Introduction to Machine Learning [Fall 2022]

Perceptron

October 20, 2022

Lerrel Pinto

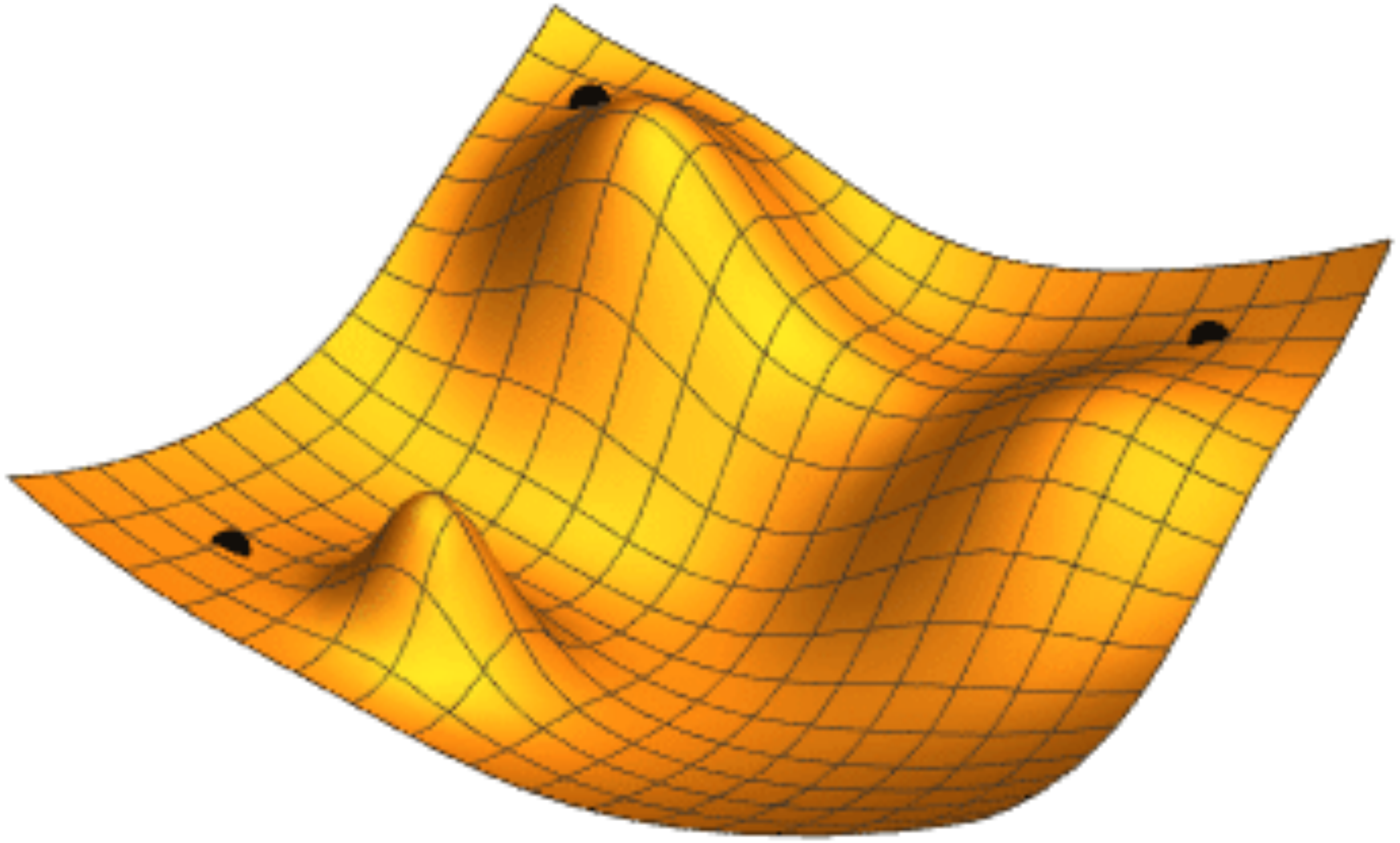
Topics for today

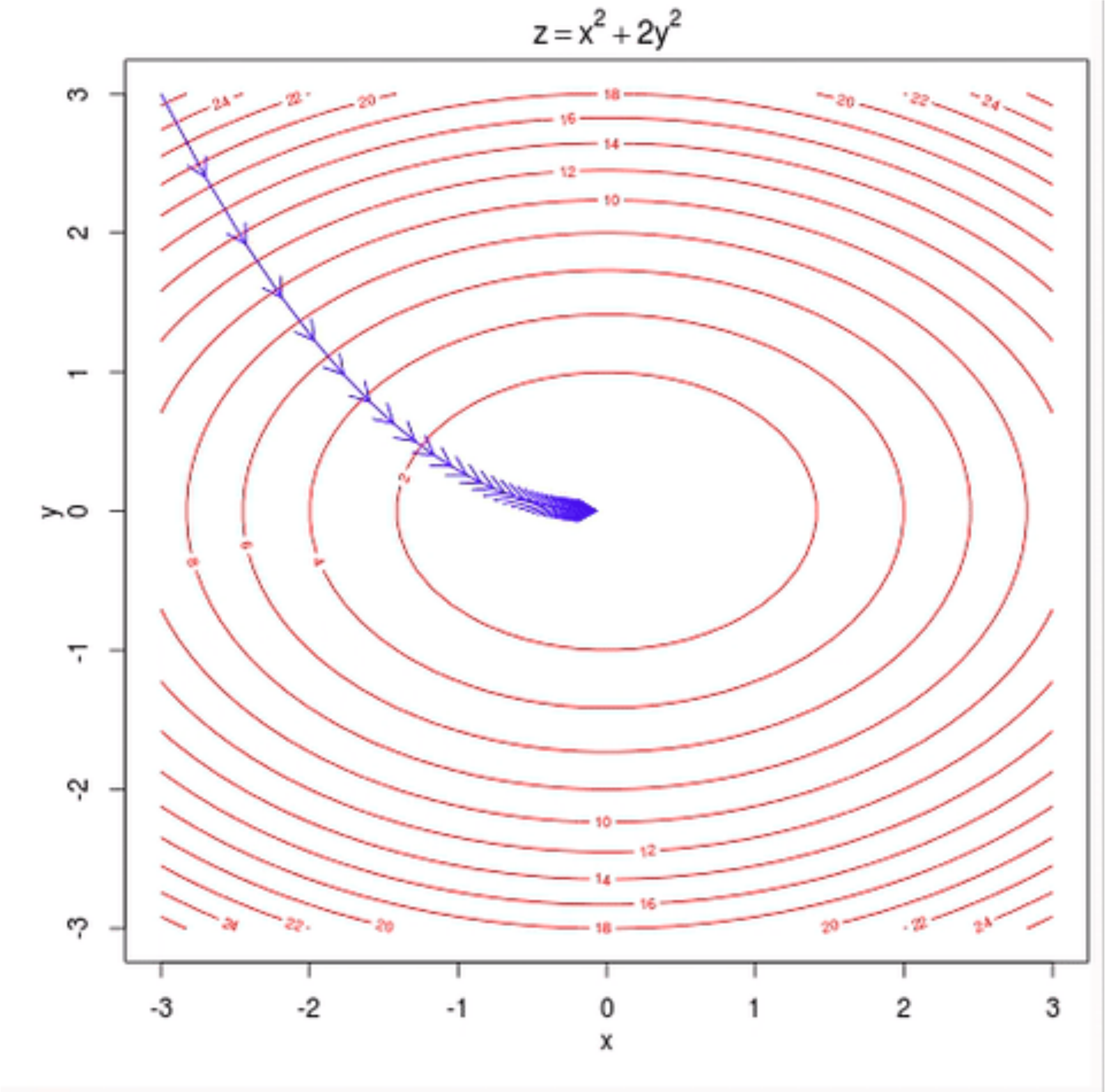
- A new view of doing Machine Learning
- Introducing the foundations of deep learning

Recap: Gradient Descent

- Given: cost / loss/ objective function $f(\vec{\theta}, D)$. Where $\vec{\theta} \in \mathbb{R}^d$.
- Goal: find $\vec{\theta}*$ such that $f(\vec{\theta}*, D) = \min_{\vec{\theta}} f(\vec{\theta}, D)$.
- Gradient descent solution:
    - Start from initial guess $\vec{\theta}^0$ and learning rate α
    - Update $\vec{\theta}^{i+1} \leftarrow \vec{\theta}^i - \alpha \nabla f(\vec{\theta}, D)$
    - Repeat until change in θ is small, or maximum number of steps reached.

