STEVENS
INSTITUTE OF TECHNOLOGY
1870

PEP 559 Machine Learning in Quantum Physics

Dr. Chunlei Qu

Spring 2025

Four Modules

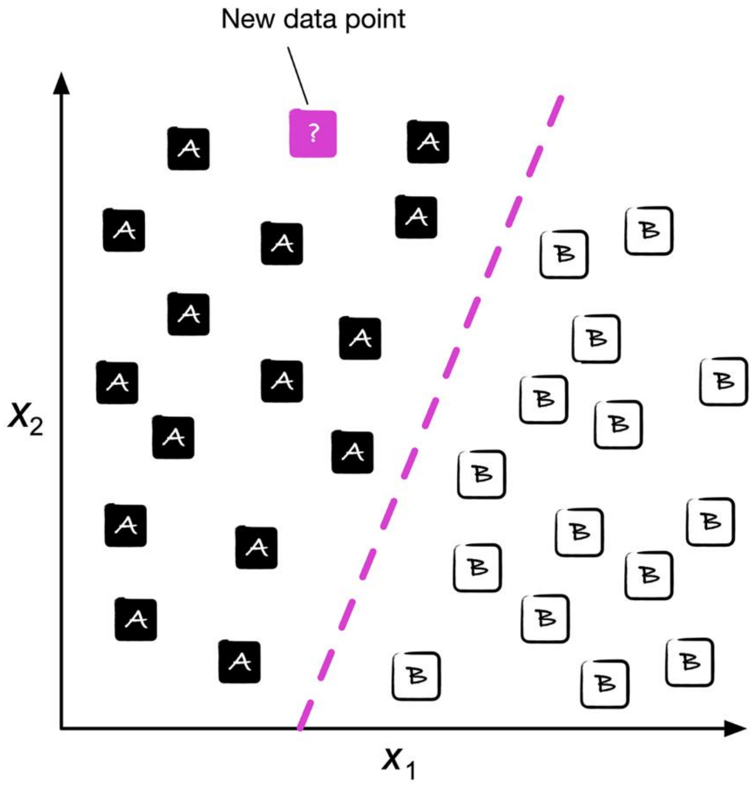
- <u>Model A: Machine Learning</u>
- Module B: Deep Learning
- Module C: Quantum Information
- Module D: Machine Learning for Quantum Physics

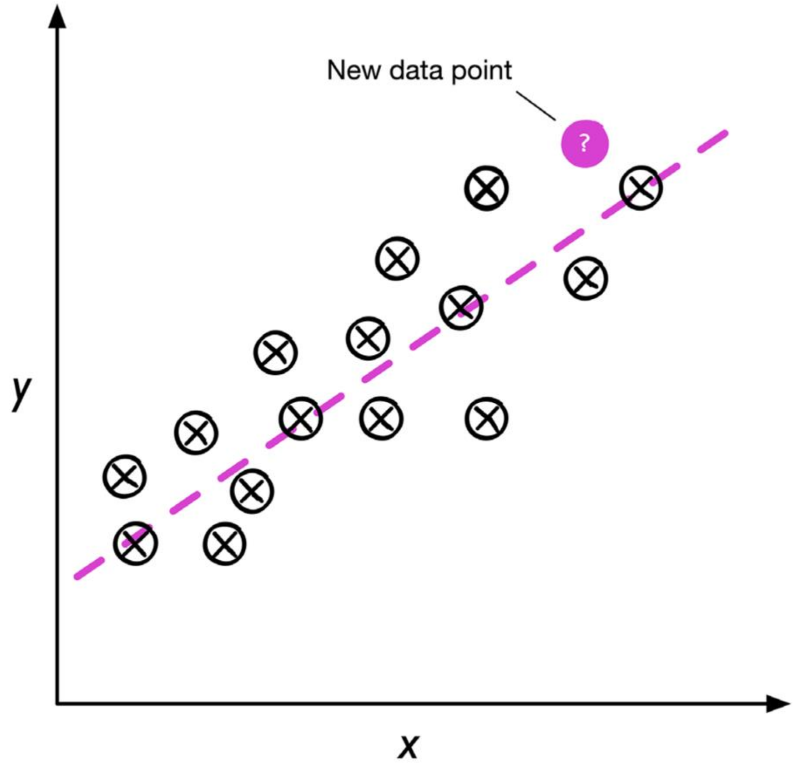
Three types of machine learning

Supervised learning	Labeled data Direct feedback Predict outcome/future
Unsupervised learning	No labels/targets No feedback Find hidden structure in data
Reinforcement learning	Decision process Reward system Learn series of actions

