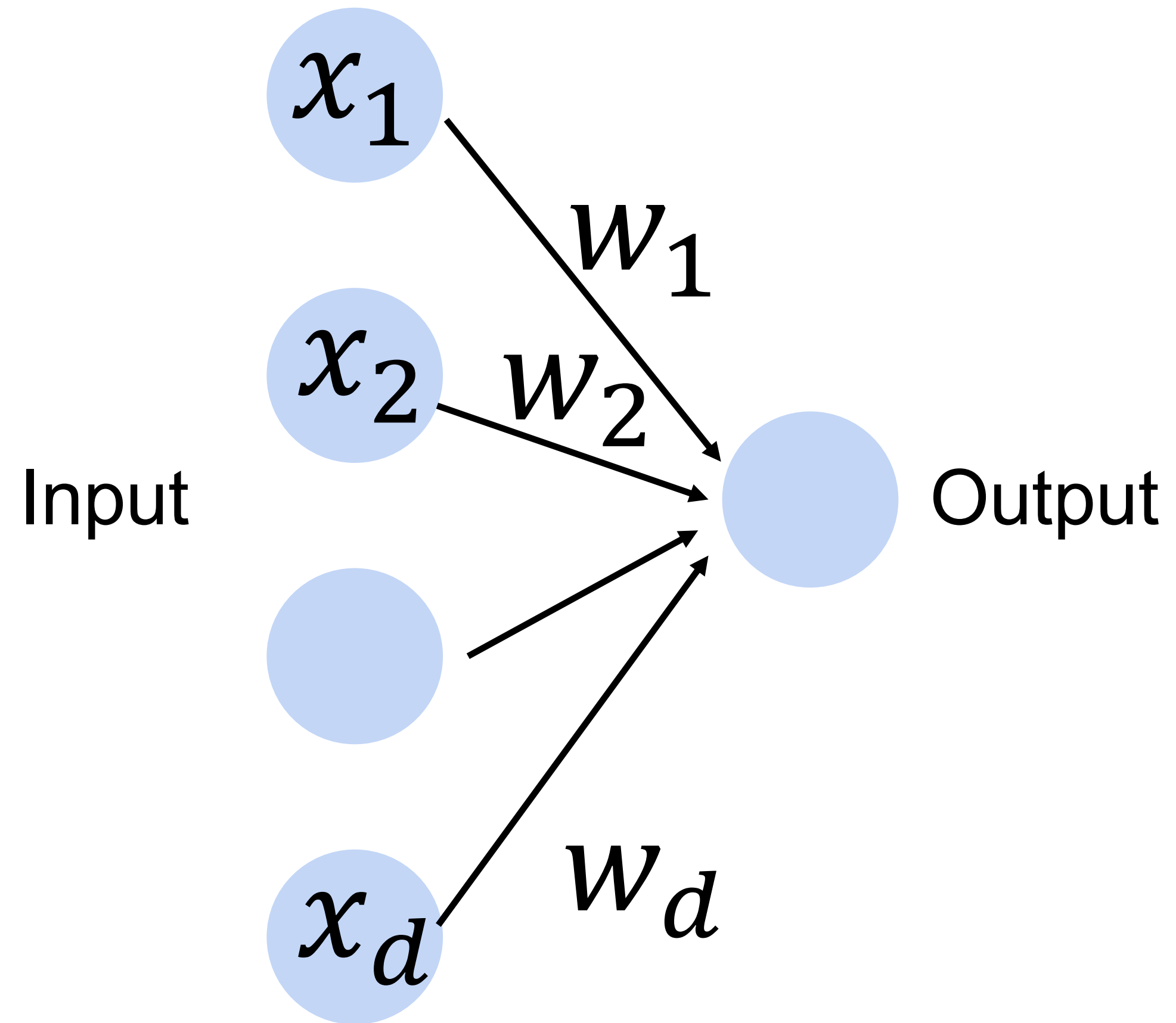


(wikipedia)



Starting point: The Perceptron

Cats vs. dogs?



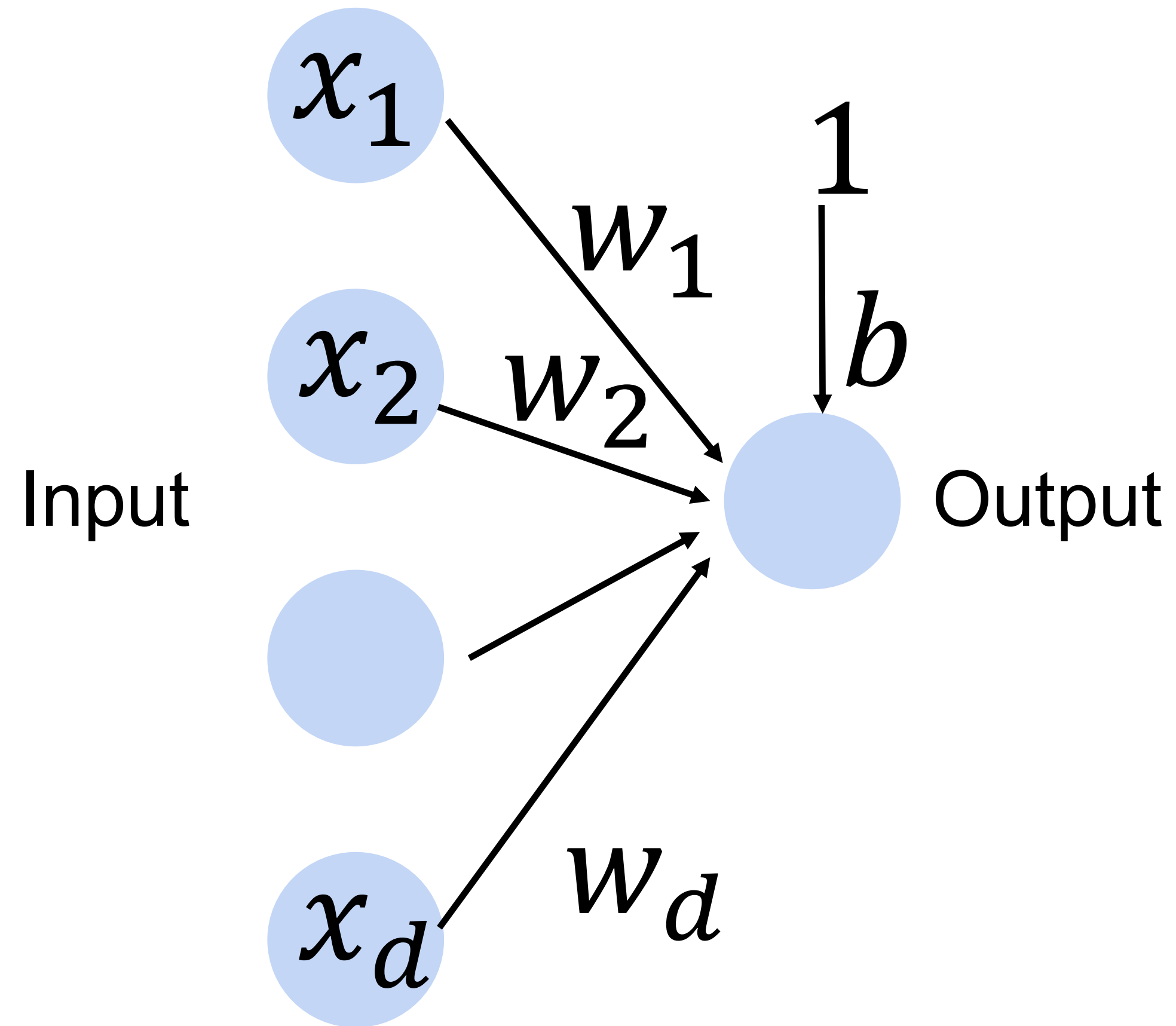
Starting point: The Perceptron

Given input $\mathbf{x}$, weight $\mathbf{w}$, and bias b , the perceptron outputs:

$$\hat{y} = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$
$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{otherwise} \end{cases}$$

activation function

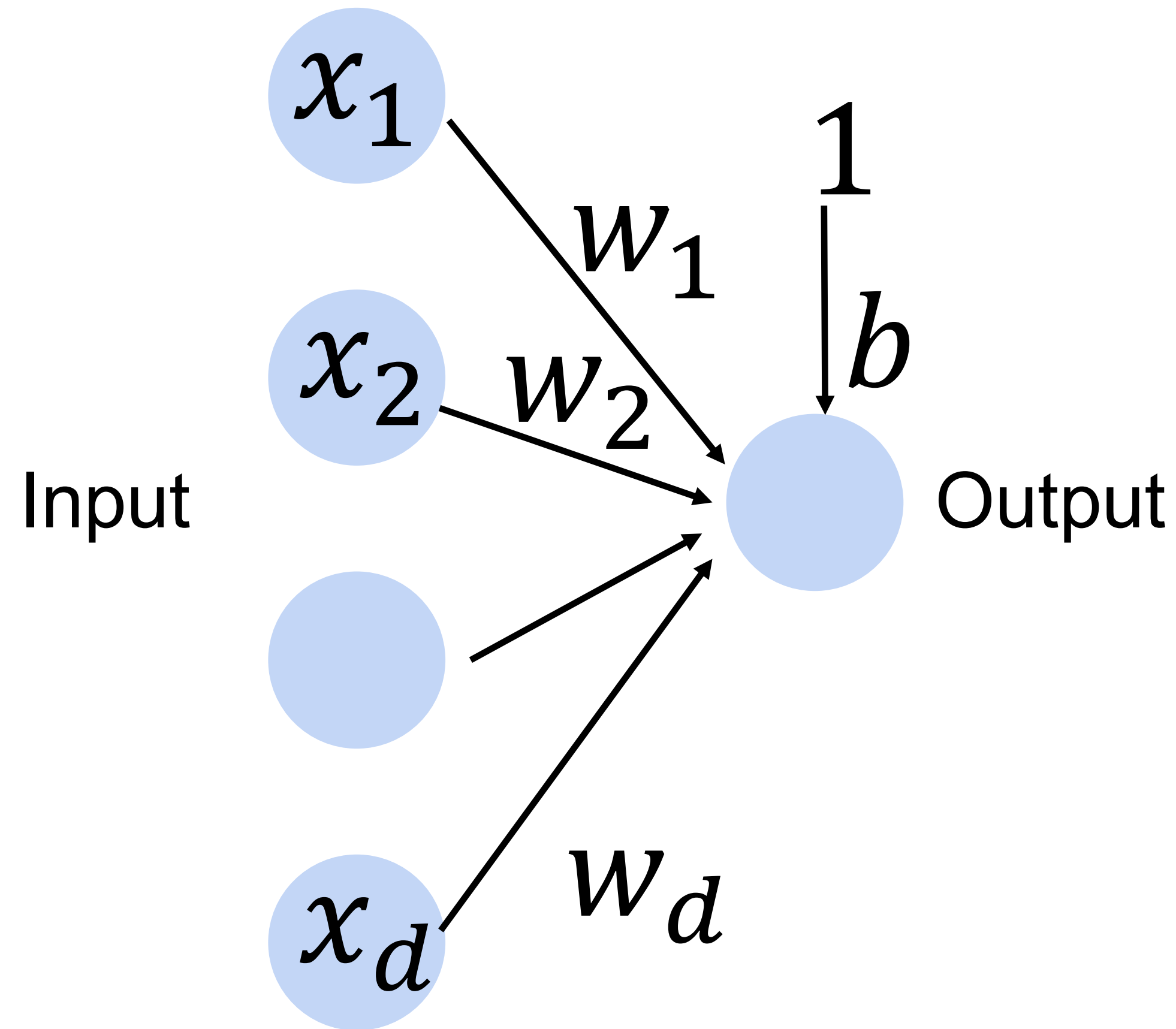
Cats vs. dogs?



Starting point: The Perceptron

Goal: learn parameters $\mathbf{w} = \{w_1, w_2, \dots, w_d\}$ and b to minimize the classification error

Cats vs. dogs?



Training the Perceptron

- An iterative stochastic algorithm to decrease loss on training data
 - Start with initial weights w_0, b_0
 - Randomly pick next training example
 - When perceptron misclassifies a point, update the weights

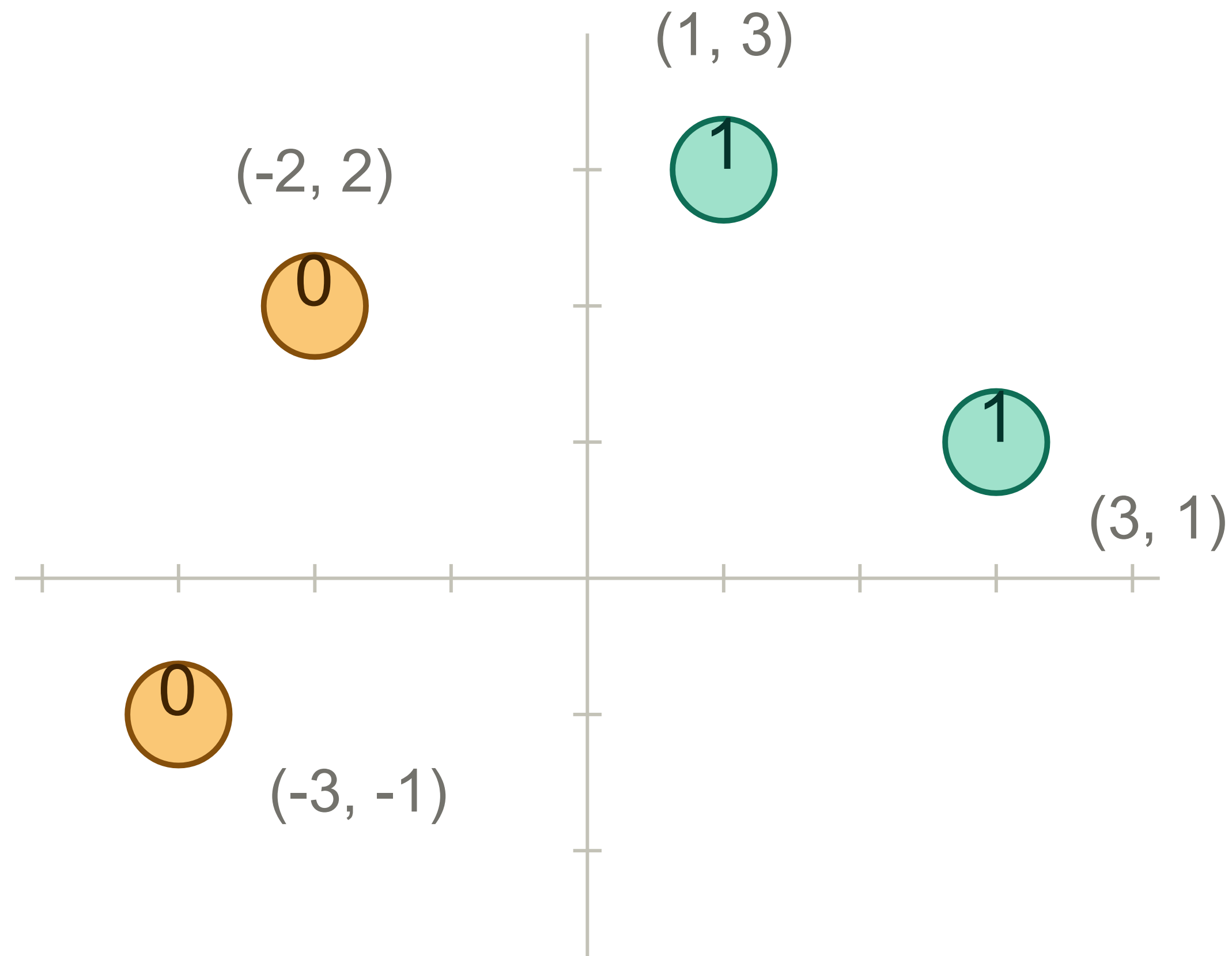
Perceptron Learning Algorithm

Input: dataset (X, y)

number of steps T , step size η

1. Initialize w_0, b_0
2. For $t = 1, 2, \dots, T$
3. Pick random (x_i, y_i)
4. Predict $\hat{y}_i \leftarrow \sigma(\langle w_{t-1}, x_i \rangle + b_{t-1})$
5. If $\hat{y}_i \neq y_i$:
6. $w_t \leftarrow w_{t-1} + \eta(y_i - \hat{y}_i)x_i$
7. $b_t \leftarrow b_{t-1} + \eta(y_i - \hat{y}_i)$
8. Return w_T

Perceptron



Training set. Each point is marked with its label y .
Goal: find w with $\langle w, x \rangle > 0$ exactly when $y = 1$.

Take $\eta = 1$

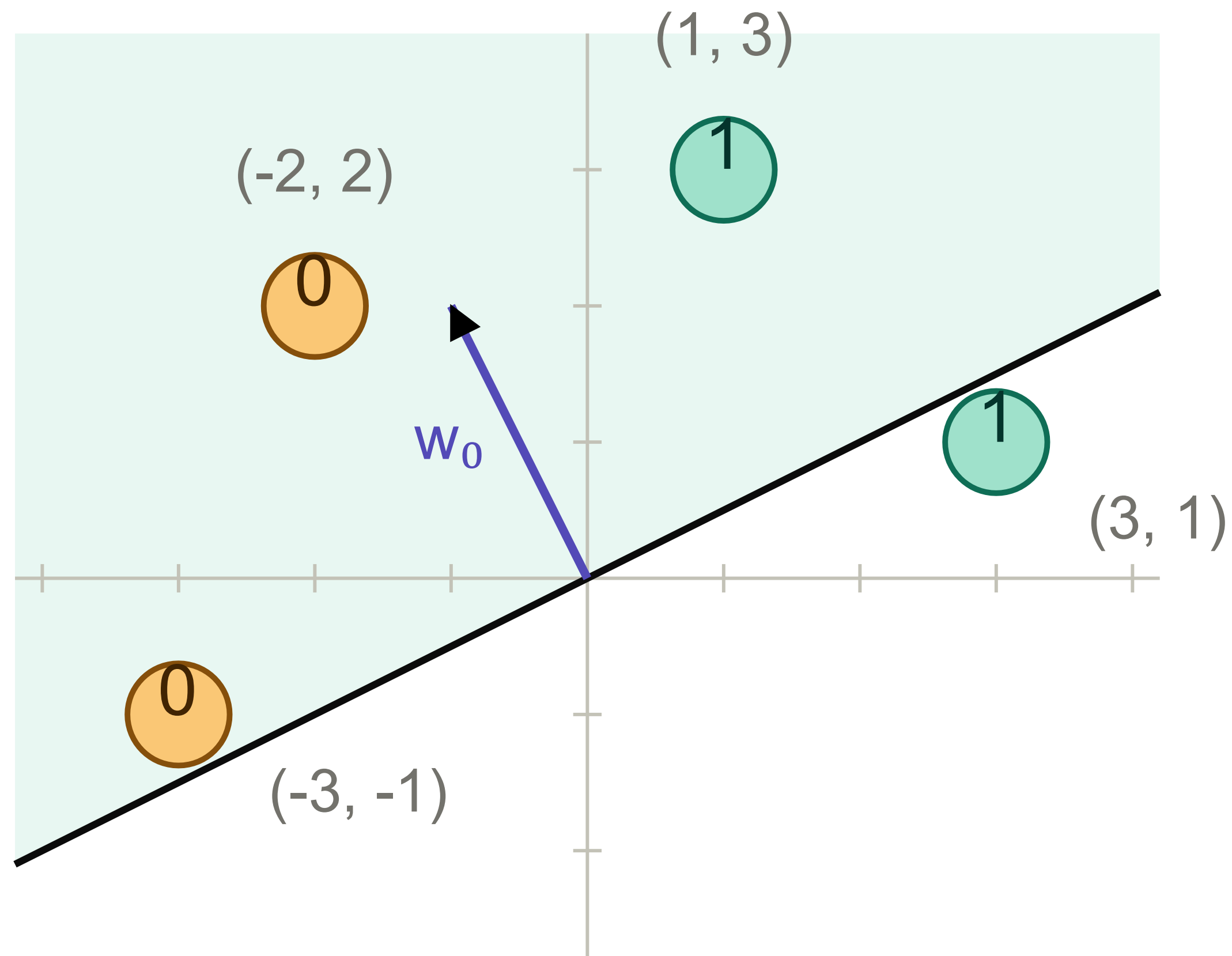
Perceptron Learning Algorithm

Input: dataset (X, y)

number of steps T , step size η

1. Initialize w_0, b_0
2. For $t = 1, 2, \dots, T$
3. Pick random (x_i, y_i)
4. Predict $\hat{y}_i \leftarrow \sigma(\langle w_{t-1}, x_i \rangle + b_{t-1})$
5. If $\hat{y}_i \neq y_i$:
6. $w_t \leftarrow w_{t-1} + \eta(y_i - \hat{y}_i)x_i$
7. $b_t \leftarrow b_{t-1} + \eta(y_i - \hat{y}_i)$
8. Return w_T

Perceptron



Initialize $w_0 = (-1, 2)$. The boundary $\langle w_0, x \rangle = 0$ is perpendicular to w_0 . Shaded region is $\hat{y} = 1$.

