Segmentation with Machine Learning

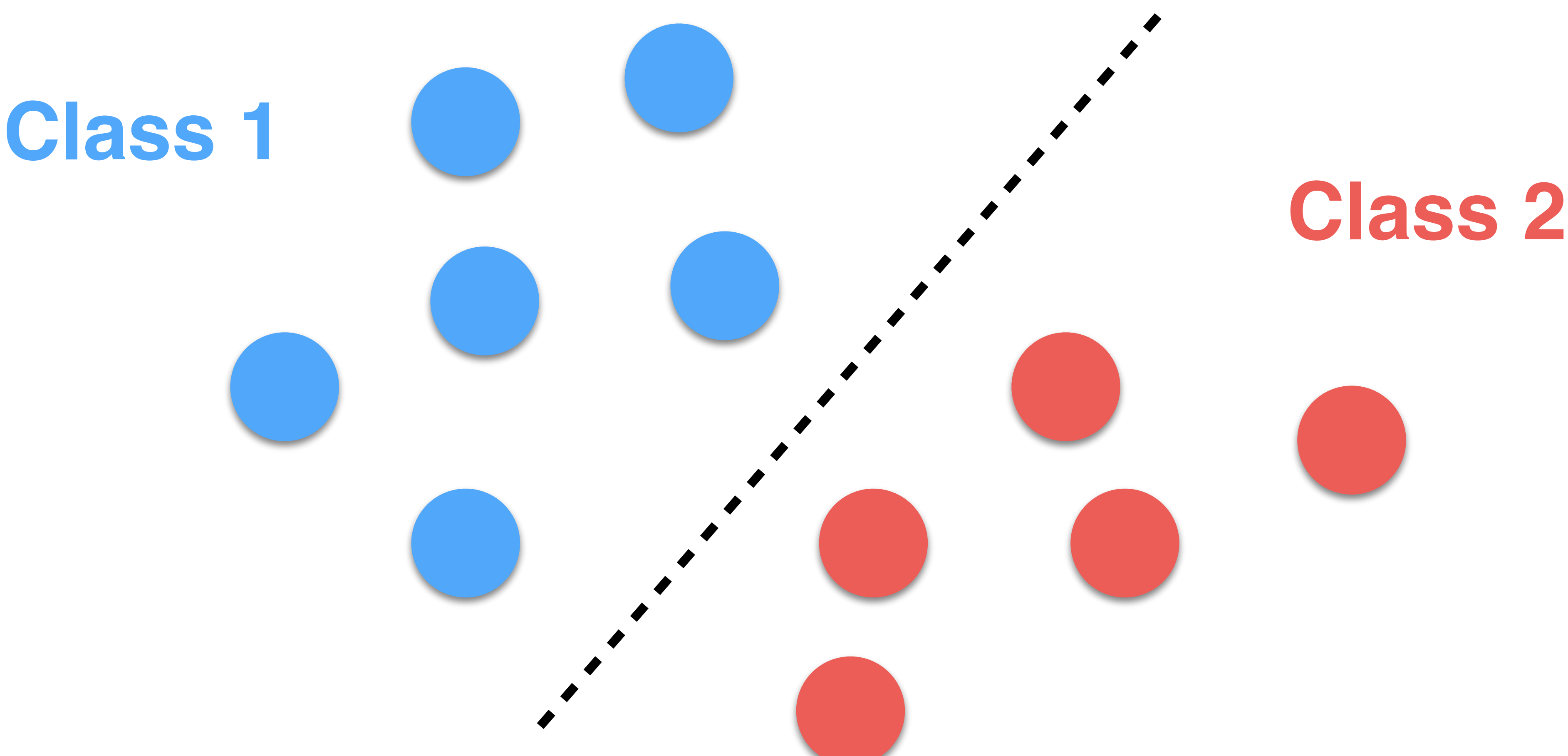
Francesco Banterle, Ph.D.
francesco.banterle@isti.cnr.it

Machine Learning

- Machine learning algorithms:
  - The use of computers algorithm that may improve automatically through **experience** and/or **the use of data**.
  - **Unsupervised**: we do not have labels.
  - **Supervised**: we have labelled data:
    - Neural Networks.

Machine Learning

- Machine learning algorithms work very well for classification: drawing a plane or hyperplane to divide samples into classes.
- Similarly to k-Means (**unsupervised**) this works for segmentation too!



Machine Learning

Training Set

Model

Learning Method

Machine Learning: Learning

- **Training set**: a dataset of n couples: input and output.
  - The larger the better:
    - at least 10,000 couples for high-quality segmentation.
  - This represents a **knowledge** to be trained. "*Learn by example*"; i.e., *supervised learning*.

Machine Learning: Learning

- **Learning Method**: a mathematical model/function that transfers the **knowledge** of the training set to the model:
  - It is a mix between:
    - Minimization method (i.e., Gradient Descent);
    - Loss function (how to minimize the differences).

Machine Learning: Learning

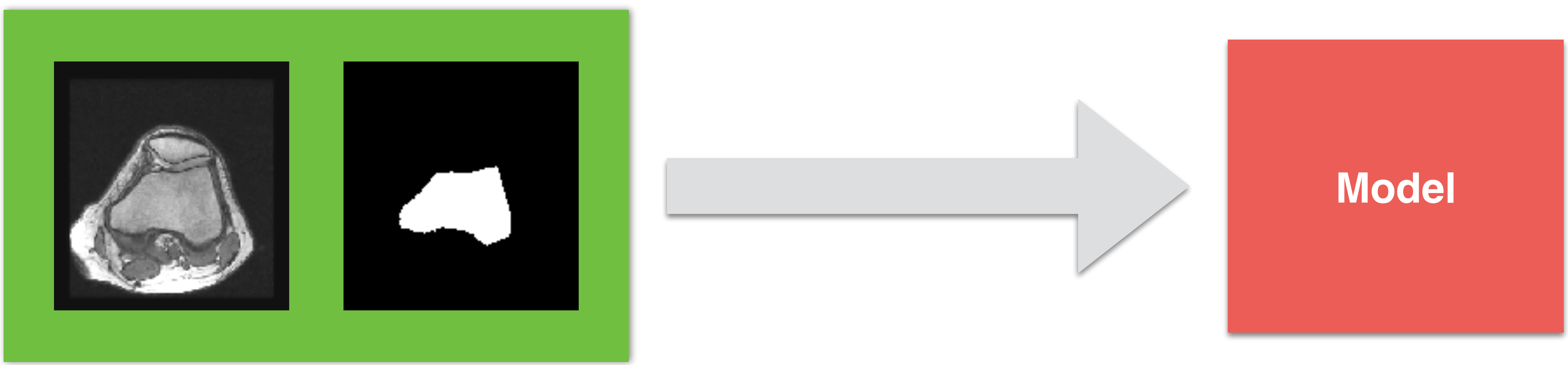
- **Model**: a mathematical model that can store the **knowledge** of the dataset into its parameters (called ***weights***).
- For example:
  - A neural network;
  - A decision tree/forest.

Machine Learning: Supervised Learning

- There are two steps:
  - Learning
  - Prediction/Evaluation

Machine Learning: Supervised Learning

- We need to collect examples and transfer that knowledge into a model.