Supervised learning

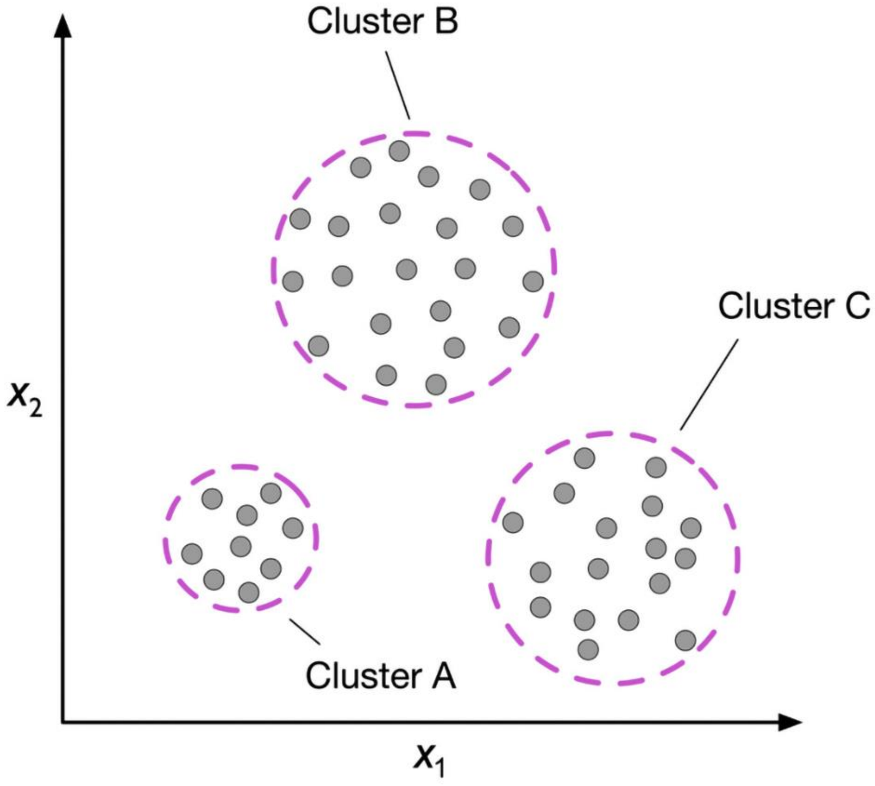
- **Classification:** labels are **discrete**, e.g., email spam detection is a binary classification task
- **Regression:** labels are **continuous**, e.g., house price vs. size