Take $\eta = 1$

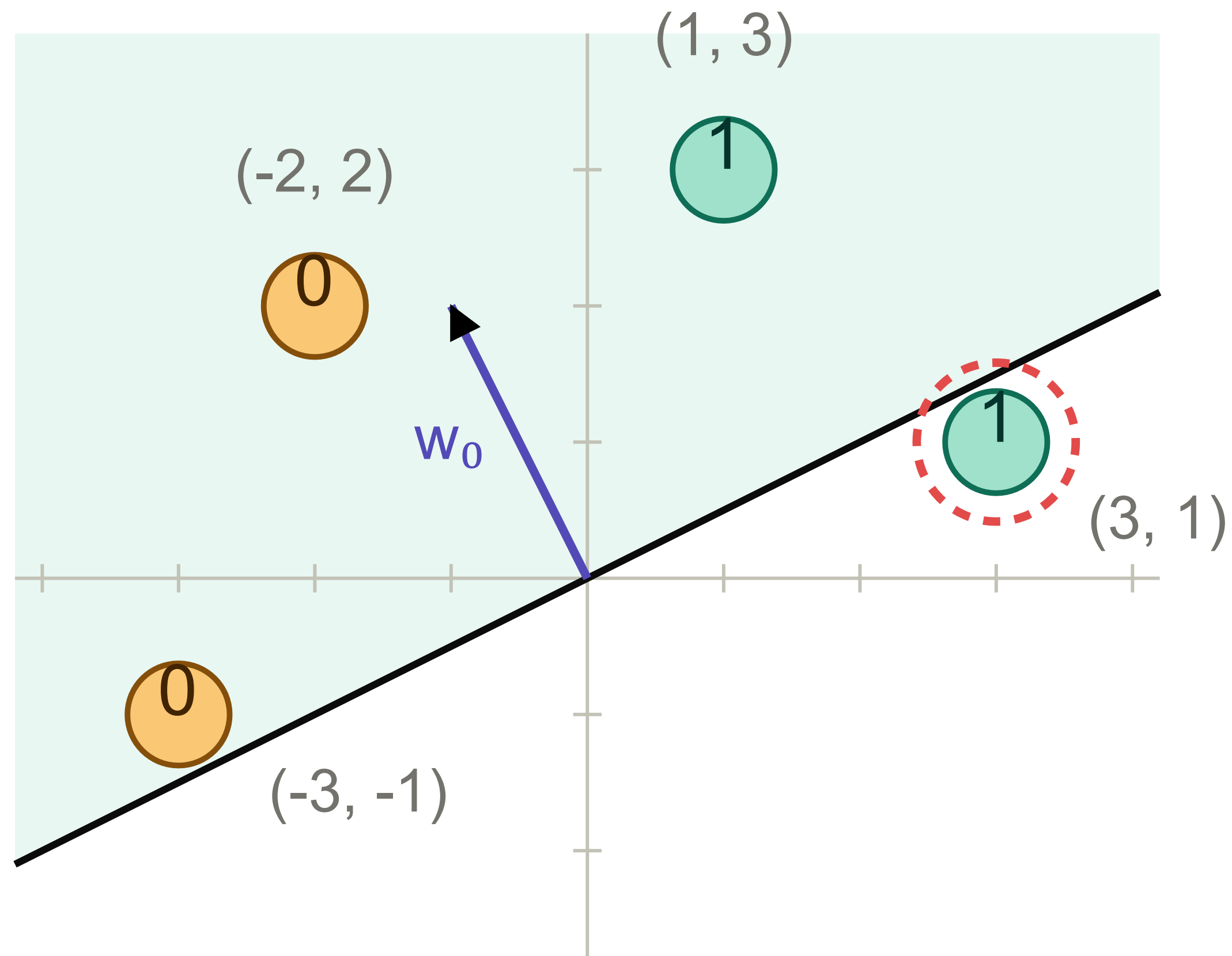
Perceptron Learning Algorithm

Input: dataset (X, y)

number of steps T , step size η

1. Initialize w_0, b_0
2. For $t = 1, 2, \dots, T$
3. Pick random (x_i, y_i)
4. Predict $\hat{y}_i \leftarrow \sigma(\langle w_{t-1}, x_i \rangle + b_{t-1})$
5. If $\hat{y}_i \neq y_i$:
6. $w_t \leftarrow w_{t-1} + \eta(y_i - \hat{y}_i)x_i$
7. $b_t \leftarrow b_{t-1} + \eta(y_i - \hat{y}_i)$
8. Return w_T

Perceptron



$t = 1$: pick $x = (3, 1)$, $y = 1$.

$\hat{y} = \sigma(-3 + 2) = 0 \neq y$, so this is a mistake.

Take $\eta = 1$

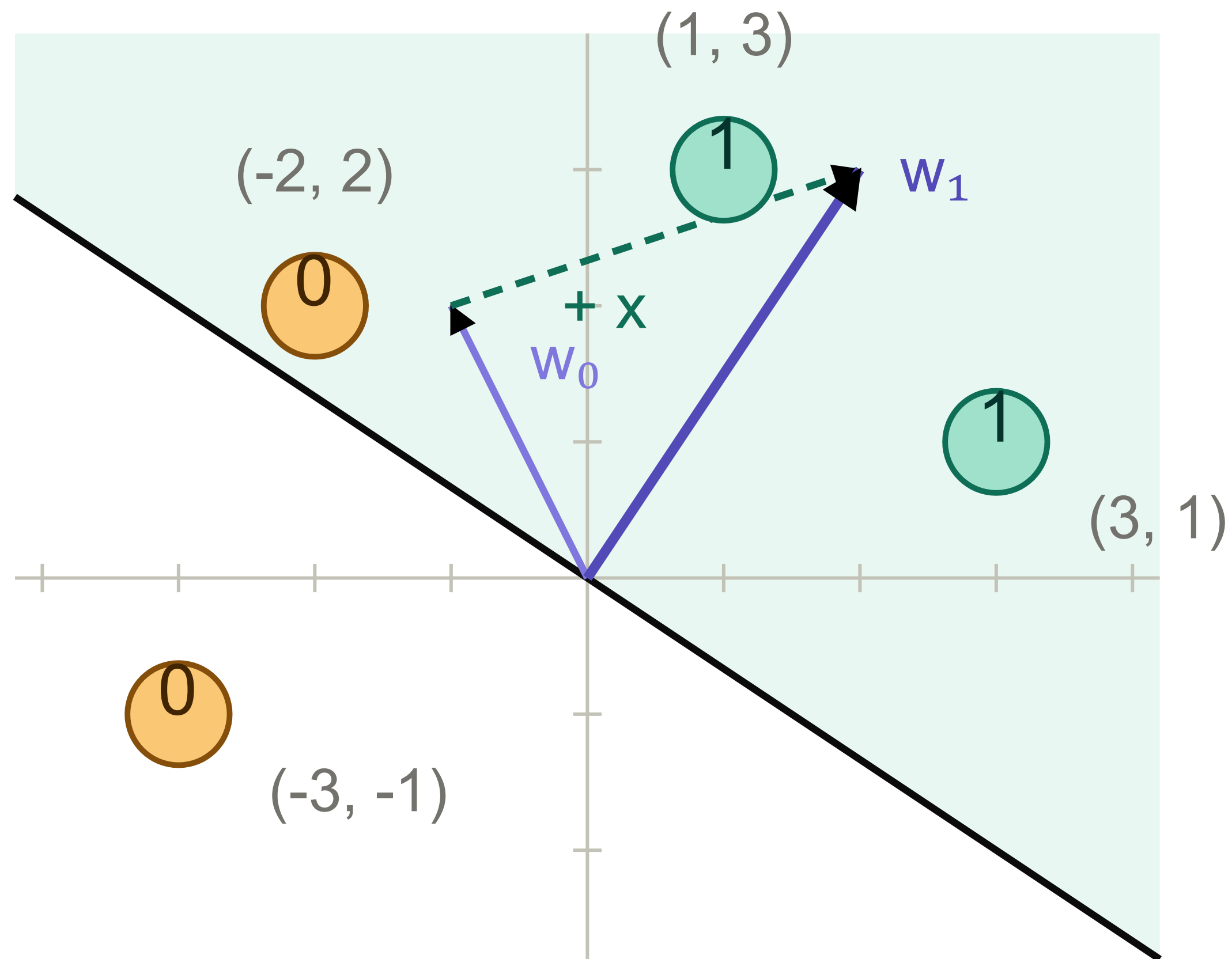
Perceptron Learning Algorithm

Input: dataset (X, y)

number of steps T , step size η

1. Initialize w_0, b_0
2. For $t = 1, 2, \dots, T$
3. Pick random (x_i, y_i)
4. Predict $\hat{y}_i \leftarrow \sigma(\langle w_{t-1}, x_i \rangle + b_{t-1})$
5. If $\hat{y}_i \neq y_i$:
6. $w_t \leftarrow w_{t-1} + \eta(y_i - \hat{y}_i)x_i$
7. $b_t \leftarrow b_{t-1} + \eta(y_i - \hat{y}_i)$
8. Return w_T

Perceptron



$$w_1 \leftarrow w_0 + \eta(y - \hat{y})x = (-1, 2) + (3, 1) = (2, 3)$$

The boundary rotates; (3, 1) is now correct.

Take $\eta = 1$

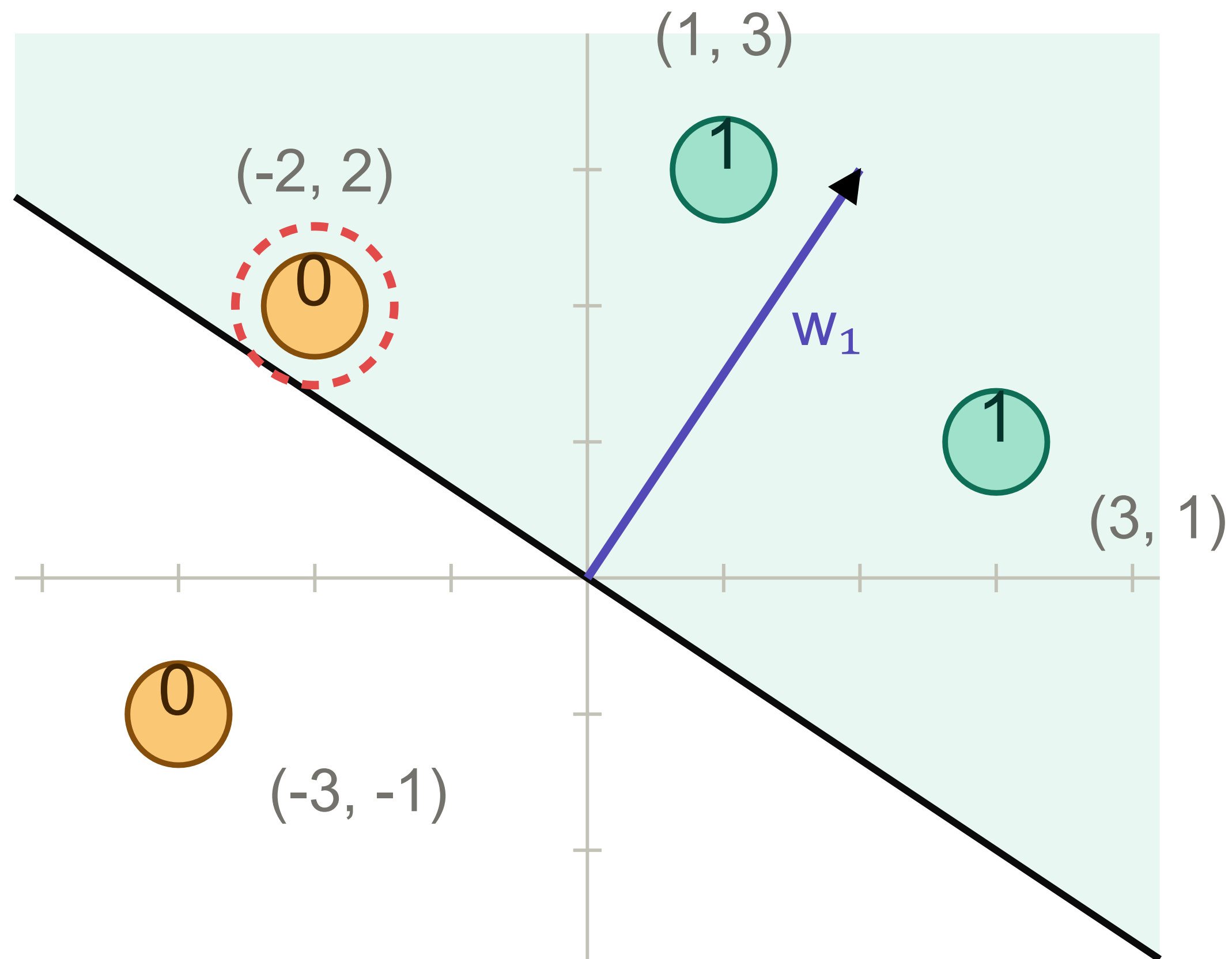
Perceptron Learning Algorithm

Input: dataset (X, y)

number of steps T , step size η

1. Initialize w_0, b_0
2. For $t = 1, 2, \dots, T$
3. Pick random (x_i, y_i)
4. Predict $\hat{y}_i \leftarrow \sigma(\langle w_{t-1}, x_i \rangle + b_{t-1})$
5. If $\hat{y}_i \neq y_i$:
6. $w_t \leftarrow w_{t-1} + \eta(y_i - \hat{y}_i)x_i$
7. $b_t \leftarrow b_{t-1} + \eta(y_i - \hat{y}_i)$
8. Return w_T

Perceptron



$t = 2$: pick $x = (-2, 2)$, $y = 0$.

$\hat{y} = \sigma(-4 + 6) = 1 \neq y$, another mistake.

Take $\eta = 1$

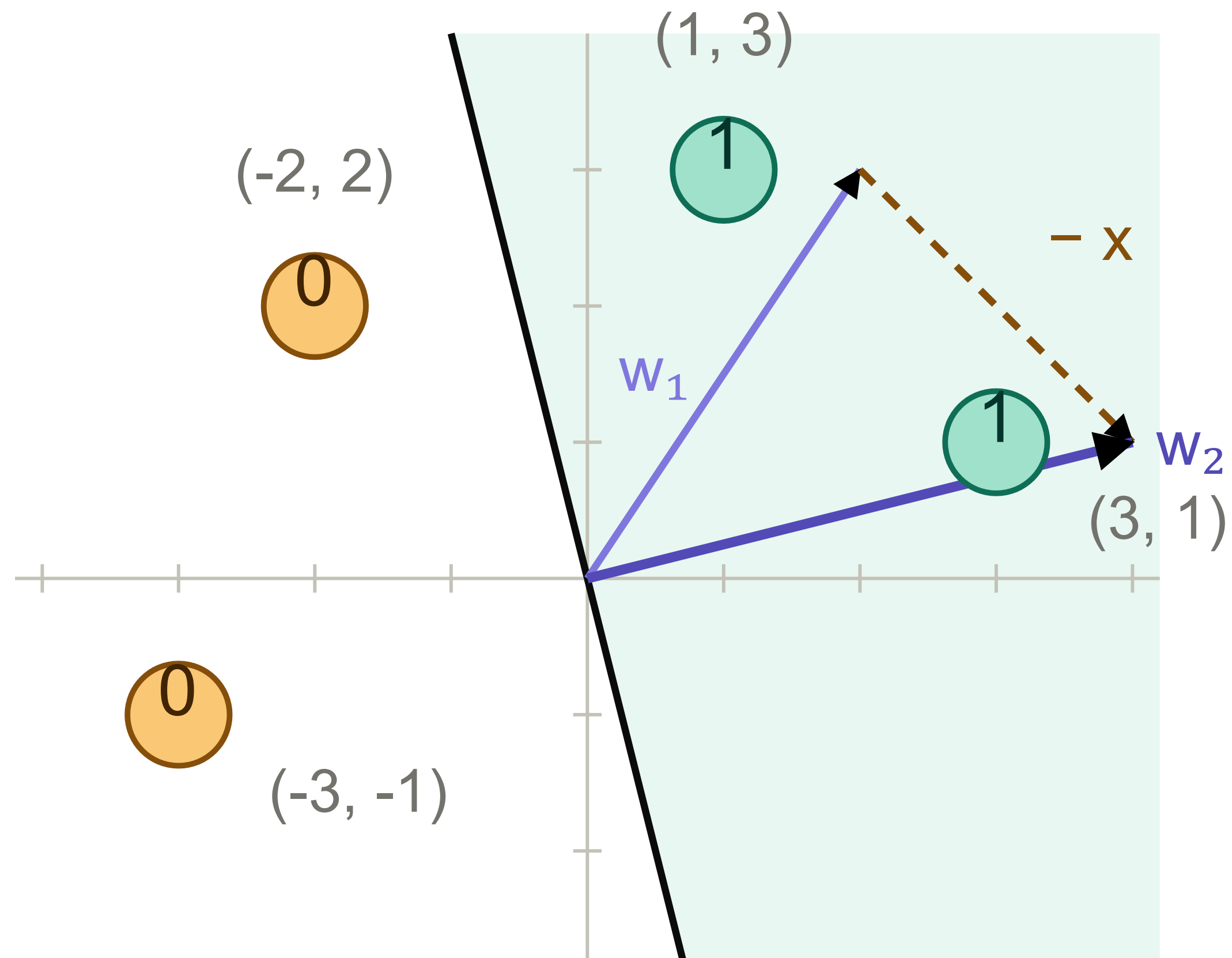
Perceptron Learning Algorithm

Input: dataset (X, y)

number of steps T , step size η

1. Initialize w_0, b_0
2. For $t = 1, 2, \dots, T$
3. Pick random (x_i, y_i)
4. Predict $\hat{y}_i \leftarrow \sigma(\langle w_{t-1}, x_i \rangle + b_{t-1})$
5. If $\hat{y}_i \neq y_i$:
6. $w_t \leftarrow w_{t-1} + \eta(y_i - \hat{y}_i)x_i$
7. $b_t \leftarrow b_{t-1} + \eta(y_i - \hat{y}_i)$
8. Return w_T

Perceptron



$$w_2 \leftarrow w_1 + \eta(y - \hat{y})x = (2, 3) - (-2, 2) = (4, 1)$$

All four points correct, so no update fires.

