Segmentation with Machine Learning

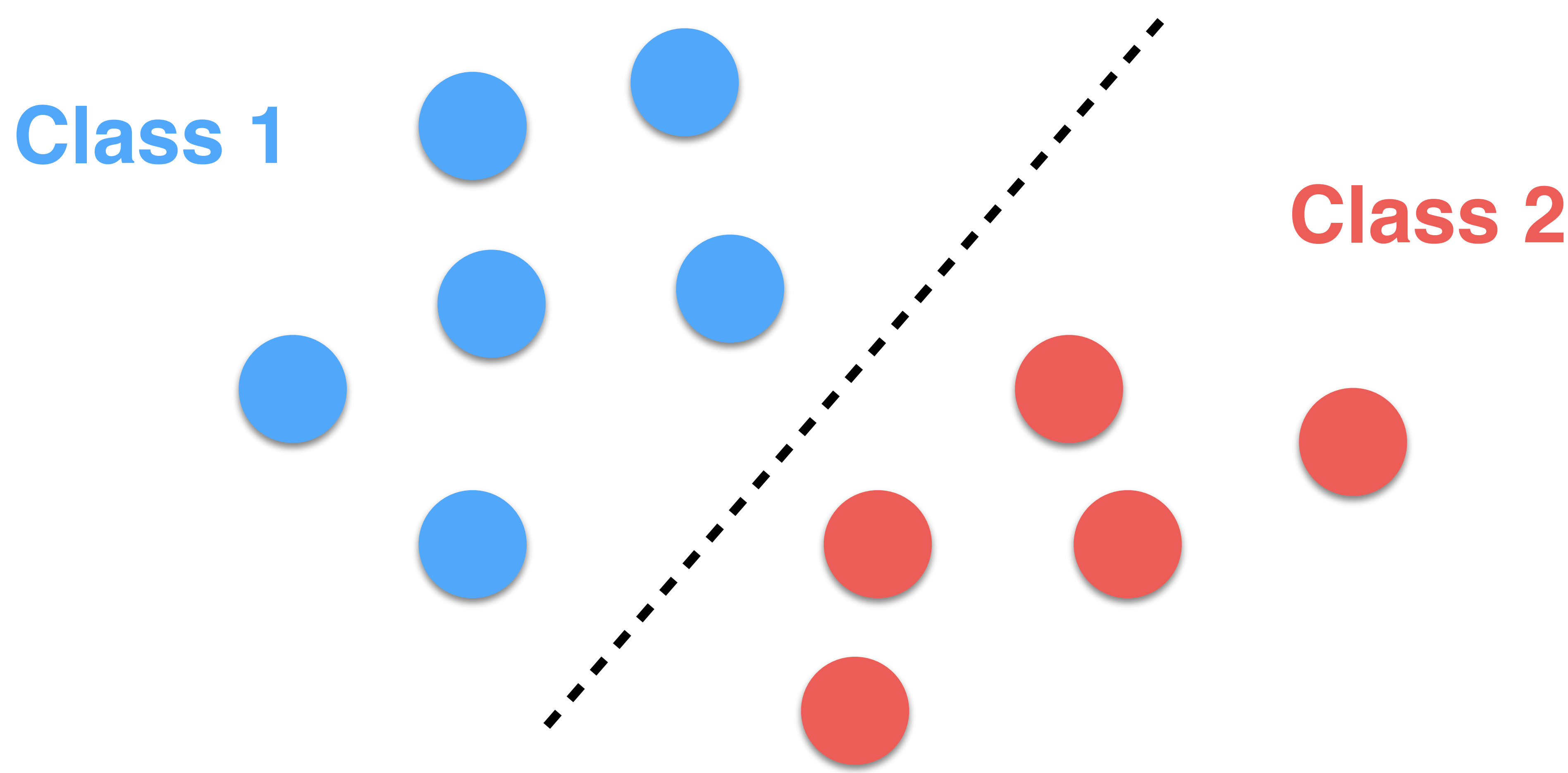
Francesco Banterle, Ph.D.
francesco.banterle@isti.cnr.it