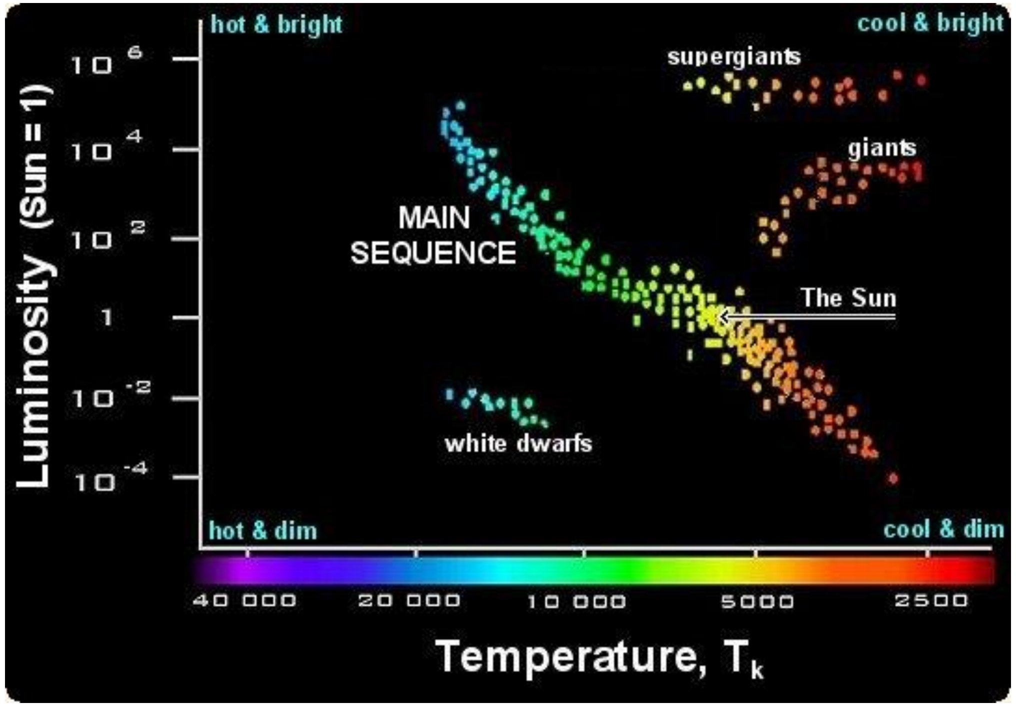




Unsupervised learning

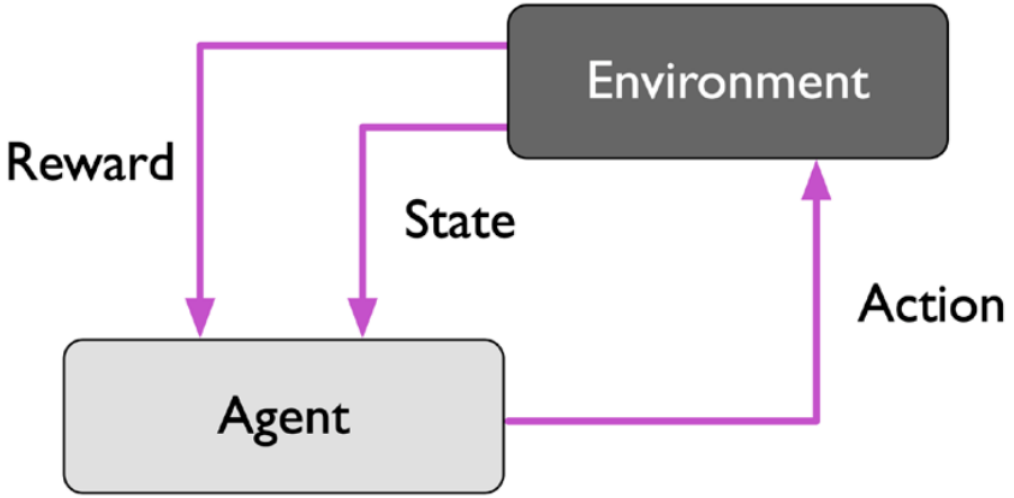
- **Clustering**: Discovering hidden structure of **unlabeled** data
- For example, the **Hertzsprung-Russell diagram** groups stars by temperature and luminosity