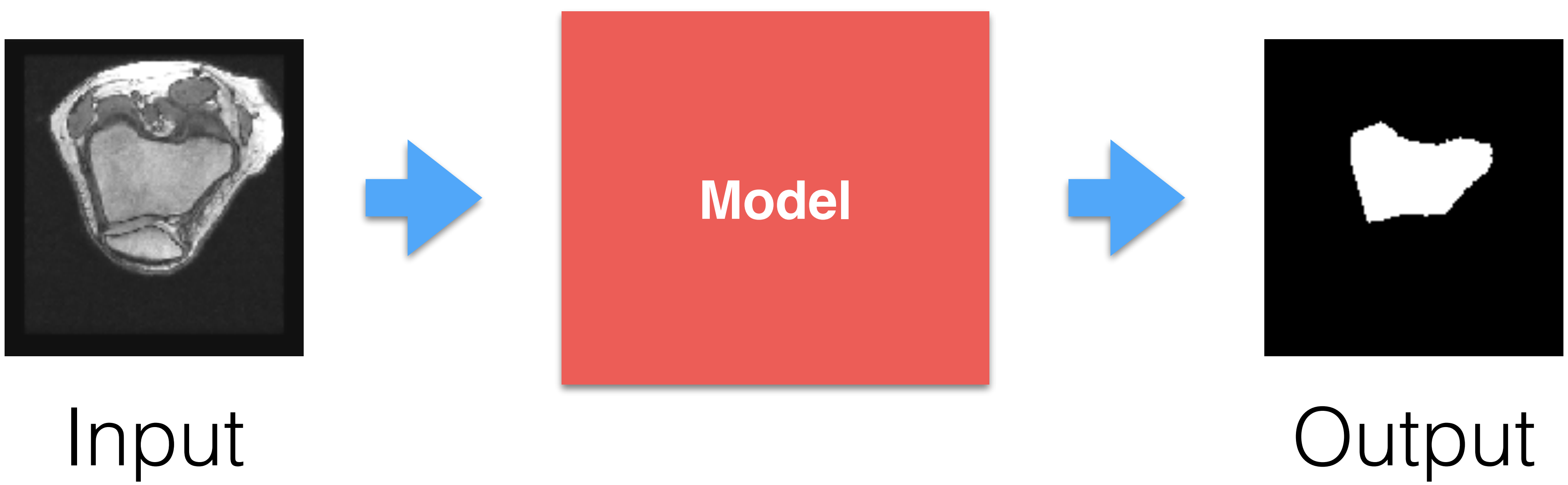


An example

Learning Method

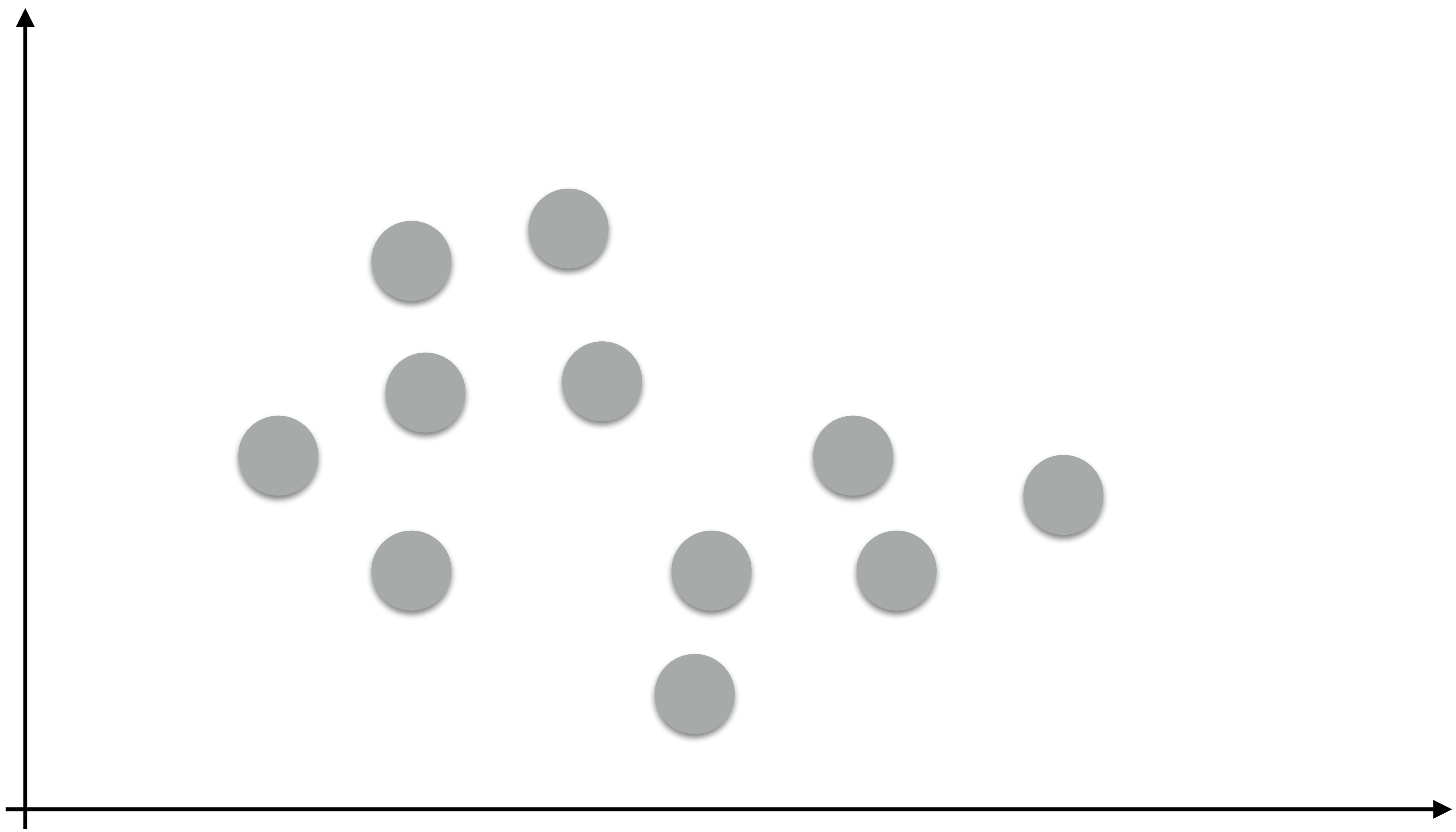
Machine Learning: Supervised Prediction/Evaluation

- After learning the dataset, we just need to pass data to the model (i.e., we evaluate it) to get results:

