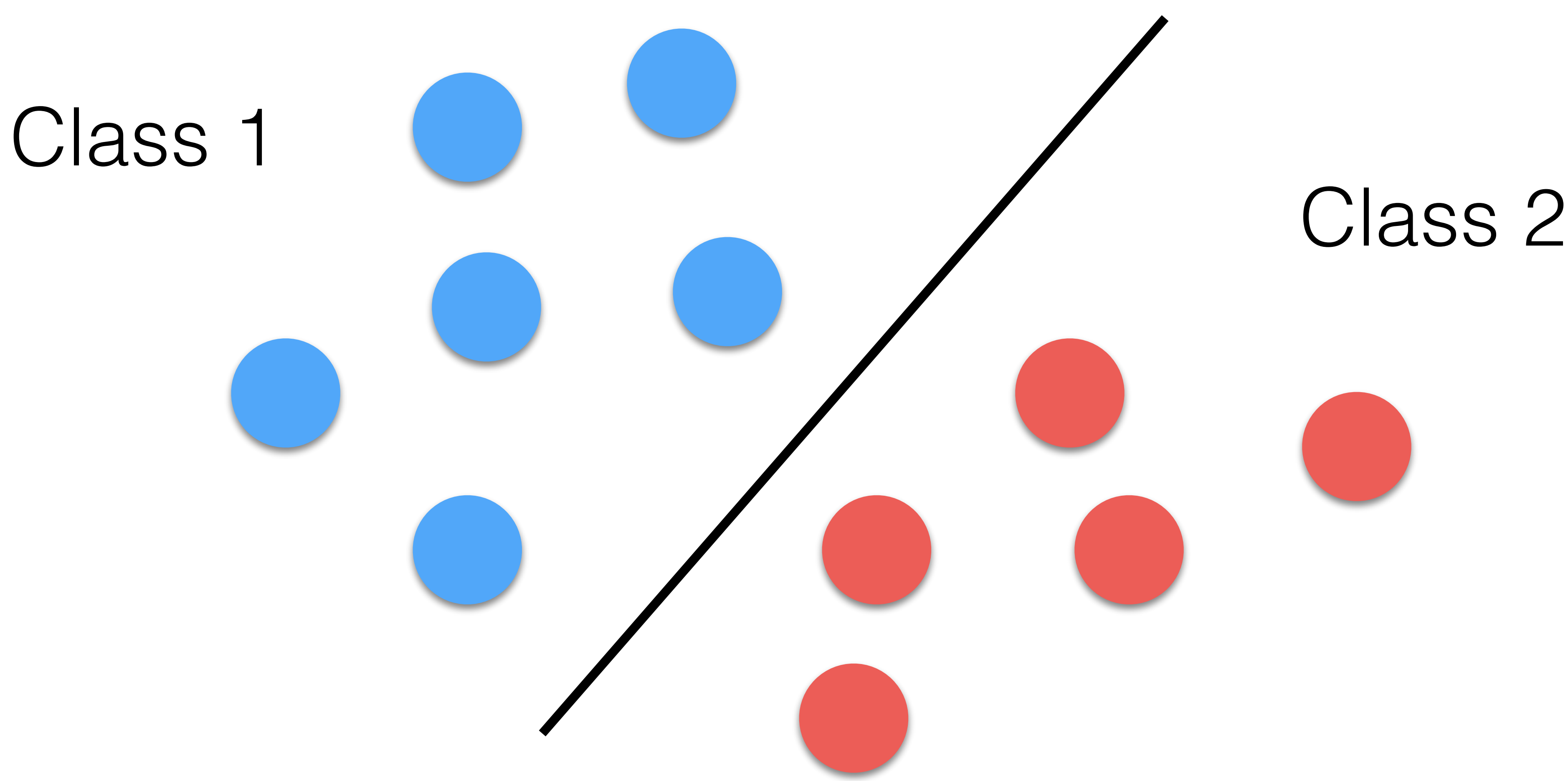
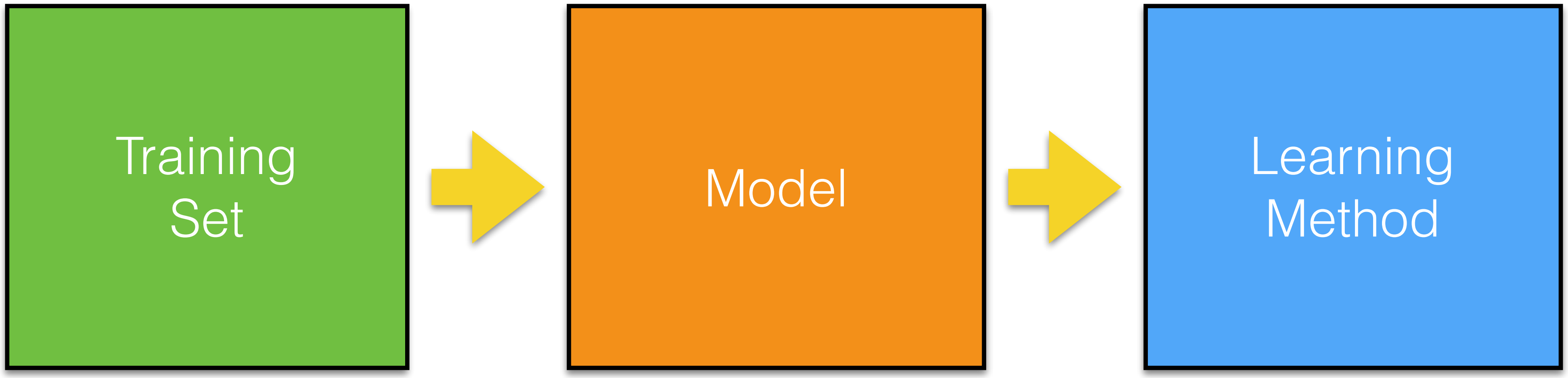
Segmentation with Machine Learning

Machine Learning

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



Machine Learning



Machine Learning

- **Training set**: a dataset of n couples: input and output. The bigger 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.

- **Learning Method**: an algorithm that transfers the ***knowledge*** of the training set to the model.