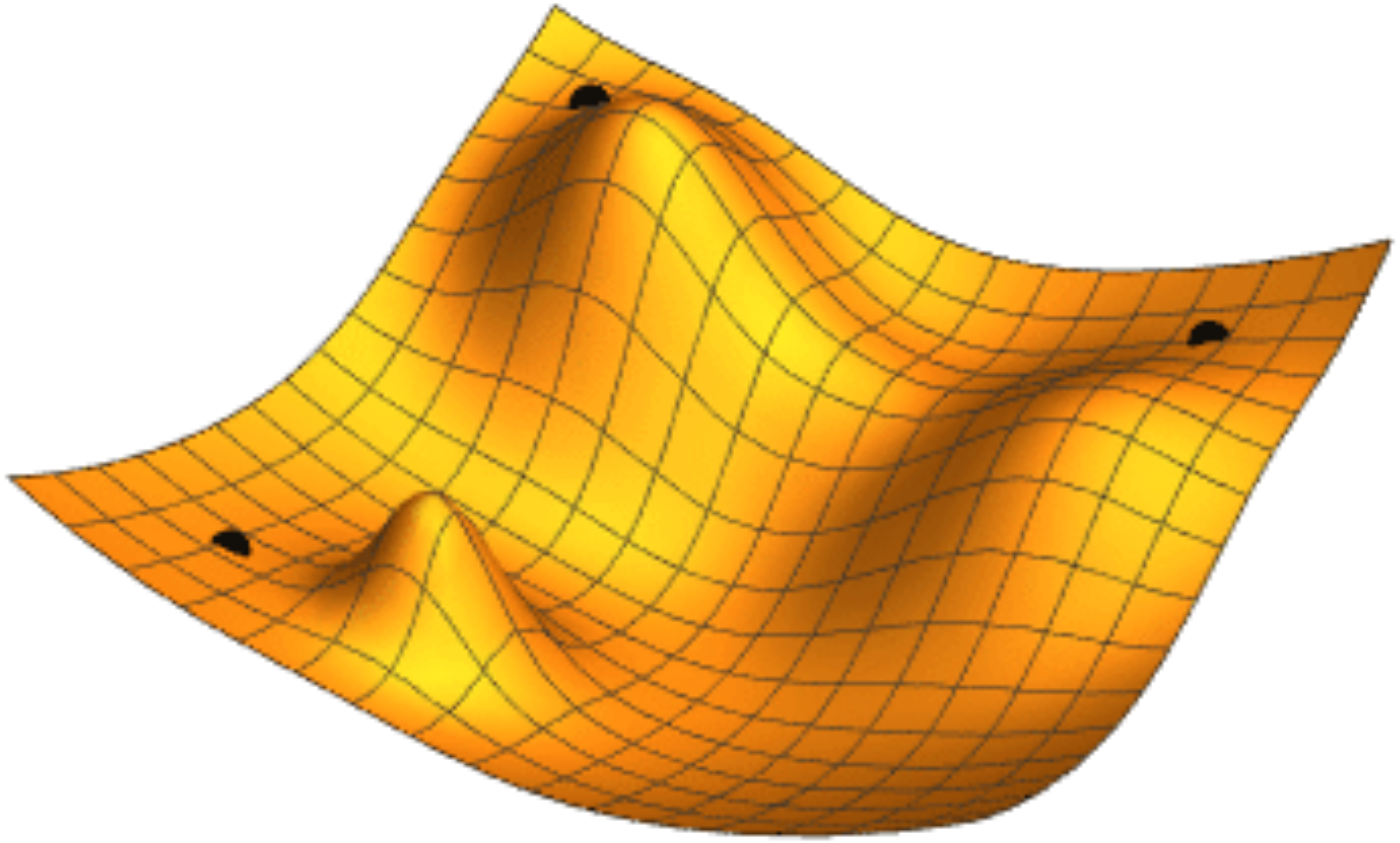
Recap: Gradient Descent

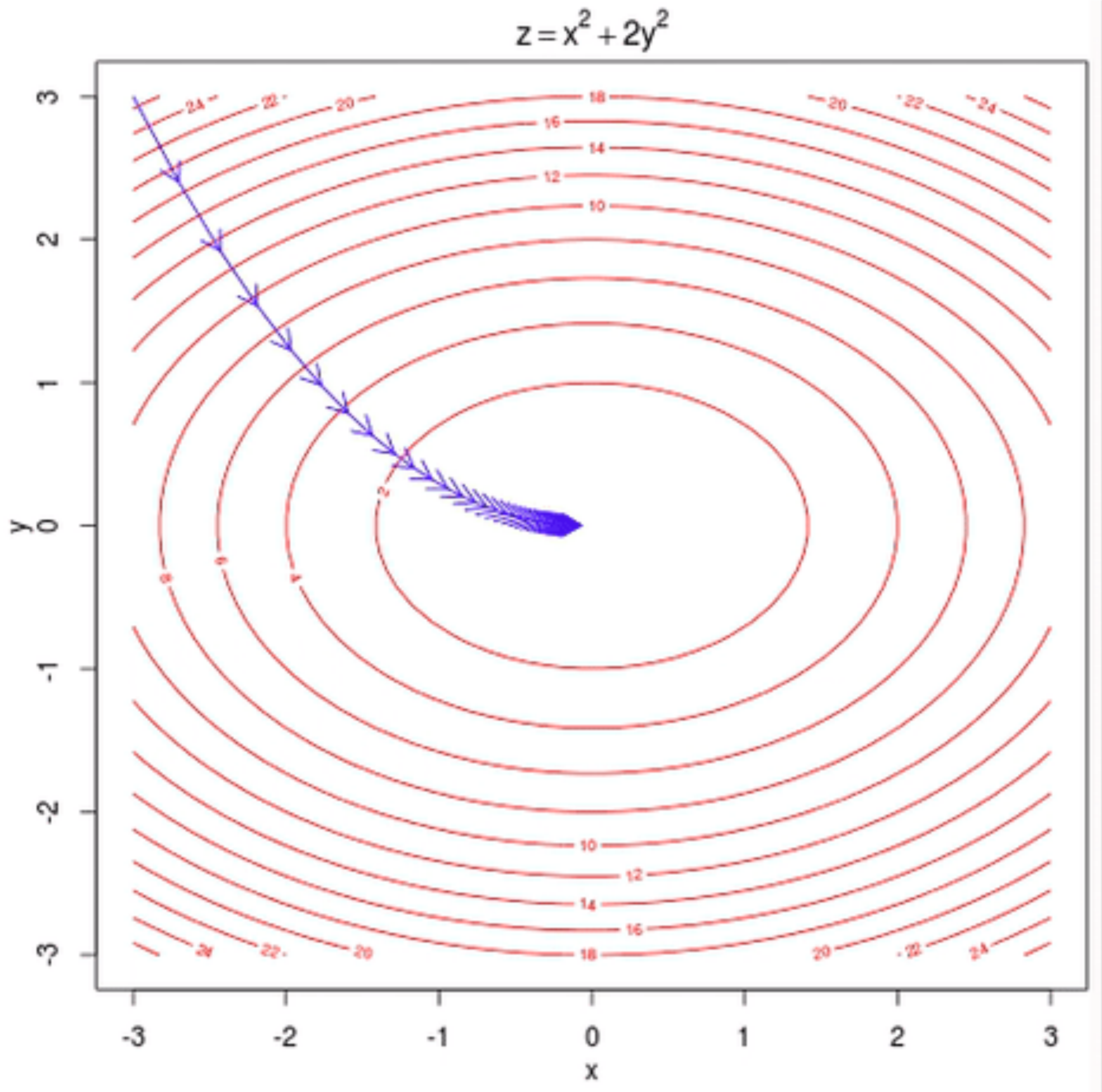




Credits: Wikimedia, Hoang Duong

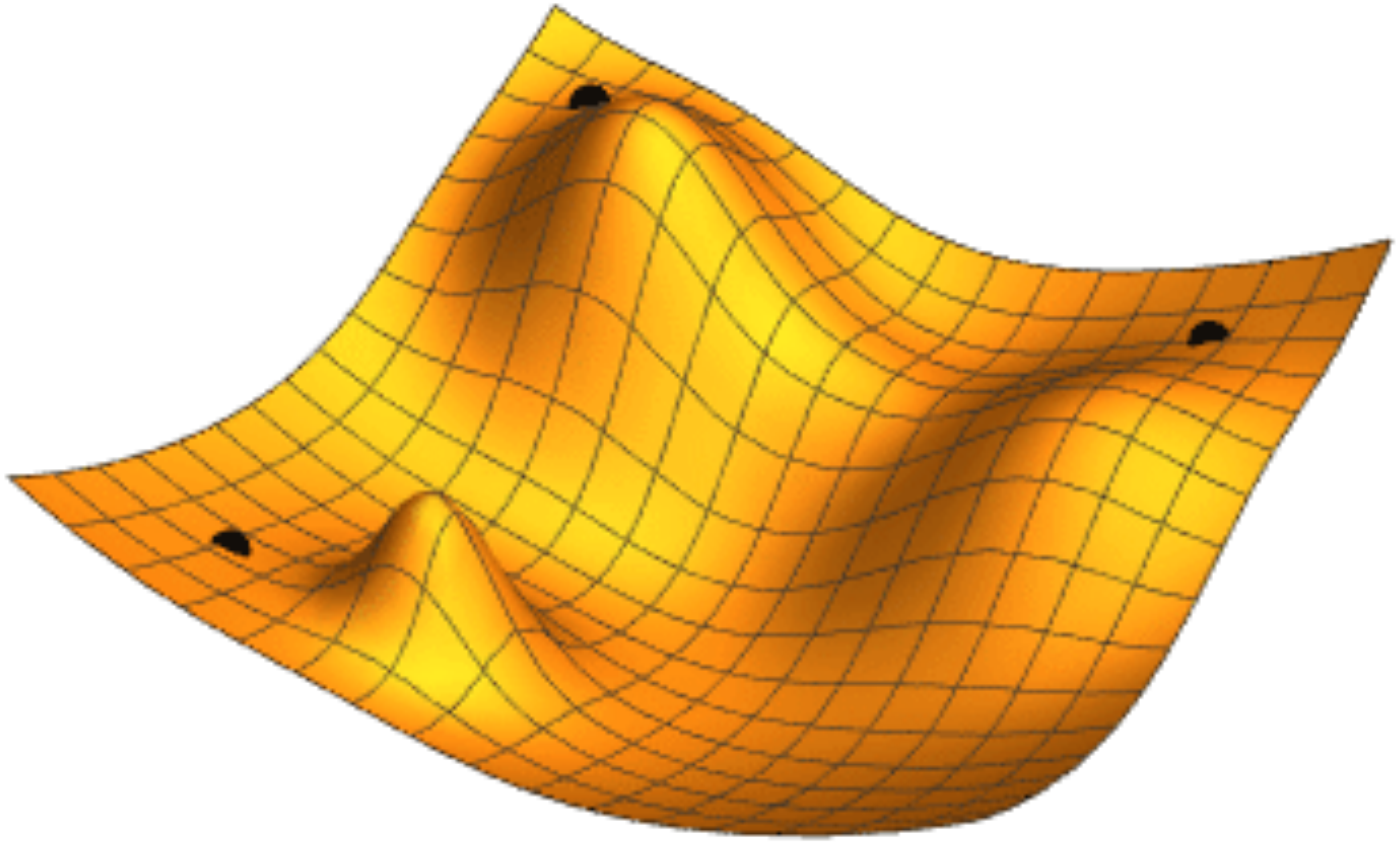
Recap: Gradient Descent

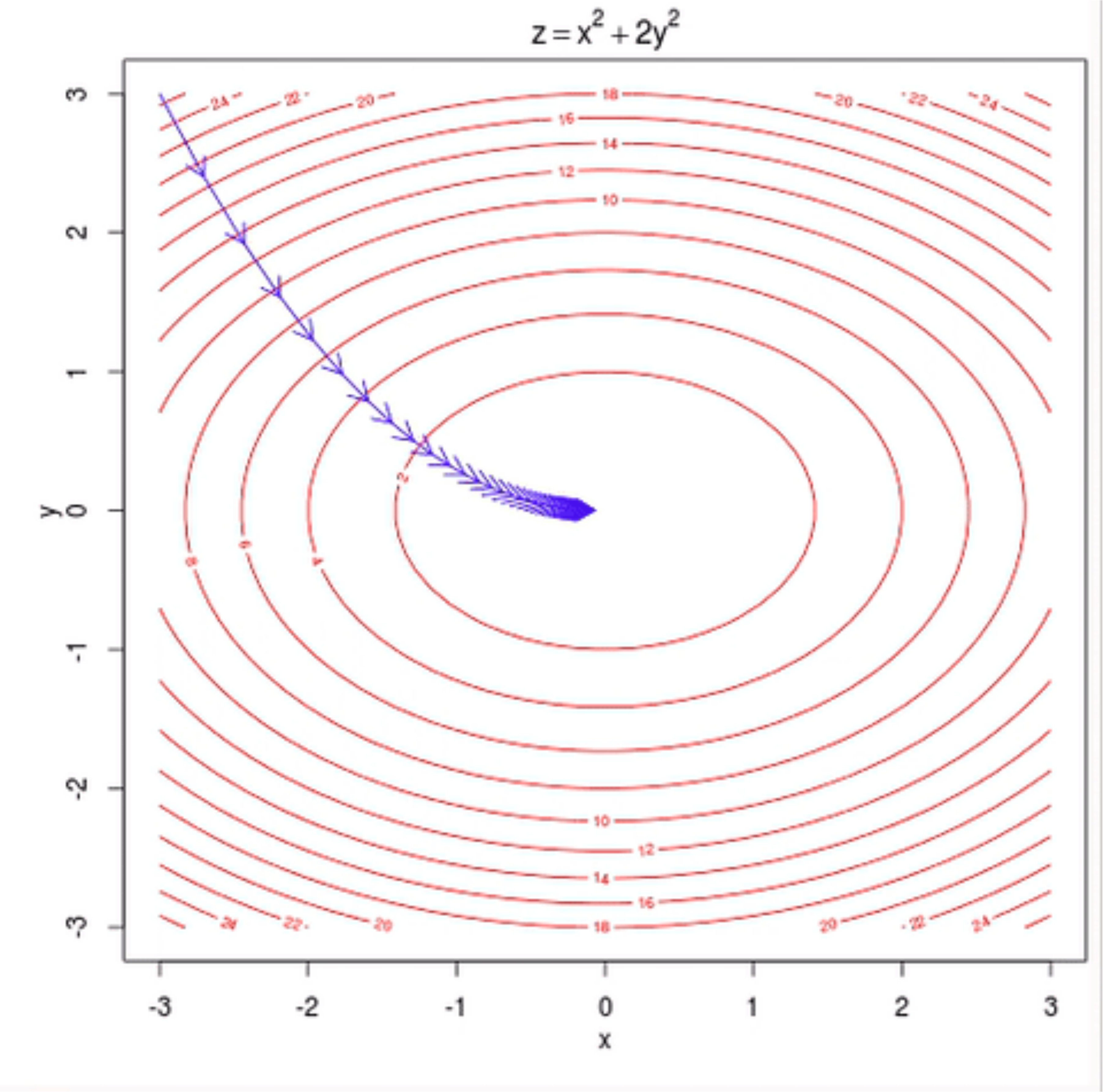




Credits: Wikimedia, Hoang Duong

Recap: Gradient Descent





Credits: Wikimedia, Hoang Duong

A general-purpose recipe for ML

- Step 1: Collect a dataset $D \equiv \{x^i, y^i\}_{i=1}^{N}$

A general-purpose recipe for ML

- Step 1: Collect a dataset $D \equiv \{x^i, y^i\}_{i=1}^{N}$

- Step 2: Choose a decision function $\hat{y} = f_\theta(x)$

A general-purpose recipe for ML

- Step 1: Collect a dataset $D \equiv \{x^i, y^i\}_{i=1}^N$
- Step 2: Choose a decision function $\hat{y} = f_\theta(x)$
- Step 3: Construct a loss function $l(\hat{y}^i, y^i)$