Take $\eta = 1$

Perceptron Learning Algorithm

Input: dataset (X, y)

number of steps T , step size η

1. Initialize w_0, b_0
2. For $t = 1, 2, \dots, T$
3. Pick random (x_i, y_i)
4. Predict $\hat{y}_i \leftarrow \sigma(\langle w_{t-1}, x_i \rangle + b_{t-1})$
5. If $\hat{y}_i \neq y_i$:
6. $w_t \leftarrow w_{t-1} + \eta(y_i - \hat{y}_i)x_i$
7. $b_t \leftarrow b_{t-1} + \eta(y_i - \hat{y}_i)$
8. Return w_T

Learning AND function using perceptron

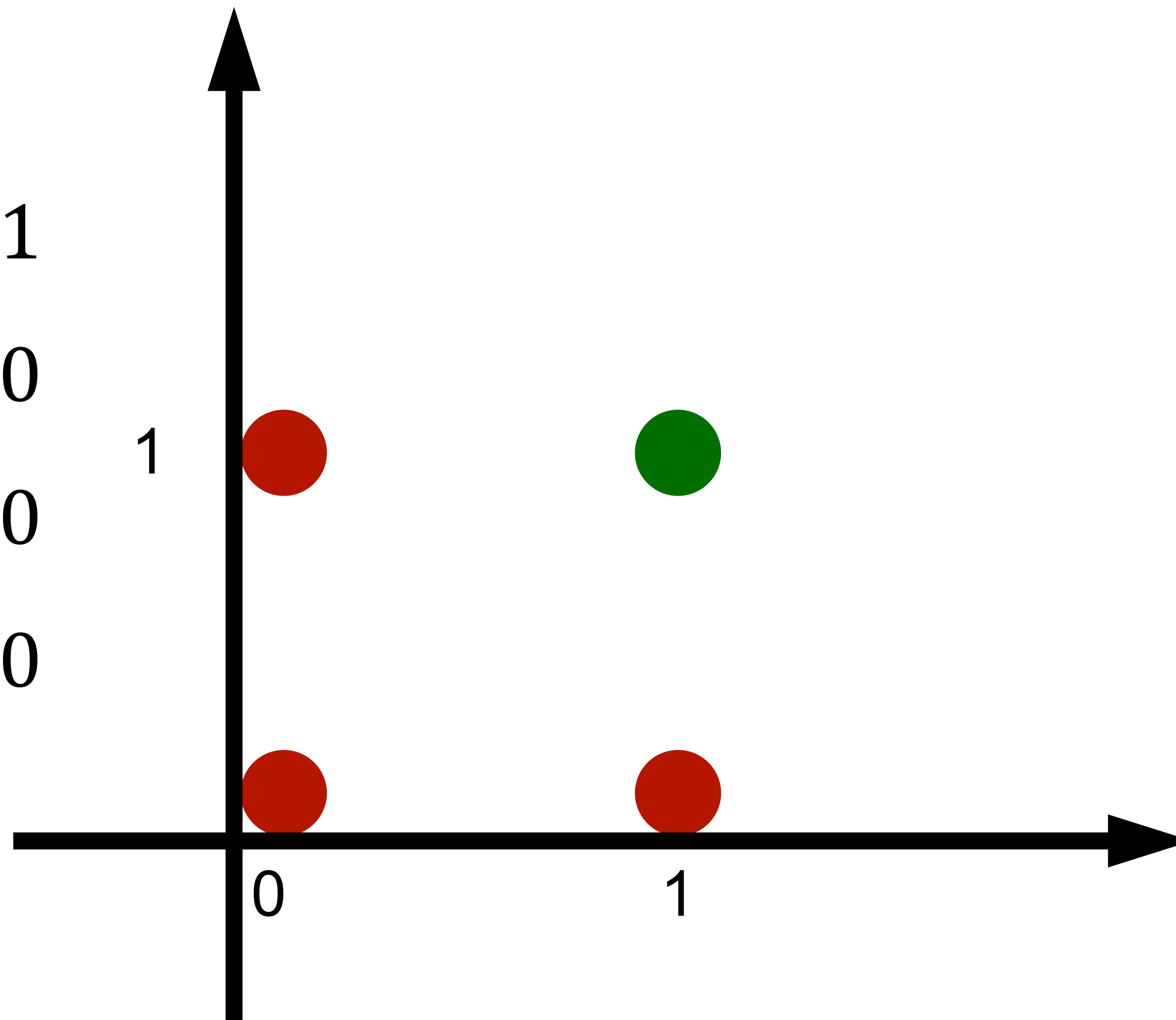
The perceptron can learn an AND function