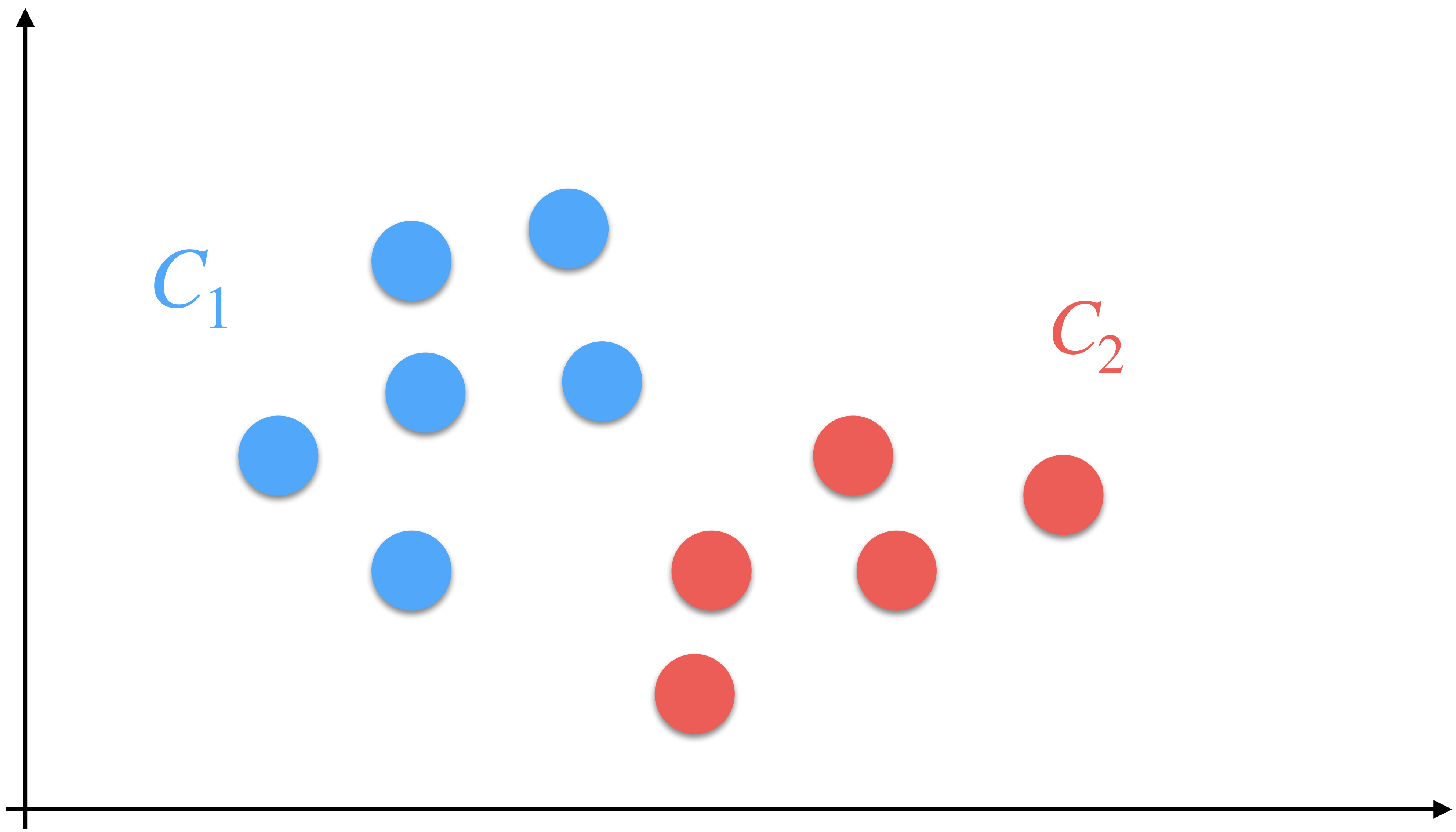


A Simple Example

Machine Learning: Binary Classification

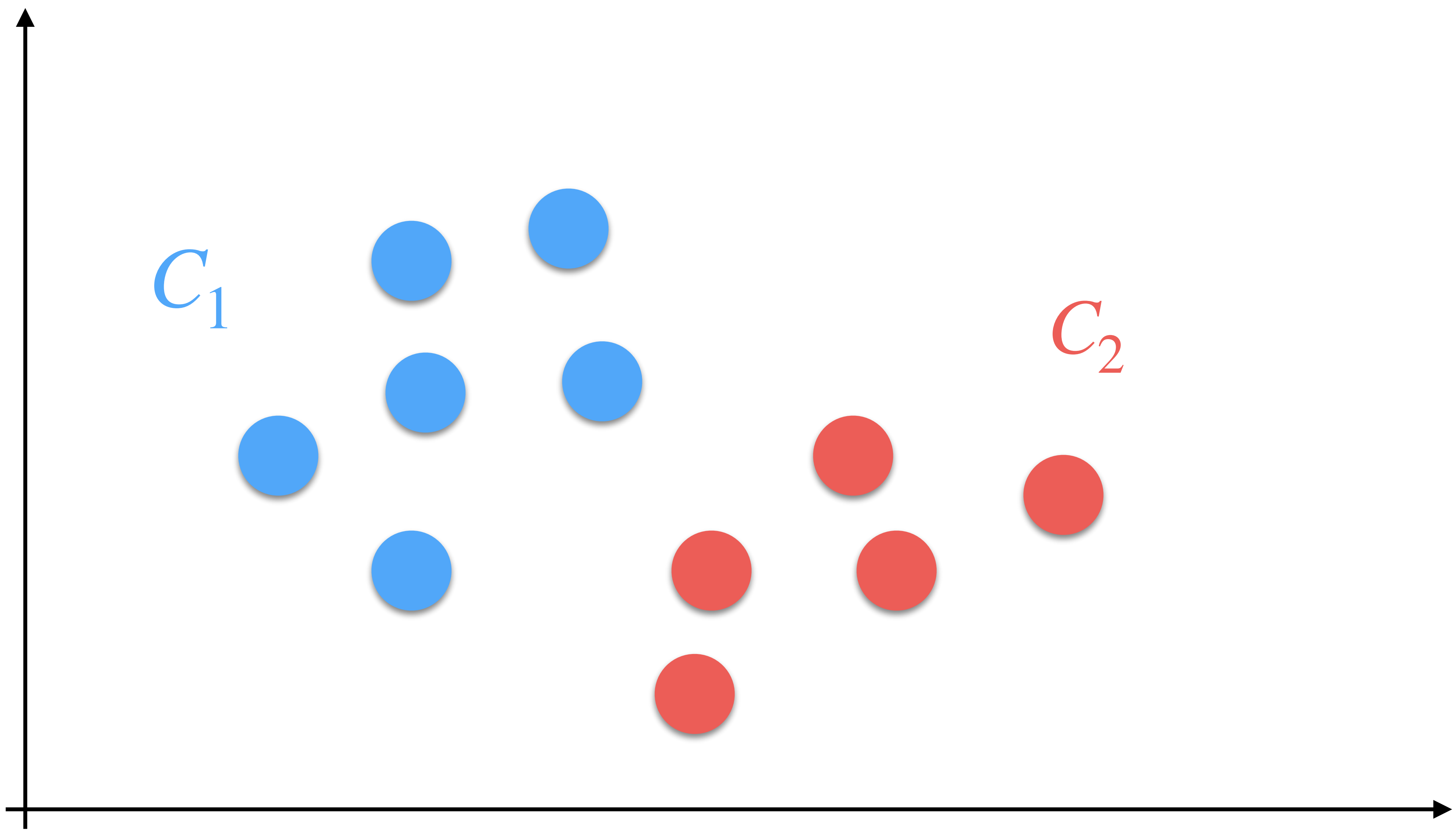


Machine Learning: Binary Classification



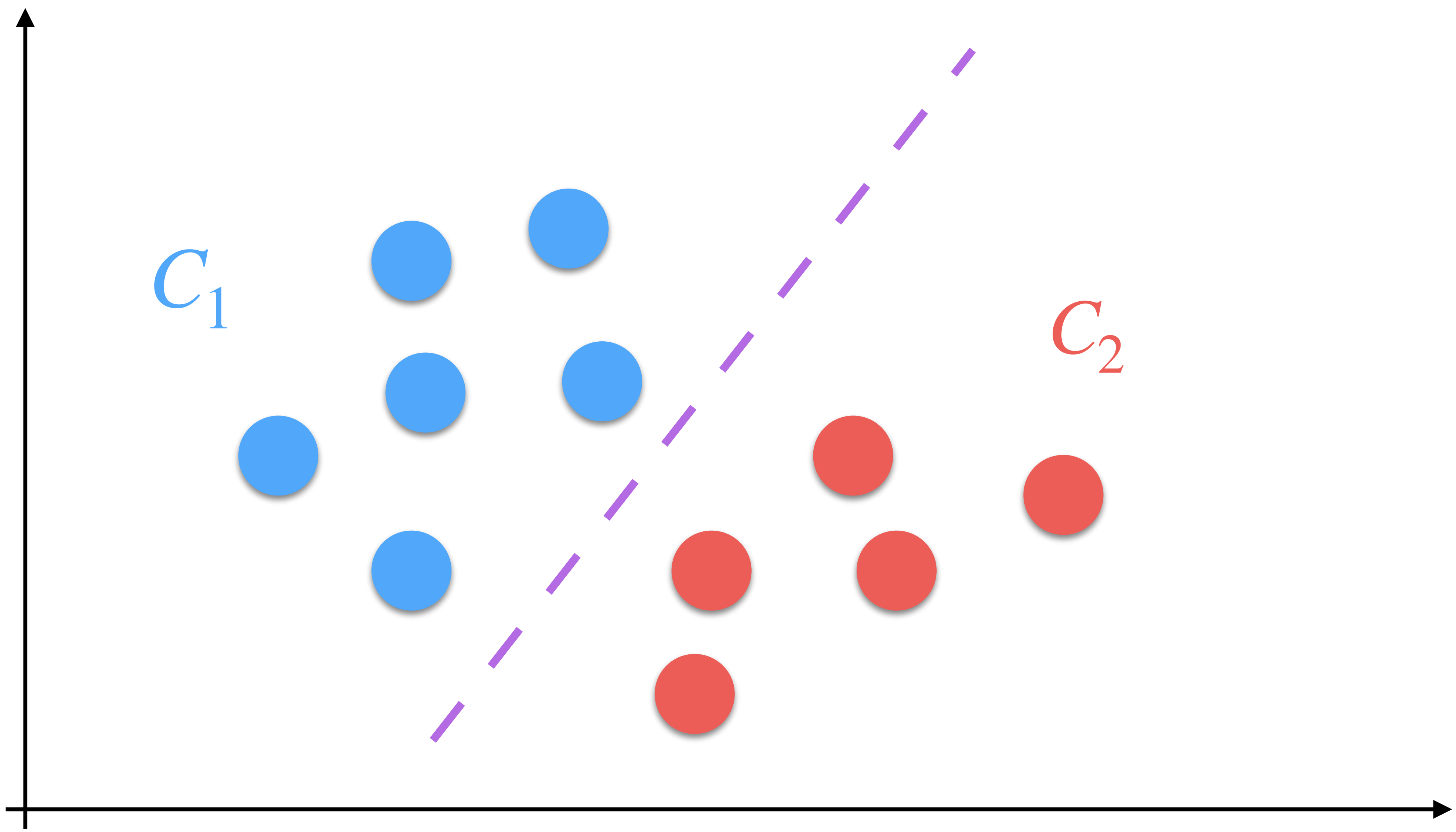
$$h : \mathbb{R}^n \to \{C_1, C_2\}$$

Machine Learning: Binary Classification