A general-purpose recipe for ML

- Step 1: Collect a dataset $D \equiv \{x^i, y^i\}_{i=1}^N$
- Step 2: Choose a decision function $\hat{y} = f_\theta(x)$
- Step 3: Construct a loss function $l(\hat{y}^i, y^i)$
- Step 4: Define goal:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} l(f_\theta(x^i), y^i)$$

A general-purpose recipe for ML

- Step 1: Collect a dataset $D \equiv \{x^i, y^i\}_{i=1}^N$
- Step 2: Choose a decision function $\hat{y} = f_\theta(x)$
- Step 3: Construct a loss function $l(\hat{y}^i, y^i)$
- Step 4: Define goal:

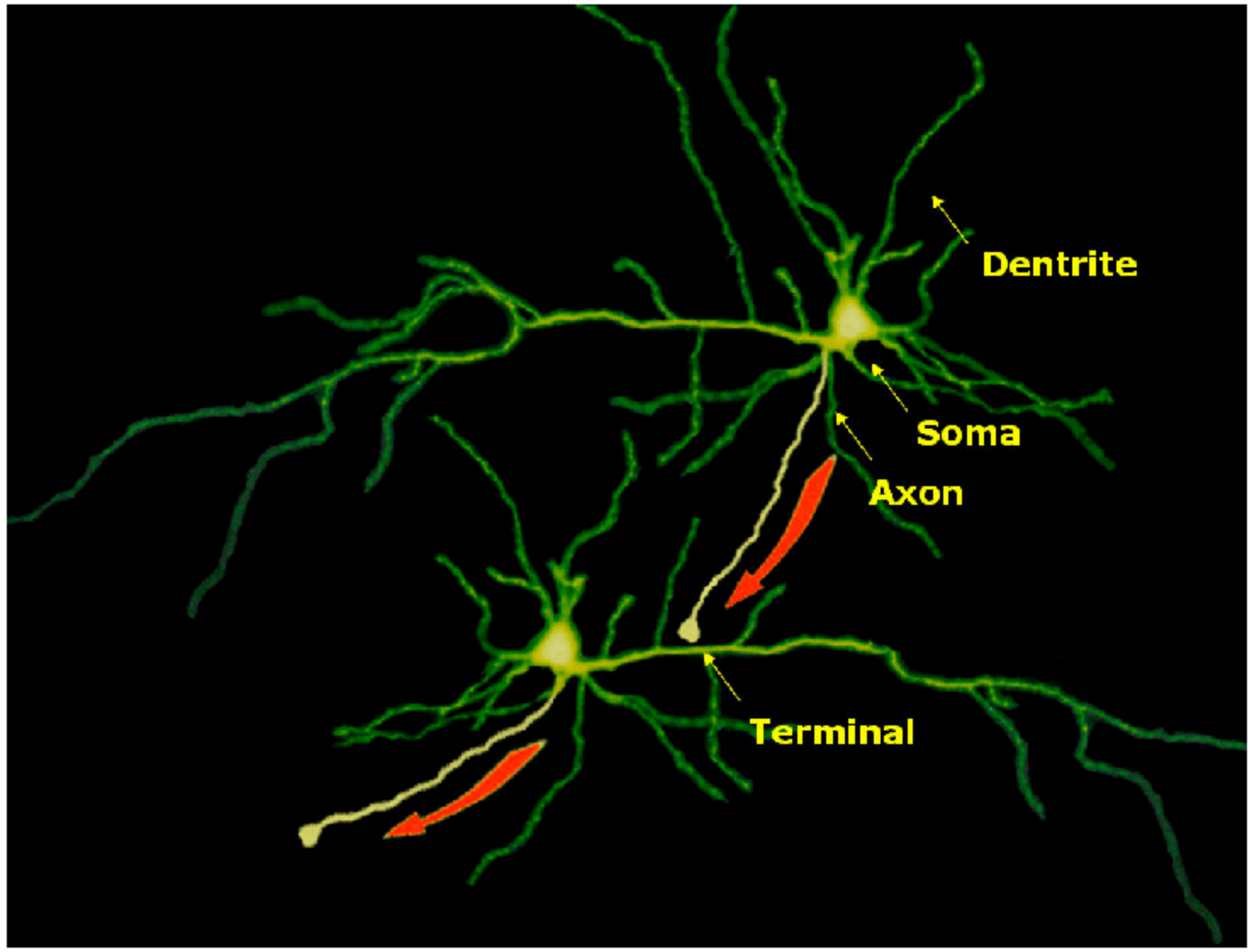
$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} l(f_\theta(x^i), y^i)$$

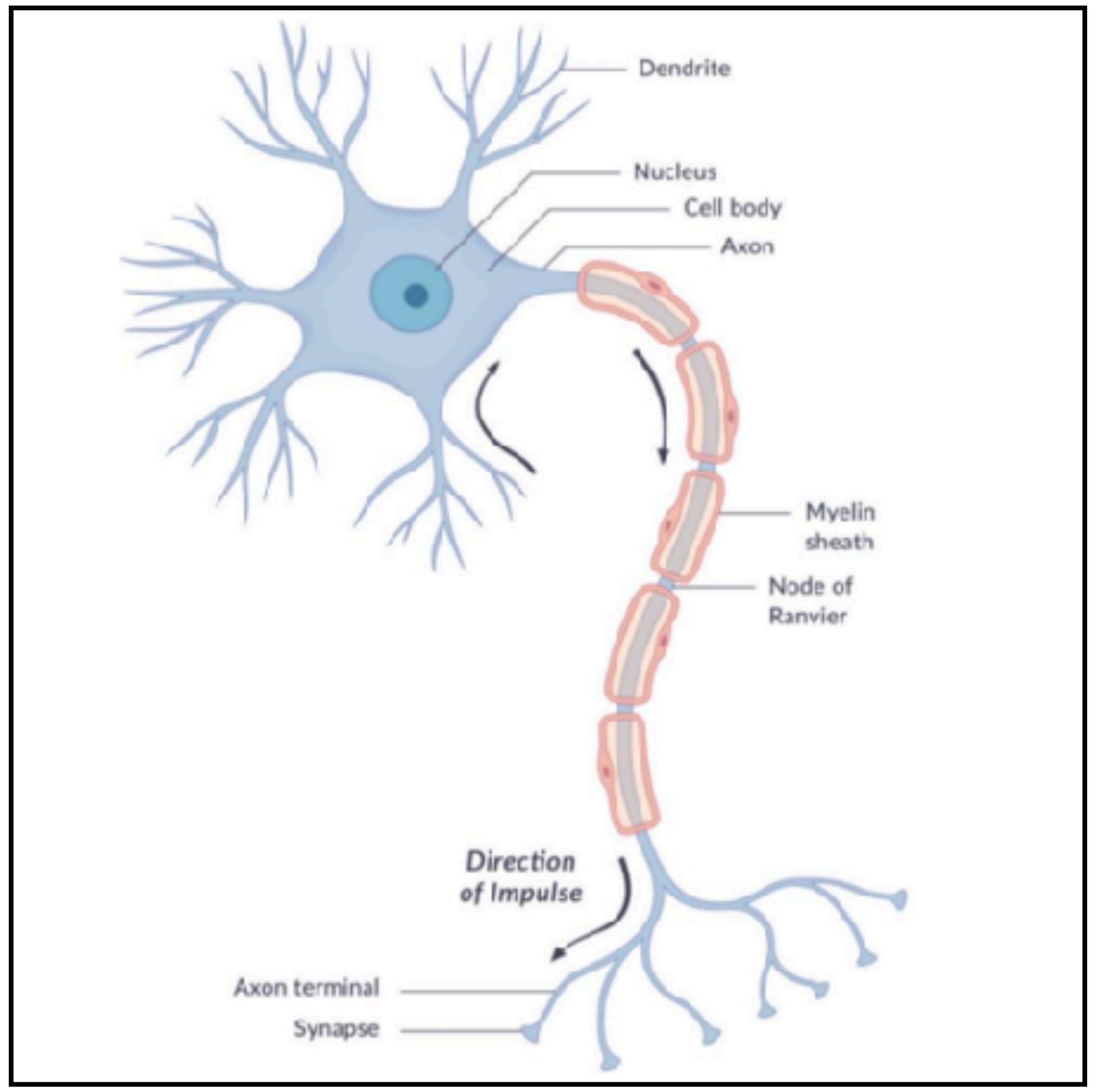
- Step 5: Train with SGD (or variants of GD).

Big question:

What should the decision function be?