$$x_1 = 1, x_2 = 1, y = 1$$

$$x_1 = 1, x_2 = 0, y = 0$$

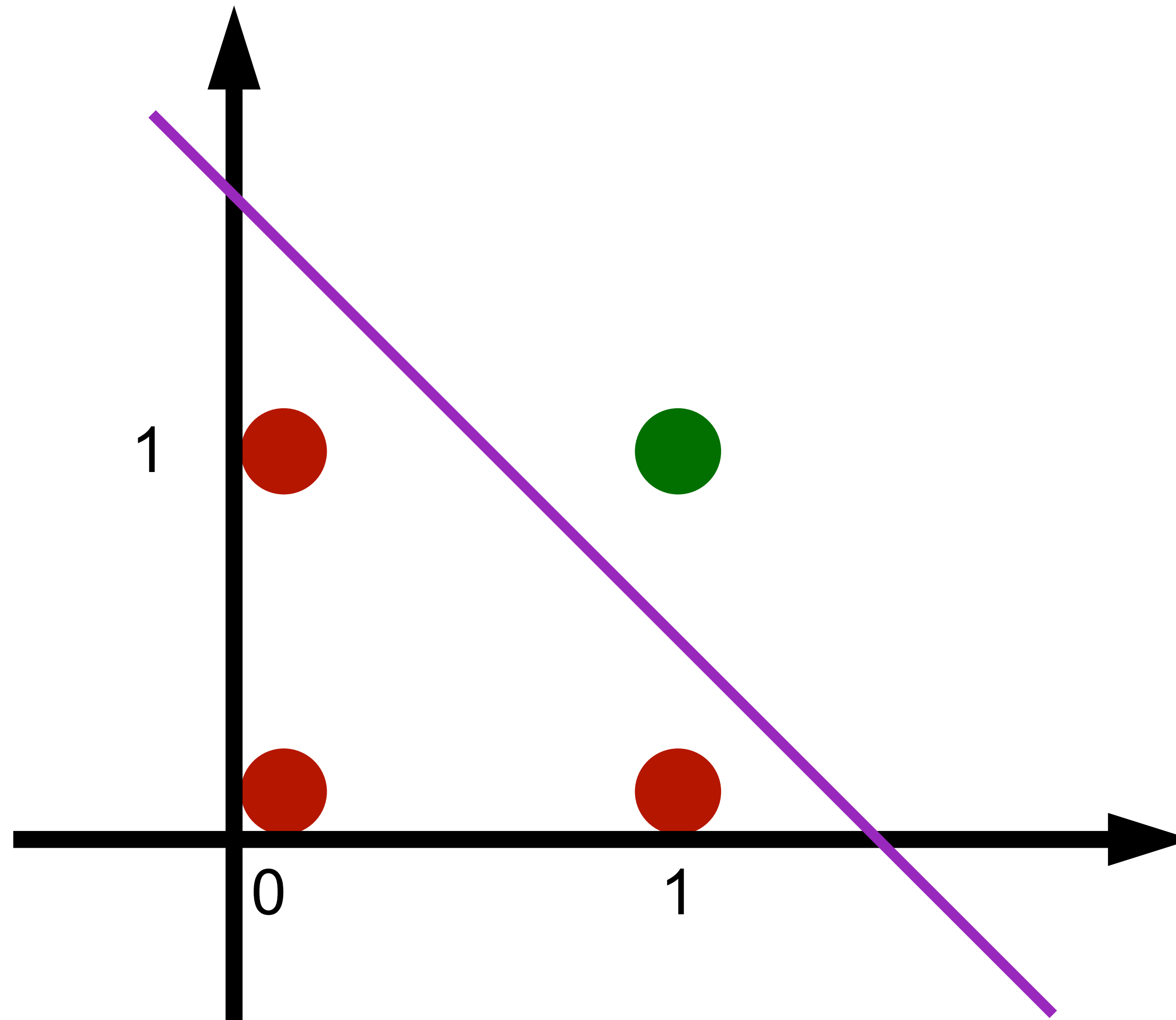
$$x_1 = 0, x_2 = 1, y = 0$$

$$x_1 = 0, x_2 = 0, y = 0$$



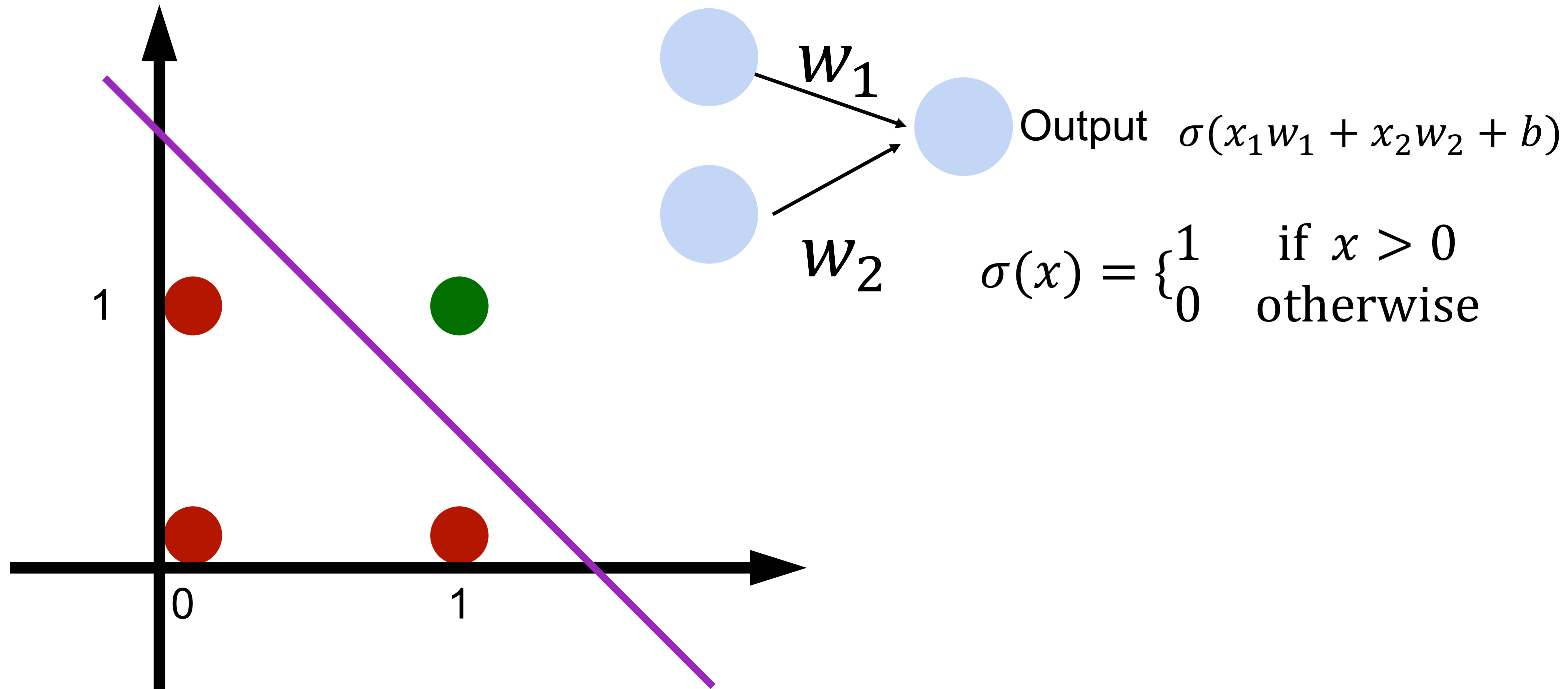
Learning AND function using perceptron

The perceptron can learn an AND function



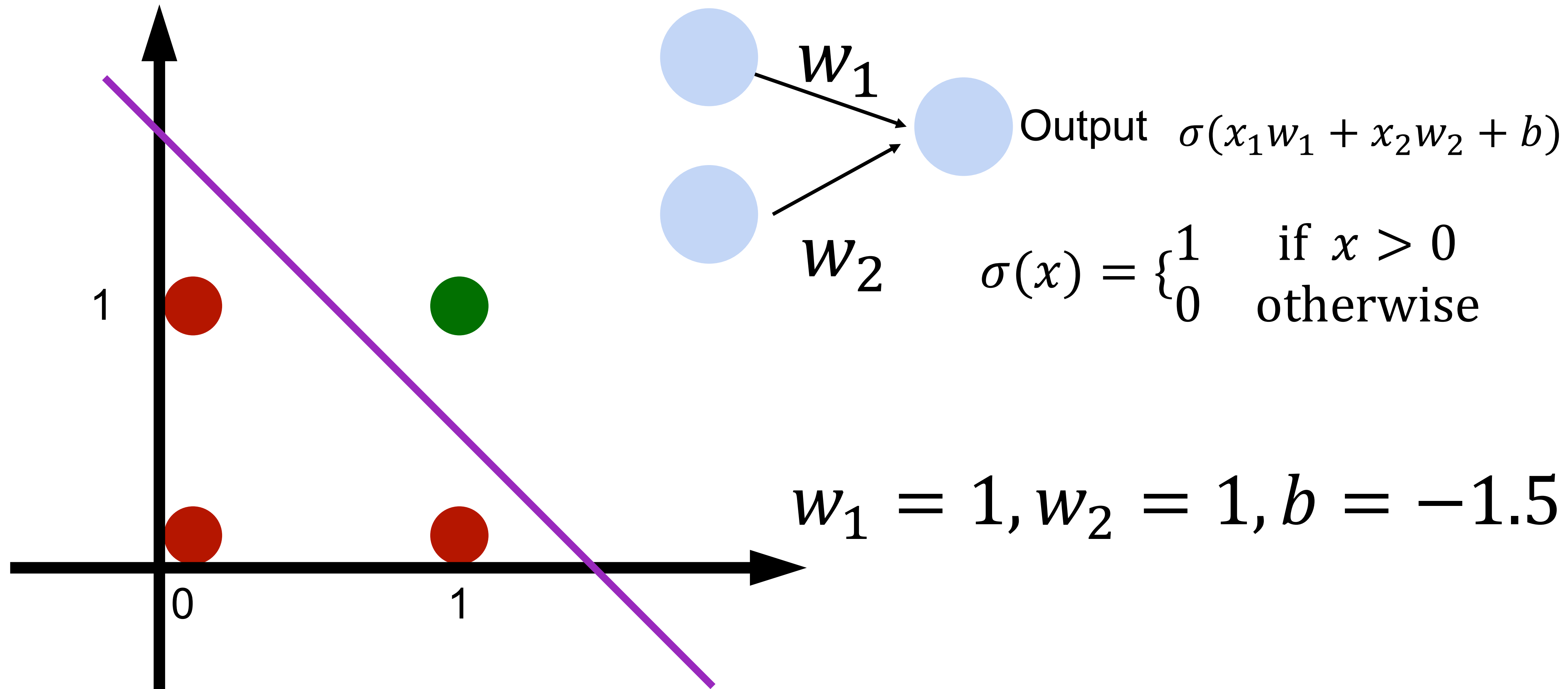
Learning AND function using perceptron

The perceptron can learn an AND function



Learning AND function using perceptron

The perceptron can learn an AND function



Learning OR function using perceptron

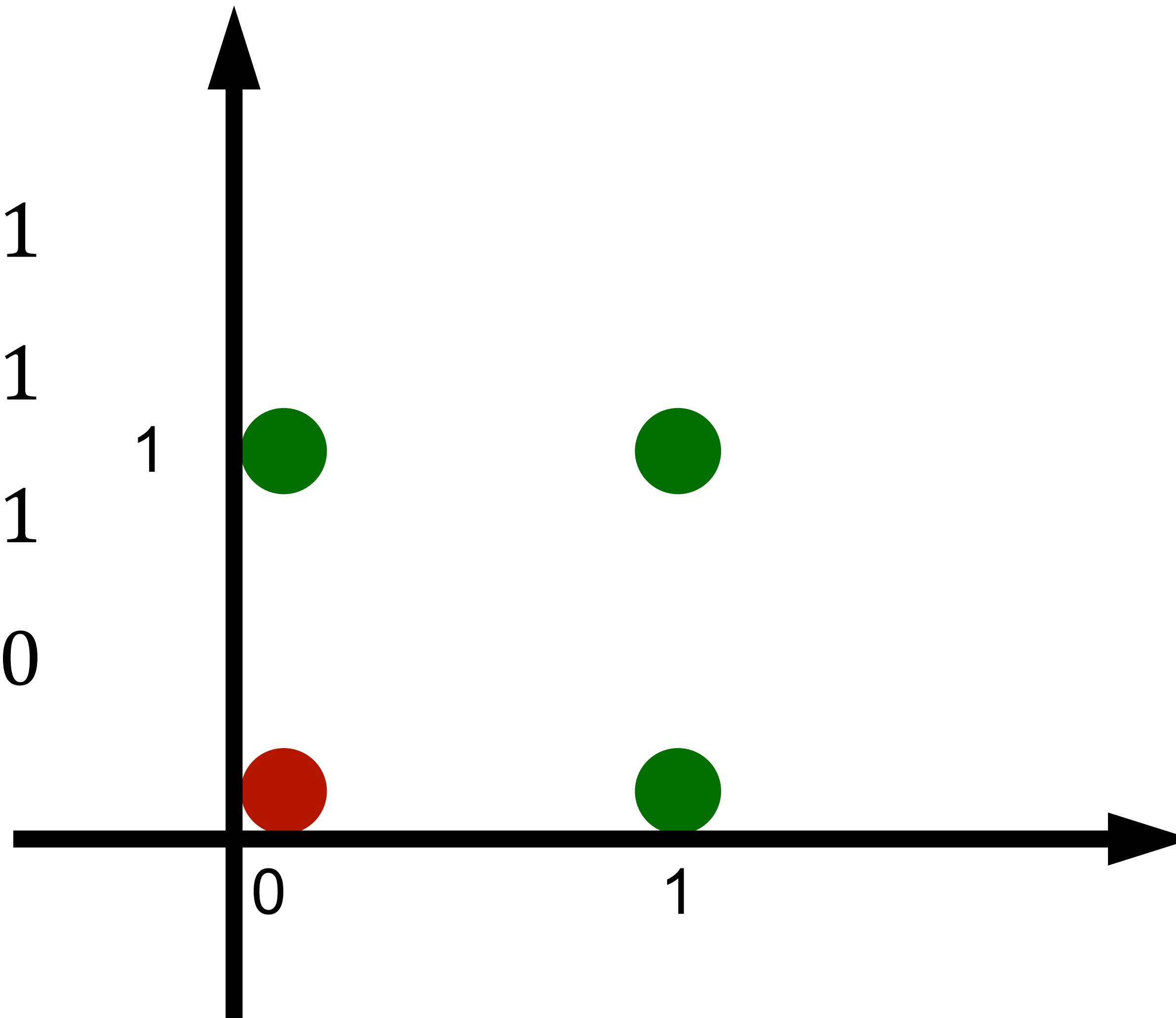
The perceptron can learn an OR function

$$x_1 = 1, x_2 = 1, y = 1$$

$$x_1 = 1, x_2 = 0, y = 1$$

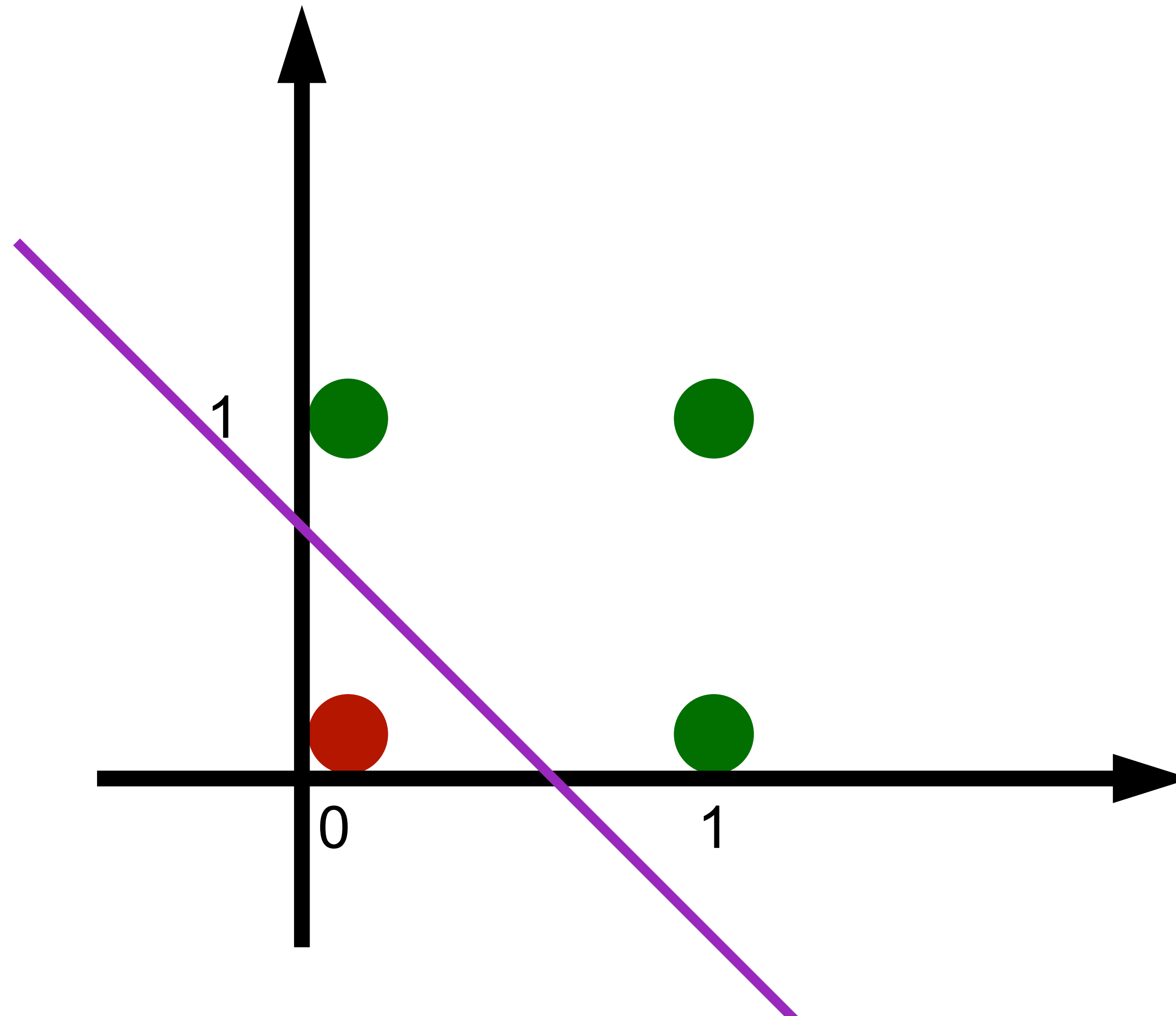
$$x_1 = 0, x_2 = 1, y = 1$$

$$x_1 = 0, x_2 = 0, y = 0$$



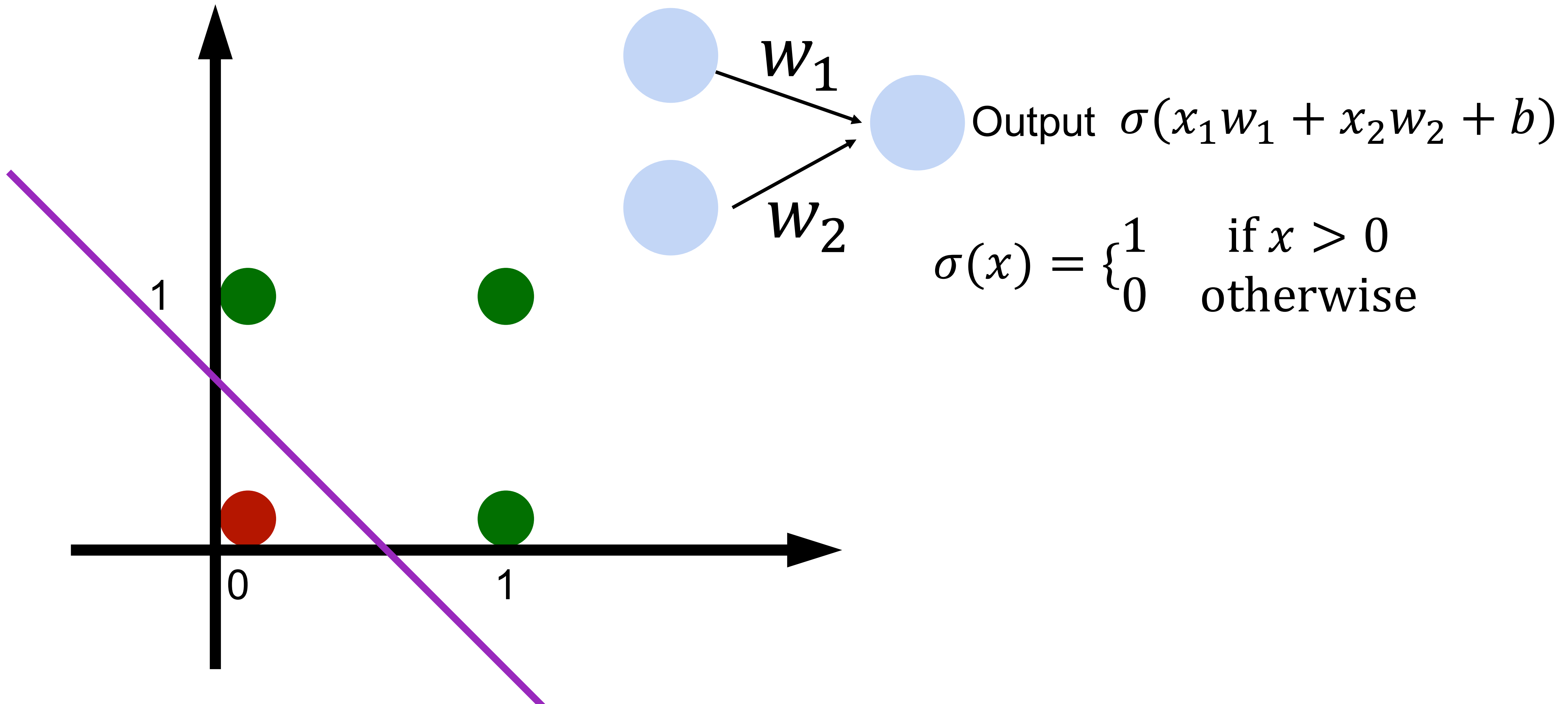
Learning OR function using perceptron

The perceptron can learn an OR function



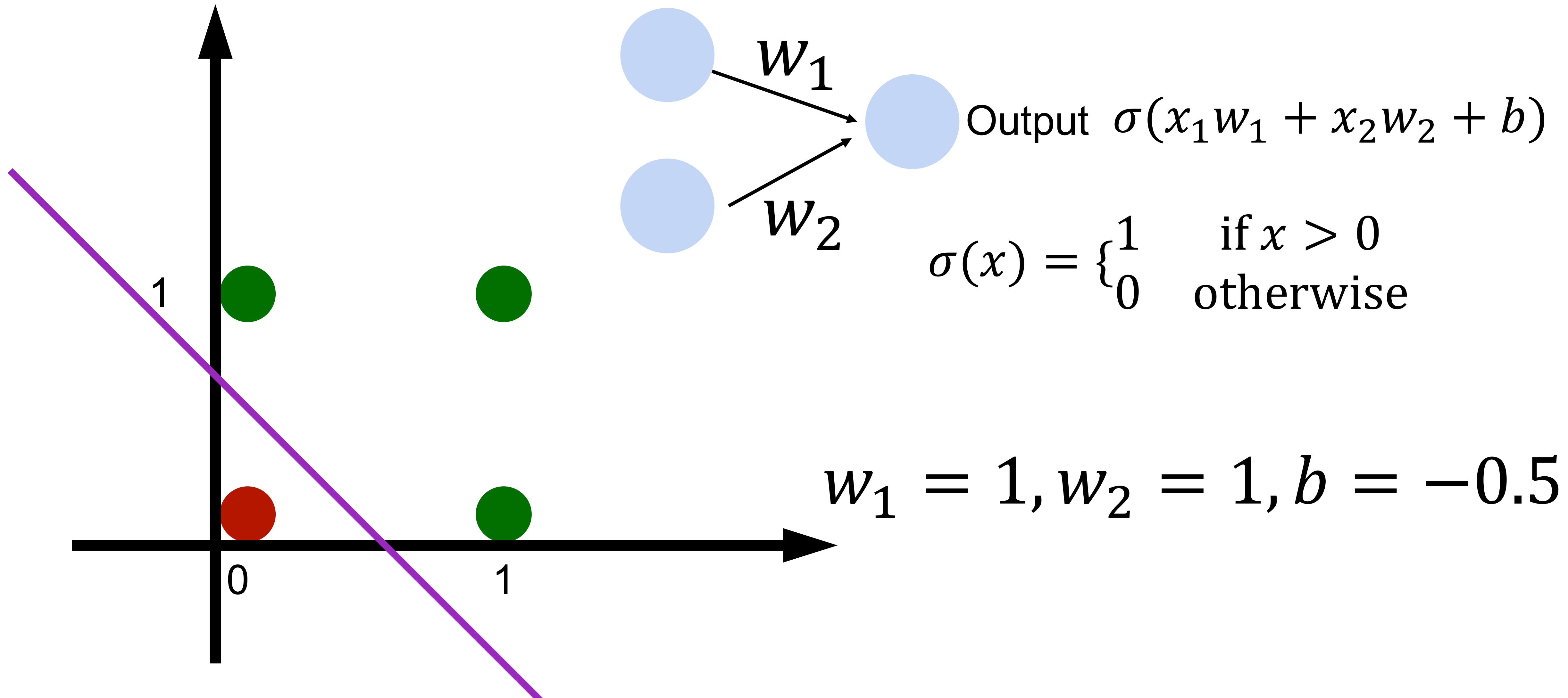
Learning OR function using perceptron

The perceptron can learn an OR function



Learning OR function using perceptron

The perceptron can learn an OR function



Learning NOT function using perceptron

The perceptron can learn NOT function (single input)



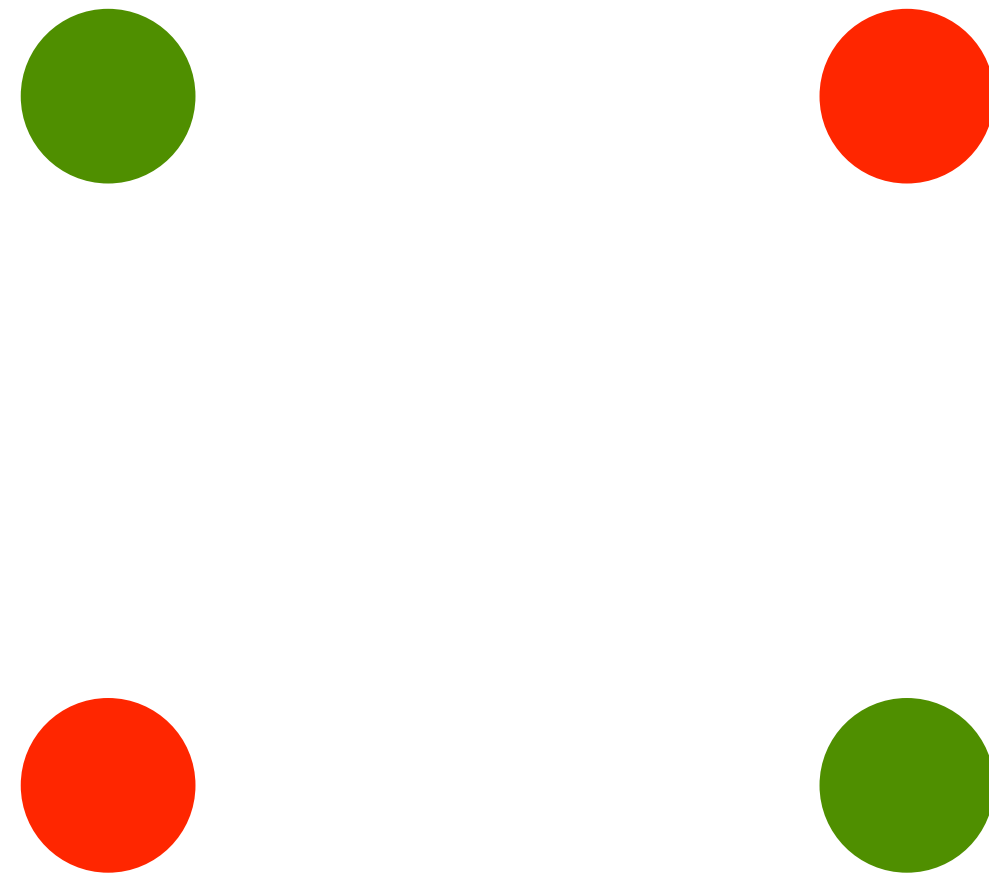
$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$



$$w_1 = -1, b = 0.5$$

XOR function (Minsky & Papert, 1969)

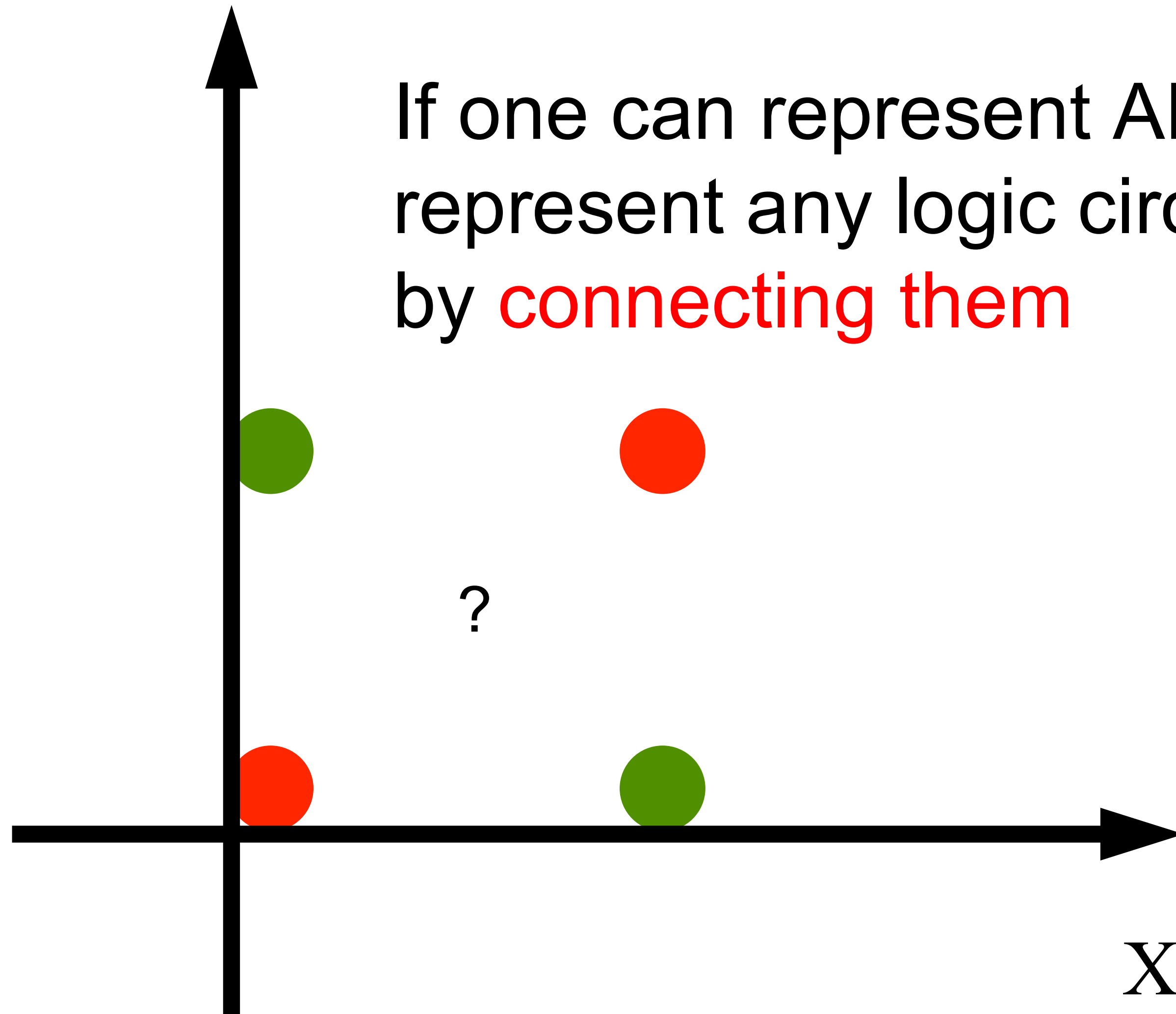
The perceptron cannot learn an XOR function
(it can only generate linear separators)



The limited power of a single neuron

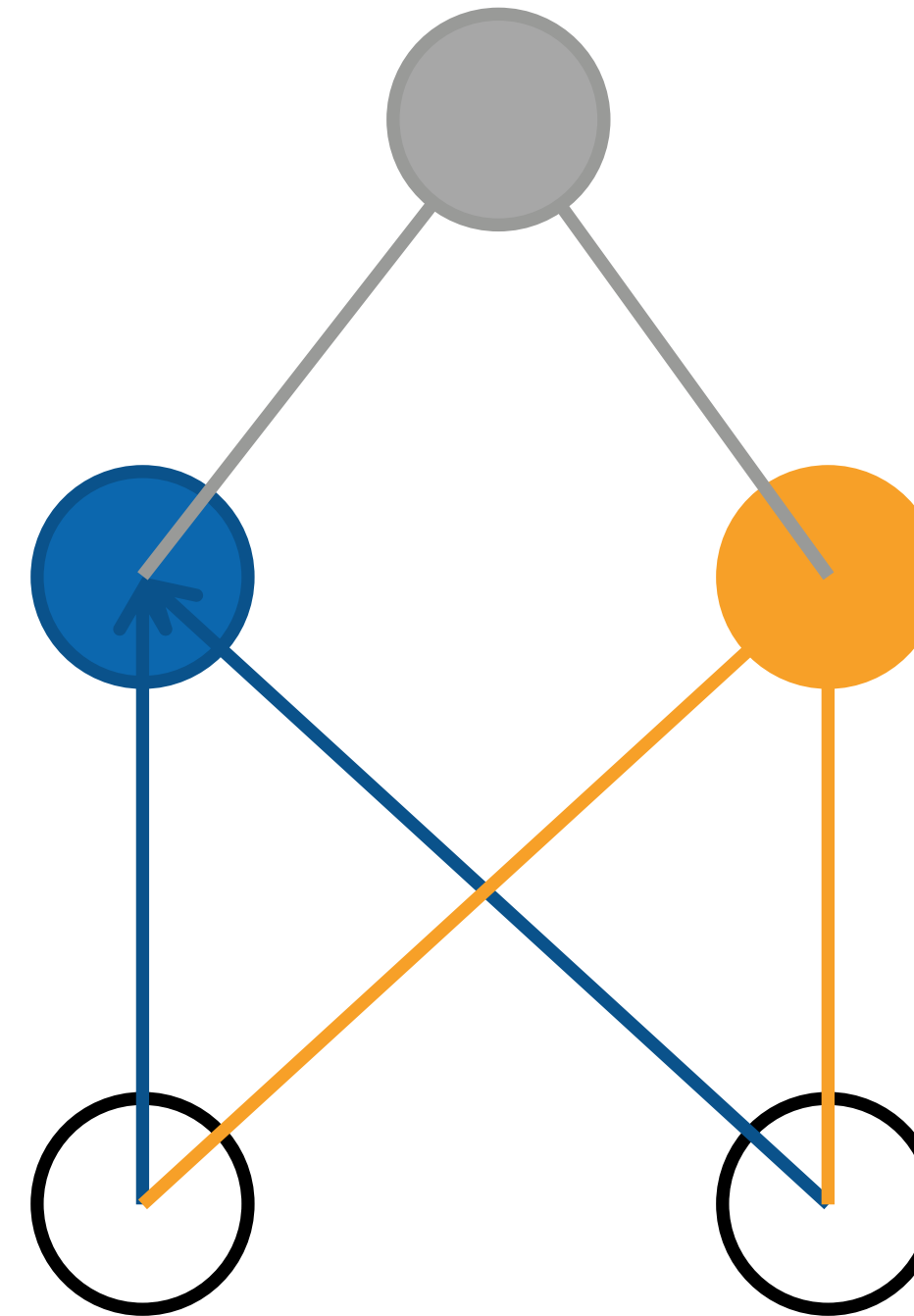
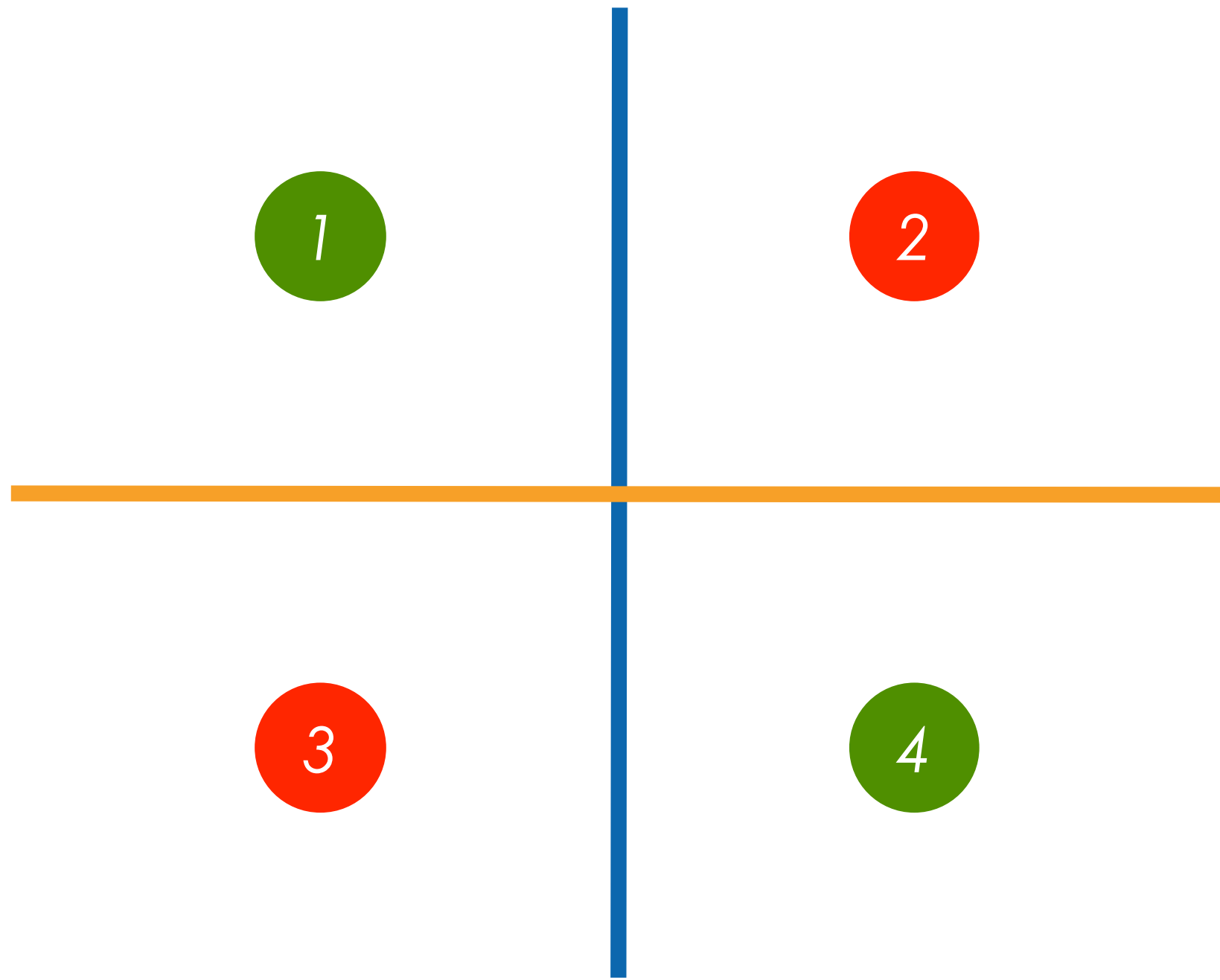
XOR problem

If one can represent AND, OR, NOT, one can represent any logic circuit (including XOR),
by **connecting them**



$$\text{XOR}(x_1, x_2) = (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$$

Learning XOR



This motivates using **multi-layer** perceptrons.