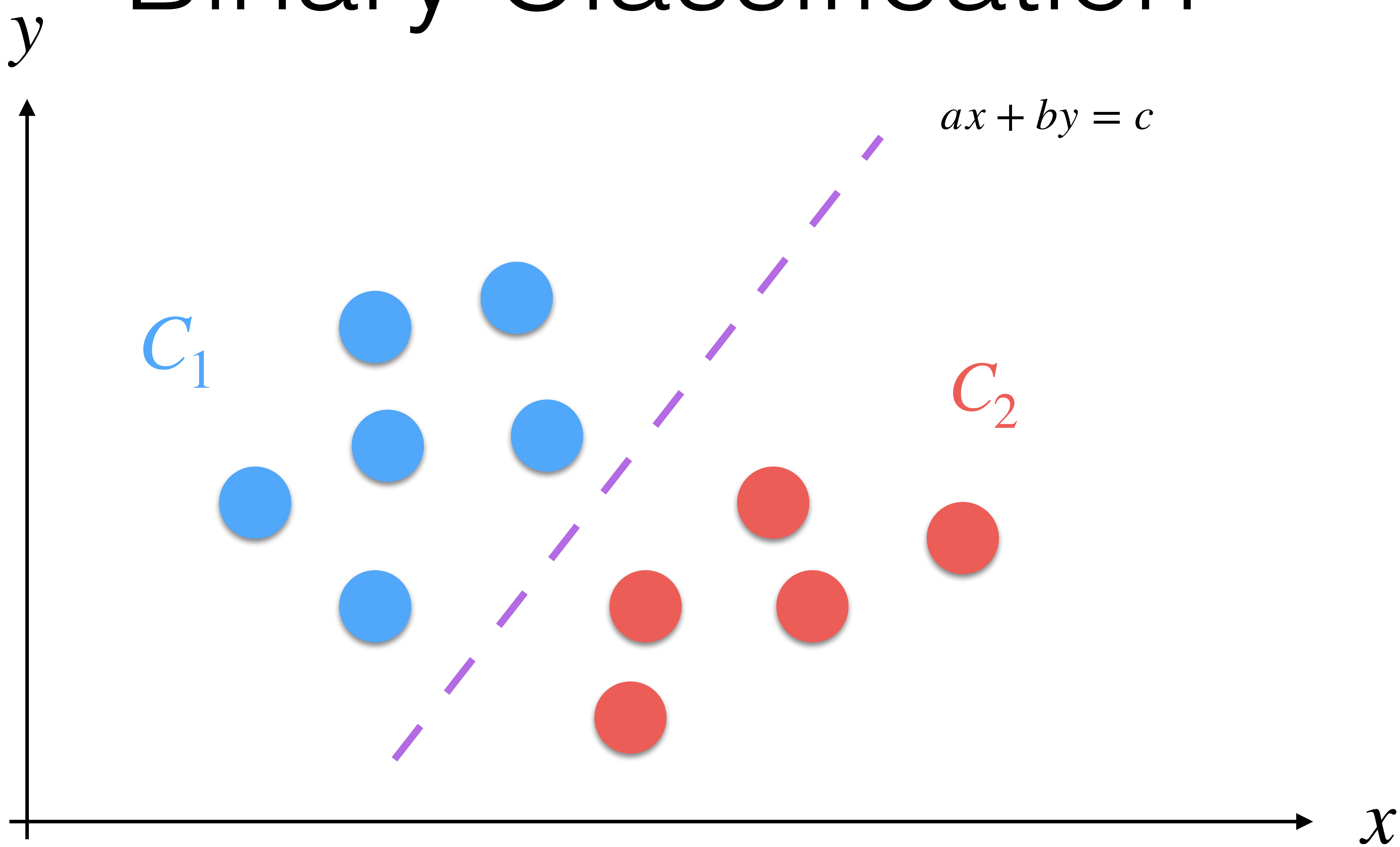
$$h : \mathbb{R}^2 \to \{0,1\}$$

Machine Learning: Binary Classification



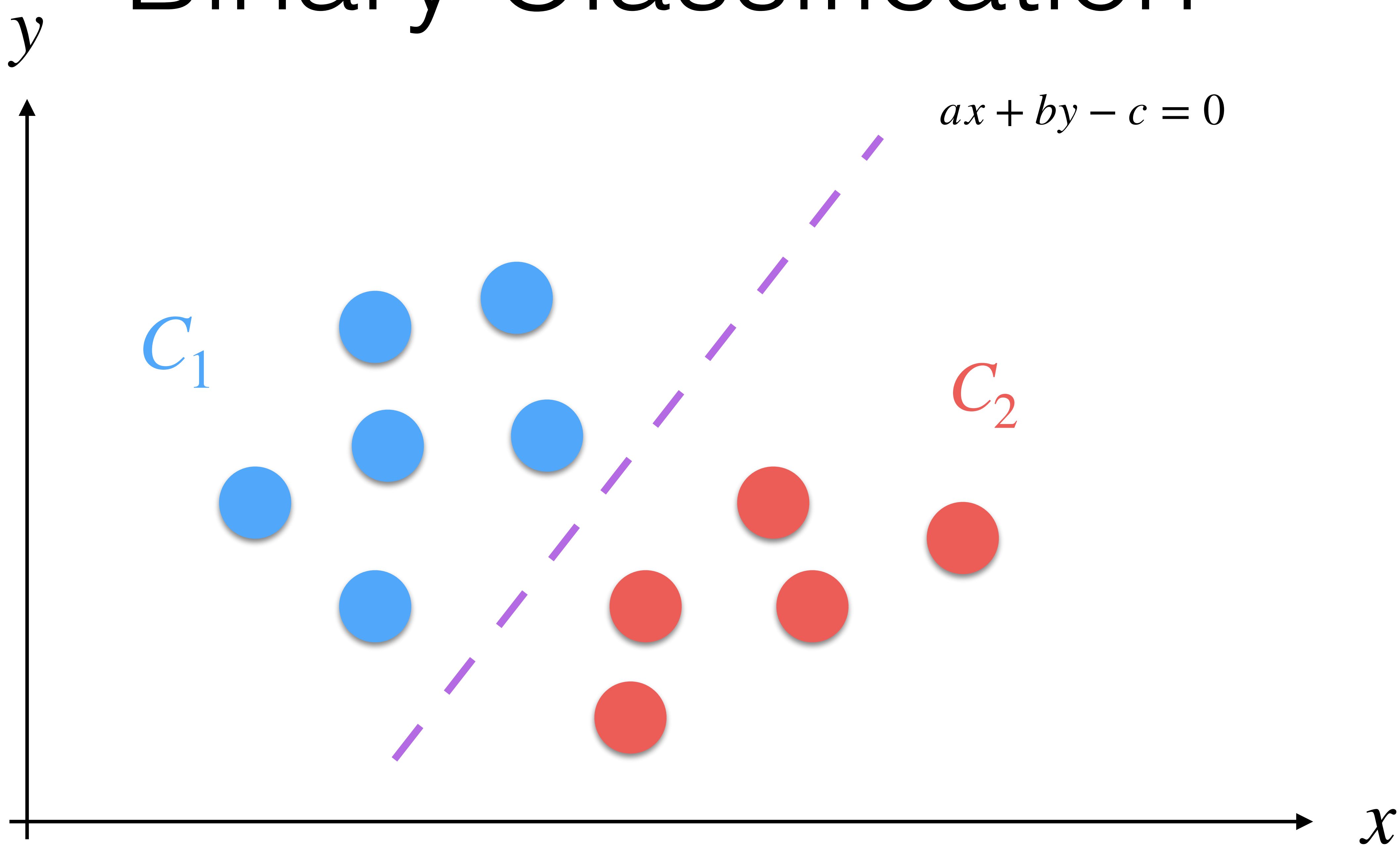
$$h : \mathbb{R}^2 \to \{0,1\}$$

Machine Learning: Binary Classification



$$h : \mathbb{R}^2 \rightarrow \{0,1\}$$