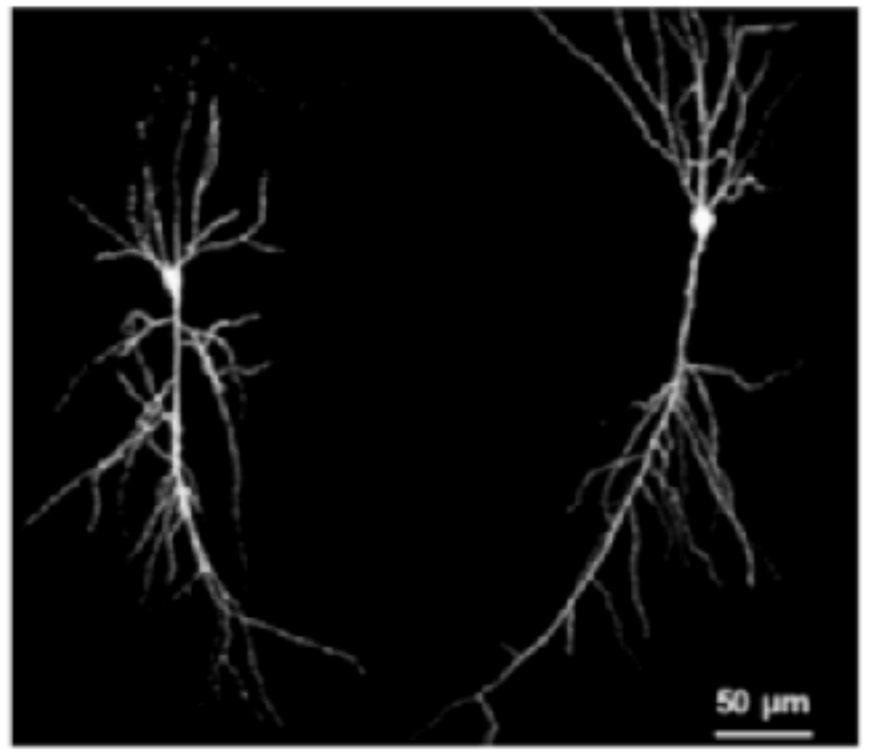
Lets take inspiration from the Brain



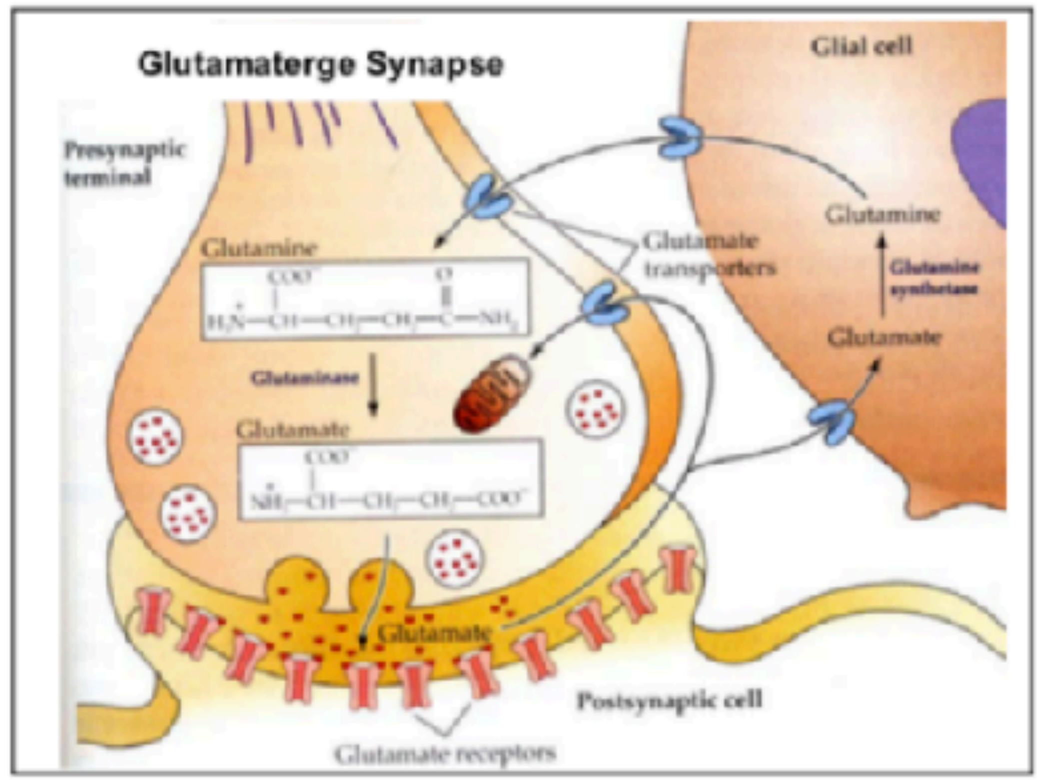


Credits: William Thomas O'Connor, wetcake / Getty Images

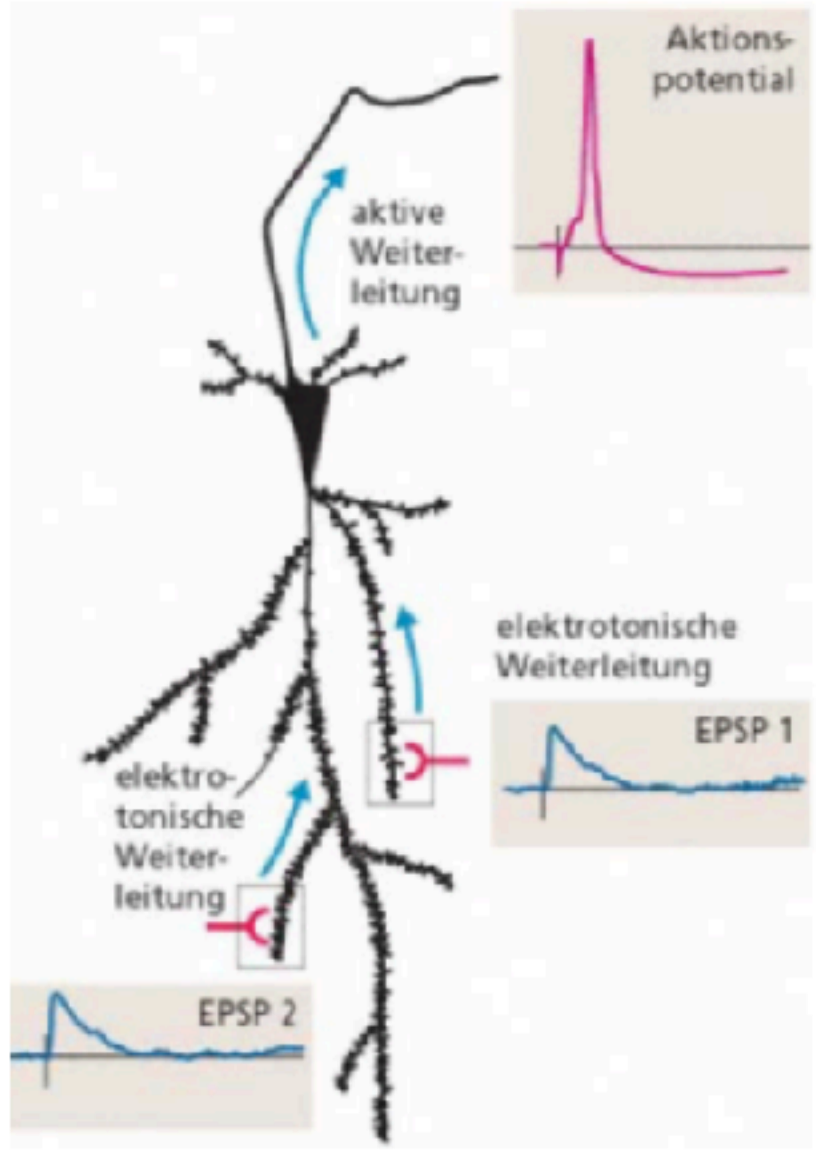
Lets take inspiration from the Brain



Pyramidal neuron cells in mouse cortex



Synaptic connection is chemical



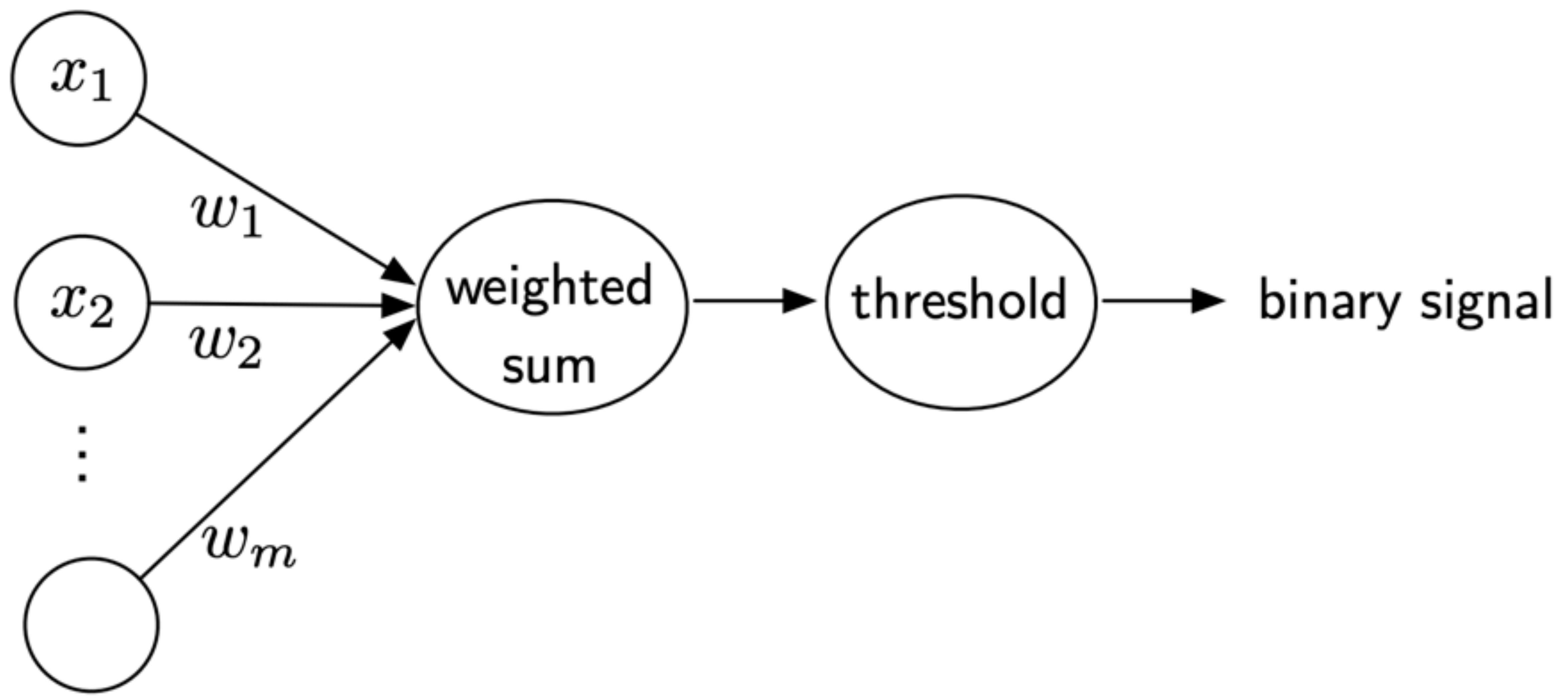
electrical postsynaptic potential accumulates; when it reaches a threshold => action potential signal