Activation Functions

Activation functions

What is an activation function?

- nonlinearity applied after taking the weighted sum
- inspired by neurons firing after exceeding a threshold
- in multi-layer networks (coming soon), they are applied **elementwise**

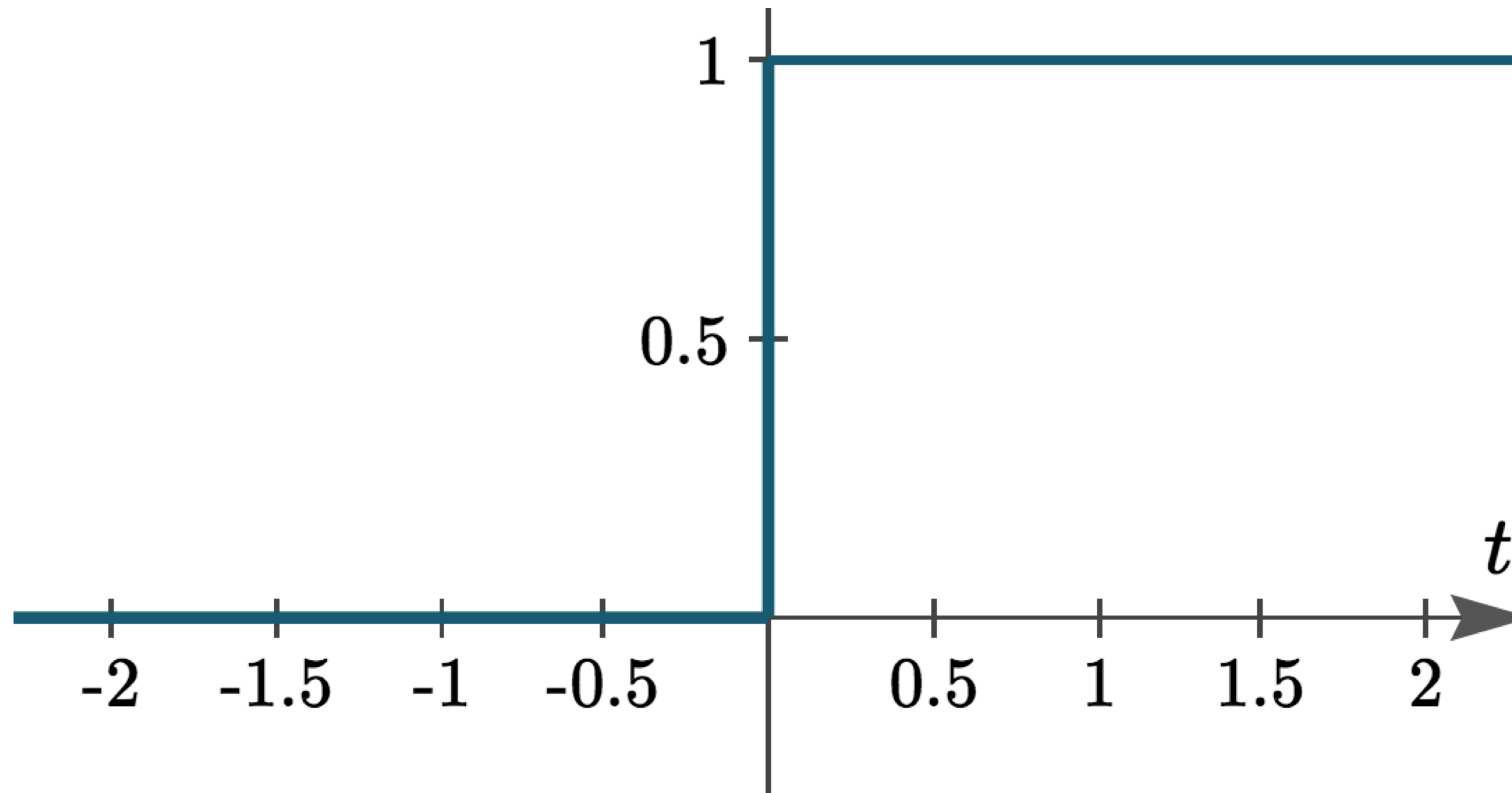
Examples:

- step functions
- sigmoid (logistic)
- tanh
- ReLU

Step Function activation (used in perceptron)

Step function is discontinuous, which cannot be used for gradient descent

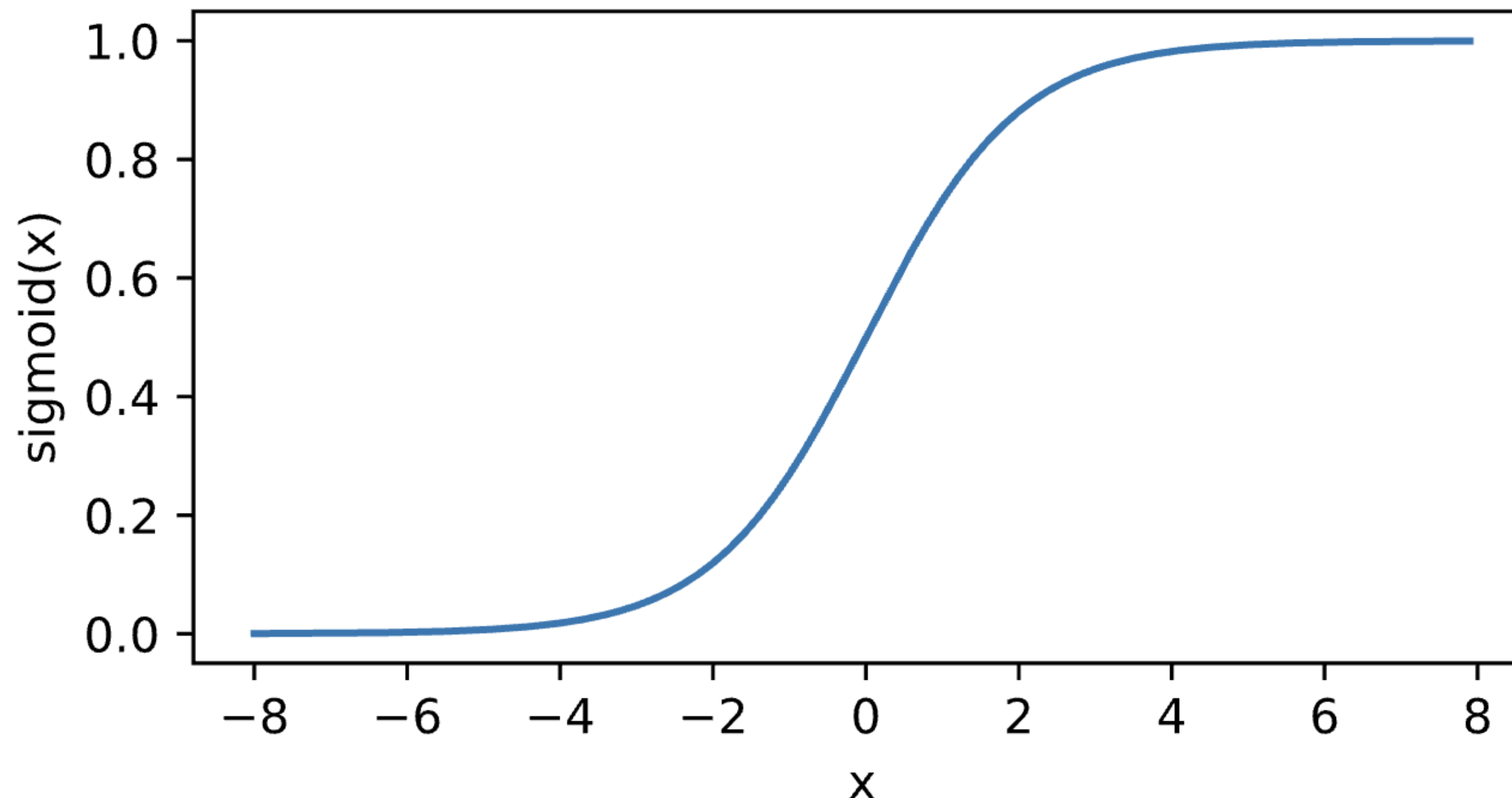
$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$



Sigmoid/Logistic Activation

Map input into $[0, 1]$, a **soft** version of $\sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$

$$\sigma(z) = \text{sigmoid}(z) = \frac{1}{1 + \exp(-z)}$$

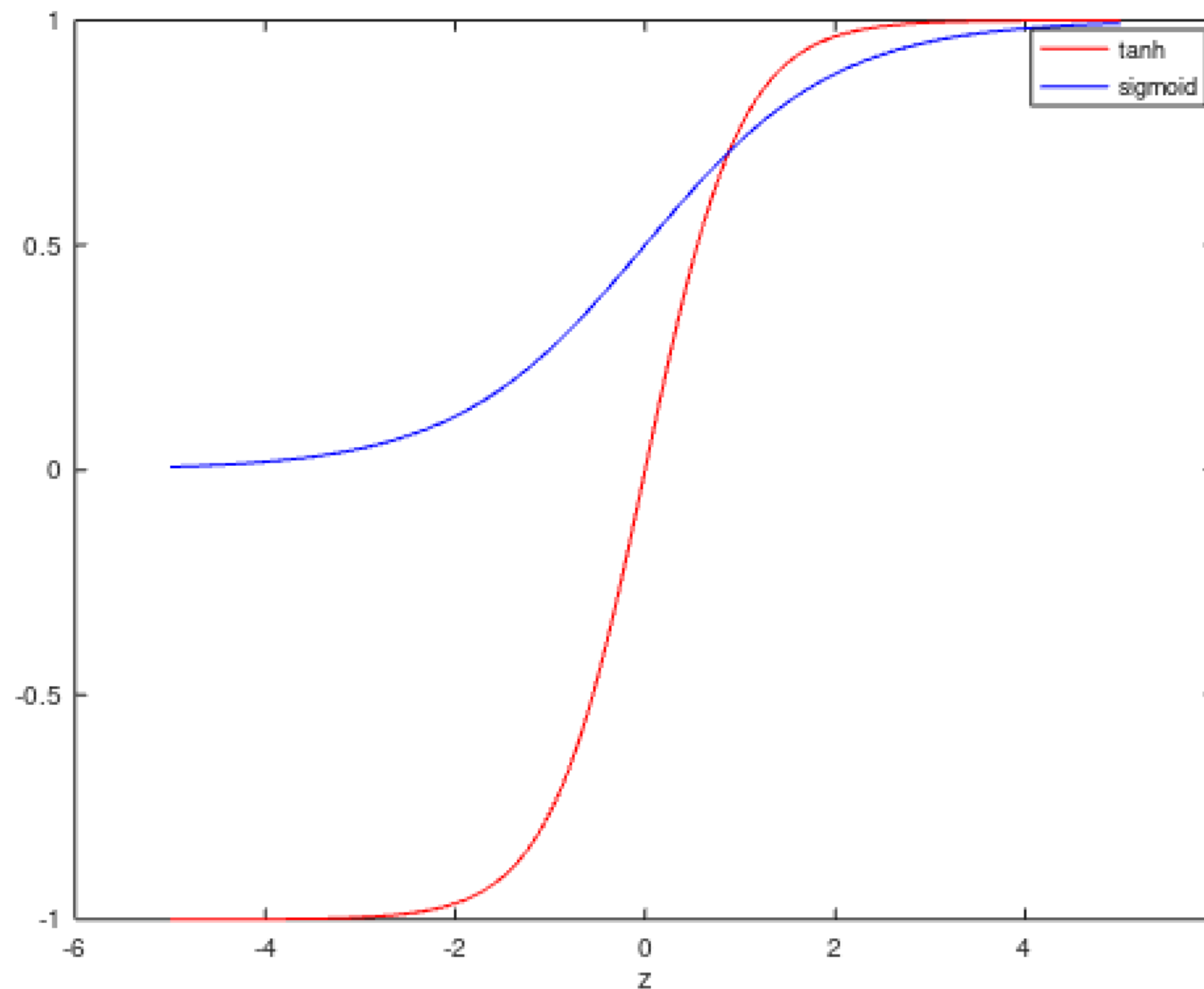


Tanh Activation

Map inputs into $(-1, 1)$

$$\sigma(z) = \tanh(z) = \frac{1 - \exp(-2z)}{1 + \exp(-2z)}$$

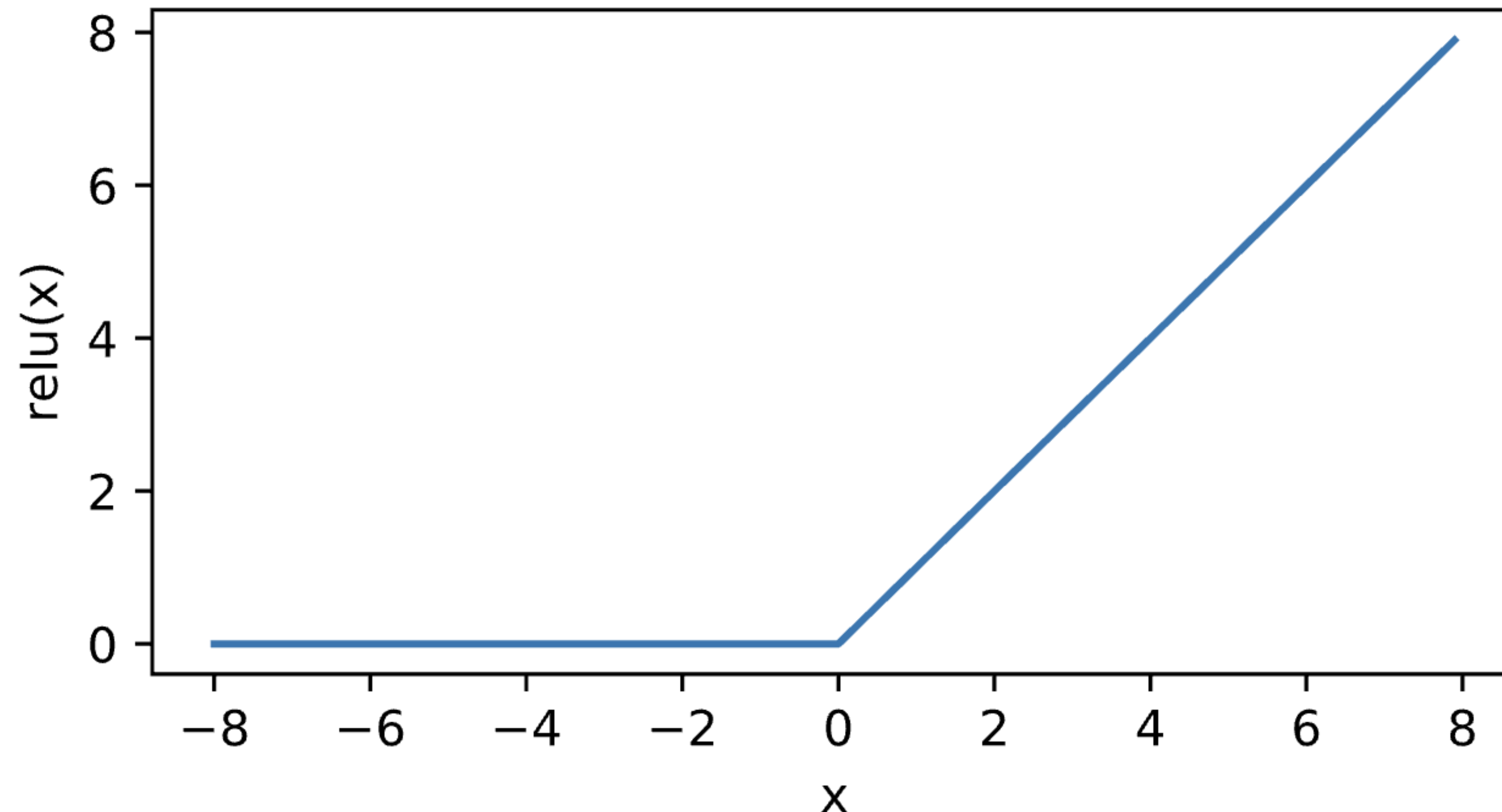
$$\tanh(z) = 2\text{sigmoid}(2z) - 1$$



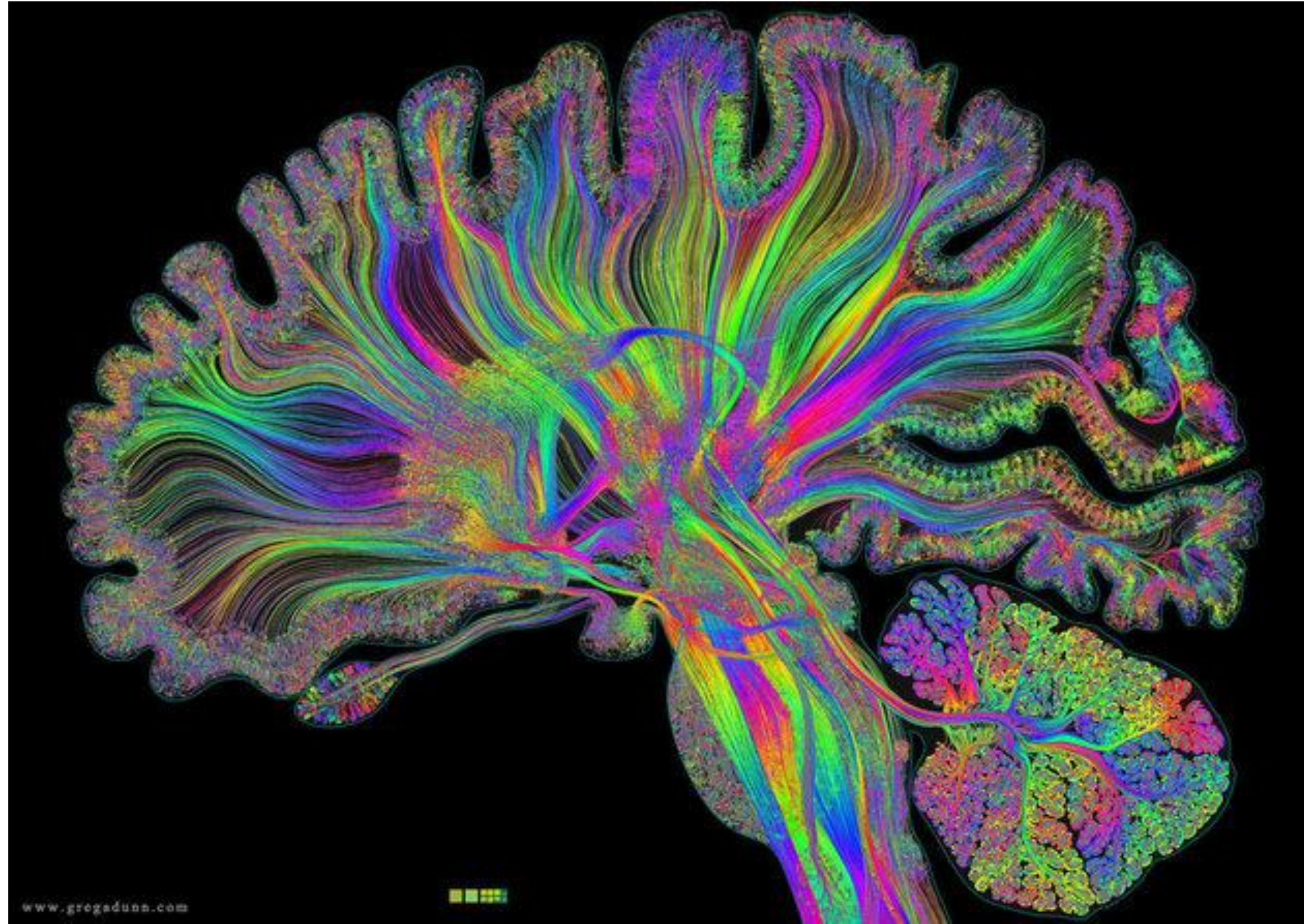
ReLU Activation

ReLU: rectified linear unit (commonly used in modern neural networks)