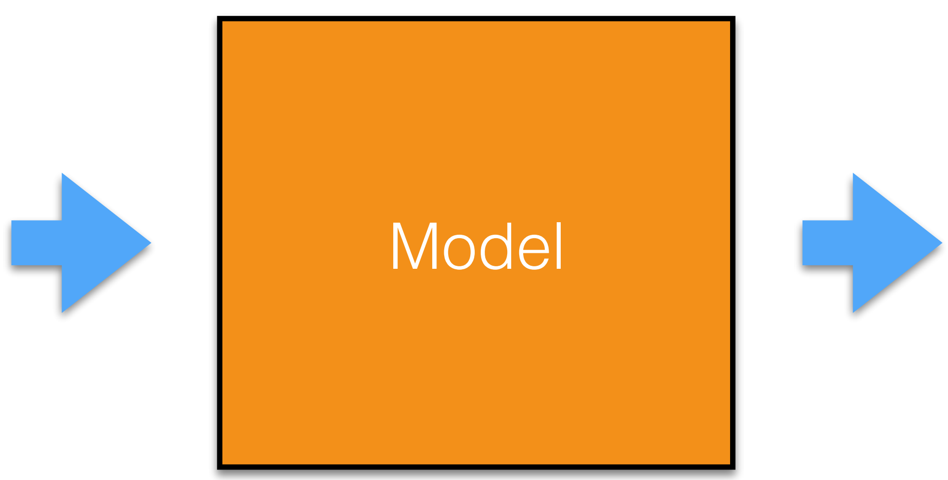
- **Model**: a mathematical model that can store the ***knowledge*** of the dataset into its parameters (called ***weights***).

Machine Learning

- There are two steps:
  - The first step, called **learning**, where the model has to be learnt using a dataset (input and output);
  - The second step, called **evaluation**, in which we give in input to the trained model a novel input (not in the dataset).

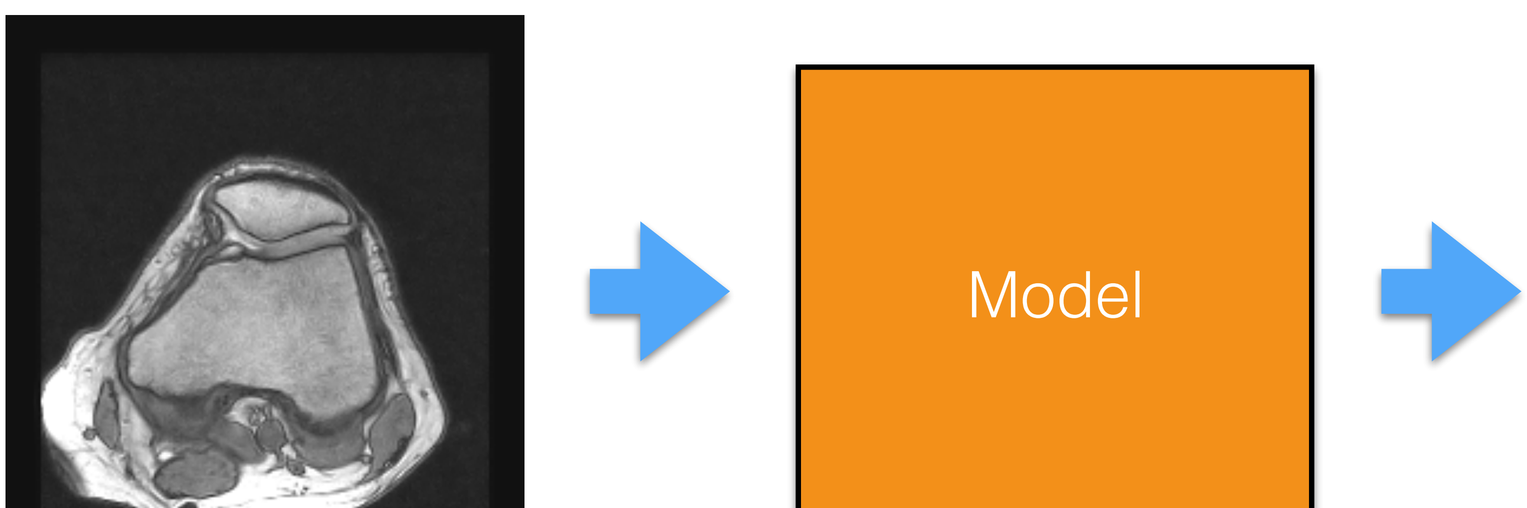
Machine Learning: Evaluation

- After the mode has learnt the dataset using a learning method. We just need to pass data to the model (i.e., we evaluate it) to get results!



Machine Learning: Evaluation

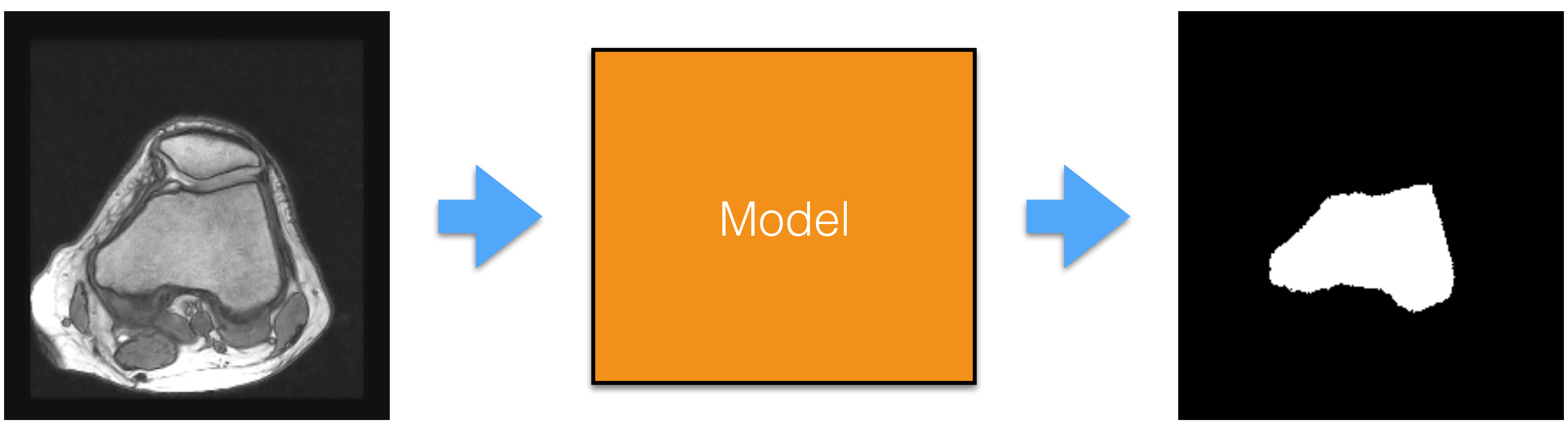
- After the mode has learnt the dataset using a learning method. We just need to pass data to the model (i.e., we evaluate it) to get results!