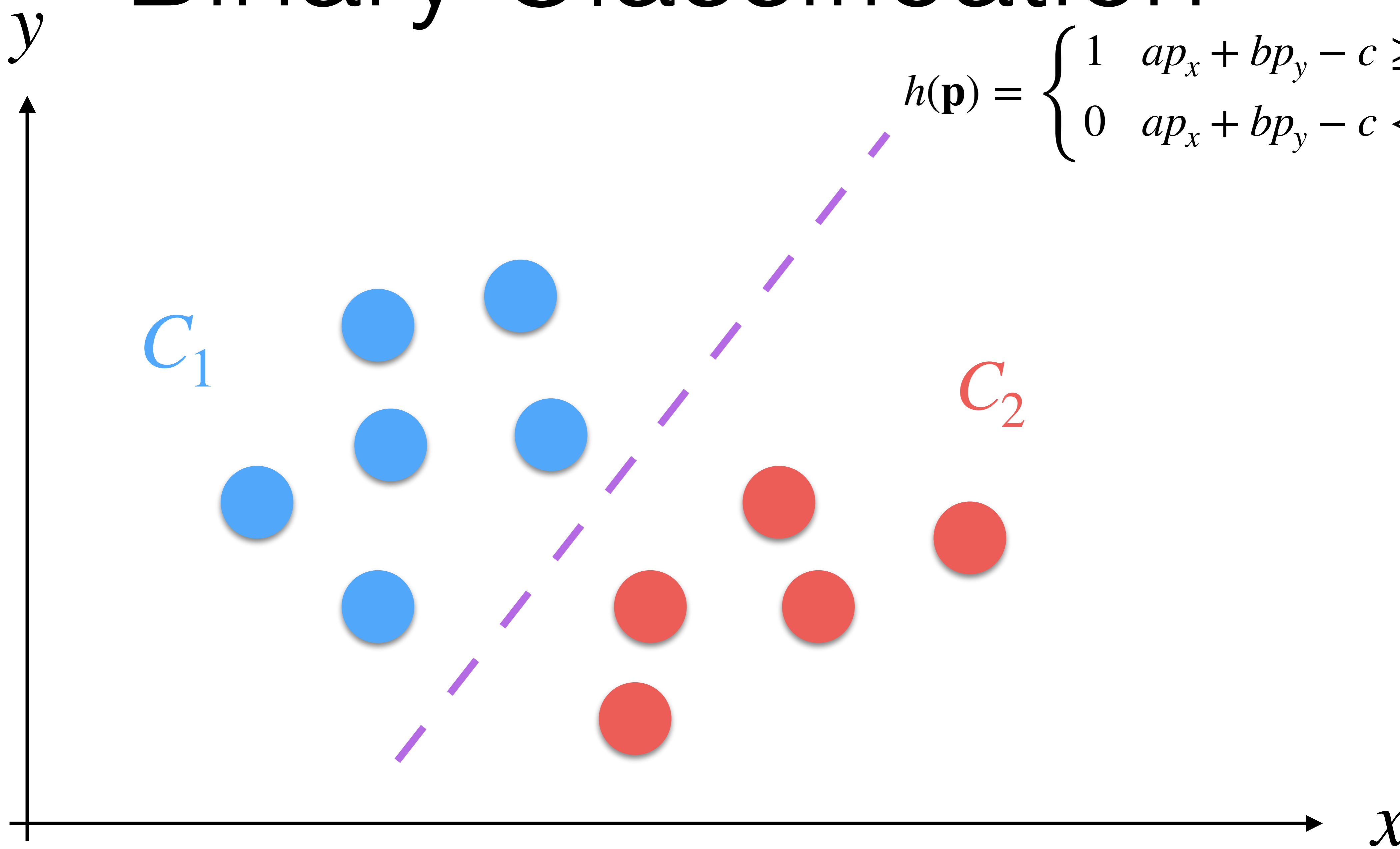
Machine Learning: Binary Classification



$$h : \mathbb{R}^2 \to \{0,1\}$$

Machine Learning: Binary Classification

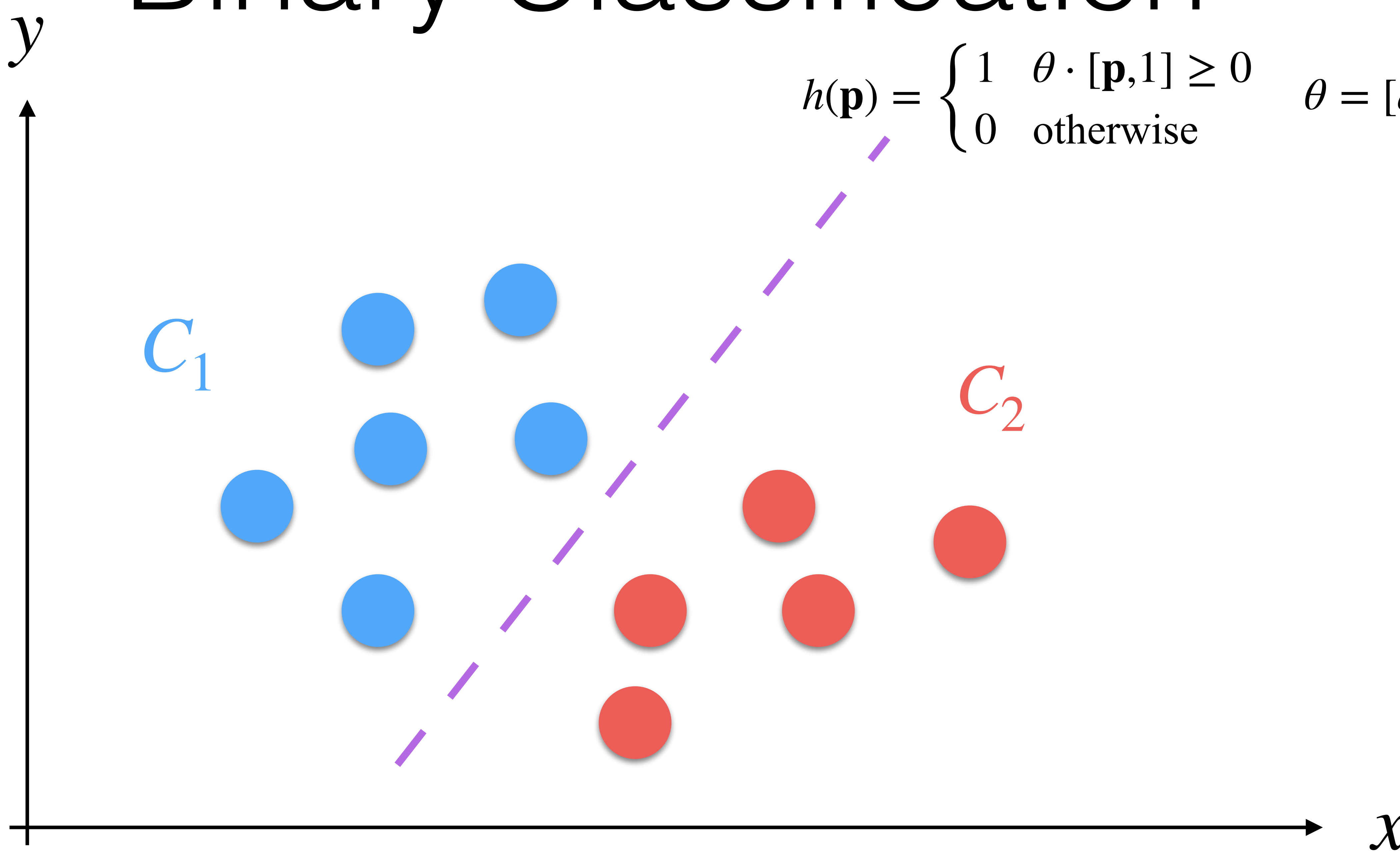
$$h(\mathbf{p}) = \begin{cases} 1 & ap_x + bp_y - c \geq 0 \\ 0 & ap_x + bp_y - c < 0 \end{cases}$$