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

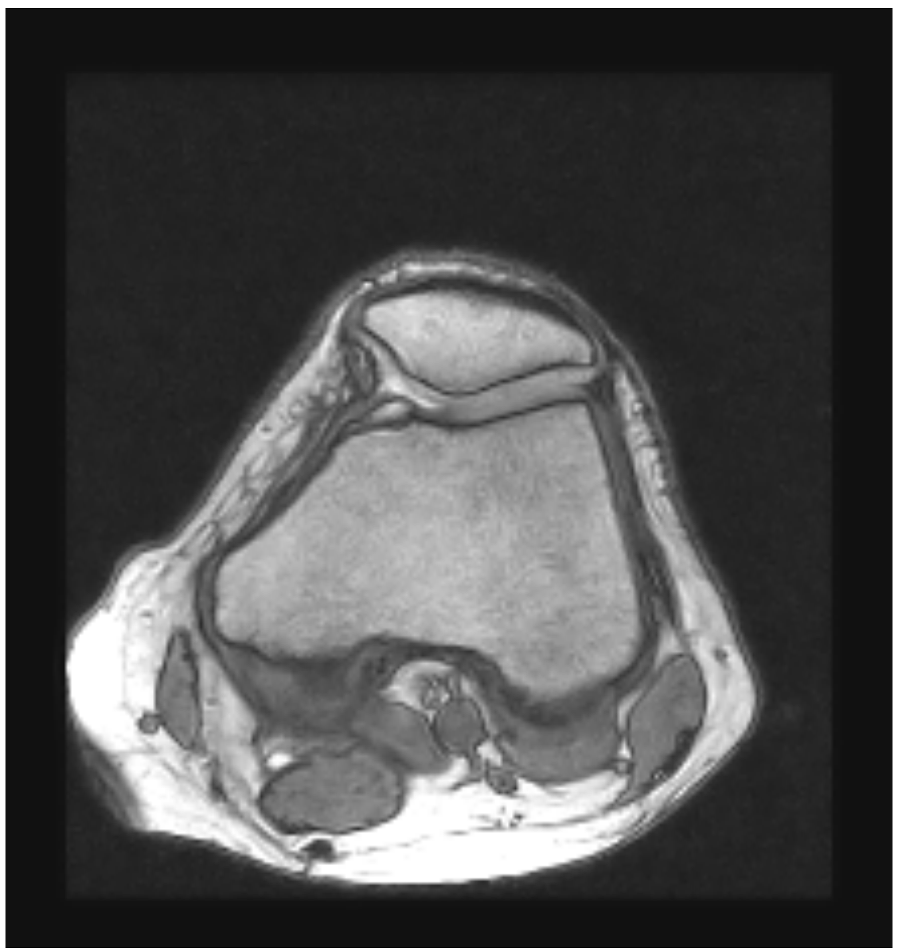
Model

Learning Method

Machine Learning: Training Set

- **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: Training Set



Input

Output

Machine Learning: Model

- **Model**: a mathematical model, i.e. a function, that can store the **knowledge** of the dataset into its parameters or ***weights***.

- For example:

    - A neural network;