$$\text{ReLU}(x) = \max(x, 0)$$



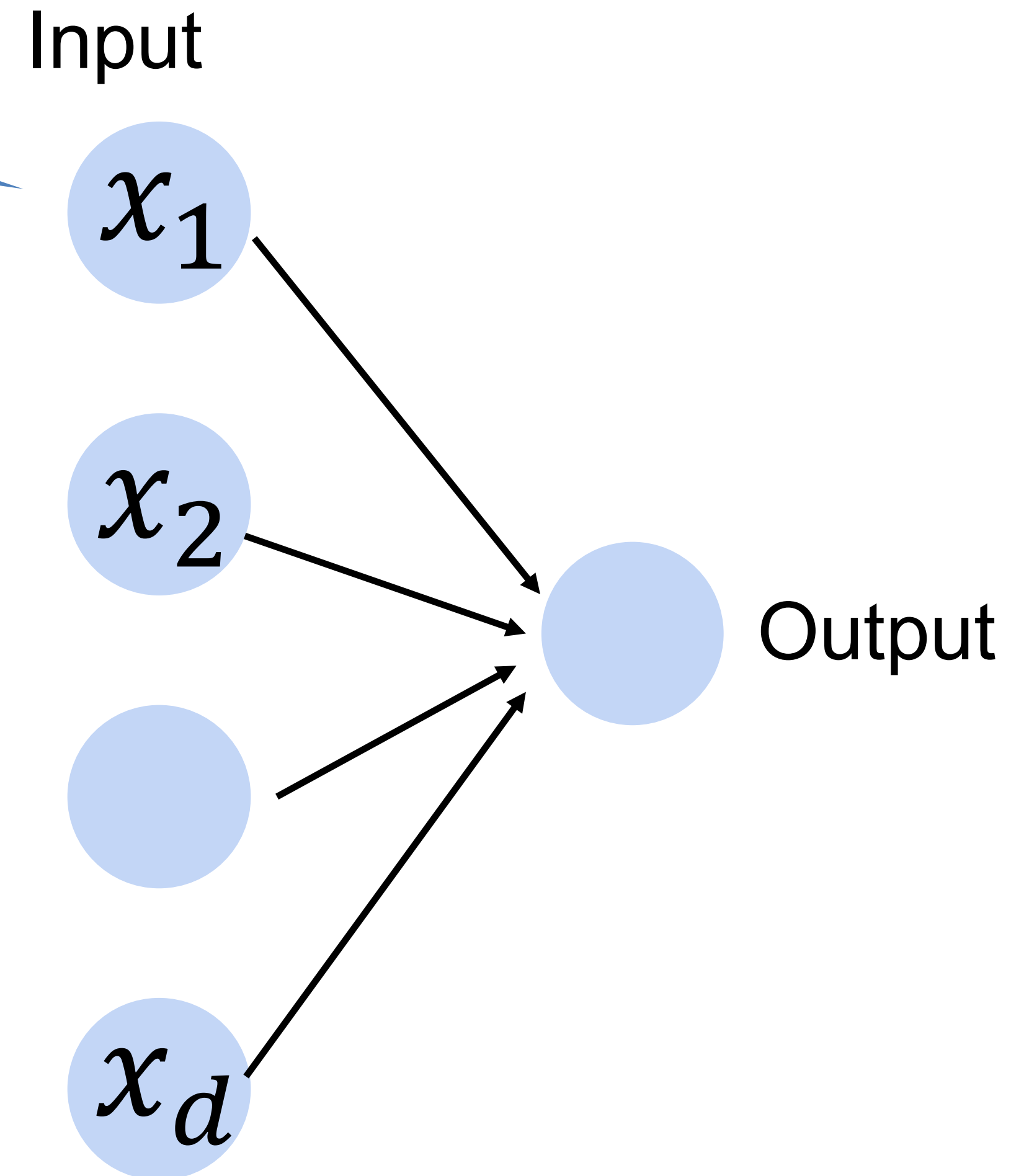
Multilayer Perceptron



The perceptron has one layer of weights

Each input node receives one scalar feature (e.g., one pixel)

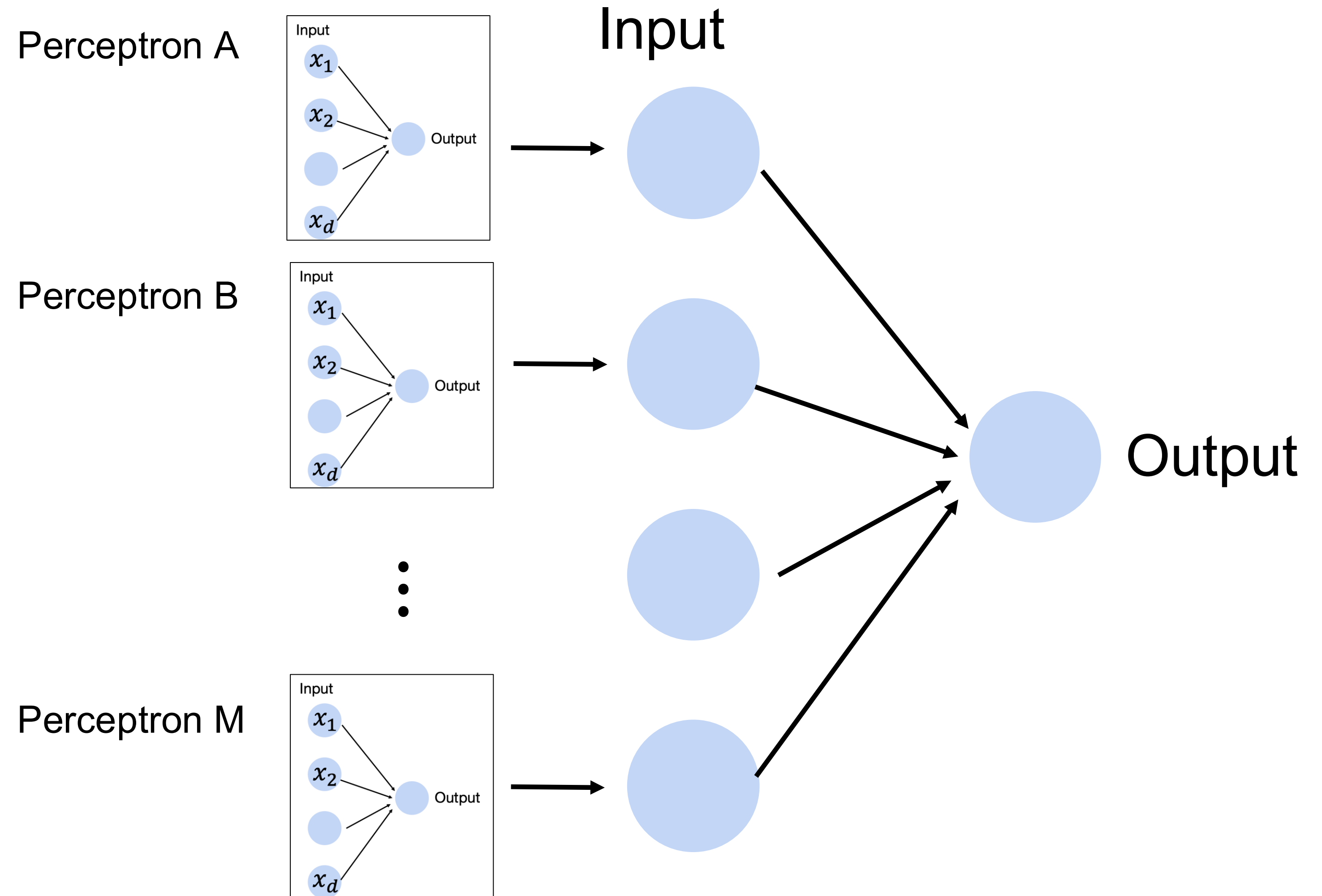
Cats vs. dogs?



Beyond one layer

Big Idea: take our inputs from other perceptrons!

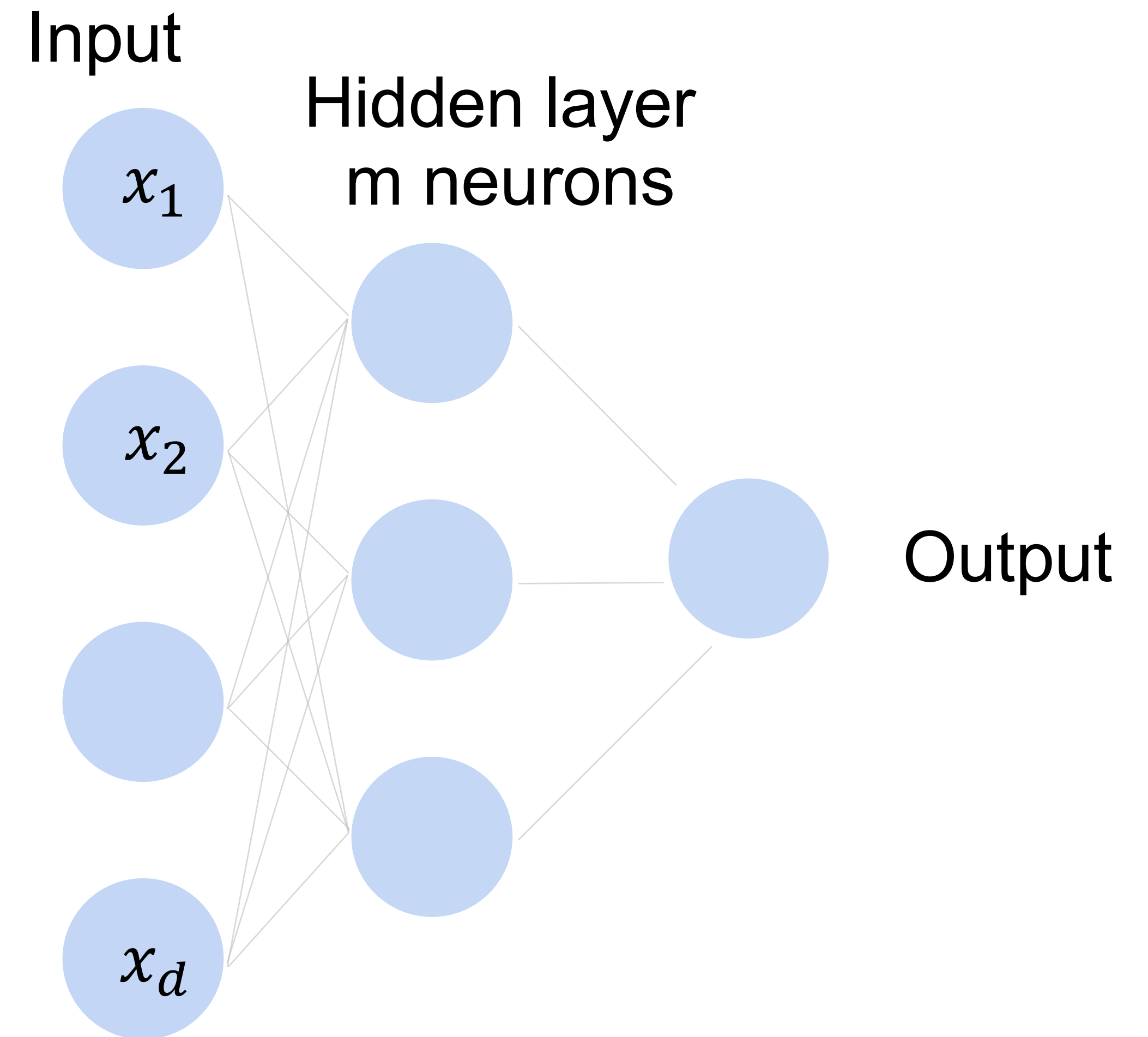
Cats vs. dogs?



Multilayer Perceptron with One "Hidden" Layer

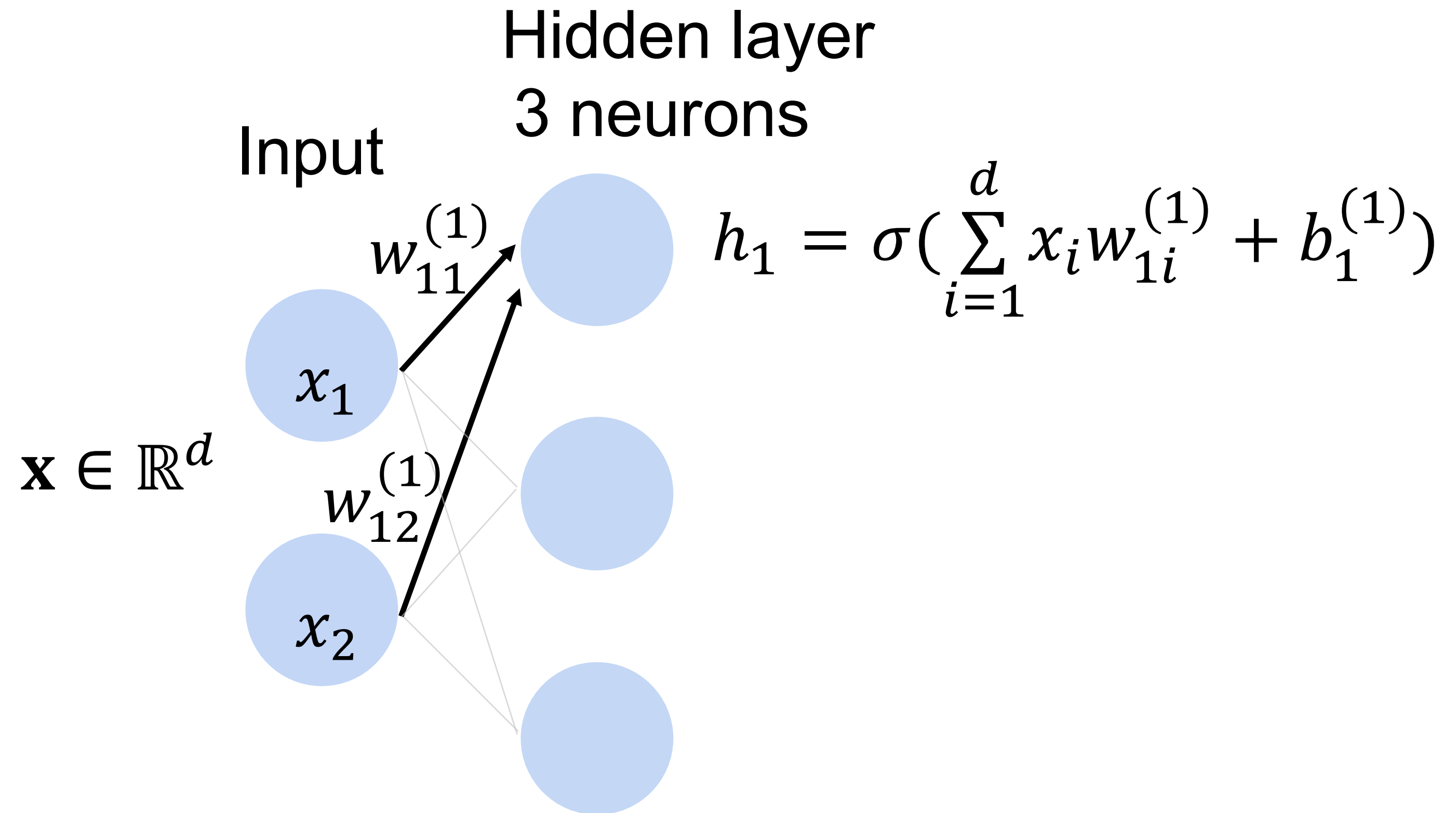
How to classify

Cats vs. dogs?



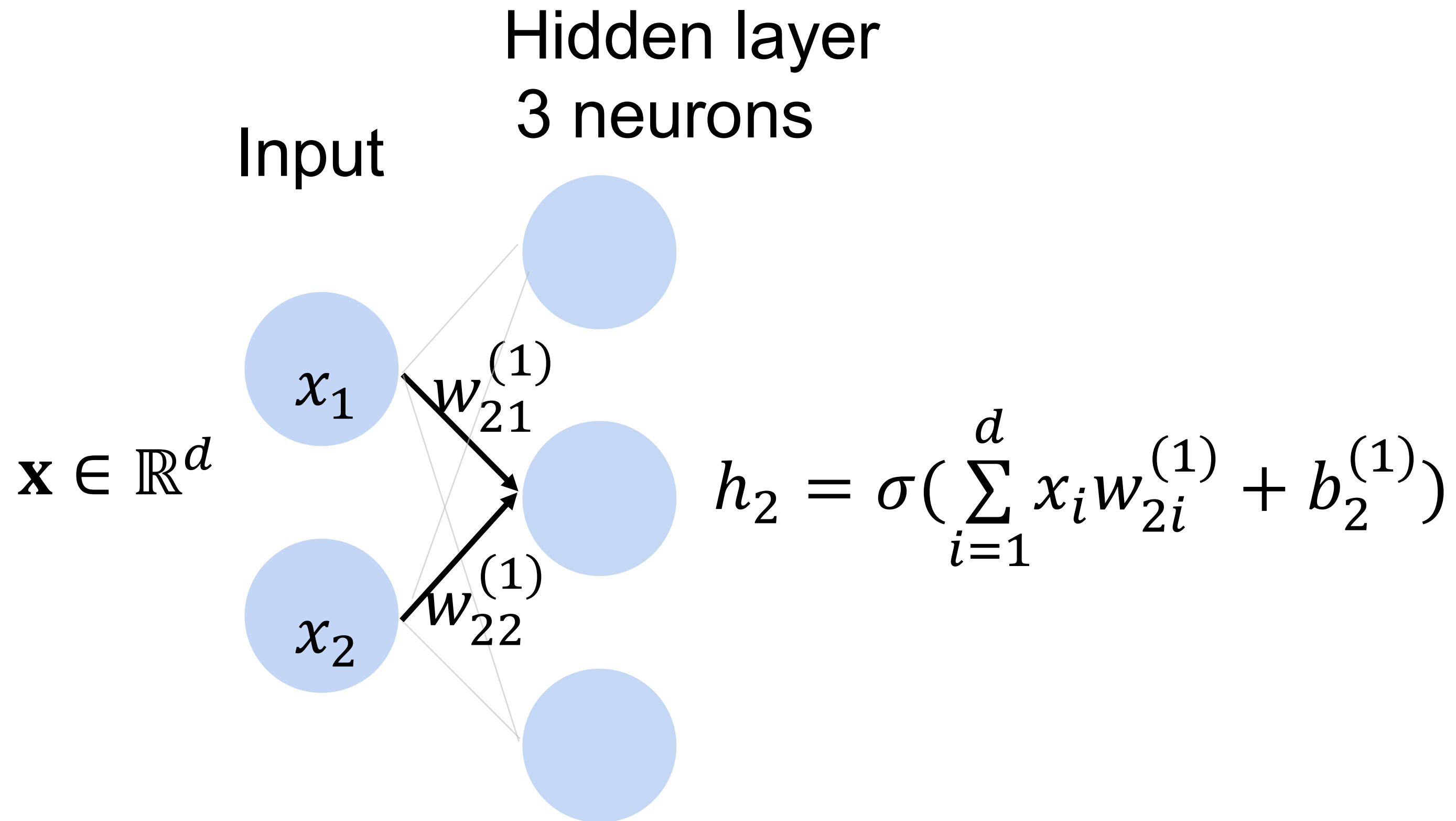
Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



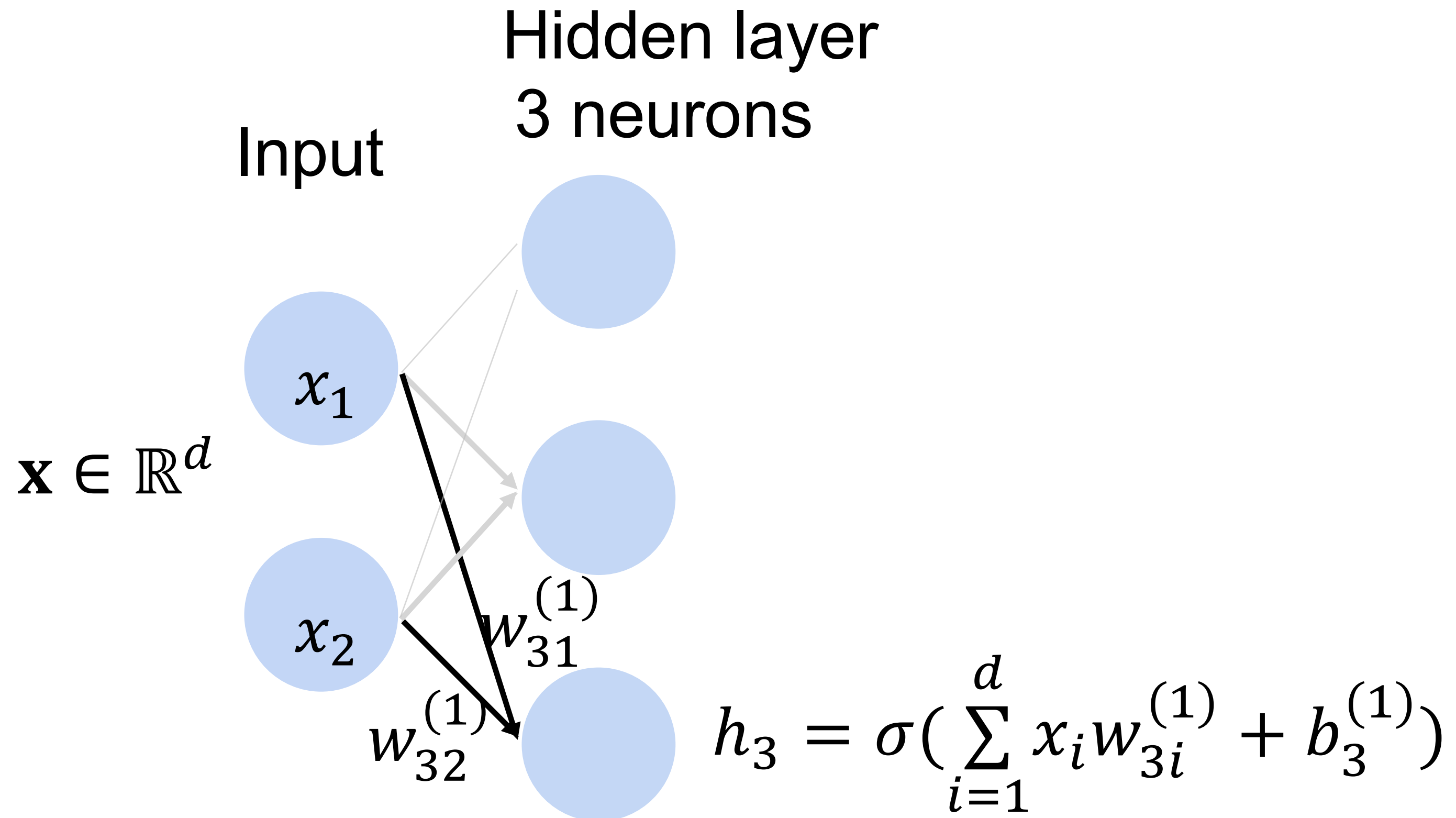
Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