$$h : \mathbb{R}^2 \rightarrow \{0,1\}$$

Machine Learning: Binary Classification

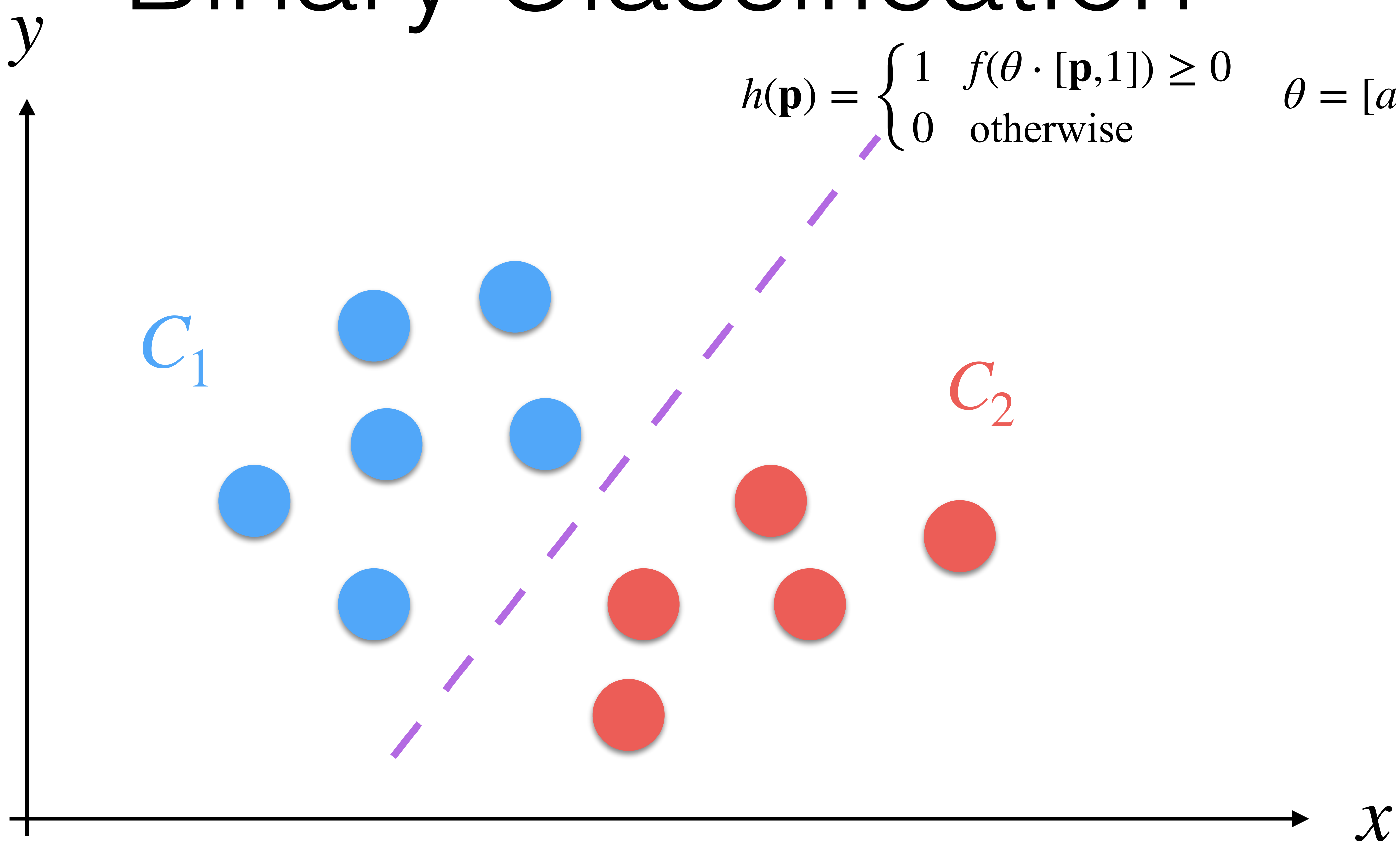
$$h(\mathbf{p}) = \begin{cases} 1 & \theta \cdot [\mathbf{p}, 1] \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad \theta = [a, b, -c]$$



$$h : \mathbb{R}^2 \rightarrow \{0,1\}$$

Machine Learning: Binary Classification

$$h(\mathbf{p}) = \begin{cases} 1 & f(\theta \cdot [\mathbf{p},1]) \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad \theta = [a, b, -c]$$



$$h : \mathbb{R}^2 \rightarrow \{0,1\}$$

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

Machine Learning: The Activation Function

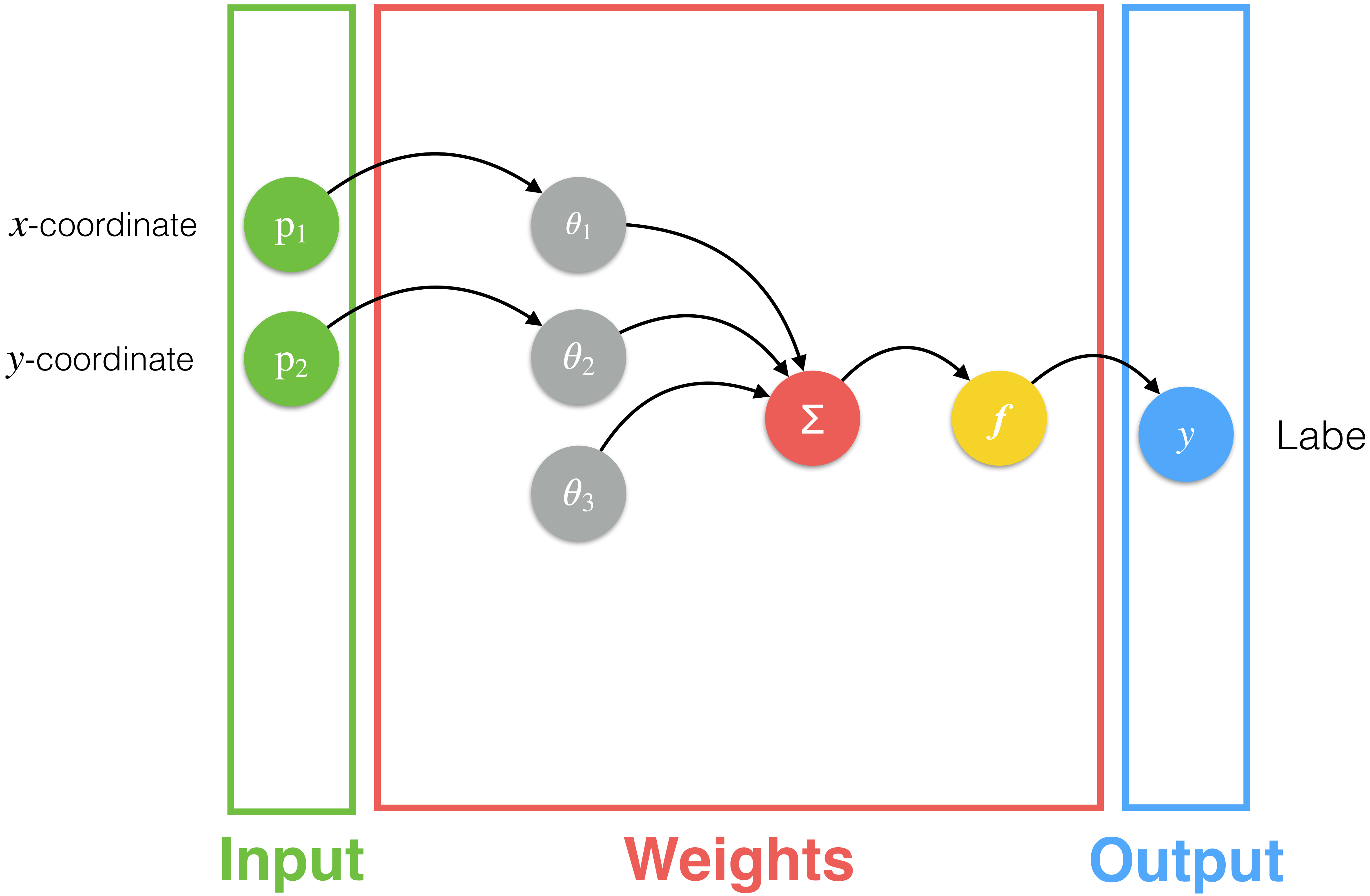
- To add non-linear effect to h, we apply a non-linear function f that is called the ***activation function***.

- It can be defined in many ways. For example:

$$f(z) = \frac{1}{1+e^{-z}} \qquad f(z) = \max\left(0,z\right)$$

- This is because the result has to be either belonging or not to a class; i.e., our area of interest.

Neural Networks: Our Model h



Machine Learning: Neural Networks

- The idea is to try to "*mimic the neurons*" in our brains:
  - A neuron receives multiple inputs or stimuli, that we can represent as a vector $\mathbf{p}$.
  - Depending on previous knowledge, θ , a neuron can react to $\mathbf{p}$, and if the stimulus is strong enough there is an activation
  - The reaction to stimuli is typically modeled as a dot product between $\mathbf{p}$ and θ. Plus the activation function to handle non-linearities.