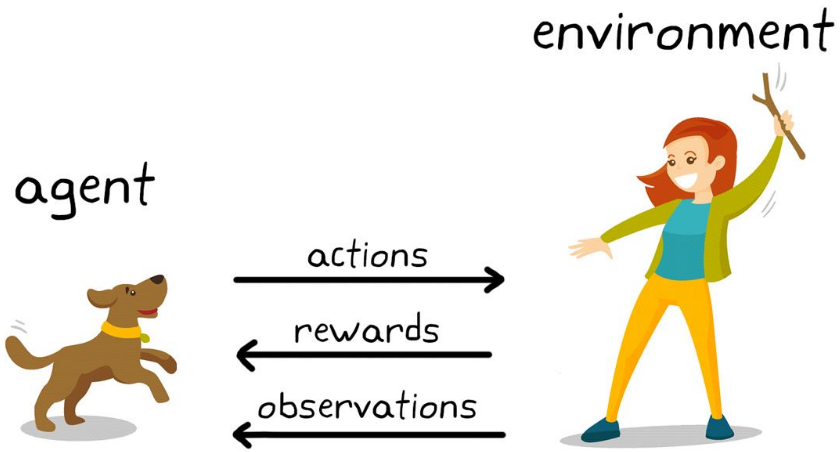




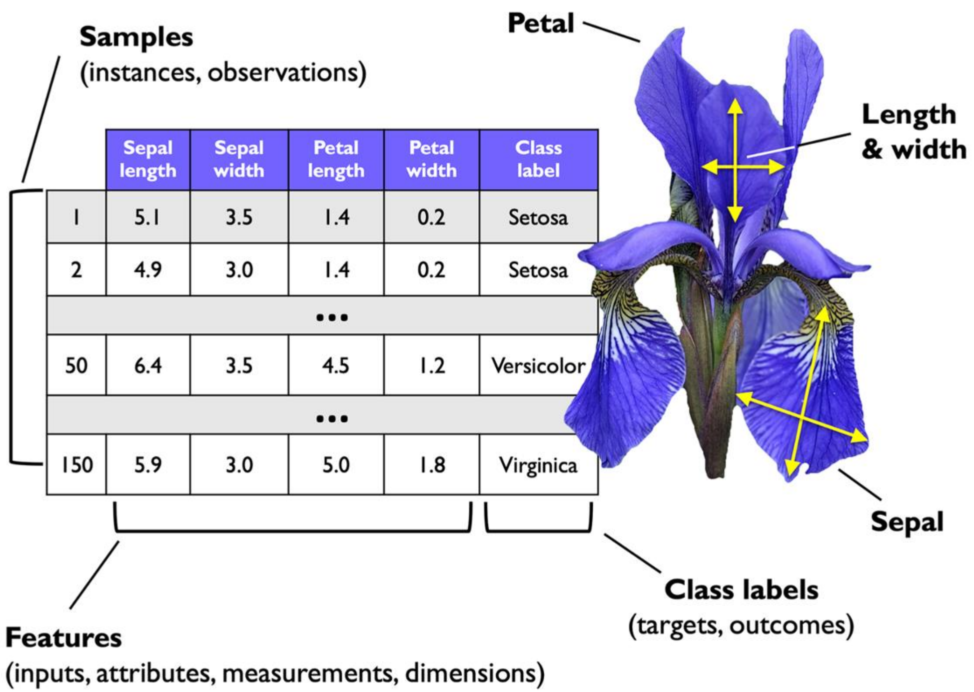
Reinforcement learning

- To develop a system (**agent**) that improves its performance based on interactions with the environment





Notation and Terminology



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

The Iris DataSet

- **4 Features**: Sepal length, Sepal width, Petal length, Petal width
- **150 Samples** or instances or observations, etc.
- **Class labels**: Setosa, Versicolor, Virginica.

Data Matrix

- Superscript = **sample** index = row index
- Subscript = **feature** index = column index