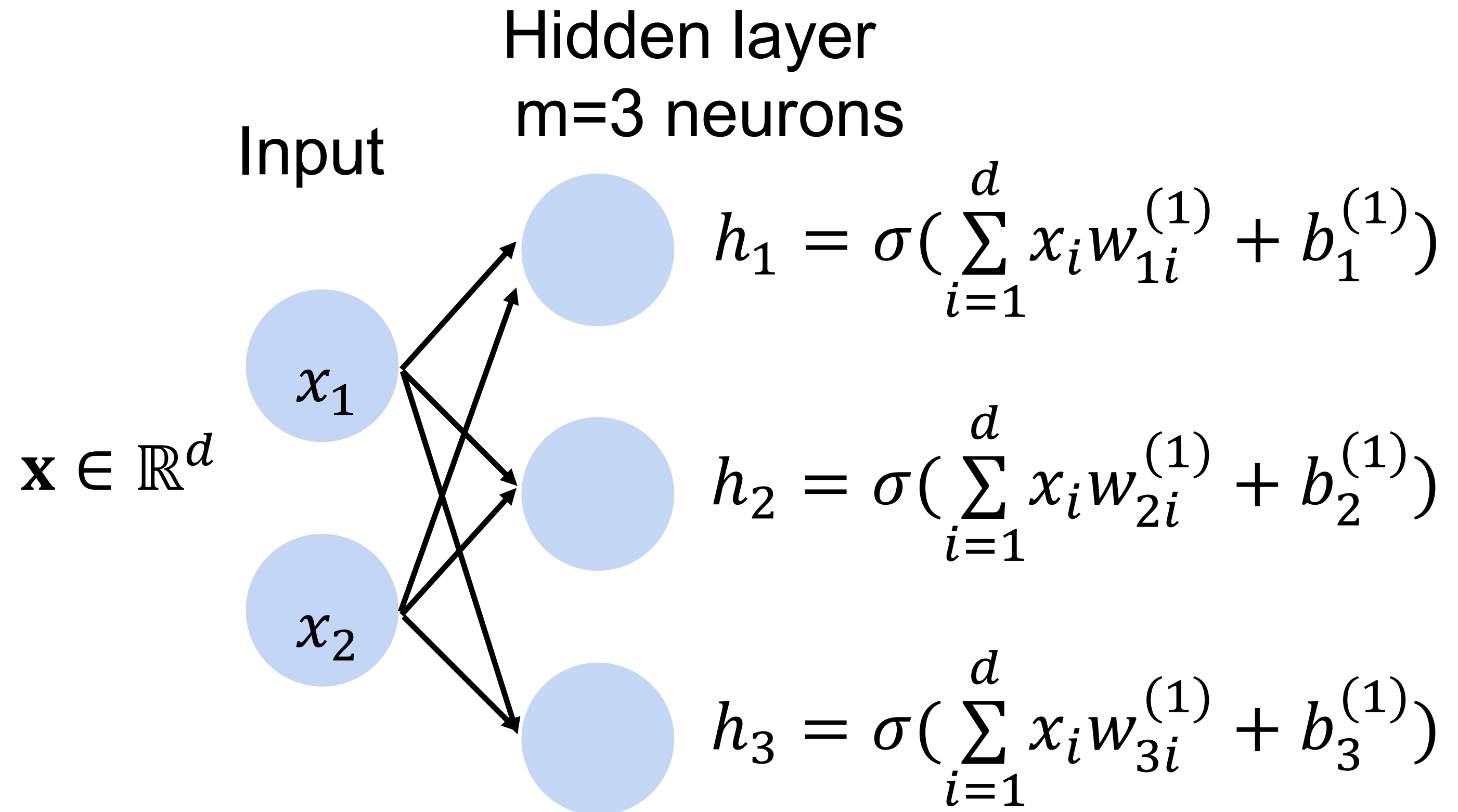
Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



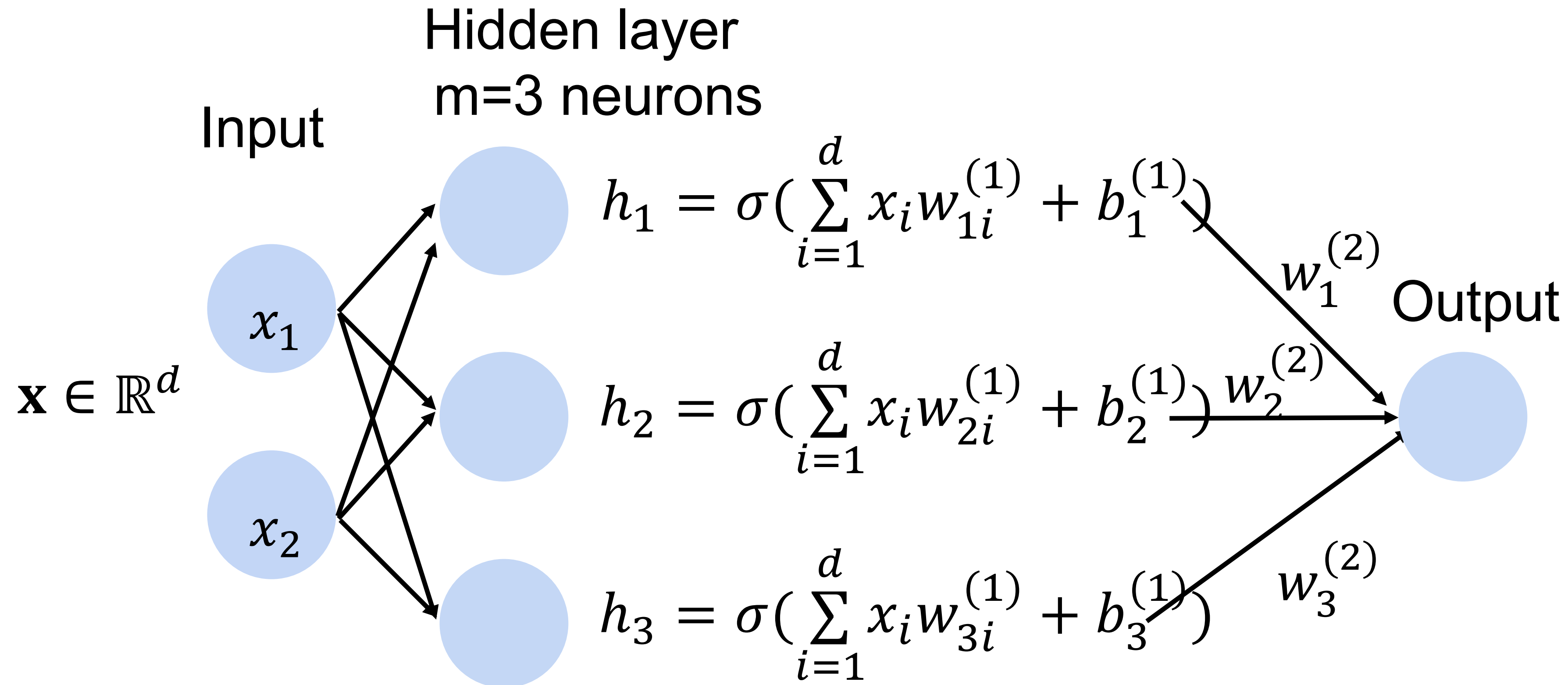
Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



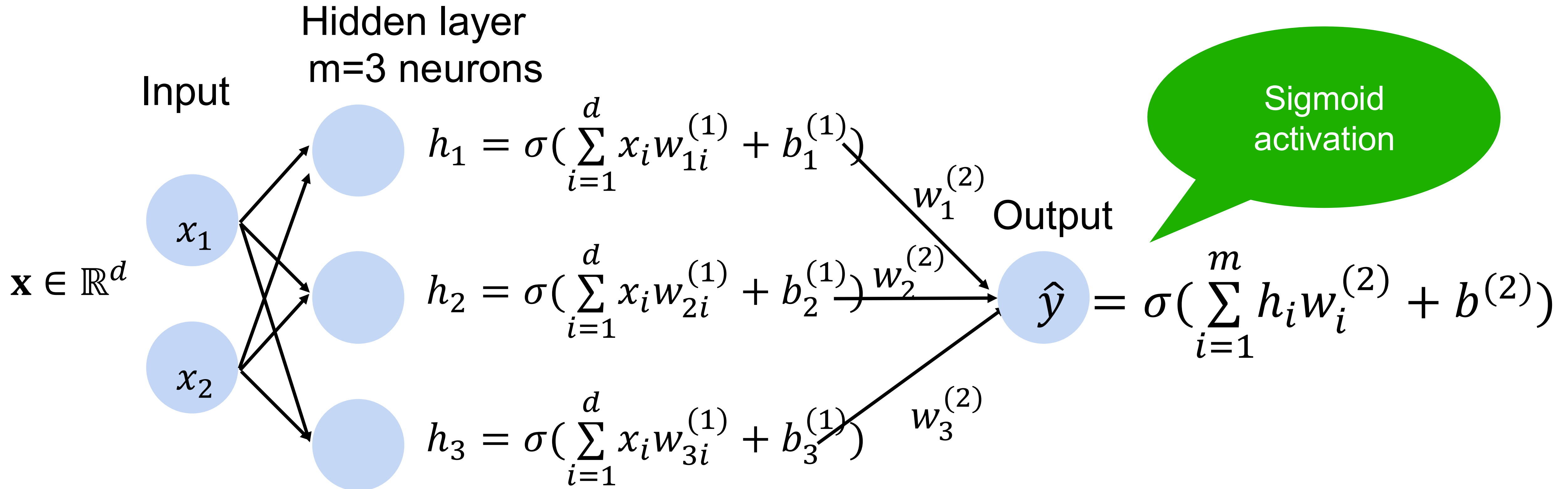
Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2

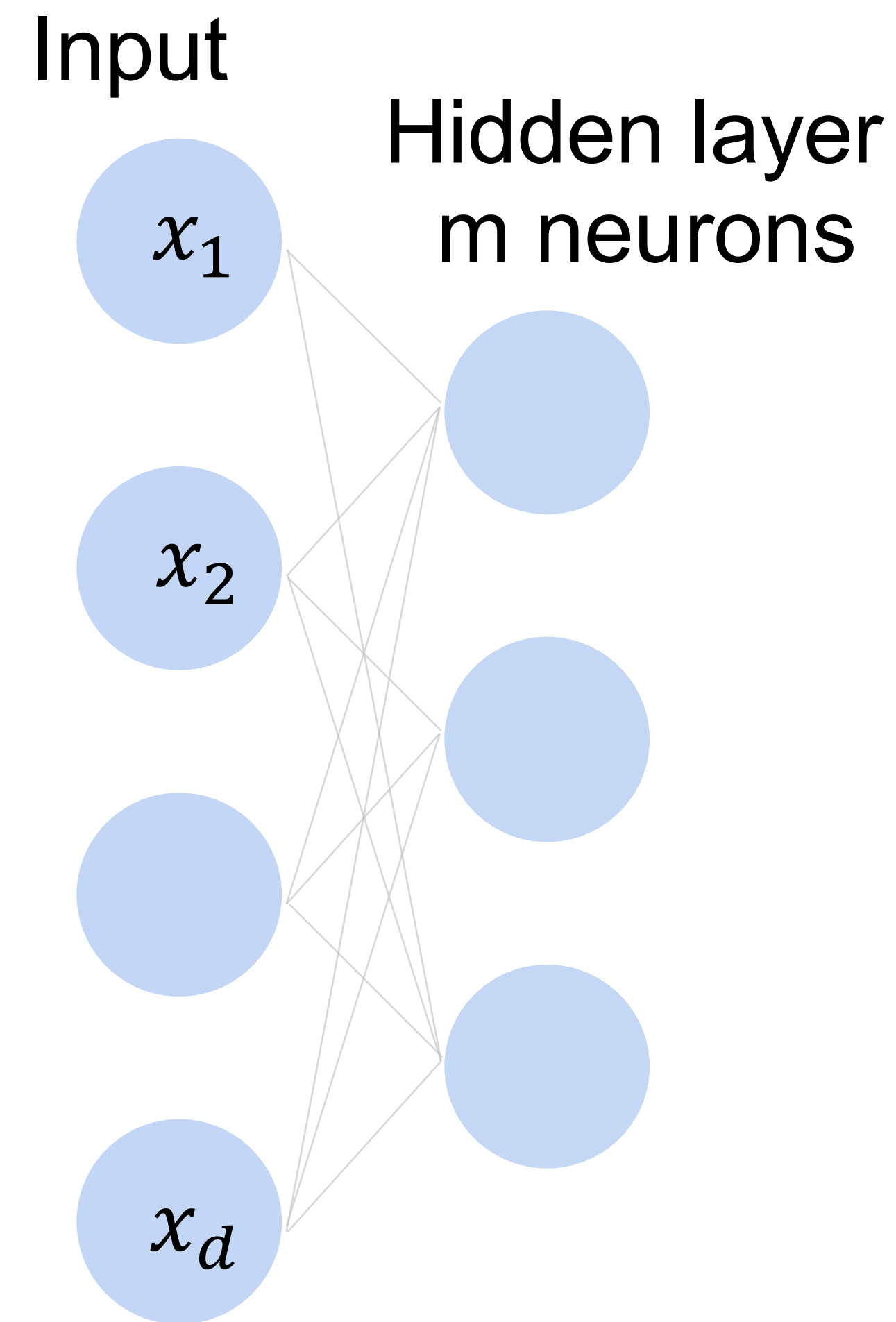


Multi-layer perceptron: Matrix Notation

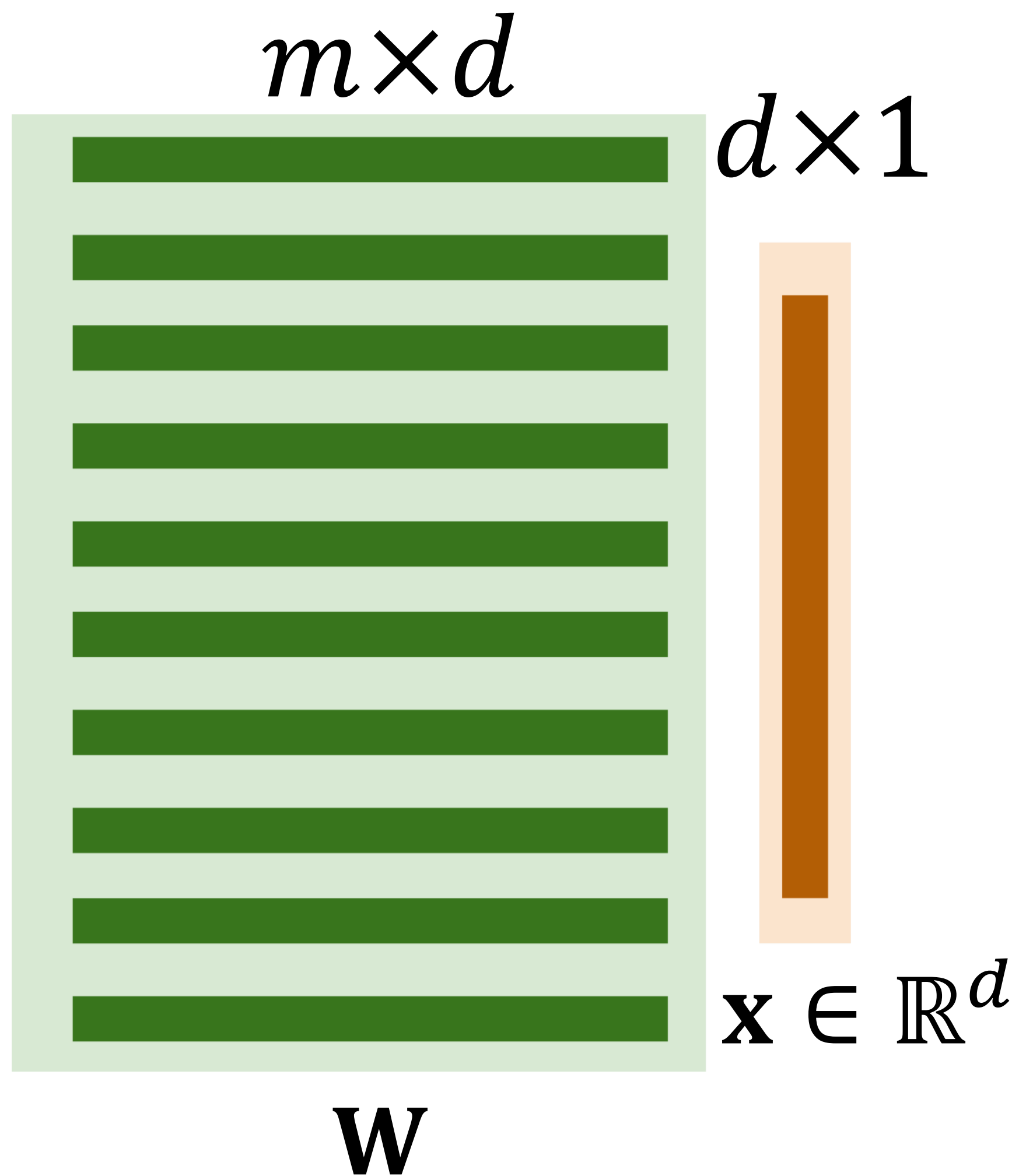
- Input: $\mathbf{x} \in \mathbb{R}^d$
- Hidden $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}$, $\mathbf{b}^{(1)} \in \mathbb{R}^m$
- Intermediate output