Input

Machine Learning: Evaluation

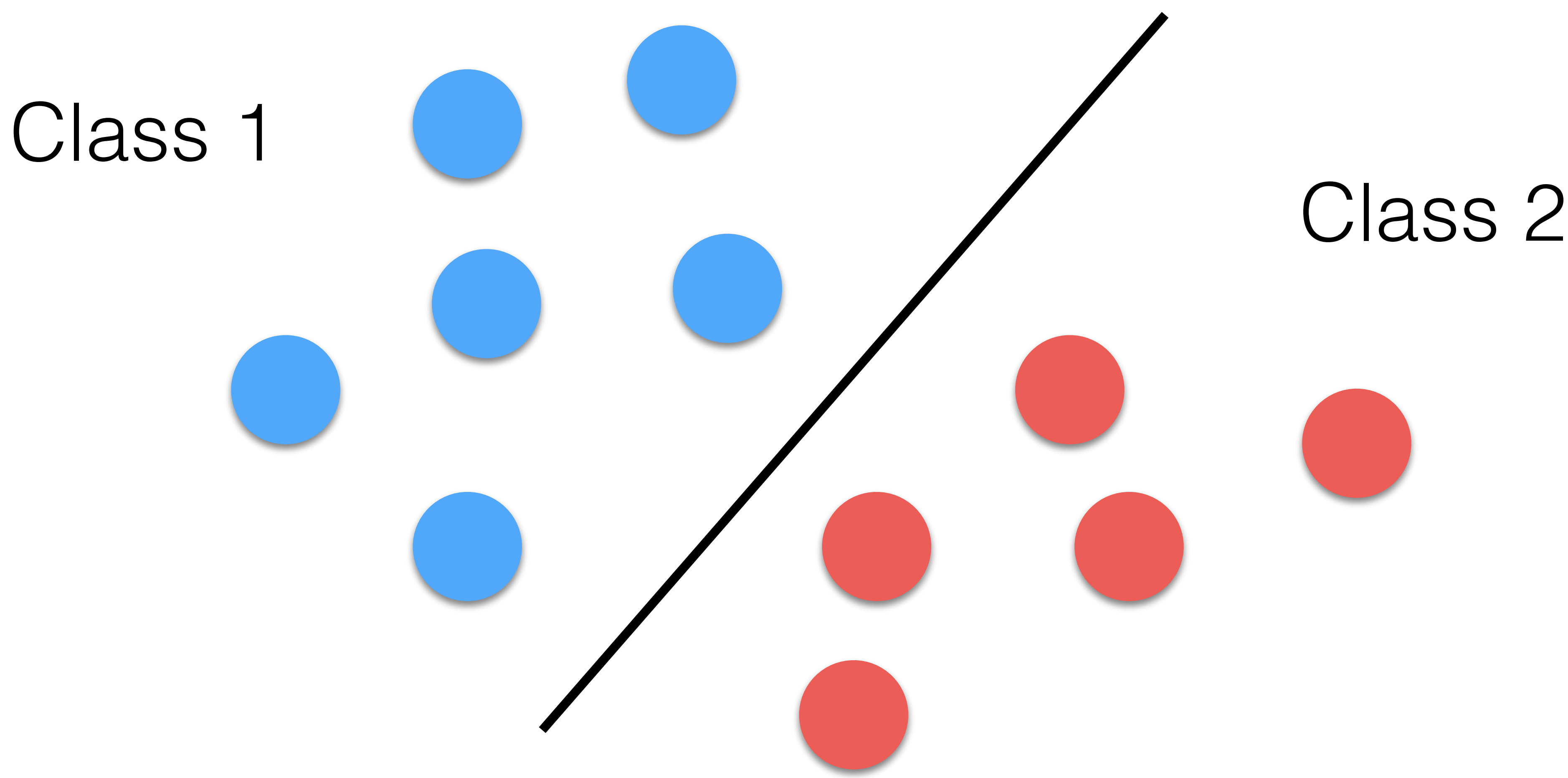
- After the mode has learnt the dataset using a learning method. We just need to pass data to the model (i.e., we evaluate it) to get results!