Credits: Sebastian Raschka

Lets take inspiration from the Brain

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH and WALTER H. PITTS 1943



Credits: Sebastian Raschka

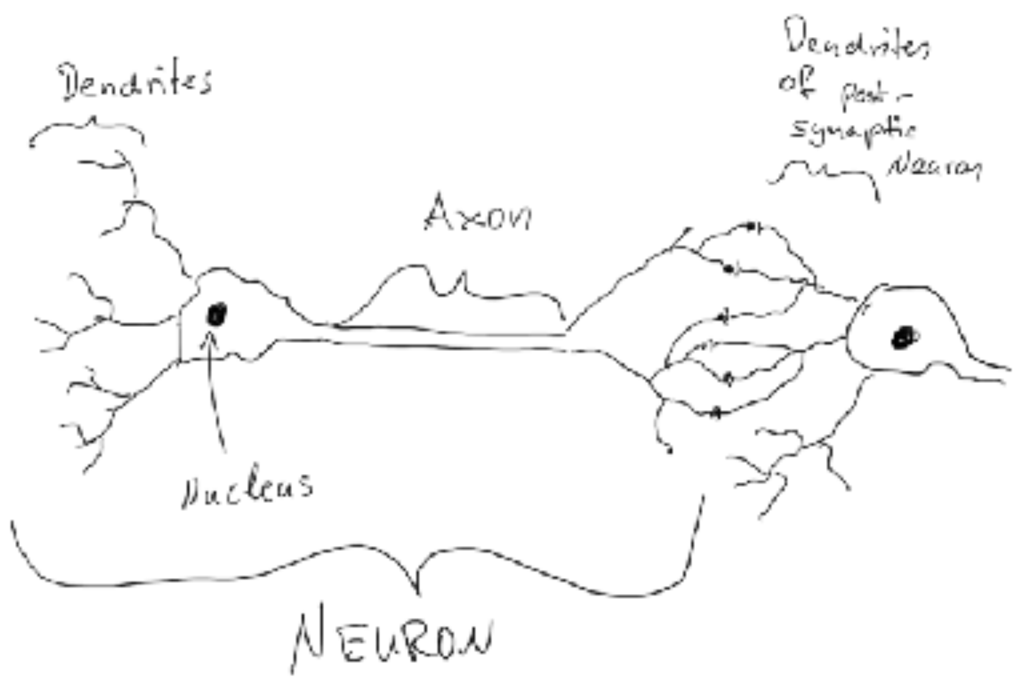
Rosenblatt's Perceptron

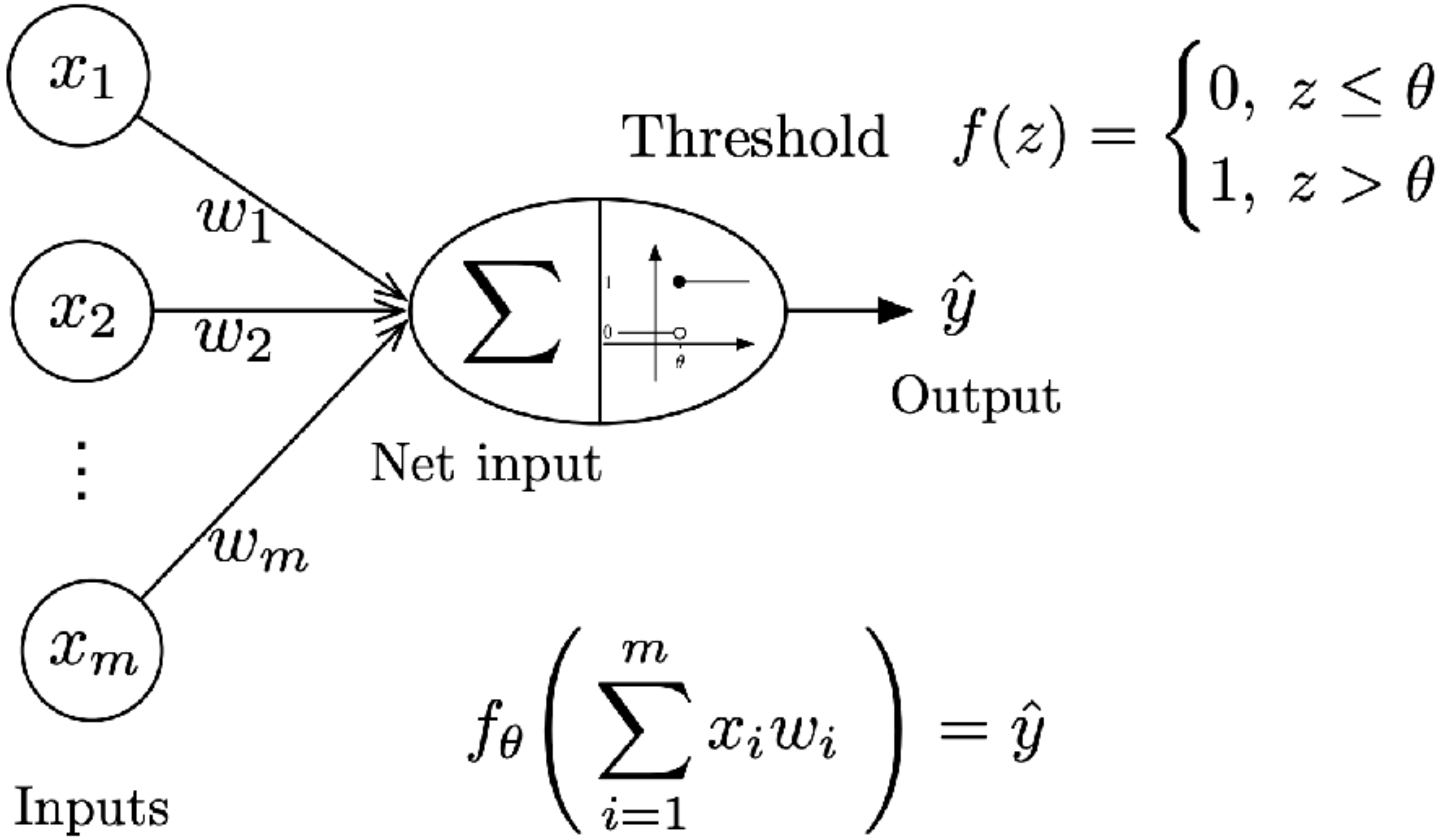
A learning rule for the computational/mathematical neuron model

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton. Project Para*. Cornell Aeronautical Laboratory.

Credits: Sebastian Raschka

Perceptron – Formulation





Credits: Sebastian Raschka

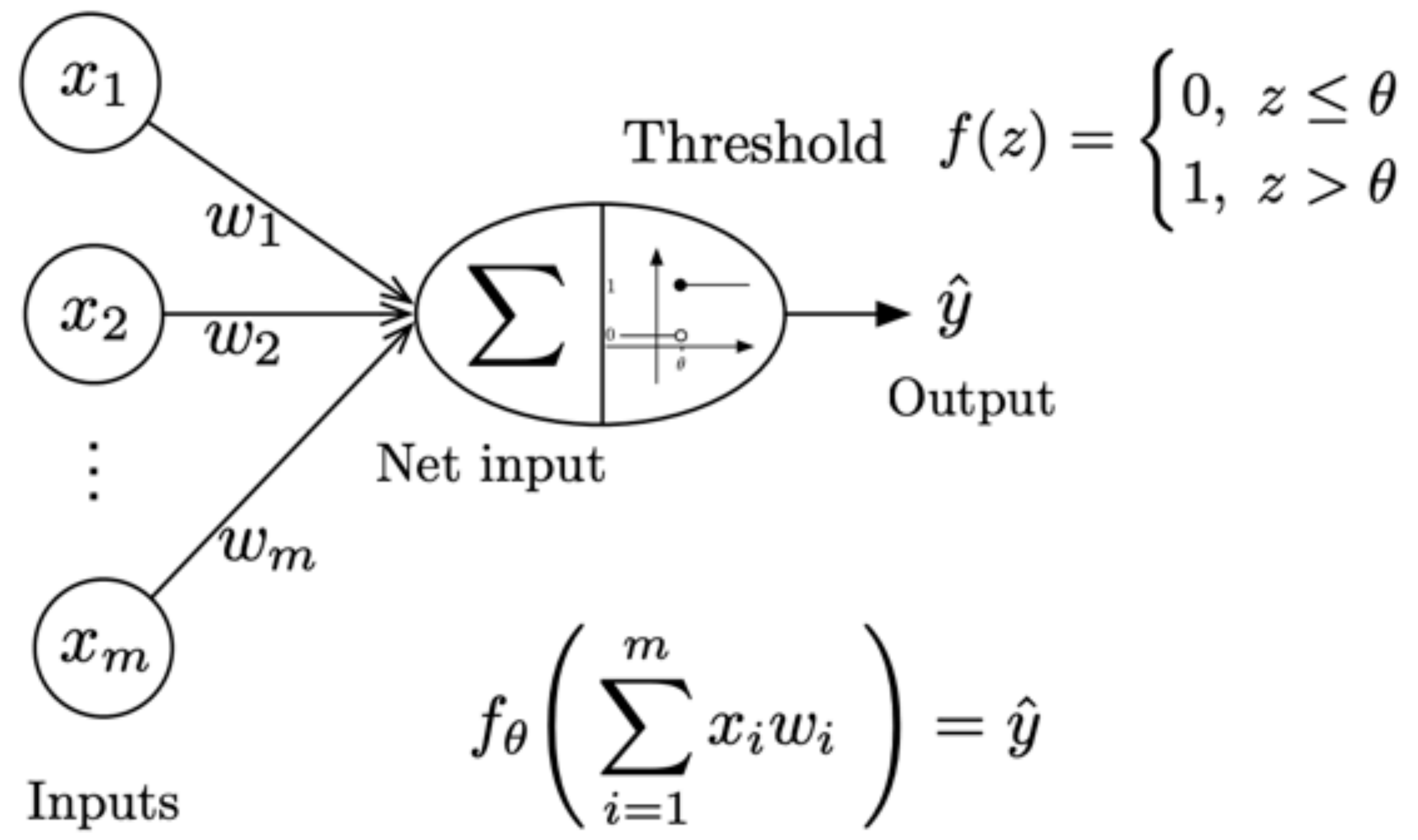
Perceptron – Terminology

General (logistic regression, multilayer nets, ...):

- Net input = weighted inputs
- Activations = activation function(net input)
- Label output = threshold(activations of last layer)

Special cases:

- In perceptron: activation function = threshold function
- In linear regression: activation = net input = output



Credits: Sebastian Raschka

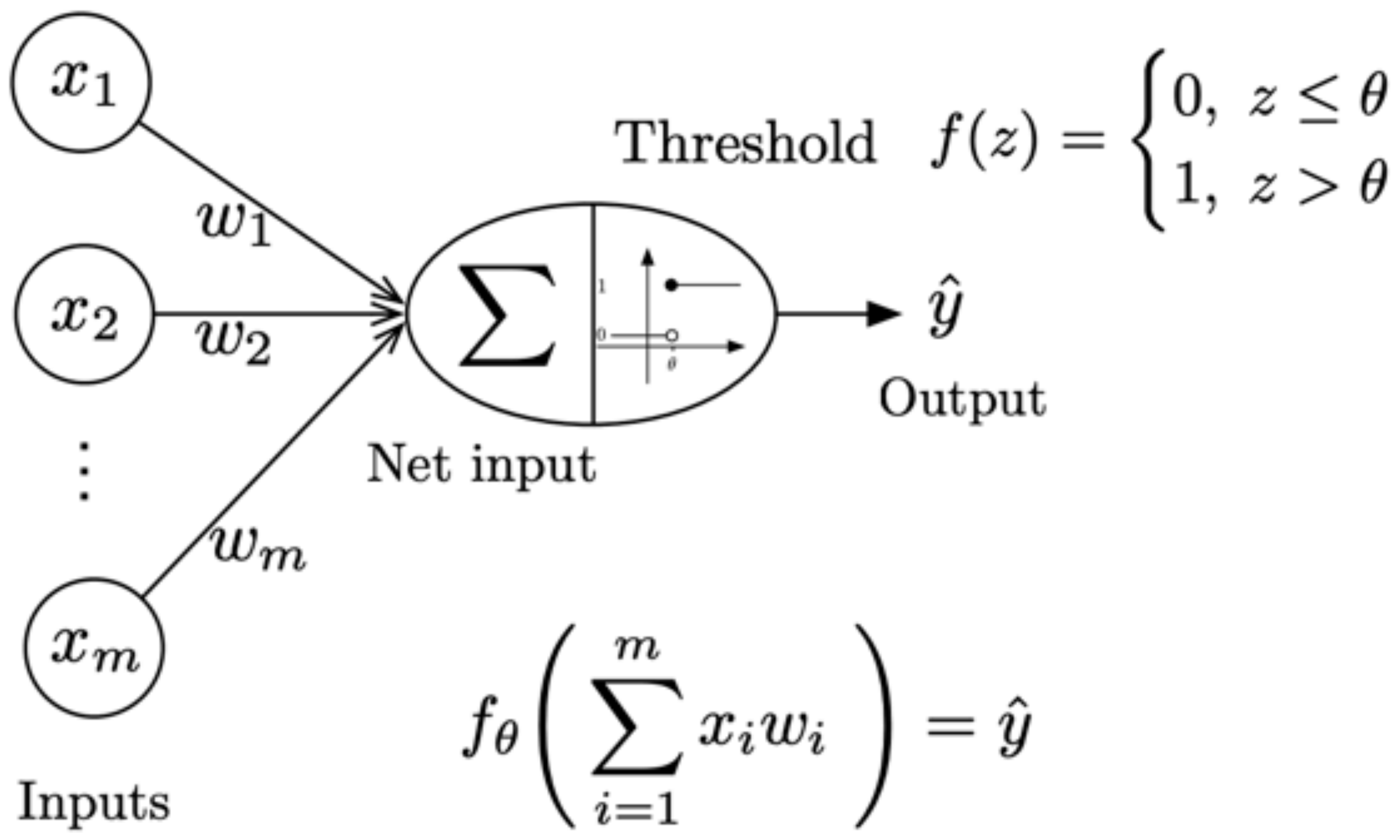
Perceptron – Terminology

General (logistic regression, multilayer nets, ...):

- Net input = weighted inputs
- Activations = activation function(net input)
- Label output = threshold(activations of last layer)

Special cases:

- In perceptron: activation function = threshold function
- In linear regression: activation = net input = output



$$\hat{y} = \begin{cases} 0, & z \leq \theta \\ 1, & z > \theta \end{cases}$$

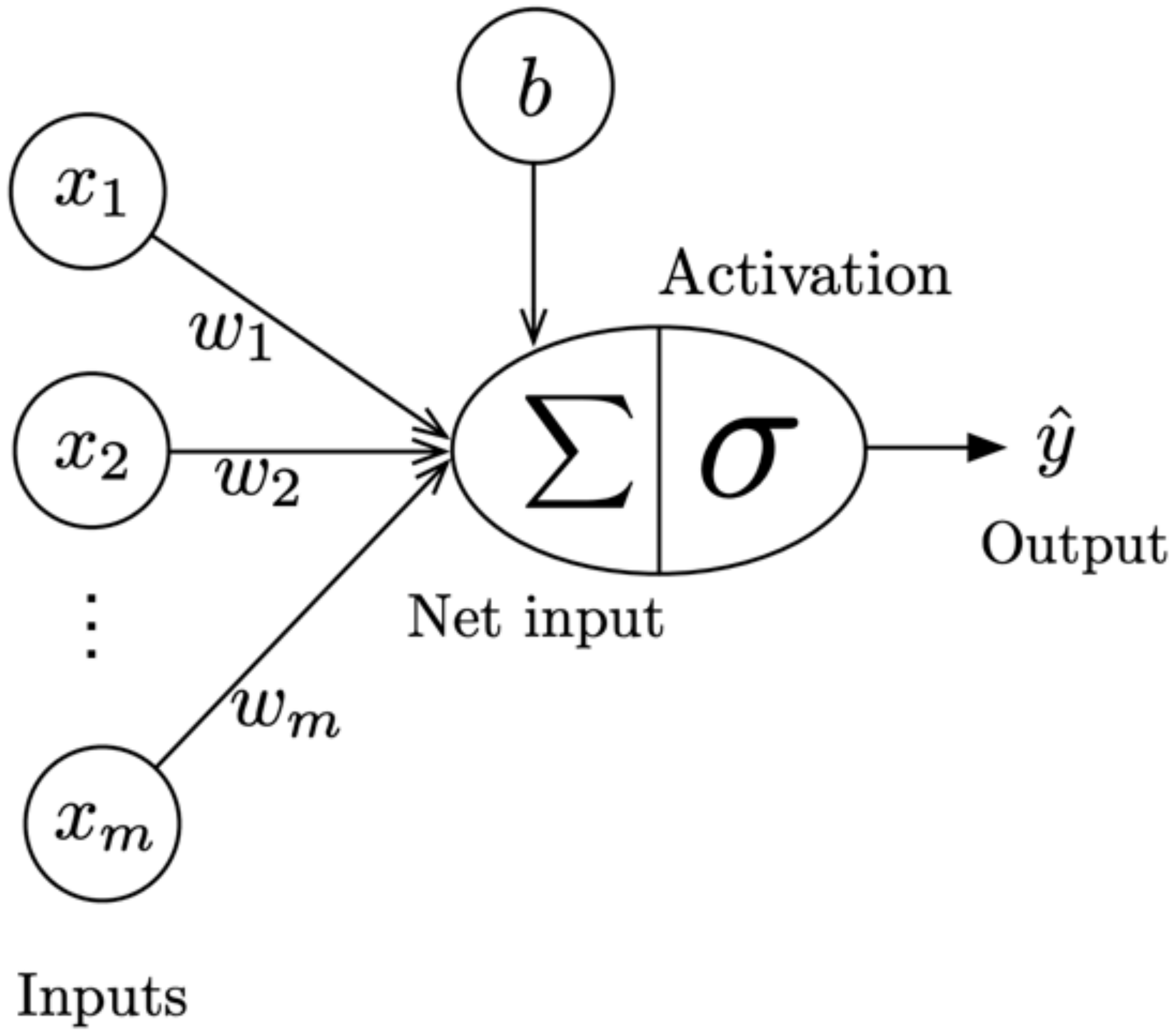
More convenient to re-arrange:

$$\hat{y} = \begin{cases} 0, & z - \theta \leq 0 \\ 1, & z - \theta > 0 \end{cases}$$

$$\boxed{-\theta = \text{"bias"}}$$

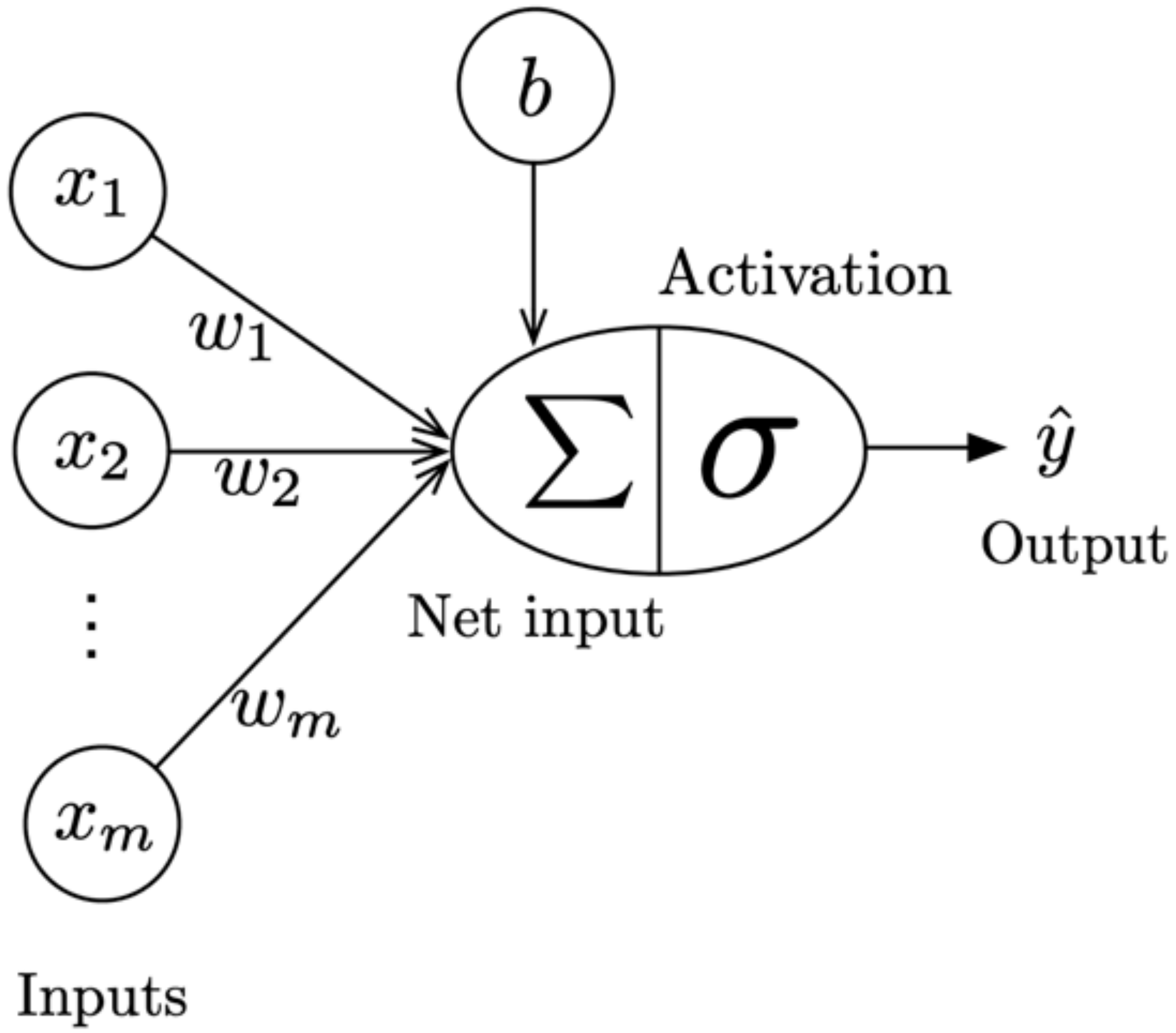
Credits: Sebastian Raschka

Perceptron – Computational Model



Credits: Sebastian Raschka

Perceptron – Computational Model



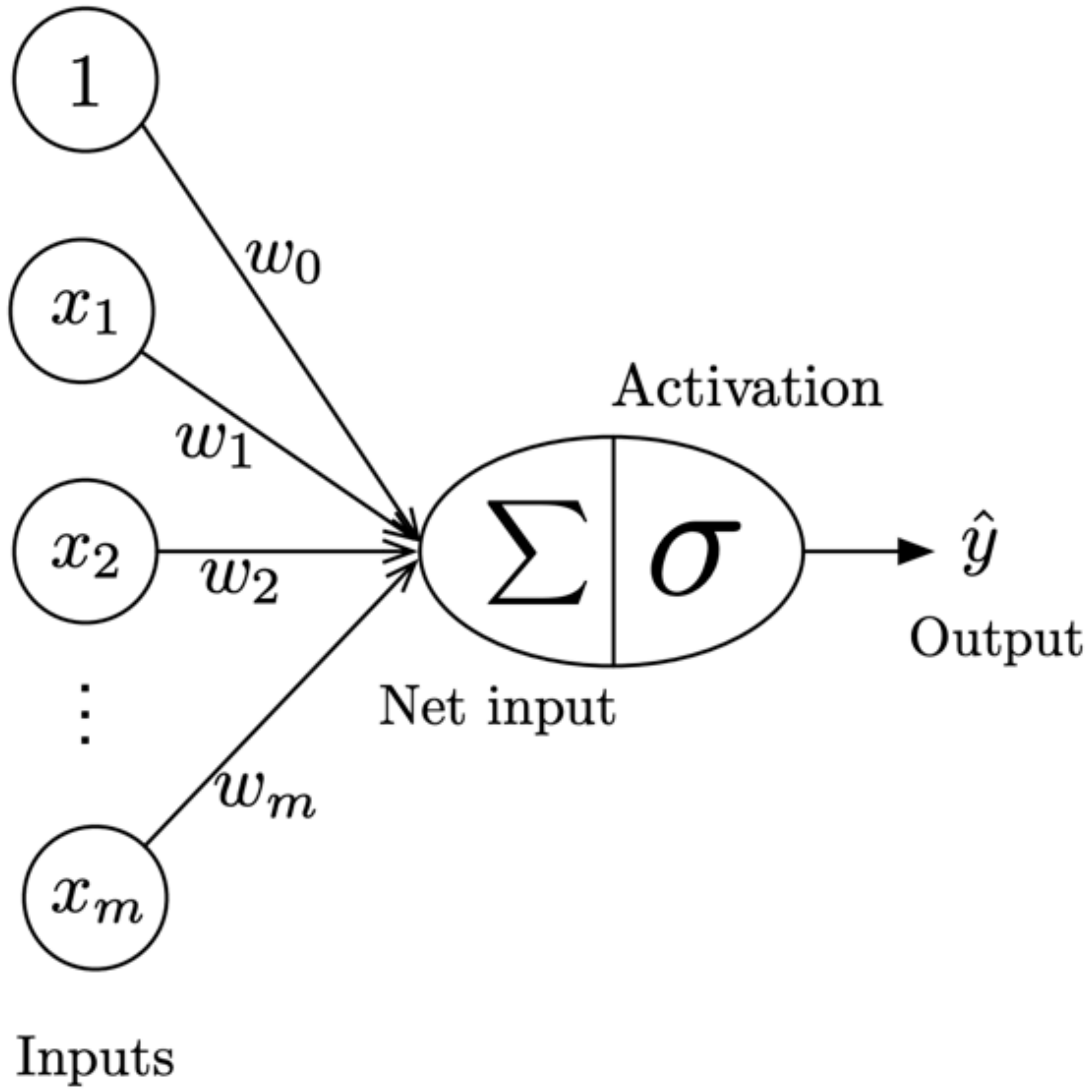
$$\sigma\left(\sum_{i=1}^{m} x_i w_i + b\right) = \sigma\left(\mathbf{x}^T \mathbf{w} + b\right) = \hat{y}$$