Machine Learning: Learning Method

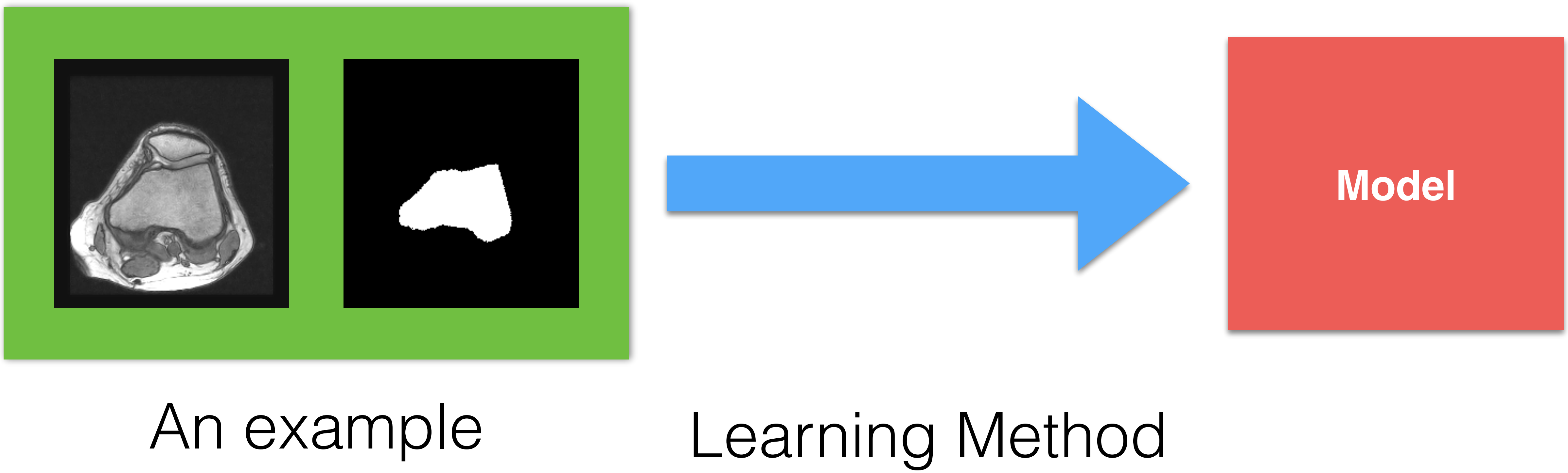
- **Learning Method**: a mathematical model/function that transfers the **knowledge** of the training set to the model:
  - It is a mix between:
    - Minimization method; i.e., Gradient Descent;
    - Loss function; i.e., how to minimize the differences.