$$\boldsymbol{X} \in \mathbb{R}^{150\times 4}$$

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}$$

sample vector

feature vector

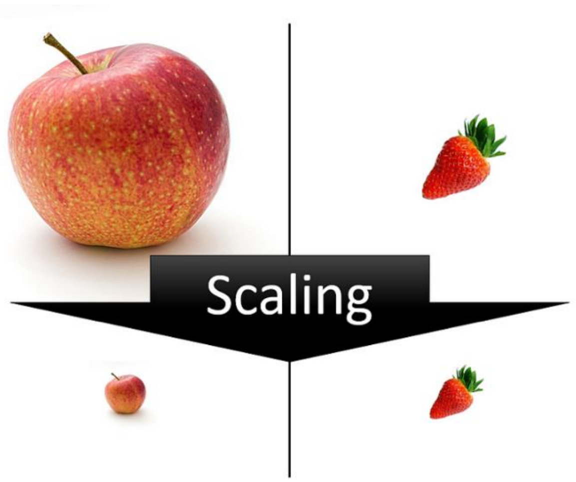
Terminology

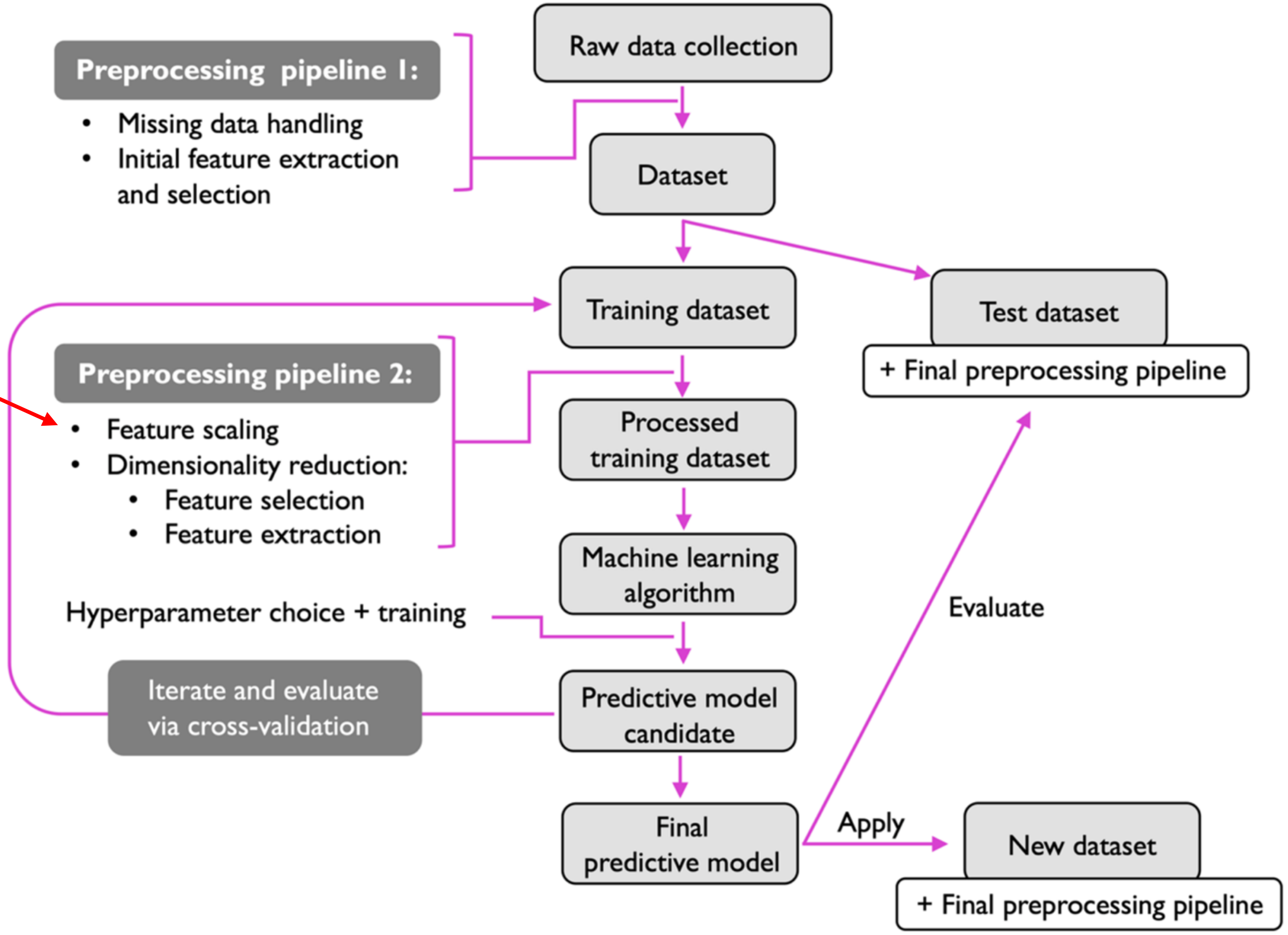
- **Training example**: a row in the data matrix, also known as an observation, record, instance, or sample
- **Feature**: a column in the data matrix, also known as predictor, variable, input, attribute
- **Target**: also known as class label, ground truth, outcome, output, etc.
- **Loss function**: also known as cost function or error function

ML typical workflow

- **Feature scaling**

The features should be on the same scale for optimal performance. Normally, we transform it to a standard distribution with zero mean and unit variance.