Input

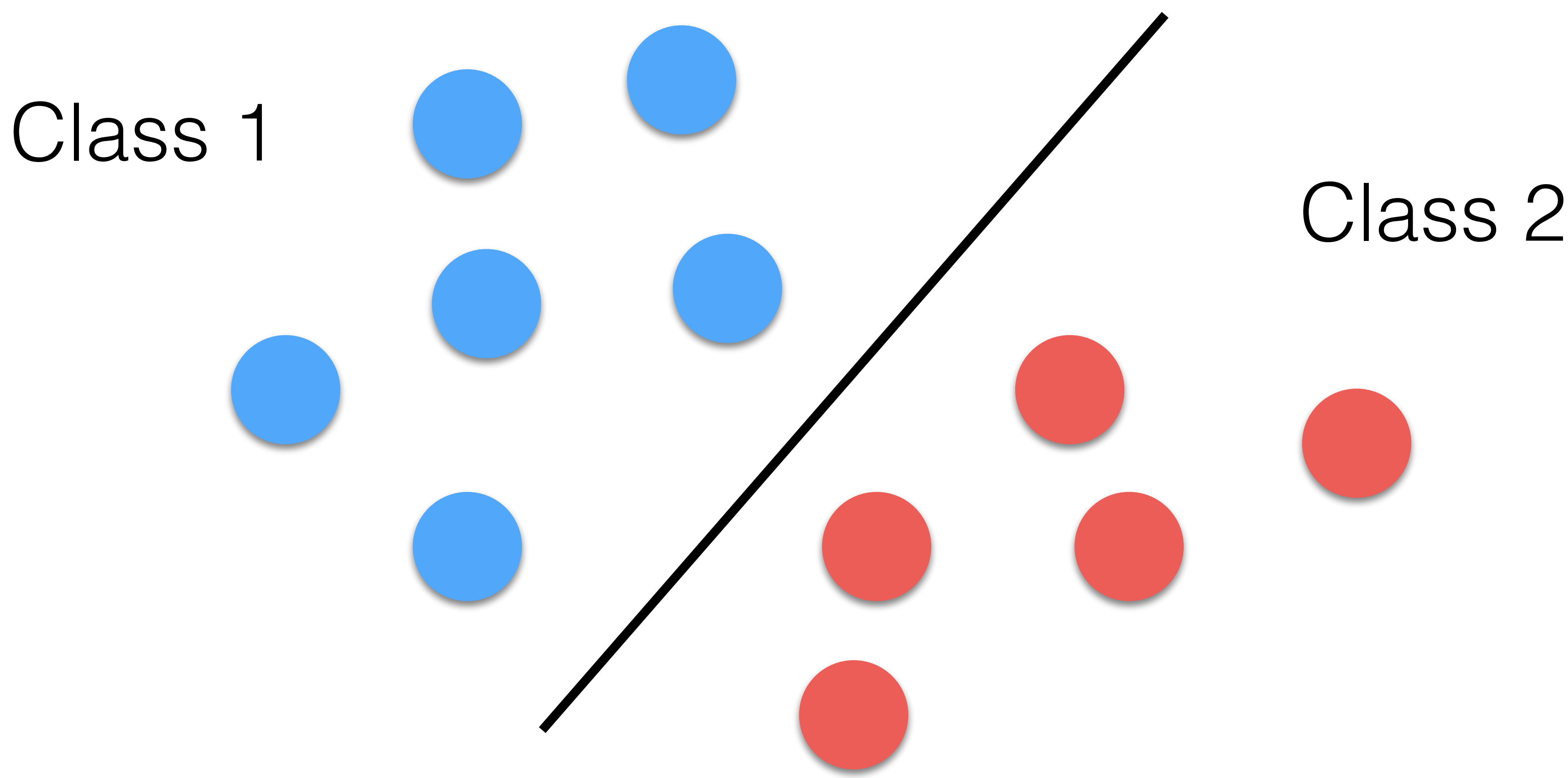
Output

Machine Learning



$$h : R^n \longrightarrow \{0, 1\}$$

Machine Learning



$$h(\mathbf{x}) = 1 \text{ if } \Theta\ \mathbf{x} + \mathrm{b} >= 0$$
$$h(\mathbf{x}) = 0 \qquad \text{otherwise.}$$

Machine Learning: Neural Networks

- The idea is to “mimic the neurons” in our brains:
  - A neuron receives multiple inputs or stimuli, that we can represent as a vector $\mathbf{x}$.
  - Depending on previous knowledge, $\boldsymbol{\Theta}$, a neuron can react to $\mathbf{x}$, and if the stimulus is strong enough there is an activation
  - The reaction to stimuli is typically modeled as a dot product between $\mathbf{x}$ and $\boldsymbol{\Theta}$.

Neural Networks: 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:

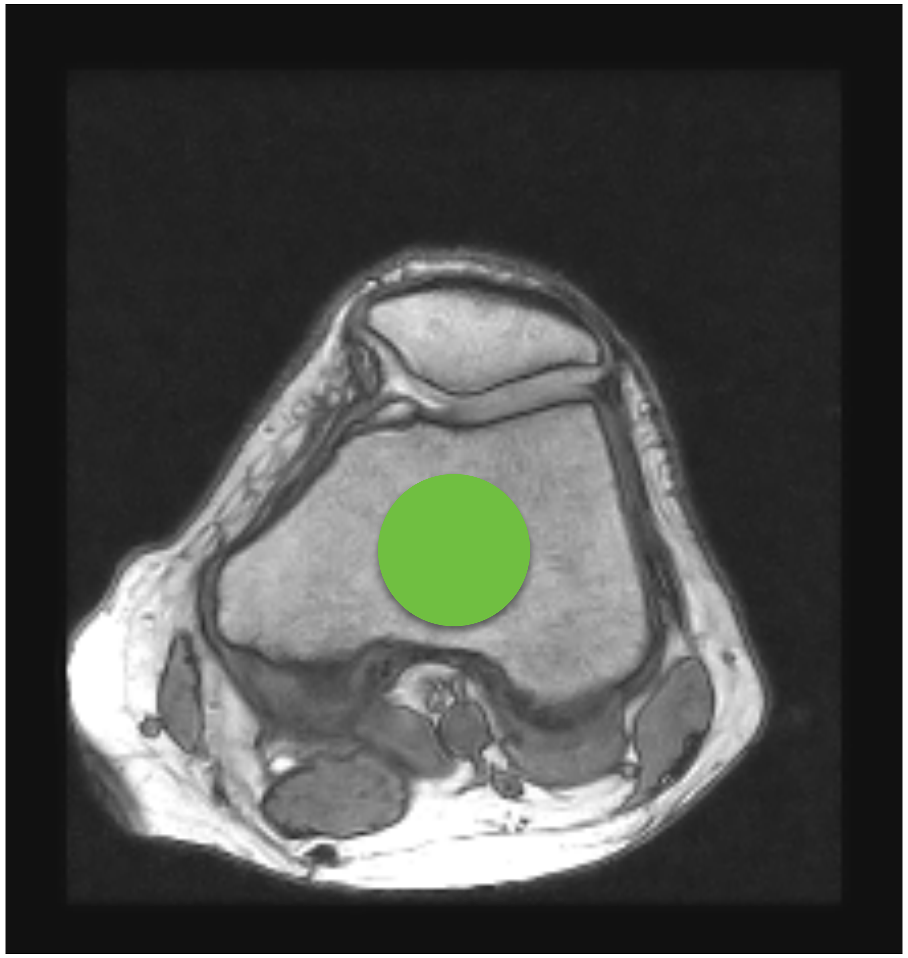
$$f(z) = \frac{1}{1 + e^{-z}} \qquad f(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