Machine Learning: Supervised Learning

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

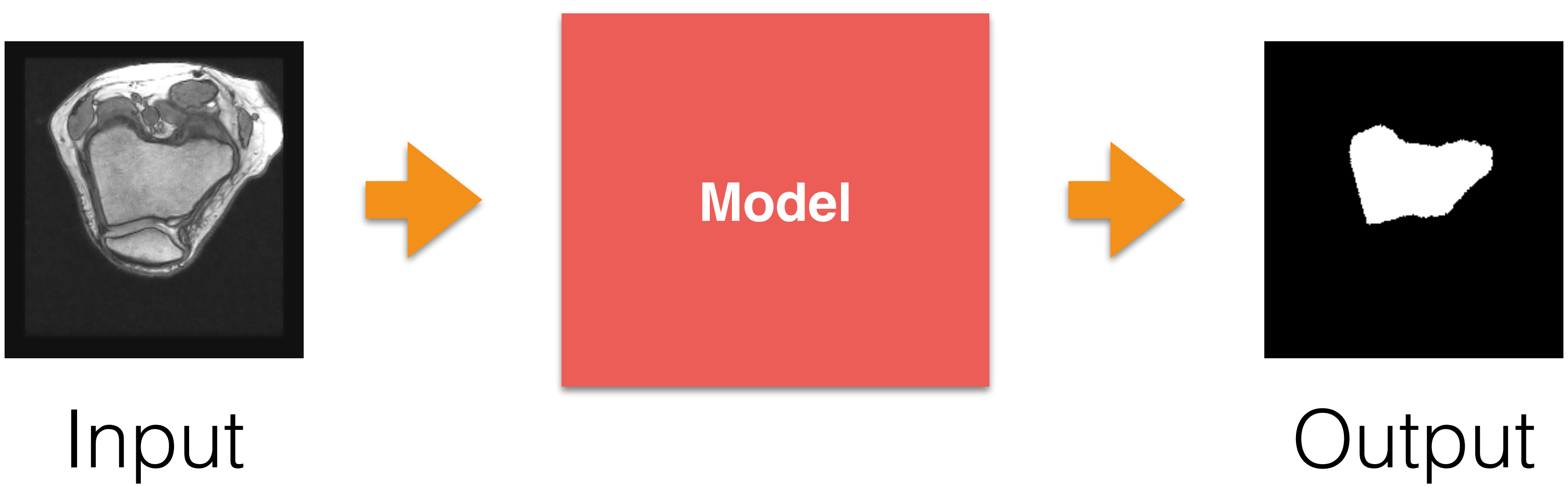
Machine Learning: Supervised Learning

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



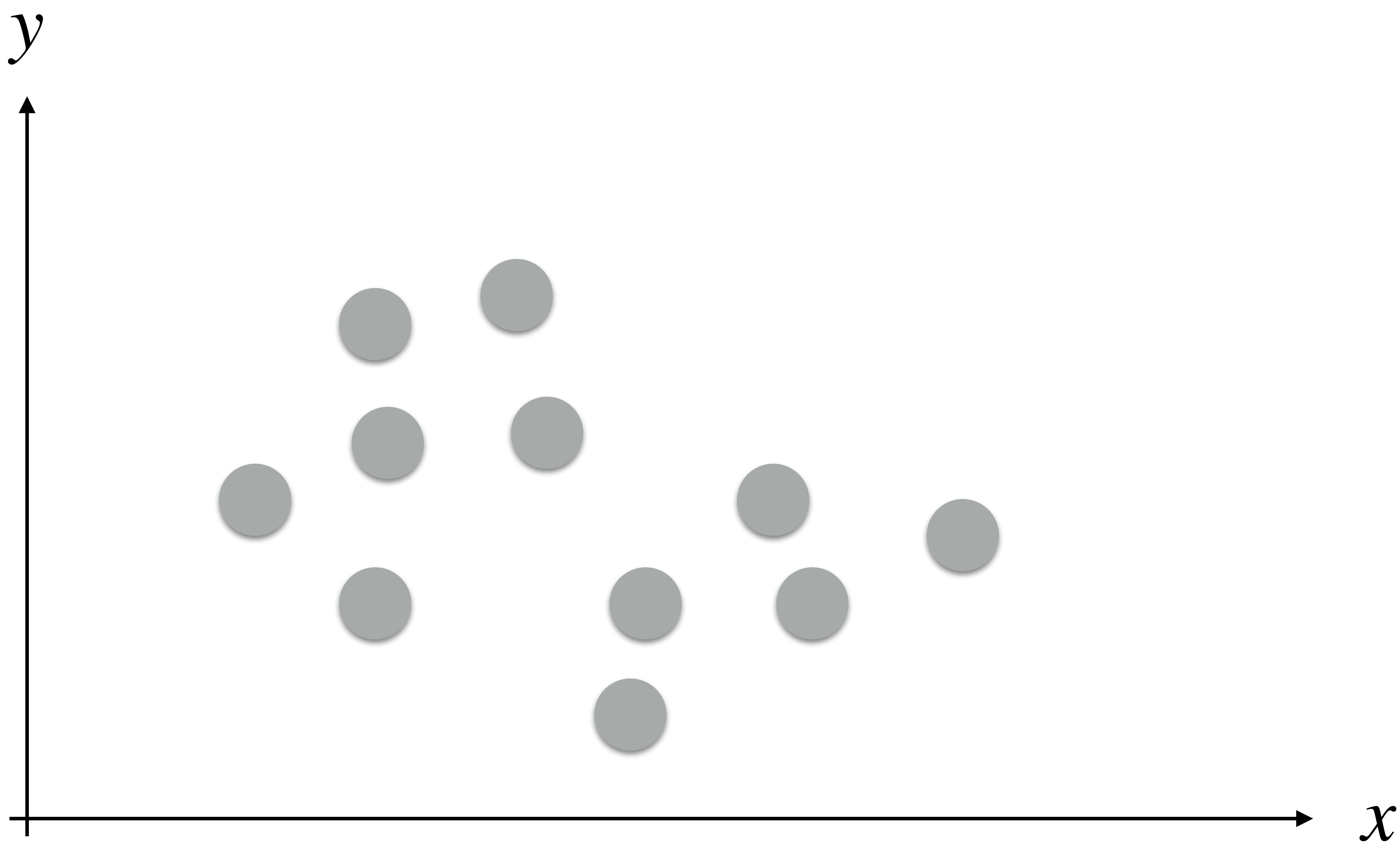
Machine Learning: Supervised Prediction/Evaluation