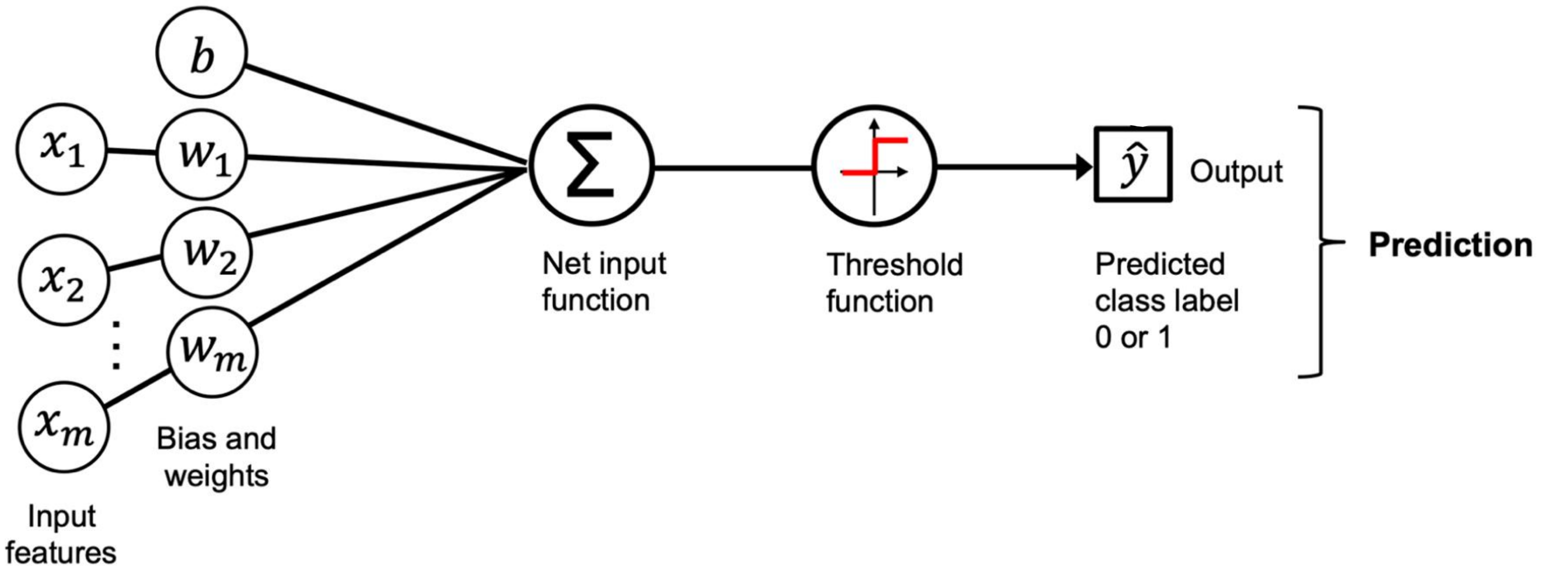


Python

- See Jupyter Notebook

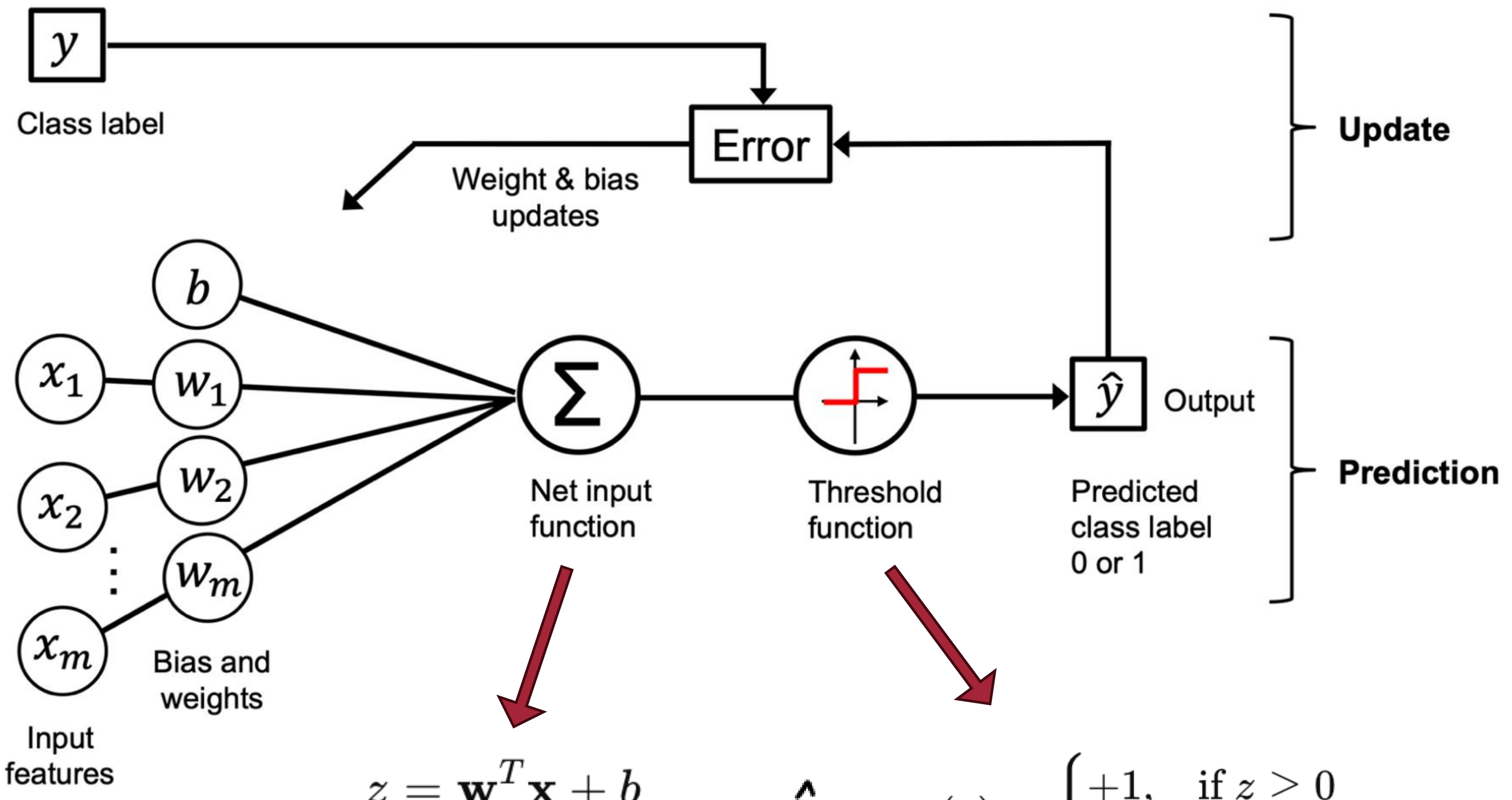
McCulloch-Pitts (MCP) neuron model

- **Pre-determined weights, no learning capability**



Rosenblatt's perceptron model

- Proposed an algorithm that would automatically **learn the optimal weight** coefficients