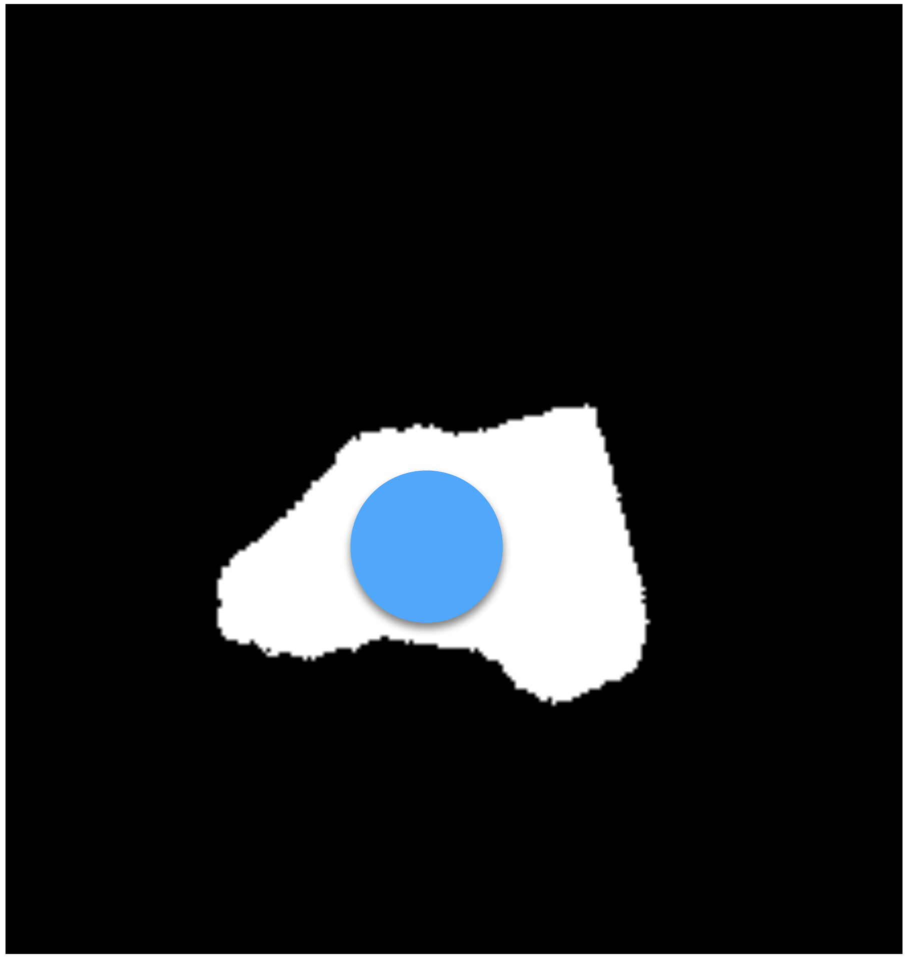
- This is because the result has to be either belonging or not to a class; i.e., our area of interest.

A Concrete Example

Neural Networks: Training Set (1)