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

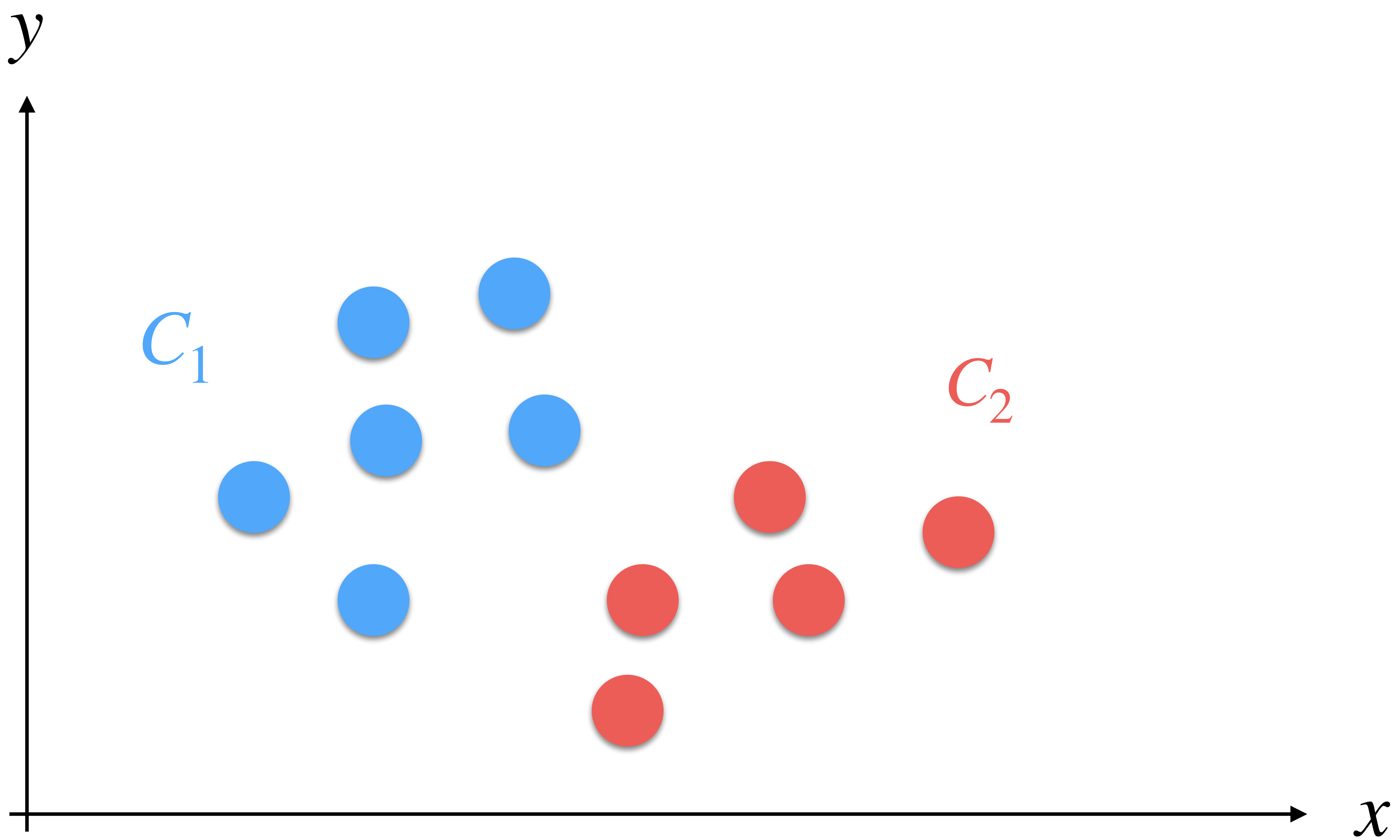


A Simple Example: Binary Classification