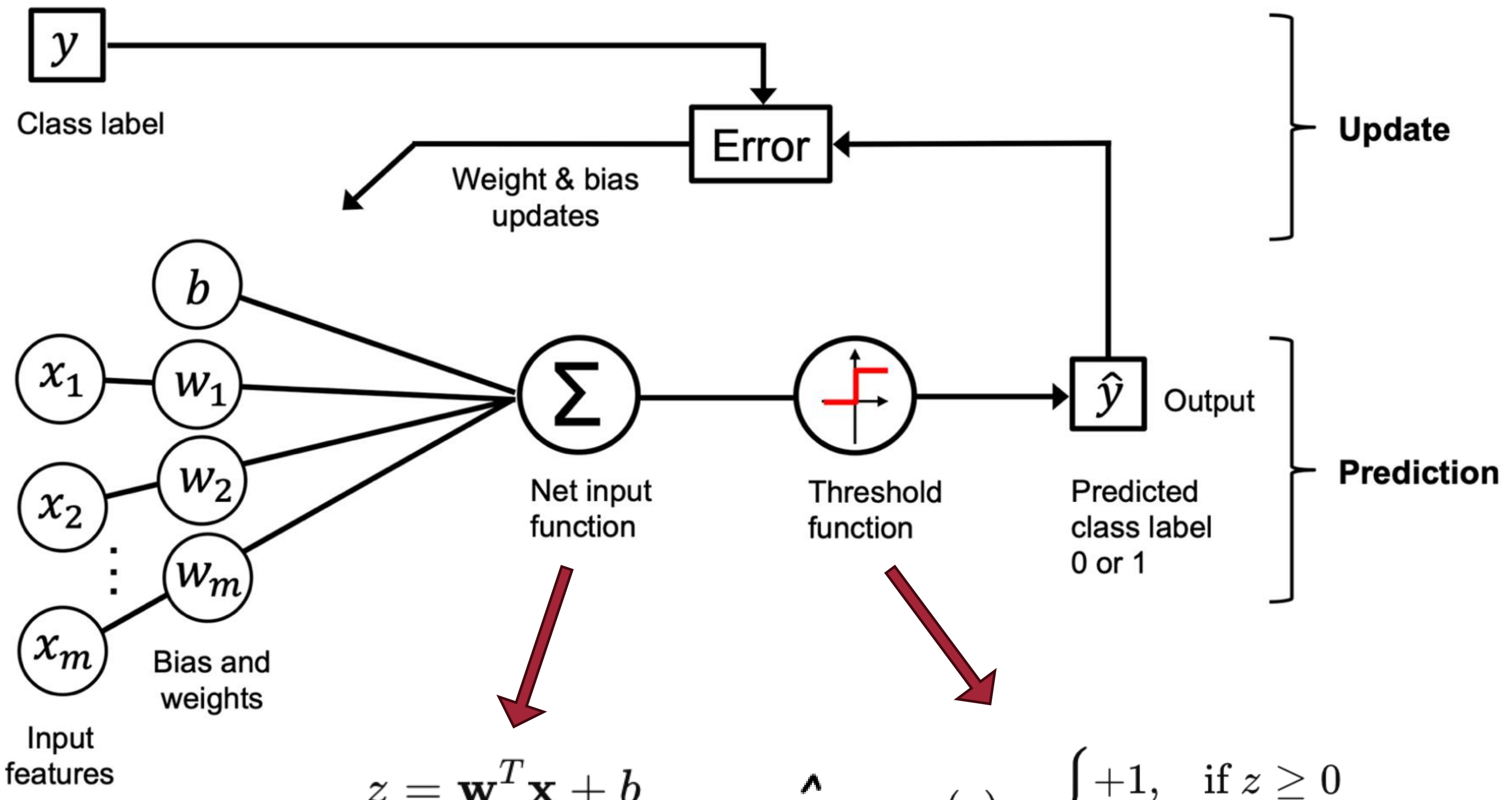


$$z = \mathbf{w}^T \mathbf{x} + b$$

$$\hat{y} = \operatorname{sgn}(z) = \begin{cases} +1, & \text{if } z \geq 0 \\ -1, & \text{if } z < 0 \end{cases}$$