Neural Networks: Supervised Learning

- We need to collect m couples $(\mathbf{x}^i, y^i)$.
- We need to minimize an error function J:

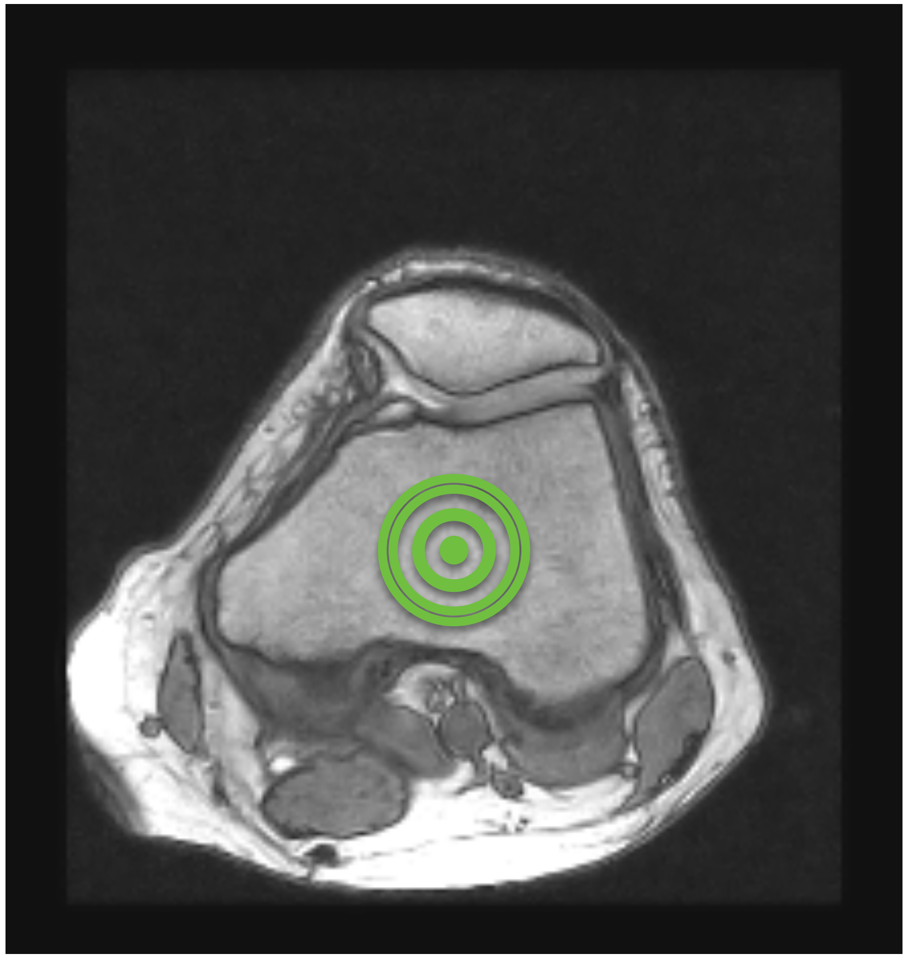
$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h(\mathbf{x}^i, \theta) - y^i\right)^2$$

- How do we minimize it?
  - Gradient descent
  - Starting solution for θ?
    - Random values in $[-1, 1]$.

A Segmentation Example

Neural Networks: Dataset Set (1)

Input



$\mathbf{p} = \{100, 100, 0.67\}$

Output

$y = 1$

Neural Networks: Dataset Set (2)

Input

$\mathbf{p}$ = {20,20,0.0}

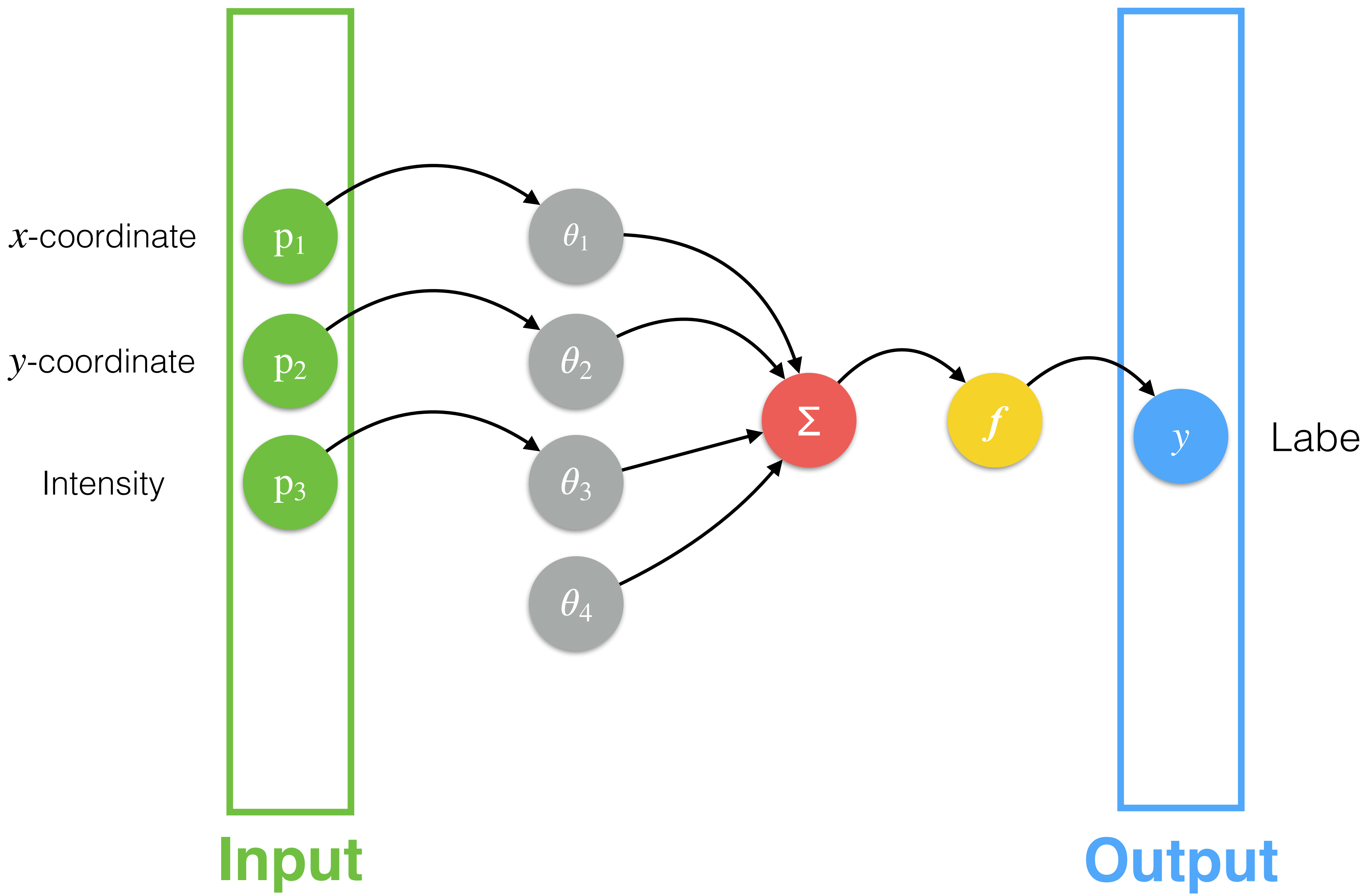
Output

y = 0

Machine Learning: Dataset Set (3)

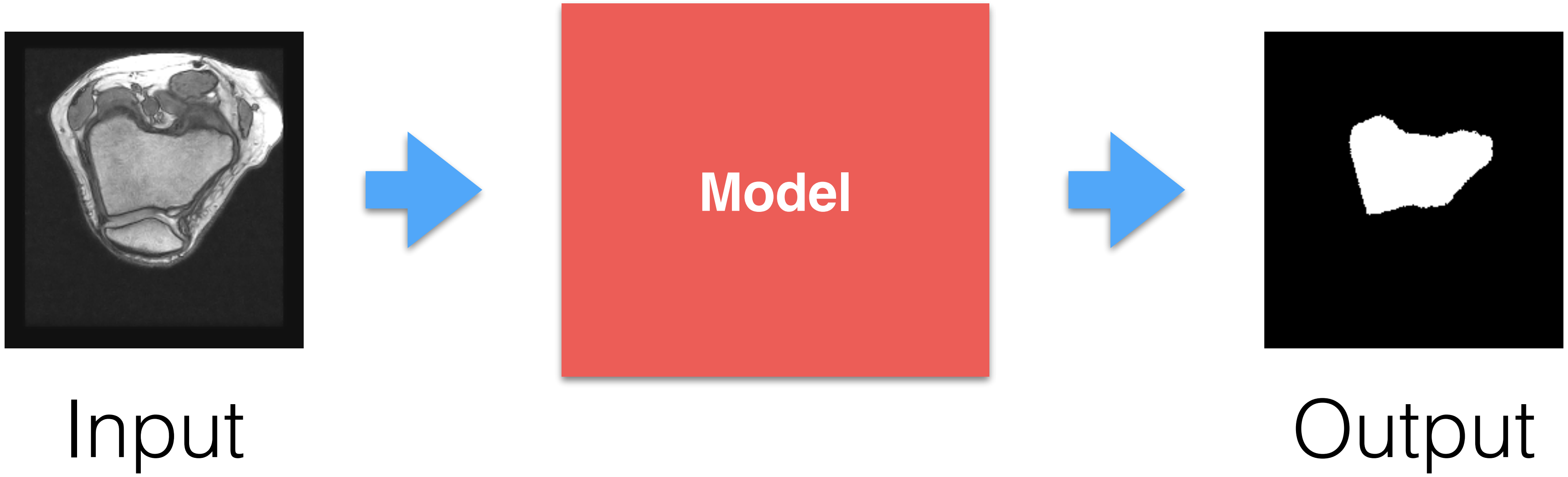
- The dataset needs to be balanced:
  - The same amount of examples for both classes: ROI and background.
- The dataset needs to be divided into:
  - Training set —> samples to train the network
  - Evaluation set —> samples to check if the model is not overfitting or under fitting.