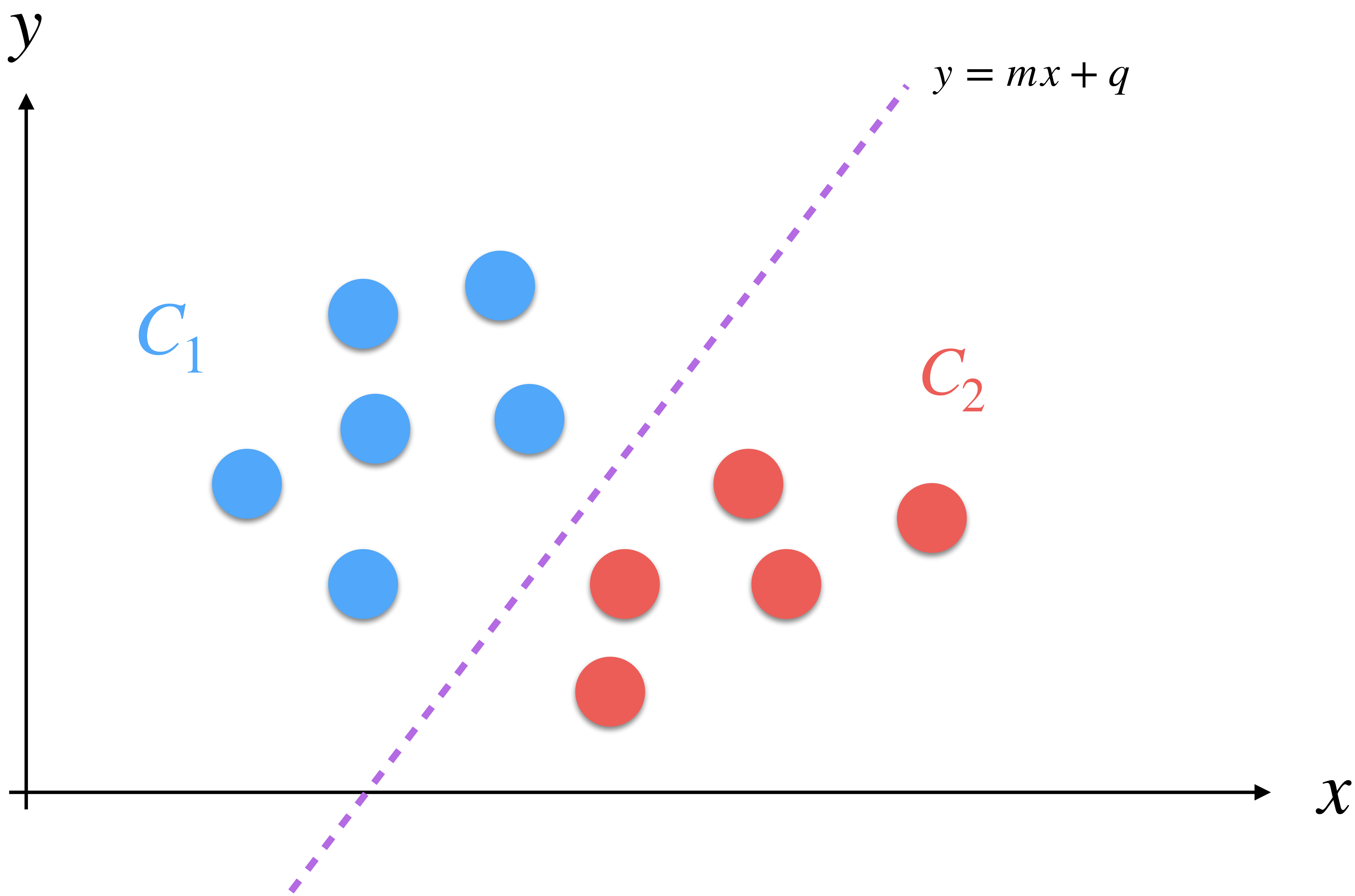
Binary Classification



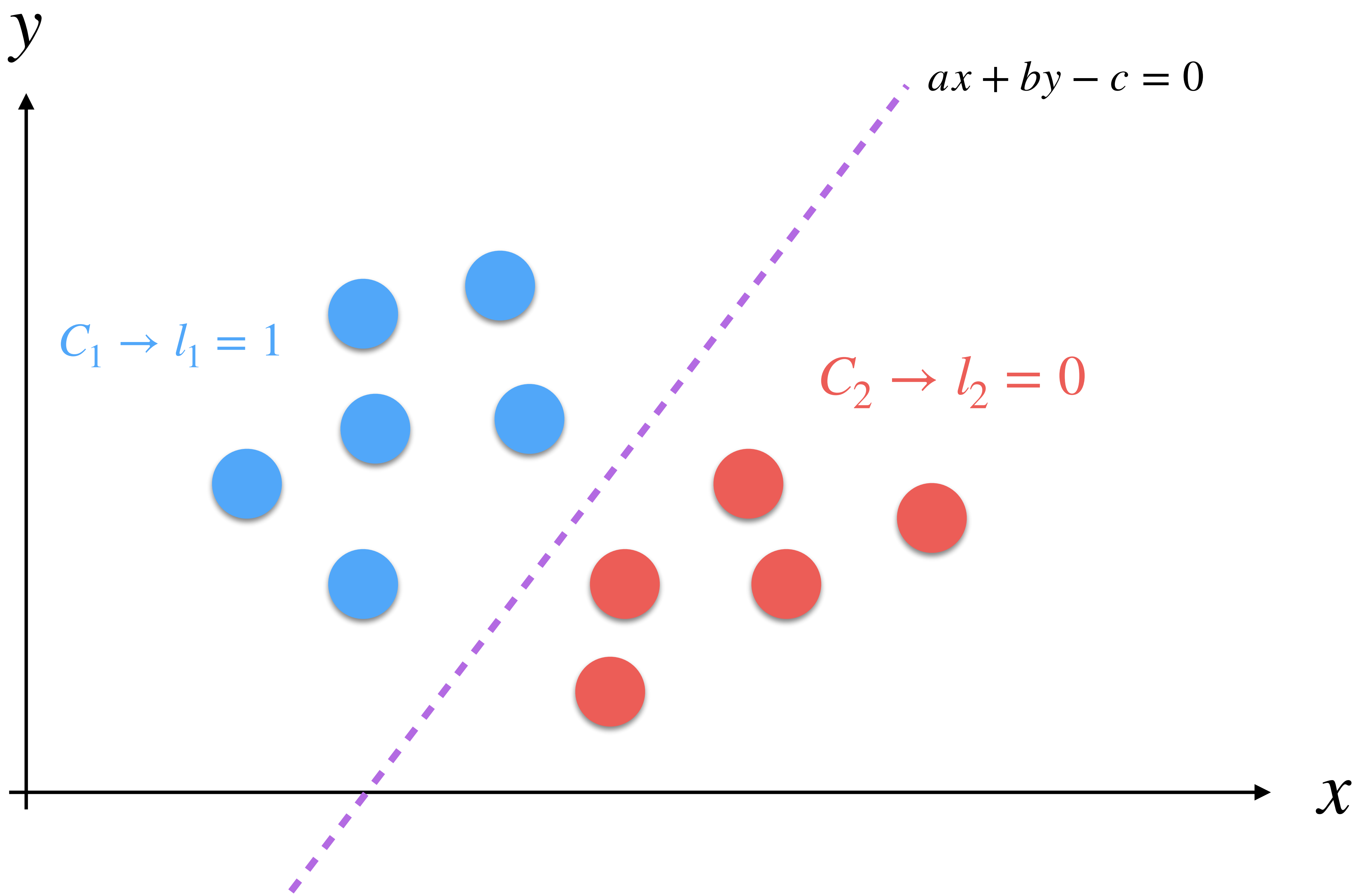