$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h} \in \mathbb{R}^m$$

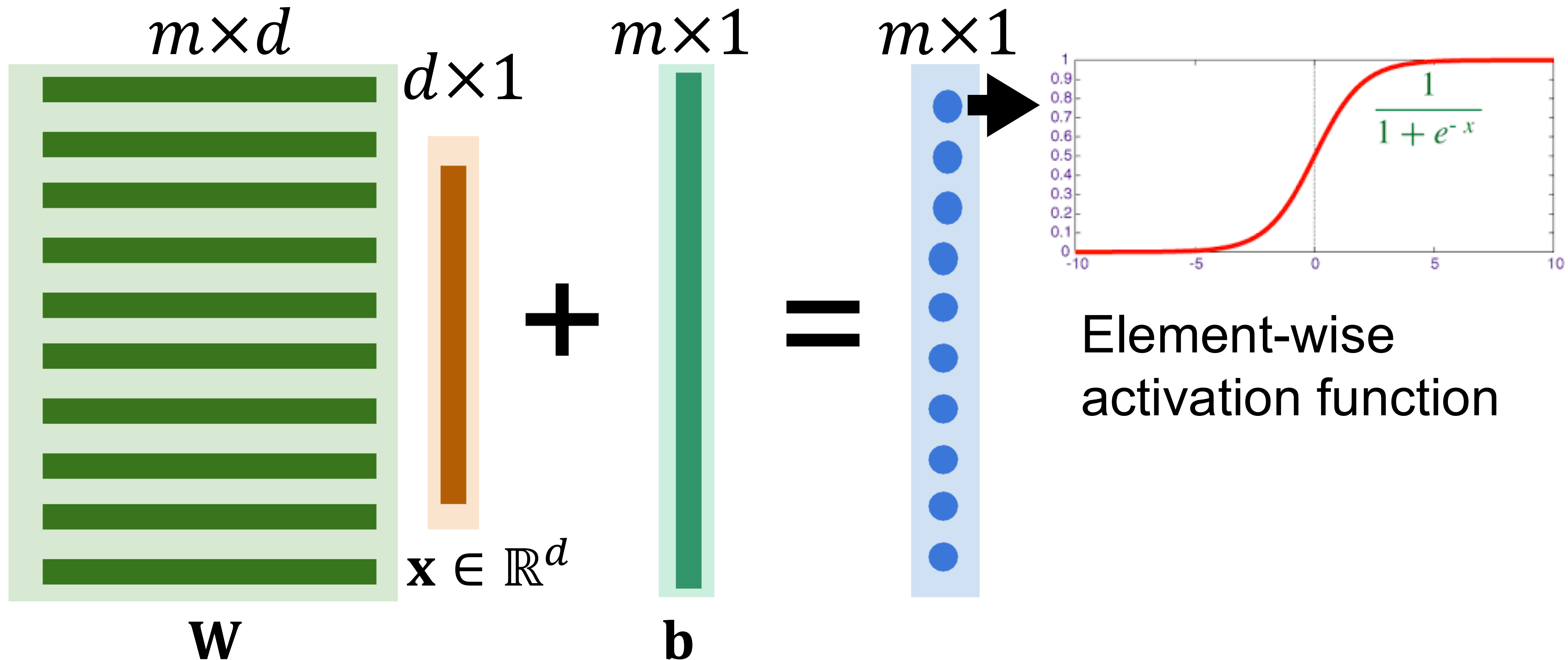


Multi-layer perceptron: Matrix Notation



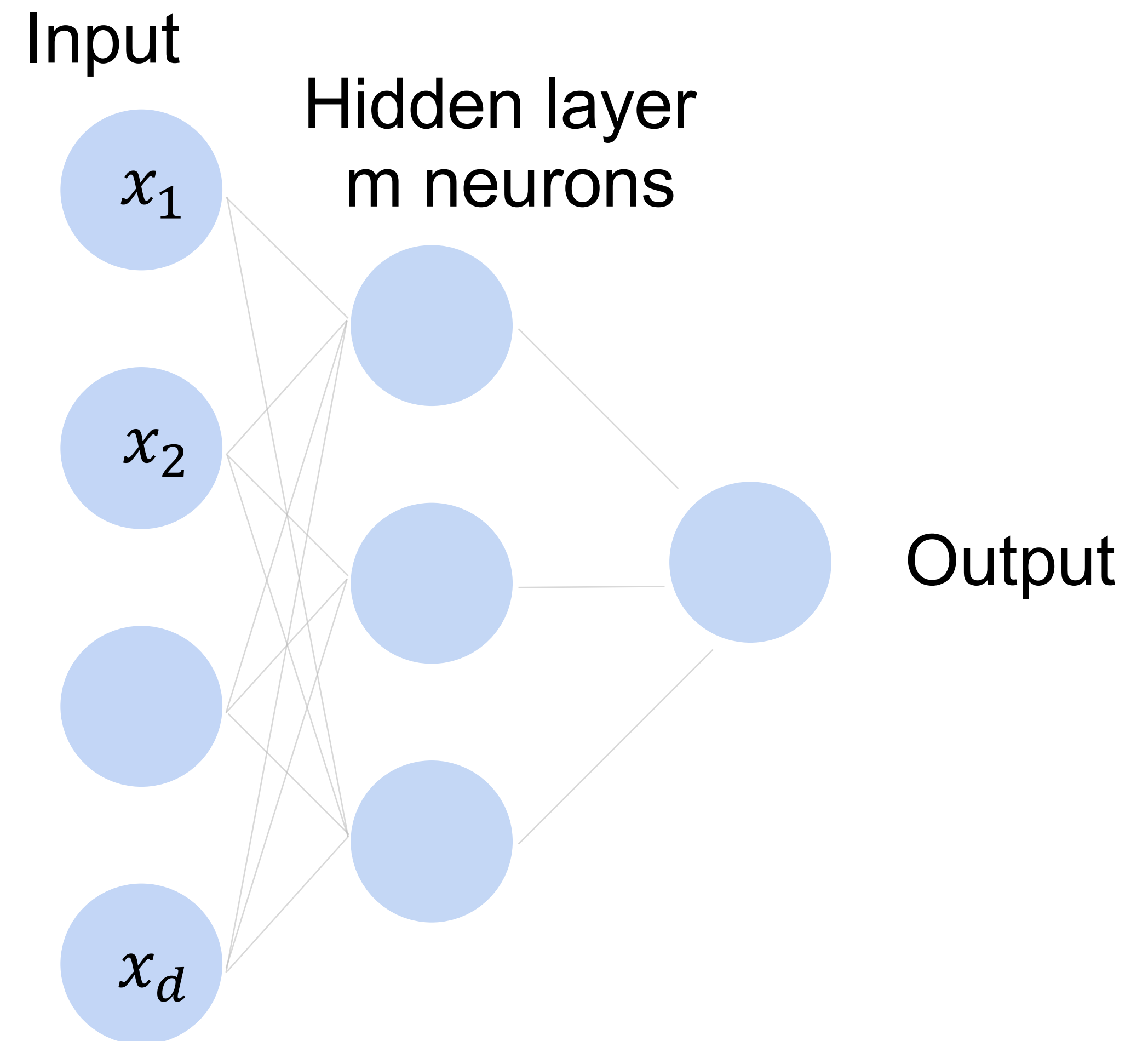
Multi-layer perceptron: Matrix Notation

Key elements: linear operations + Nonlinear activations



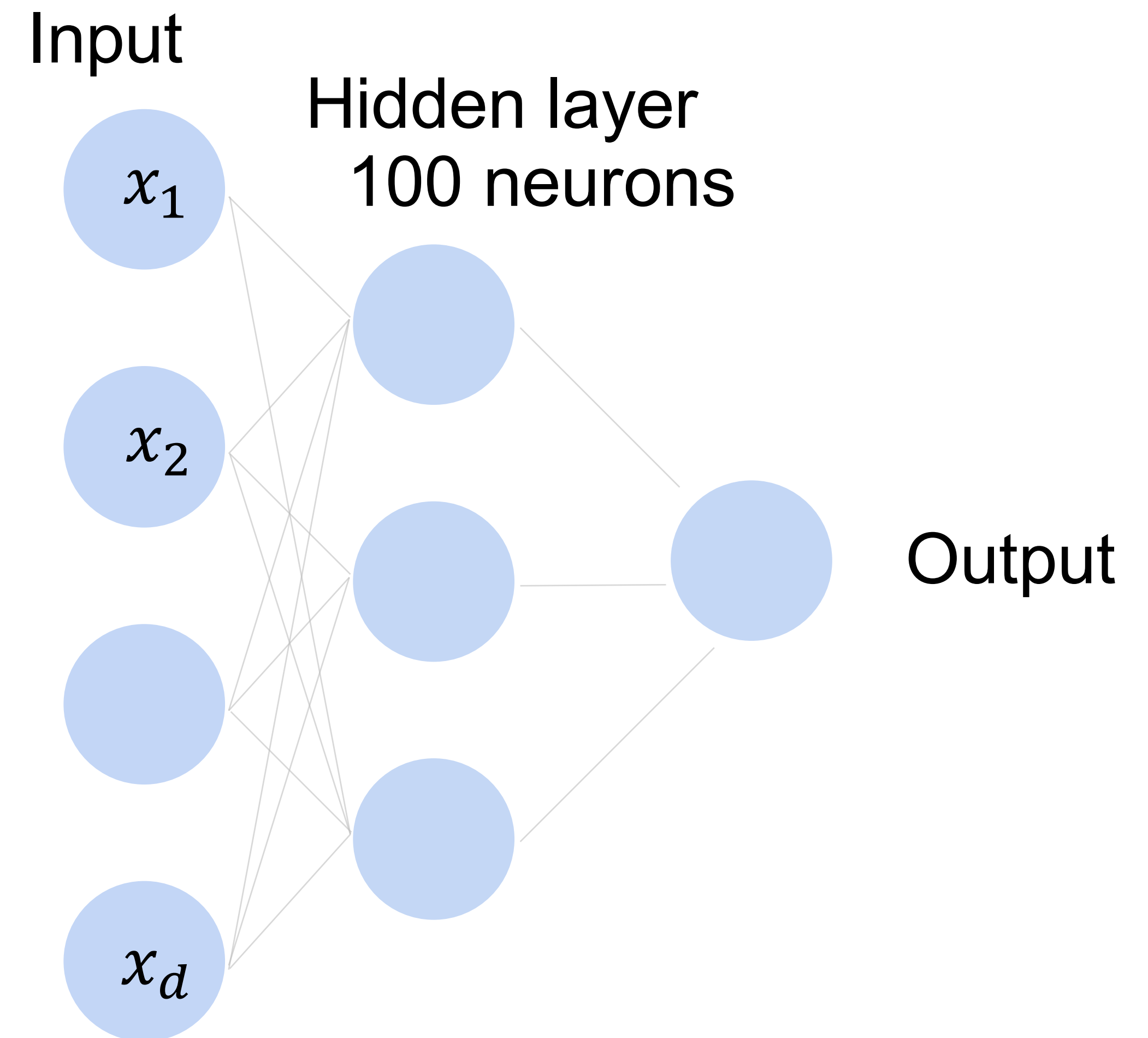
Multi-layer perceptron: Matrix Notation

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Hidden $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}$, $\mathbf{b}^{(1)} \in \mathbb{R}^m$
- Intermediate output
$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$
- Final output
$$\hat{y} = \sigma(\mathbf{w}^{(2)}\mathbf{h} + \mathbf{b}^{(2)})$$

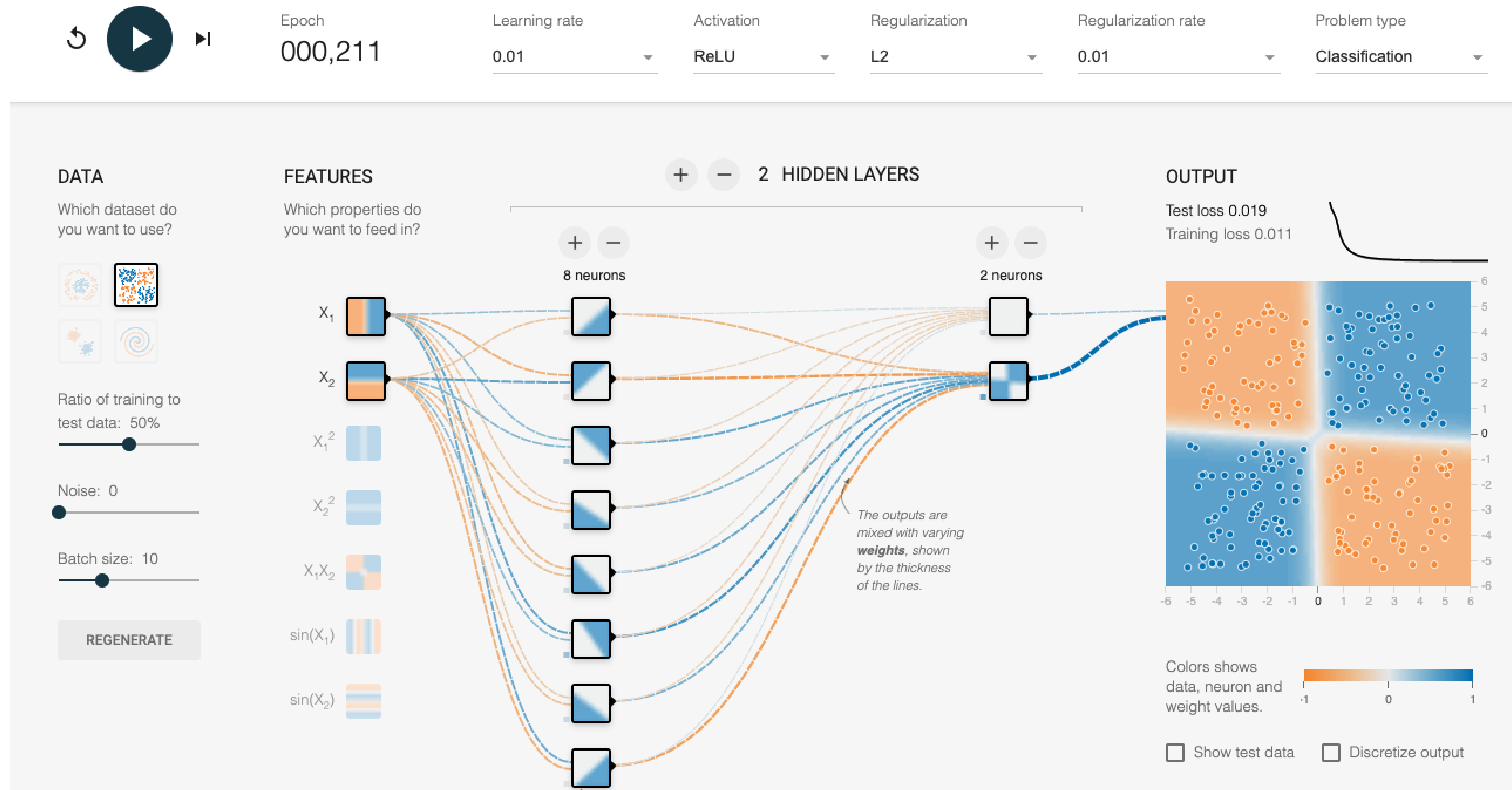


Multi-layer perceptron: Recap

Classify cats vs. dogs

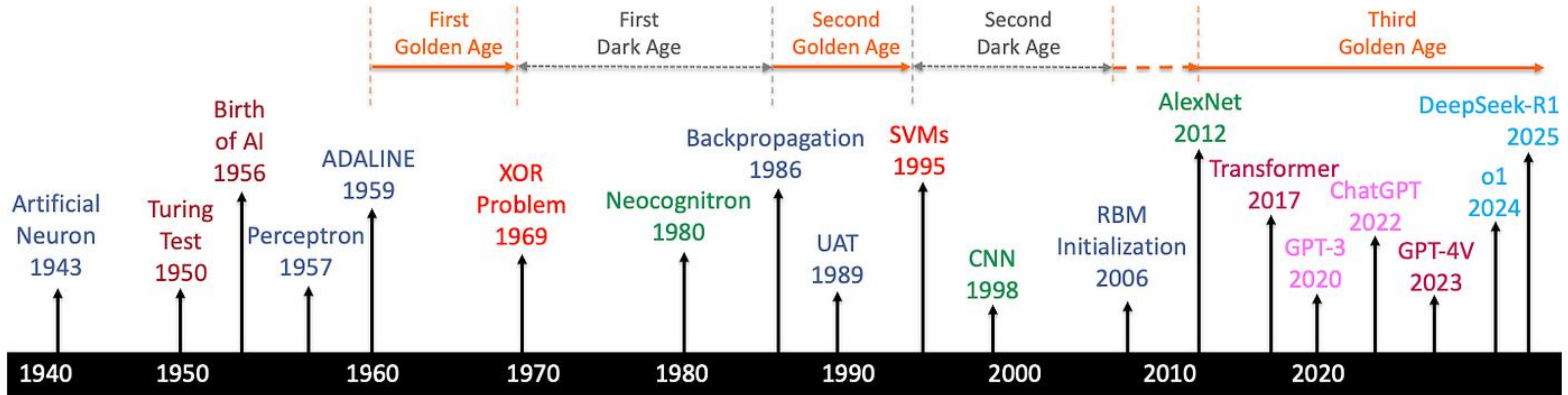


Demo: Learning XOR using neural net



<https://playground.tensorflow.org/>

A Brief History of AI with Deep Learning



McCulloch-Pitts

Rosenblatt

Widrow-Hoff

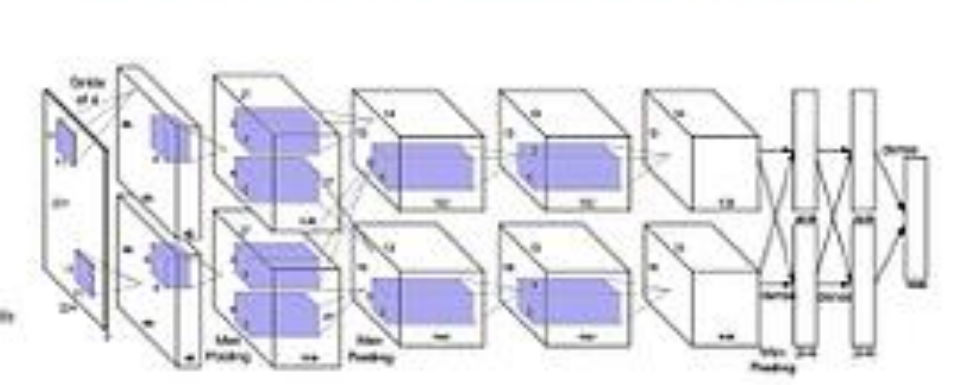
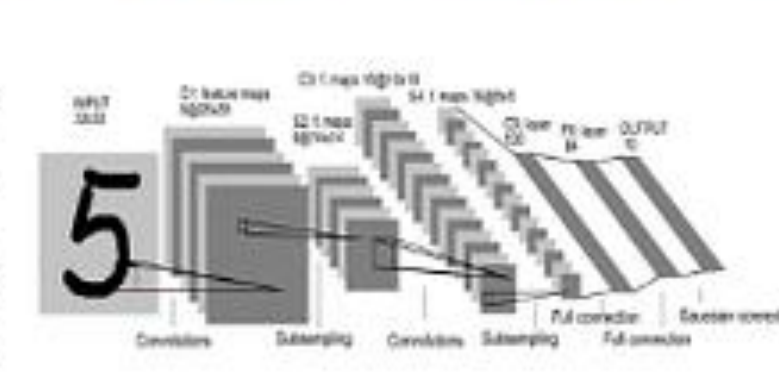
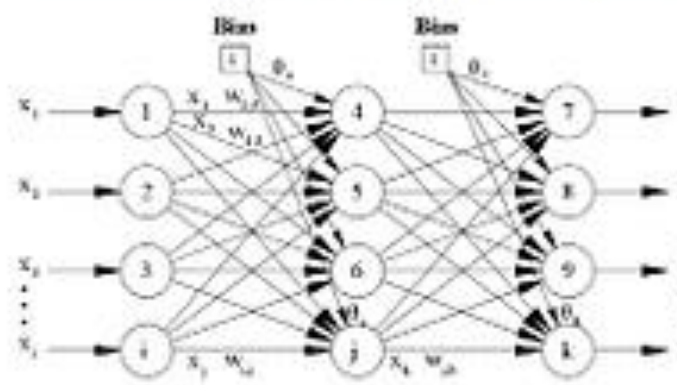
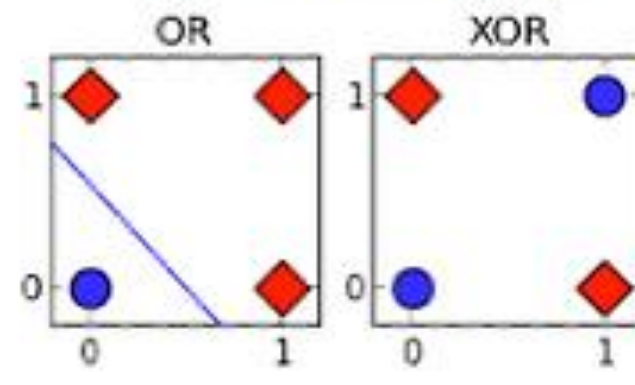
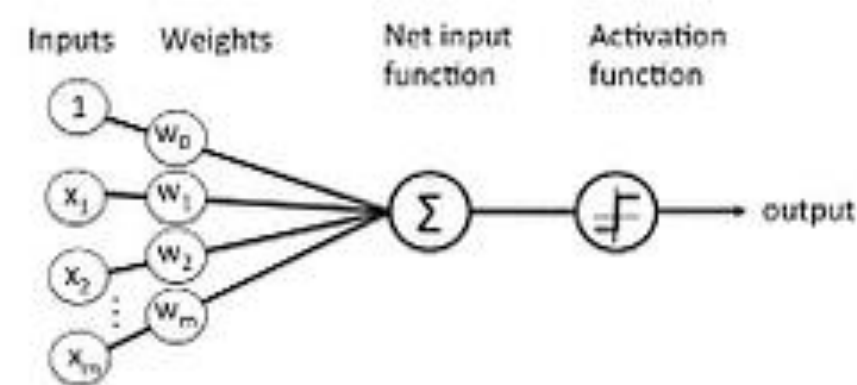
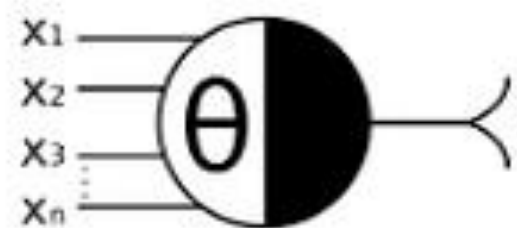
Minsky-Papert

Rumelhart, Hinton et al.

LeCun

Hinton-Ruslan Krizhevsky et al.

Vaswani et al.



What we've learned today...

- Single-layer Perceptron
 - Motivation
 - Activation function
 - Representing AND, OR, NOT
- Multilayer Perceptron
- Brief history of neural networks

Suggested Readings

- Textbook: Artificial Intelligence: A Modern Approach (4th edition). Stuart Russell and Peter Norvig. Pearson, 2020.
 - Sections 19.6.4, 21.1