Input | **Output**



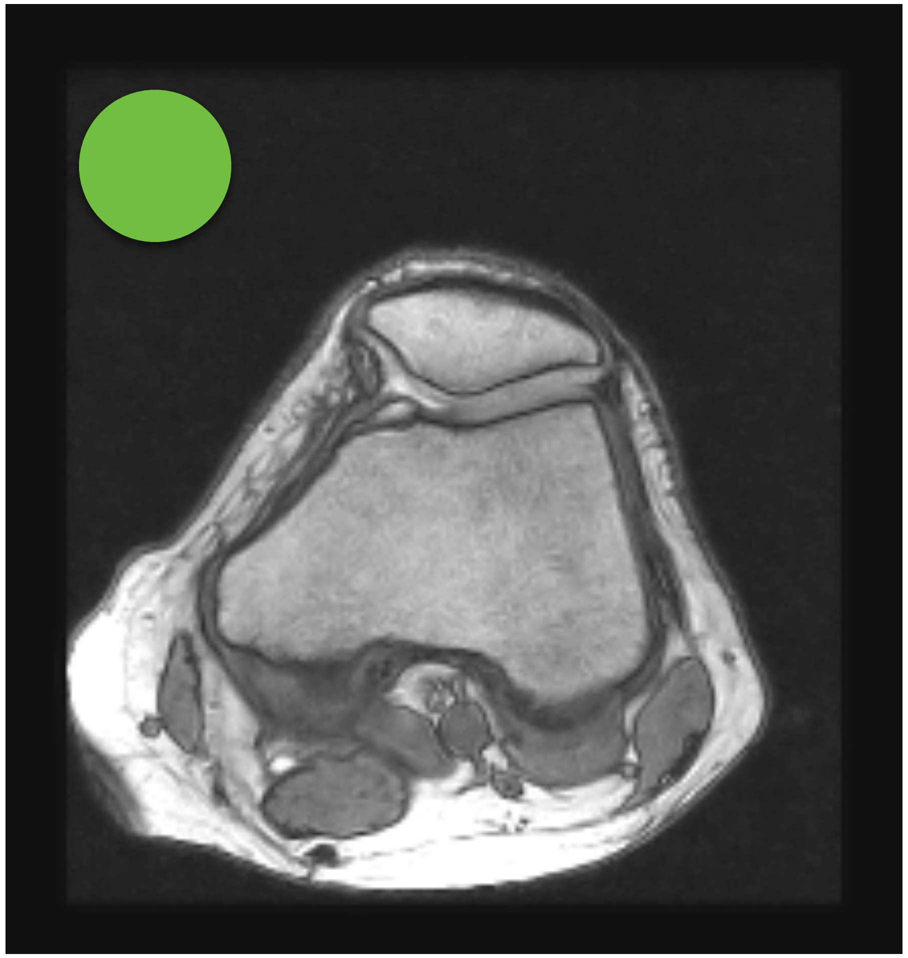


$\mathbf{x}$ = {100,100, 0.78}

y = 1

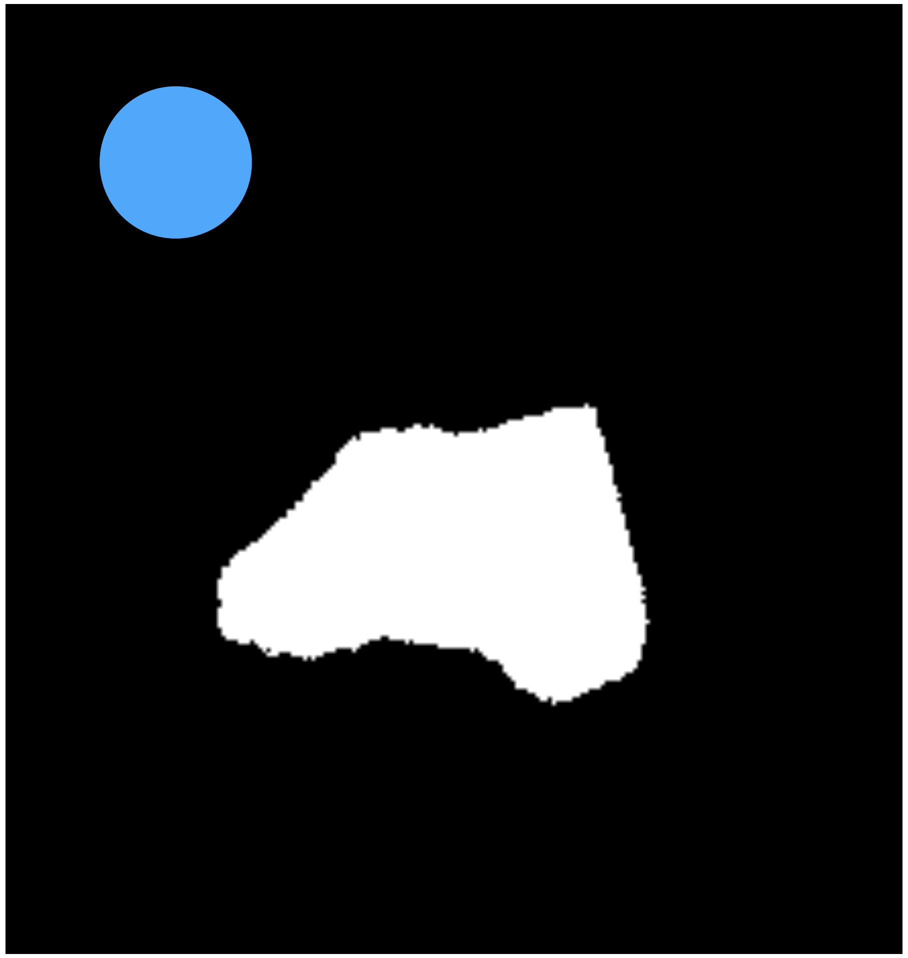
Neural Networks: Training Set (2)

Input



$\mathbf{x} = \{20, 20, 0.039\}$

Output

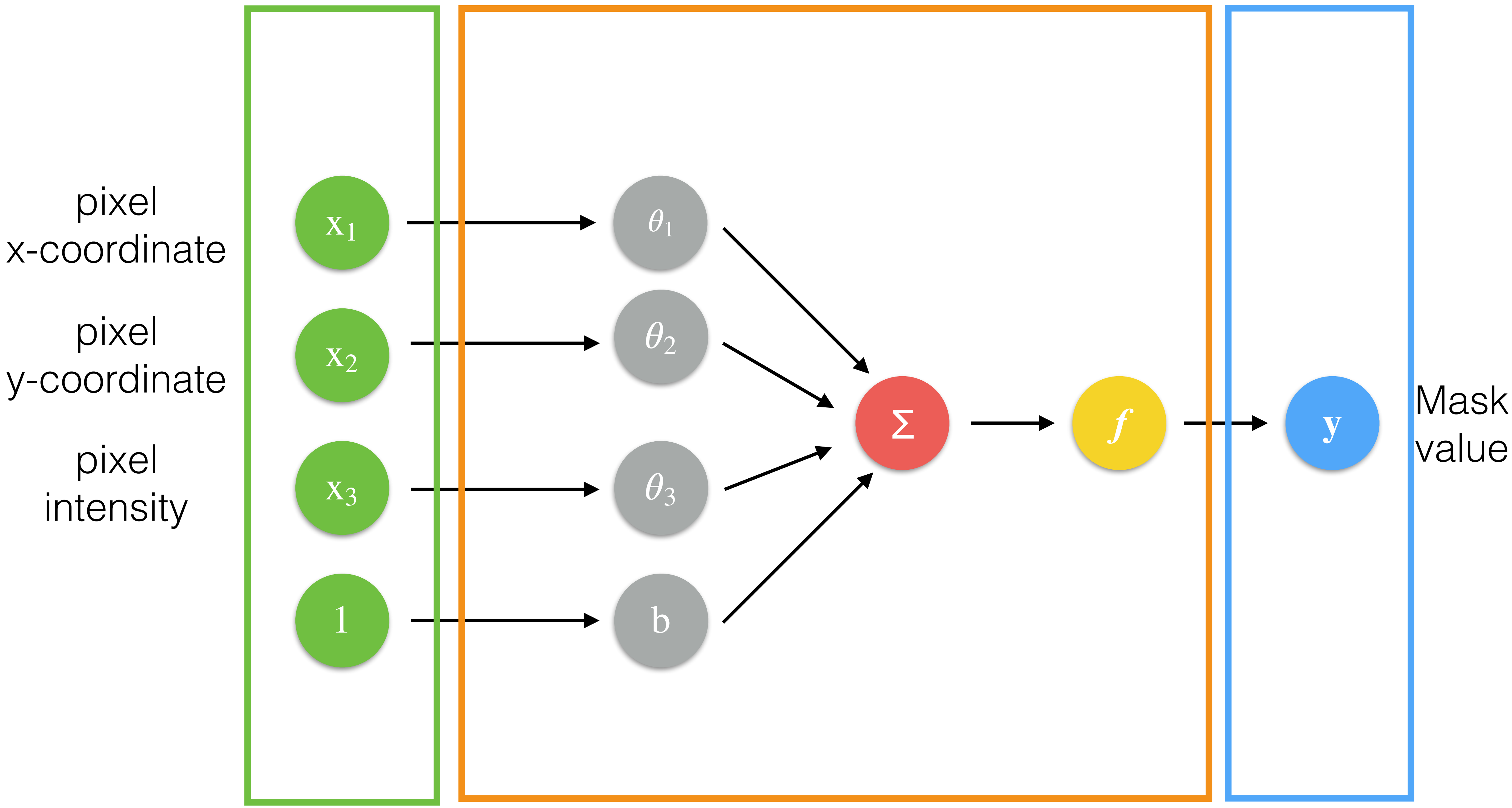


$y = 0$

Machine Learning: Training Set (3)