$$\sigma(z) = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases}$$

$$b = -\theta$$

Credits: Sebastian Raschka

Perceptron – Computational Model



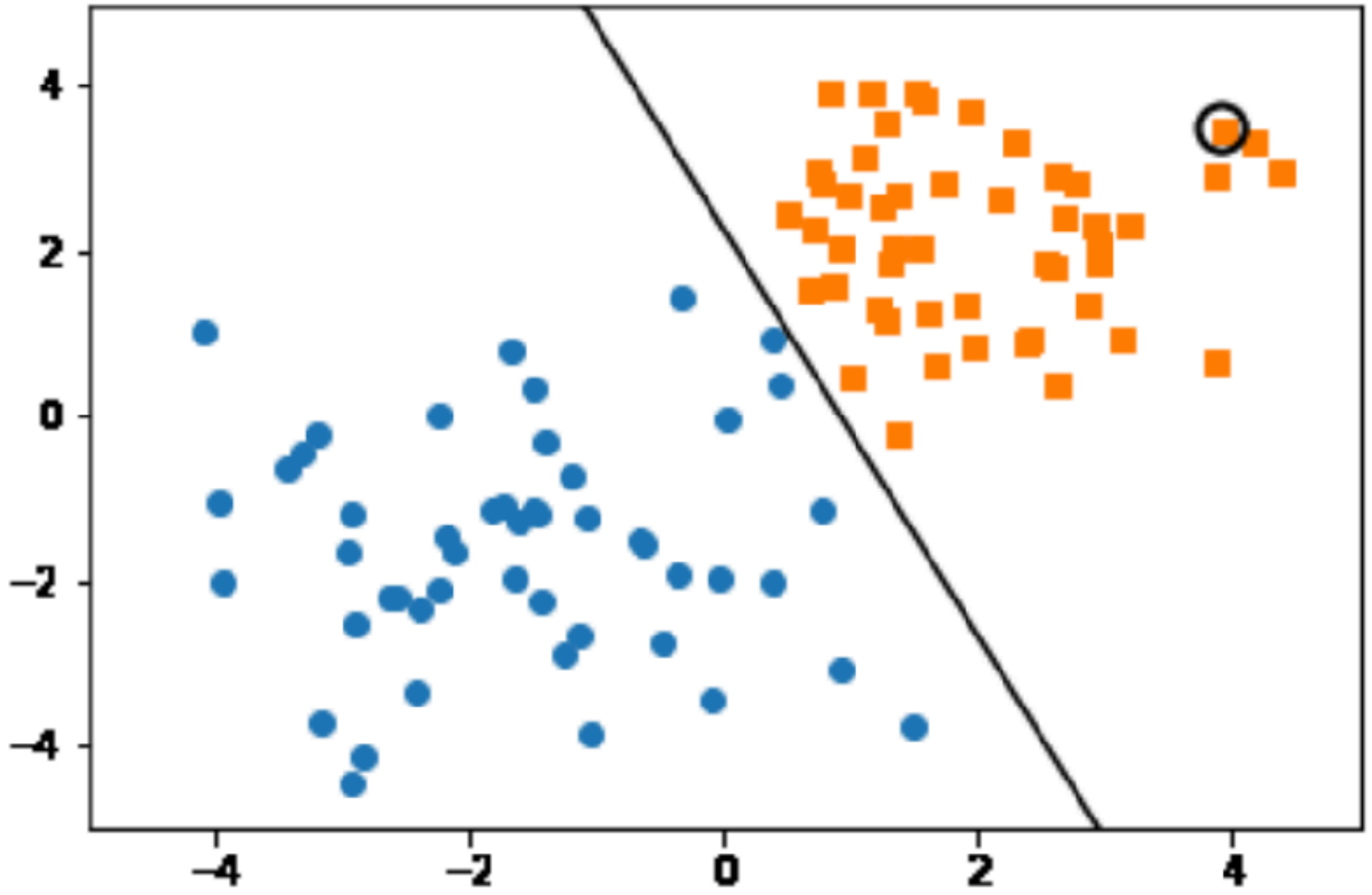
$$\sigma\left(\sum_{i=0}^{m} x_i w_i\right) = \sigma\left(\mathbf{x}^T \mathbf{w}\right) = \hat{y}$$

$$\sigma(z) = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases}$$

$$w_0 = -\theta$$

Credits: Sebastian Raschka

Perceptron – Classification



Credits: Sebastian Raschka

Perceptron – Learning Algorithm

- Let: $D \equiv \{x^i, y^i\}_{i=1}^{N}$
- Initialize $\overrightarrow{w}^0 = 0^d$
- For every training 'epoch':
  - For every $(x^i, y^i) \in D$:
    - $\hat{y}^i = \sigma(\overrightarrow{w}^T x^i)$
    - $e = (y^i - \hat{y}^i)$
    - $\overrightarrow{w}^{t+1} \leftarrow \overrightarrow{w}^t + e \times x^i$

Credits: Sebastian Raschka

Perceptron – Learning Algorithm

- Let: $D \equiv \{x^i, y^i\}_{i=1}^N$
- Initialize $\vec{w}^0 = 0^d$
- For every training 'epoch':
  - For every $(x^i, y^i) \in D$:
    - $\hat{y}^i = \sigma(\vec{w}^T x^i)$
    - $e = (y^i - \hat{y}^i)$
    - $\vec{w}^{t+1} \leftarrow \vec{w}^t + e \times x^i$

Principle:

- If there is no error, do not update.
- If output is 0 and target is 1, add input to weight vector.
- If output is 1 and target is 0, subtract input from weight vector.

Credits: Sebastian Raschka

Perceptron – Learning Algorithm

- Let: $D \equiv \{x^i, y^i\}_{i=1}^{N}$
- Initialize $\overrightarrow{w}^0 = 0^d$
- For every training 'epoch':
  - For every $(x^i, y^i) \in D$:
    - $\hat{y}^i = \sigma(\overrightarrow{w}^T x^i)$
    - $e = (y^i - \hat{y}^i)$
    - $\overrightarrow{w}^{t+1} \leftarrow \overrightarrow{w}^t + e \times x^i$

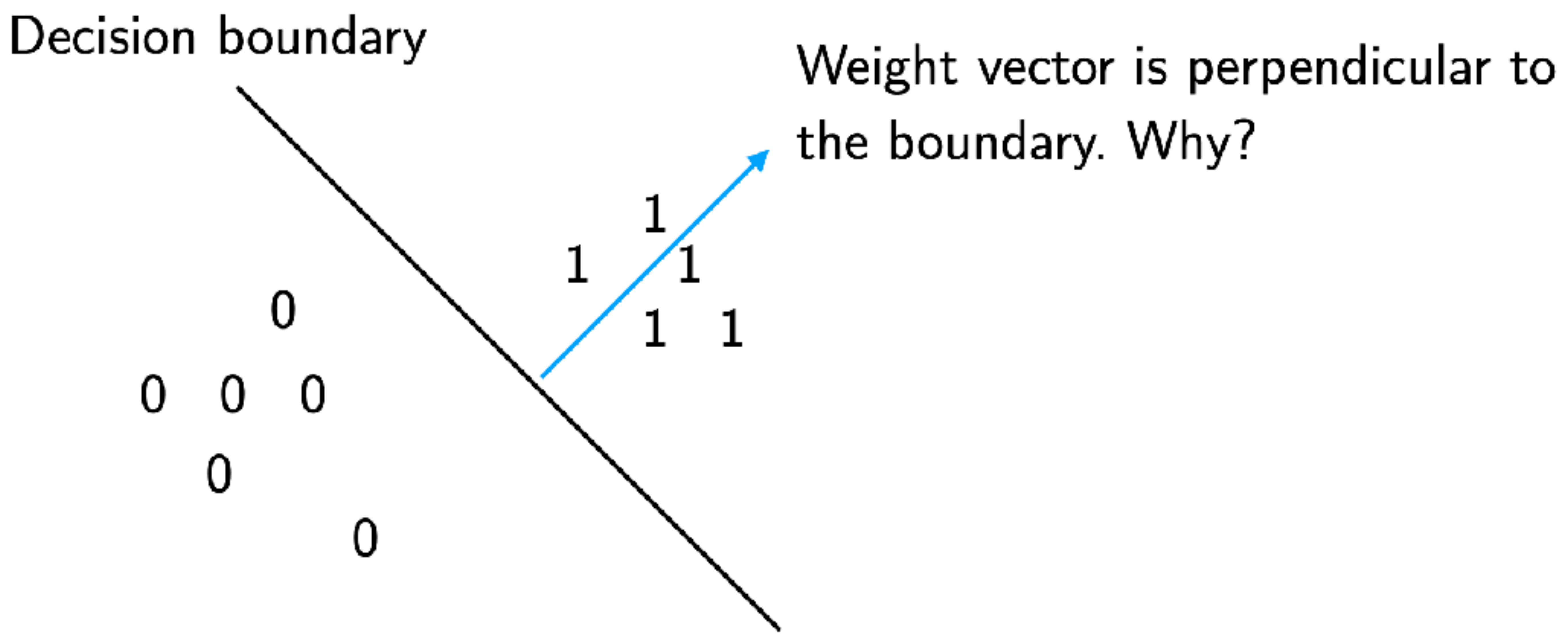
Principle:

- If there is no error, do not update.
- If output is 0 and target is 1, add input to weight vector.
- If output is 1 and target is 0, subtract input from weight vector.

Guaranteed to converge if solution exists!

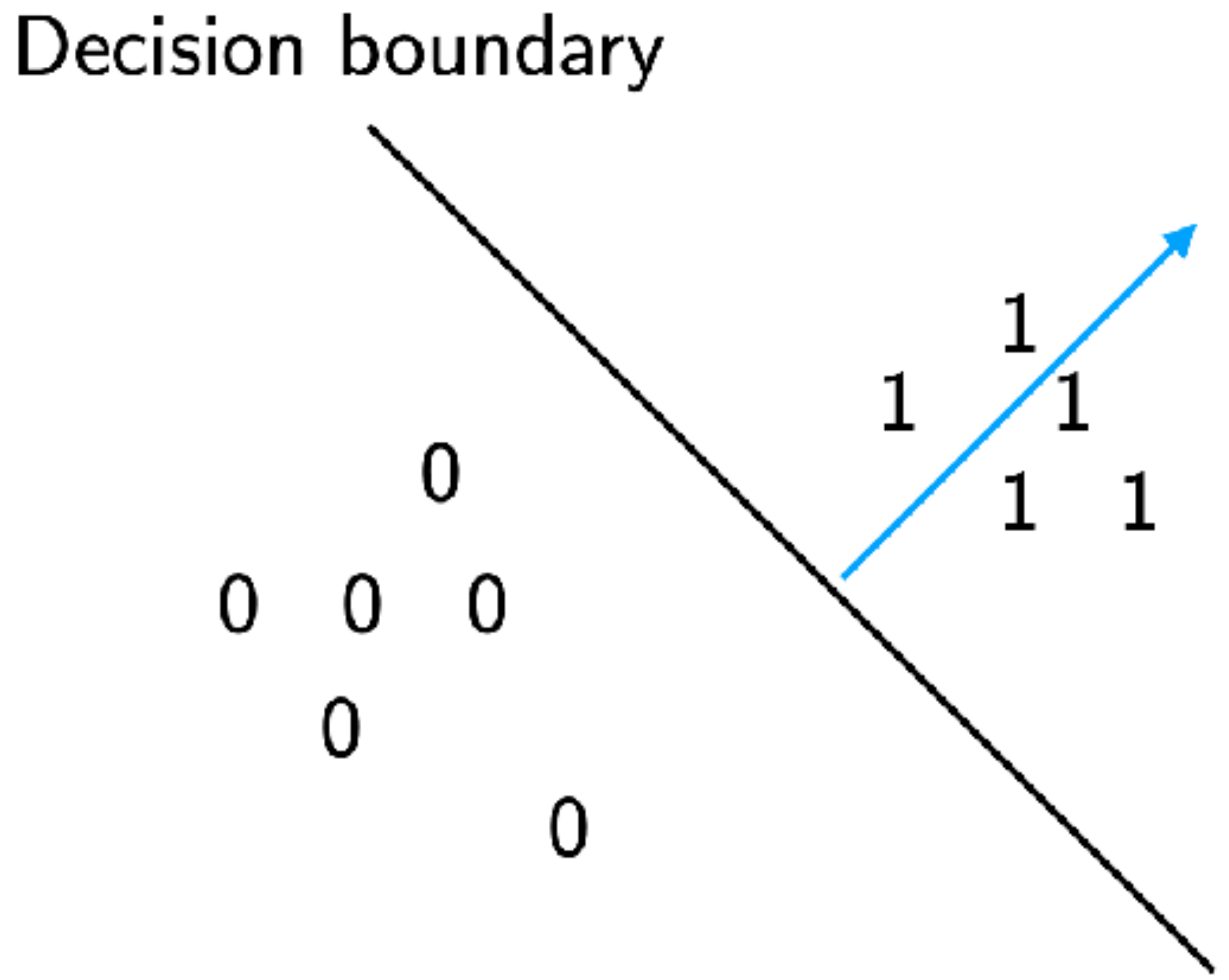
Credits: Sebastian Raschka

Perceptron – Geometric Intuition



Credits: Sebastian Raschka

Perceptron – Geometric Intuition



Weight vector is perpendicular to the boundary. Why?