Key idea to adjust the weight (and bias)

- If predicted label is 1, but the actual label is 0, we want to reduce the weight
- If predicted label is 0, but the actual label is 1, we want to enhance the weight



$$z = \mathbf{w}^T\mathbf{x} + b$$

$$\hat{y} = \mathrm{sgn}(z) = \begin{cases} +1, & \text{if } z \geq 0 \\ -1, & \text{if } z < 0 \end{cases}$$

The perceptron learning rule

1. Initialize the weights and bias unit to 0 or small random numbers
2. For each training example, $\boldsymbol{x}^{(i)}$:
   a. Compute the output value, $\hat{y}^{(i)}$
   b. Update the weights and bias unit

$$w_j := w_j + \Delta w_j$$
$$\text{and} \quad b := b + \Delta b$$

The update values ("deltas") are computed as follows:

$$\Delta w_j = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)x_j^{(i)}$$
$$\text{and} \quad \Delta b = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)$$

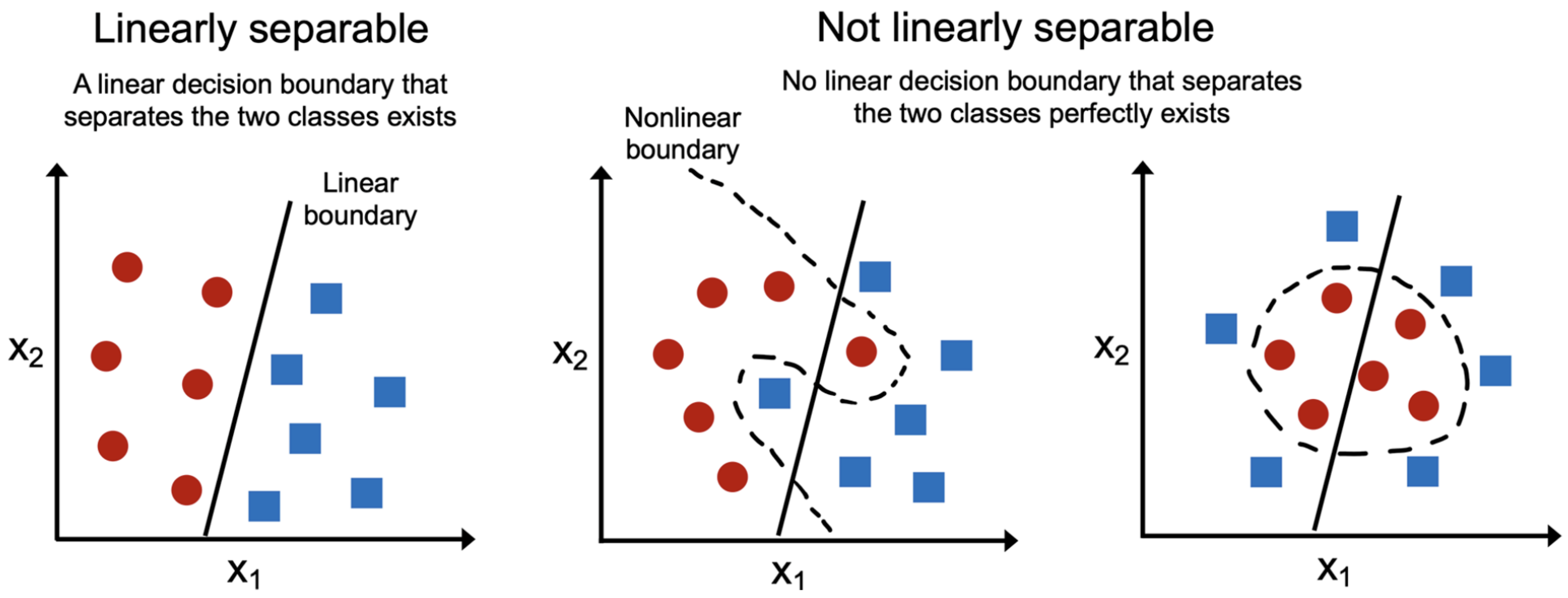
predicted class label

learning rate

actual class label

Applicable to linearly separable data only

- The algorithm finds the linear decision boundary after certain number of iterations (**epochs**)



Implementation

- See jupyter notebook