- The training set needs to be balanced:
  - The same amount of examples for both classes: ROI and background

Neural Networks: A Model



Input

$$\boxed{y} = f(\boxed{\boldsymbol{\Theta}} \cdot \boxed{\mathbf{x}}^\top + \boxed{b})$$

Output

Neural Networks: Learning

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

$$J(\boldsymbol{\Theta}) = \frac{1}{2}\sum_{i=1}^{m}\left(f(\mathbf{x}^i \cdot \boldsymbol{\Theta}^\top + b) - y^i\right)^2 \text{ with } f(x) = x$$

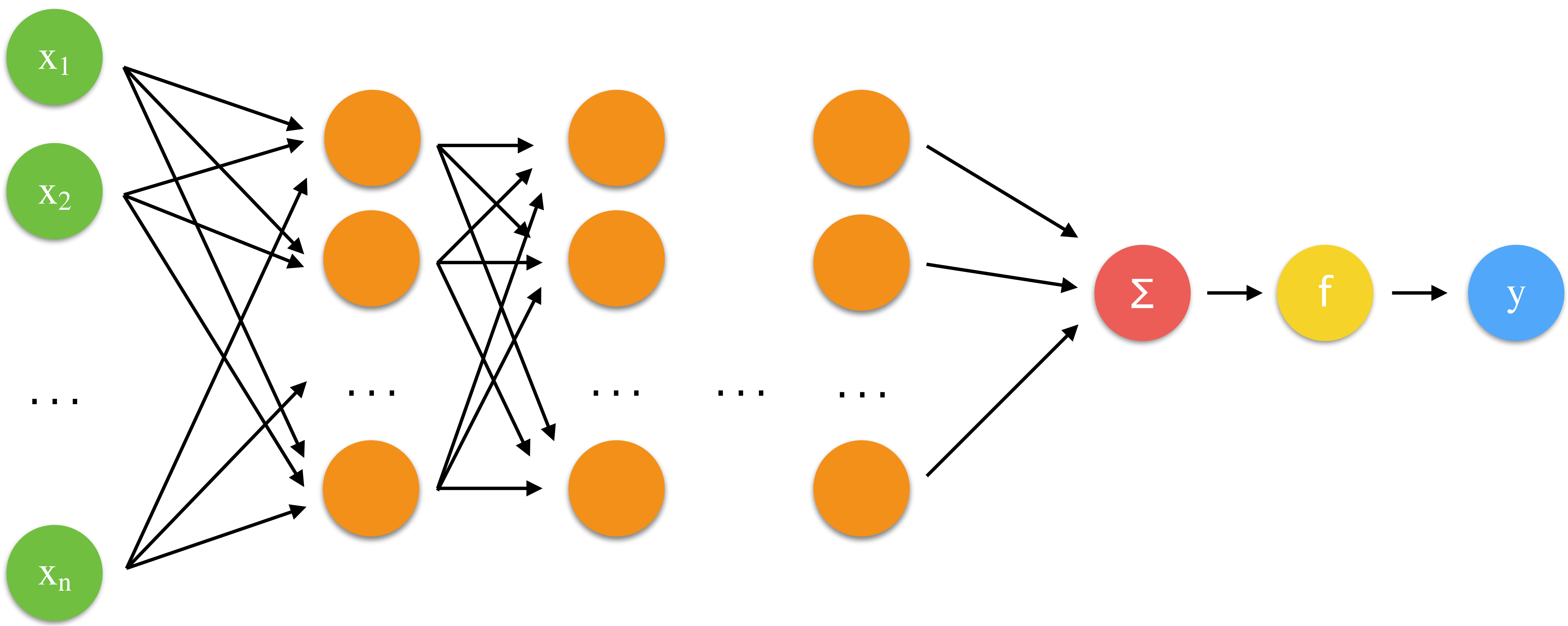
- How do we minimize it?
  - Gradient descent
  - Starting solution for theta? Random values in [0,1]!