Neural Networks: Training Phase

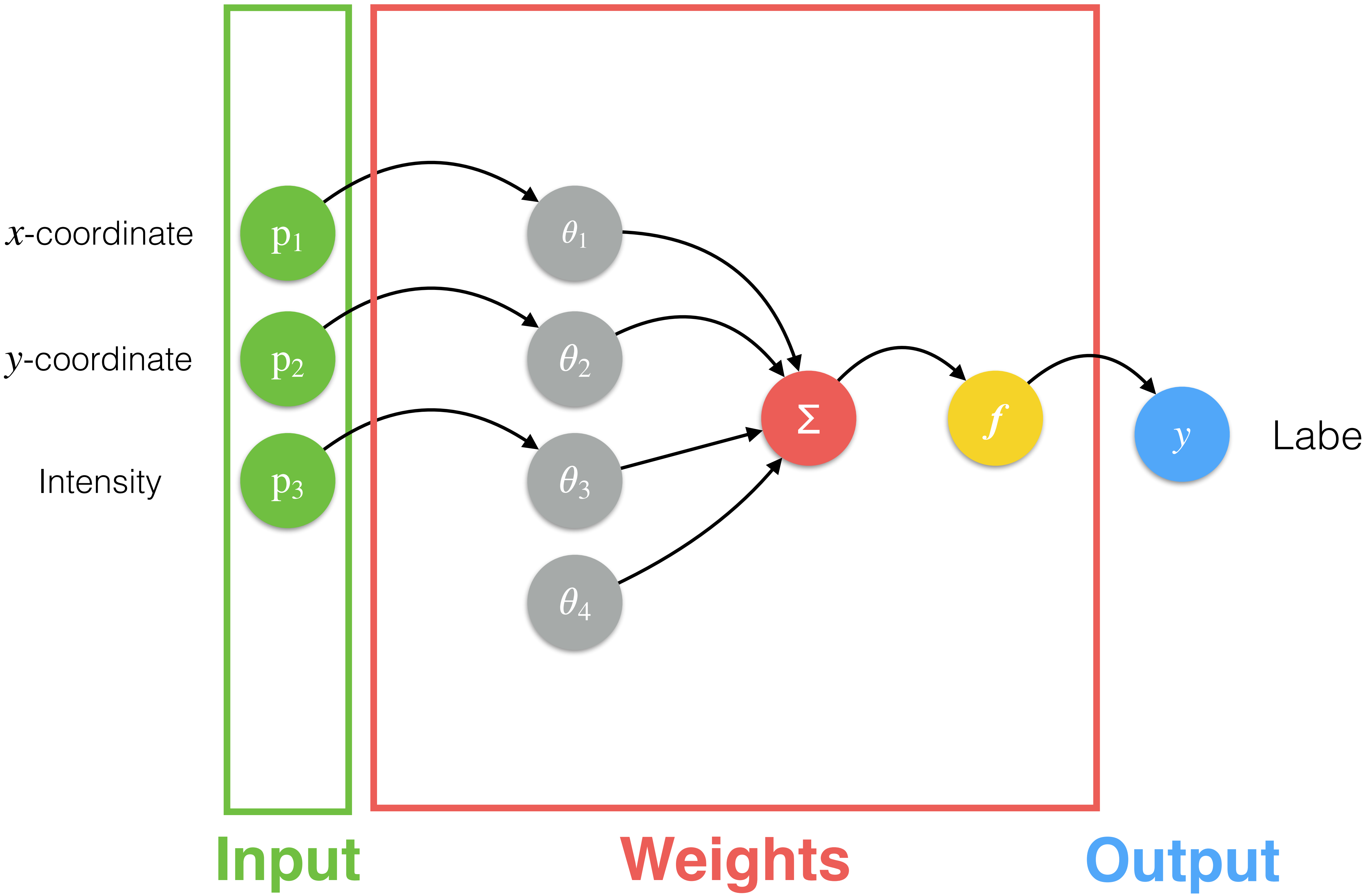


Machine Learning: Prediction Phase

- After learning, we can use our network on new images to segment the image:



Neural Networks: Prediction Phase



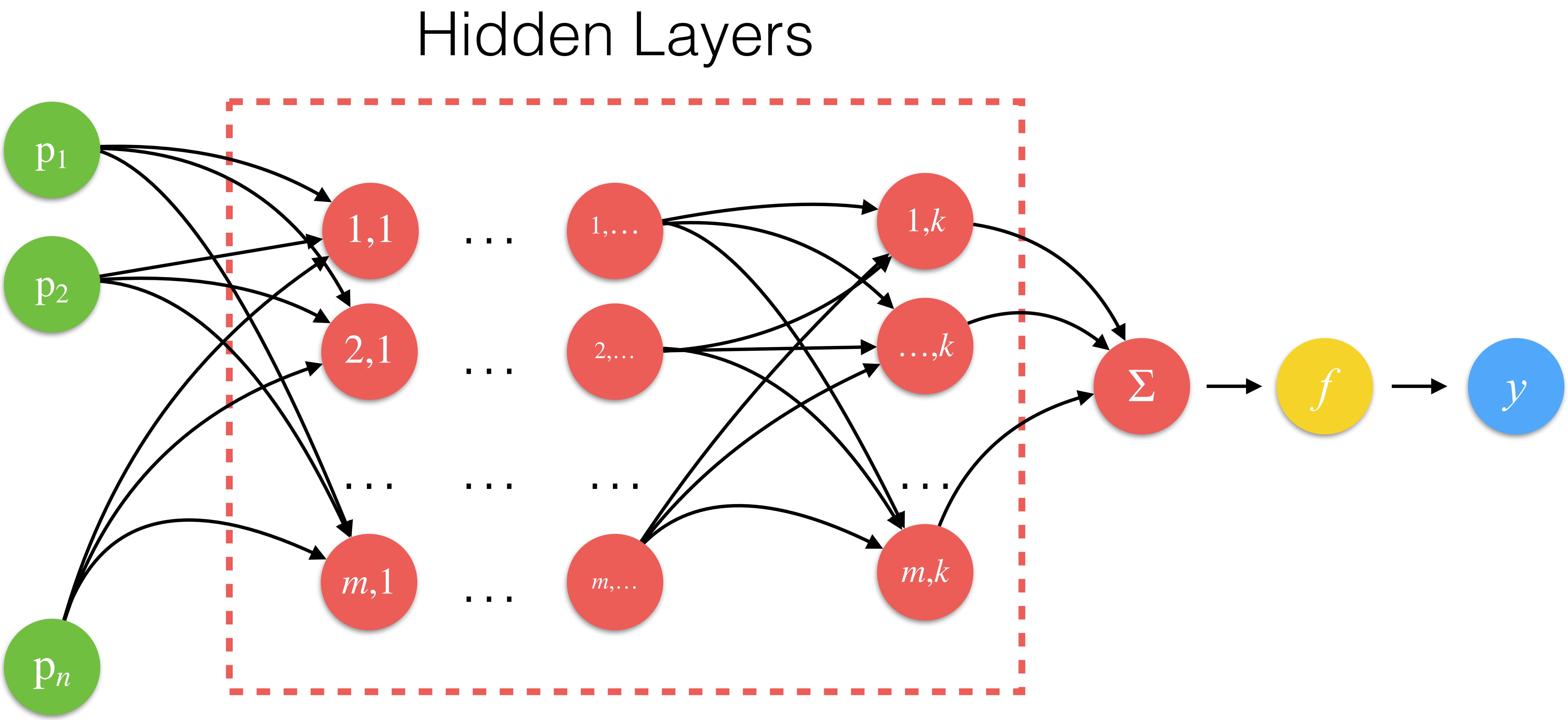
More Complex Examples

More Complex Nets

- To achieve high-quality results, a network needs to "see" and "understand" more data at the same time; not only a couple such as the pixel coordinates and its pixel intensity and its classification as in the previous example!

- We need to use more pixels/voxels at the same time:
    - How?
        - Adding and mixing more neurons

Neural Networks: Bigger Networks



$$y = h^i(\mathbf{p}, \theta)$$

Neural Networks

- Advantages:
  - fully automatic!
  - computationally fast to evaluate (not the learning though); especially using GPUs.
- Disadvantages:
  - they required many many examples:
    - more than 1,000 to get some decent results;
    - better >10,000 training example!

that’s all folks!