Remember,

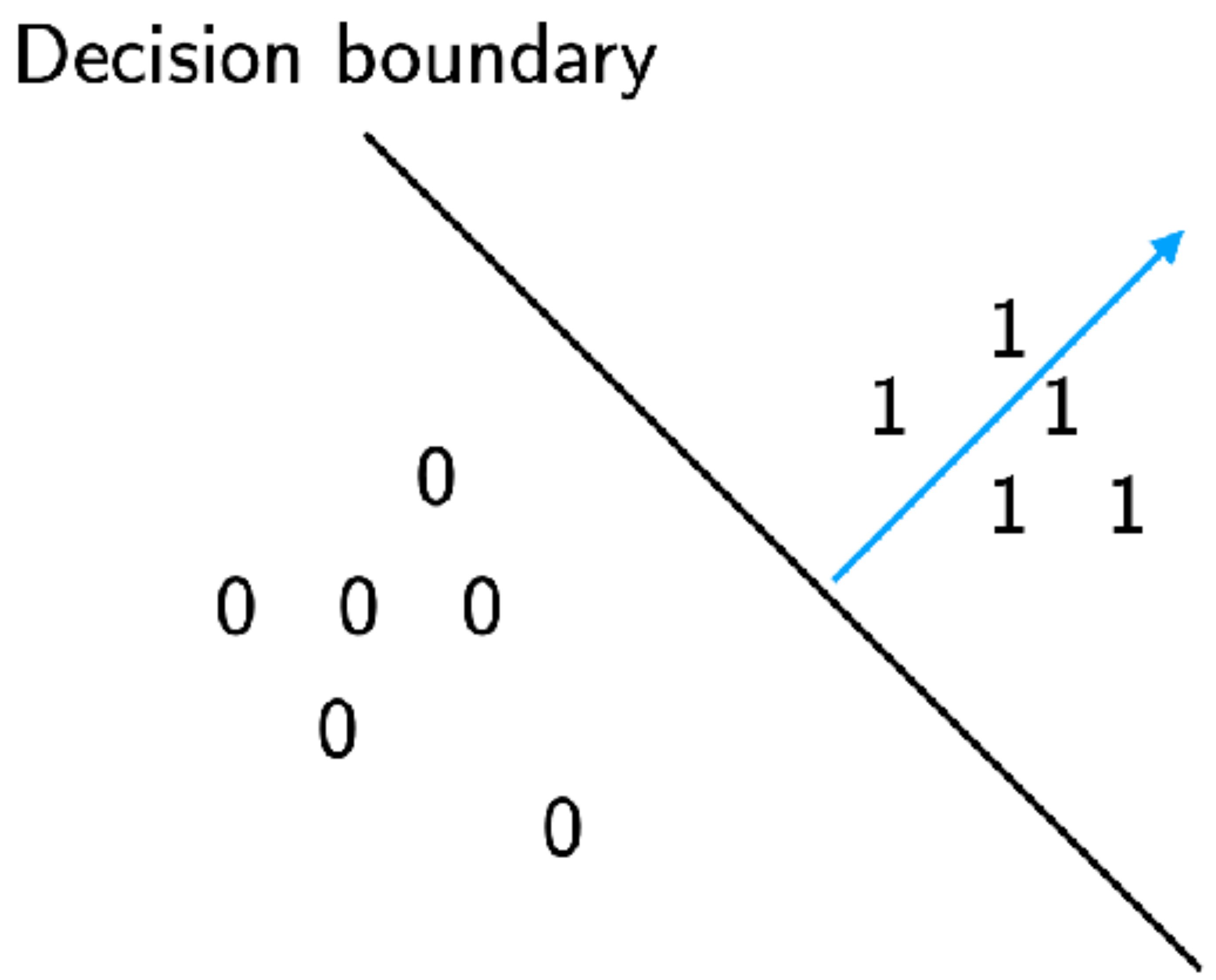
$$\hat{y} = \begin{cases} 0, & \mathbf{w}^T\mathbf{x} \leq 0 \\ 1, & \mathbf{w}^T\mathbf{x} > 0 \end{cases}$$

$$\mathbf{w}^T\mathbf{x} = ||\mathbf{w}|| \cdot ||\mathbf{x}|| \cdot \underbrace{\cos(\theta)}$$

So this needs to be 0 at the boundary, and it is zero at 90°

Credits: Sebastian Raschka

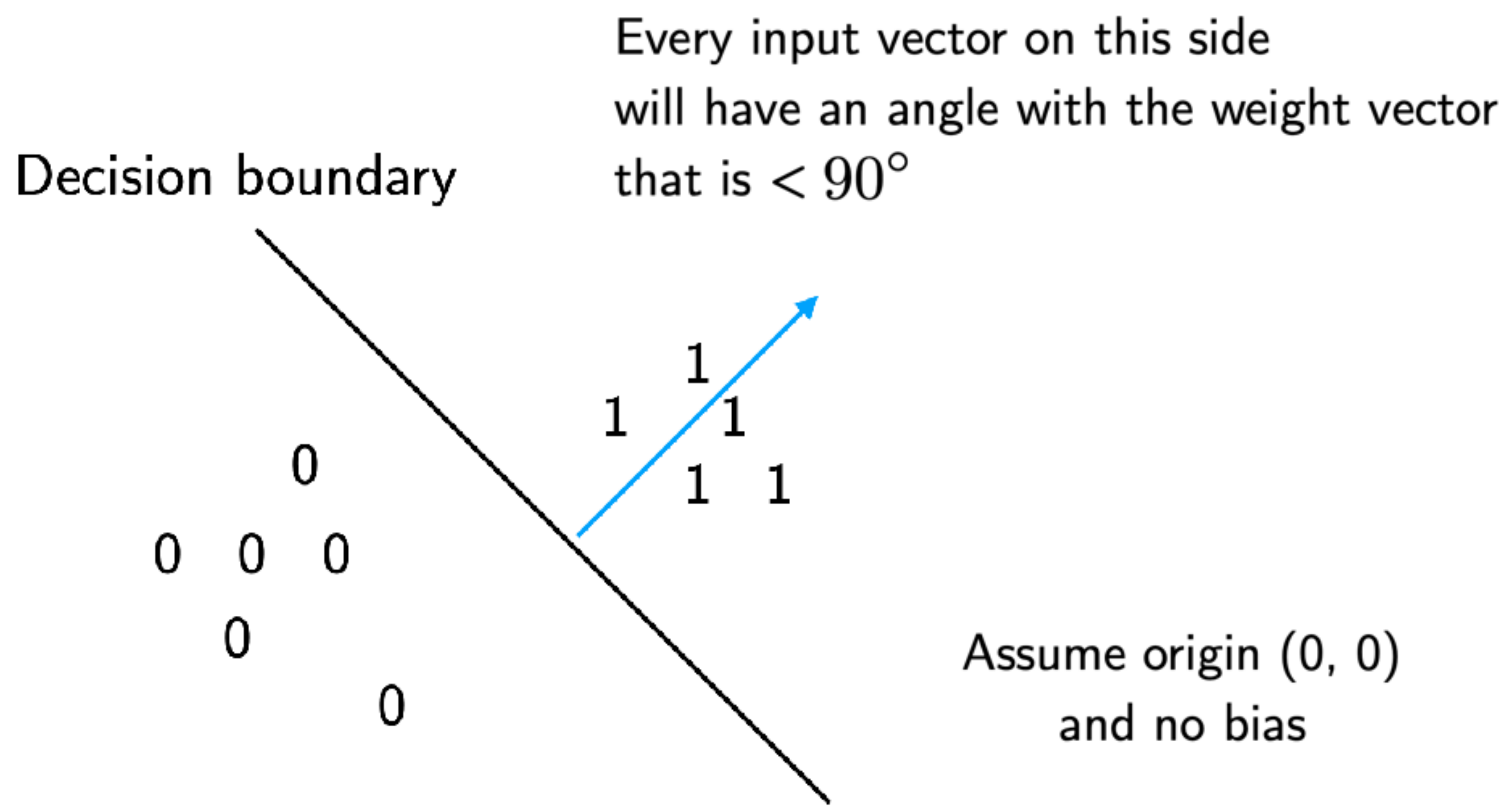
Perceptron – Geometric Intuition



Assume origin (0, 0)
and no bias

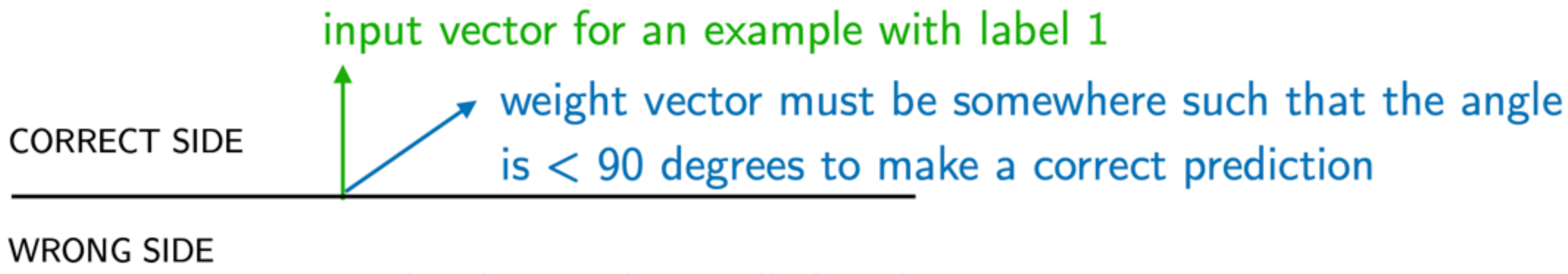
Credits: Sebastian Raschka

Perceptron – Geometric Intuition



Credits: Sebastian Raschka

Perceptron – Geometric Intuition



The dot product will then be positive, i.e., > 0, since

$$\mathbf{w}^T\mathbf{x} = ||\mathbf{w}|| \cdot ||\mathbf{x}|| \cdot \cos(\theta)$$

Credits: Sebastian Raschka

Perceptron – Geometric Intuition

Credits: Sebastian Raschka

Perceptron – Geometric Intuition

Credits: Sebastian Raschka

Perceptron – Downsides

- No non-linear boundaries possible with classical perceptron
- Does not converge when classes are non-separable
- In its current form not compatible with gradient descent

Additional Reading

- Perceptron proof: https://mlu.red/muse/52491166310
- Perceptron paper (Rosenblatt 1958): https://psycnet.apa.org/fulltext/1959-09865-001.pdf

Questions?