More Complex Examples

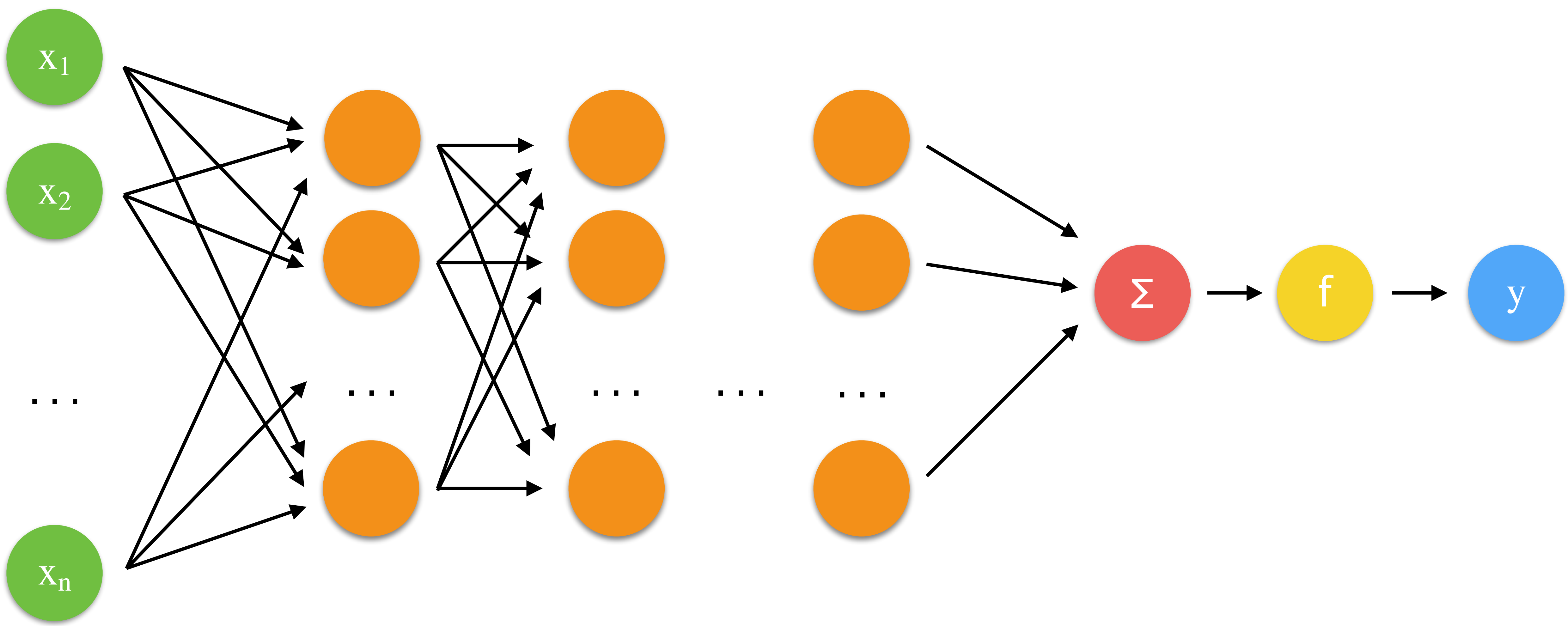
More Complex Nets

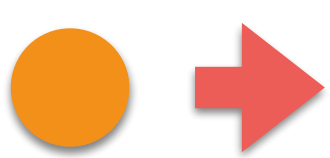
- To achieve high-quality results, a network needs to “see” and “understand” more data at the same time; not only a couple such as the pixel coordinats+pixel intensity and its classification as in the previous example!
- We need to use more pixels/voxels at the same time:
  - How?
    - Adding and mixing more neurons

Neural Networks: Bigger Networks



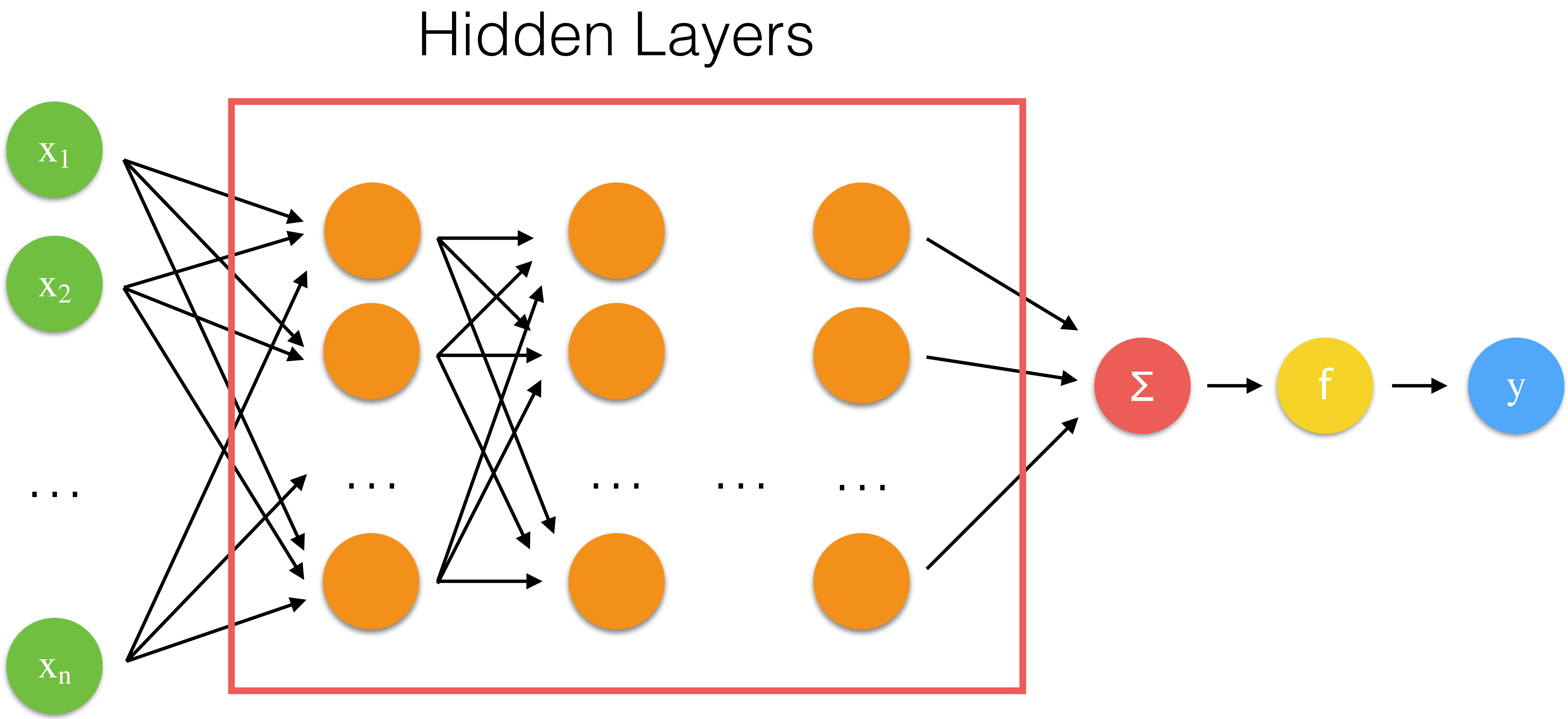
Neural Networks: Bigger Networks

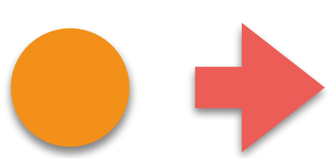




$$y = f(\mathbf{\Theta} \cdot \mathbf{x}^\top + b)$$

Neural Networks: Bigger Networks





$$y = f(\mathbf{\Theta} \cdot \mathbf{x}^\top + b)$$

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 result; better >10,000 training example!

that’s all folks!