Binary Classification



Binary Classification



Binary Classification



Binary Classification

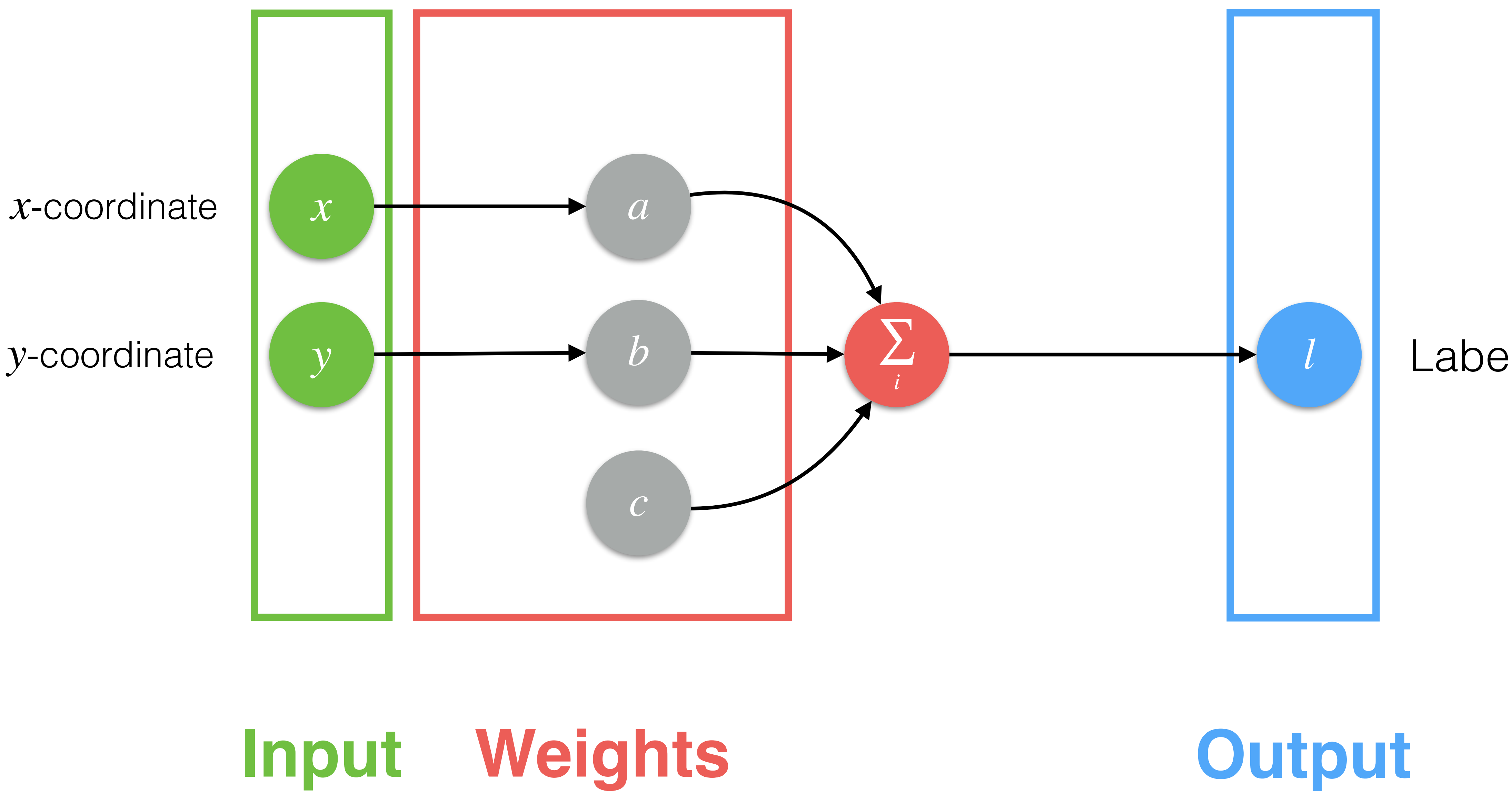
- Now, if we get a new sample $\mathbf{p}^i = (x^i, y^i)$ belongs C_1 we have:

$$ax^i + by^i - c \geq 0$$

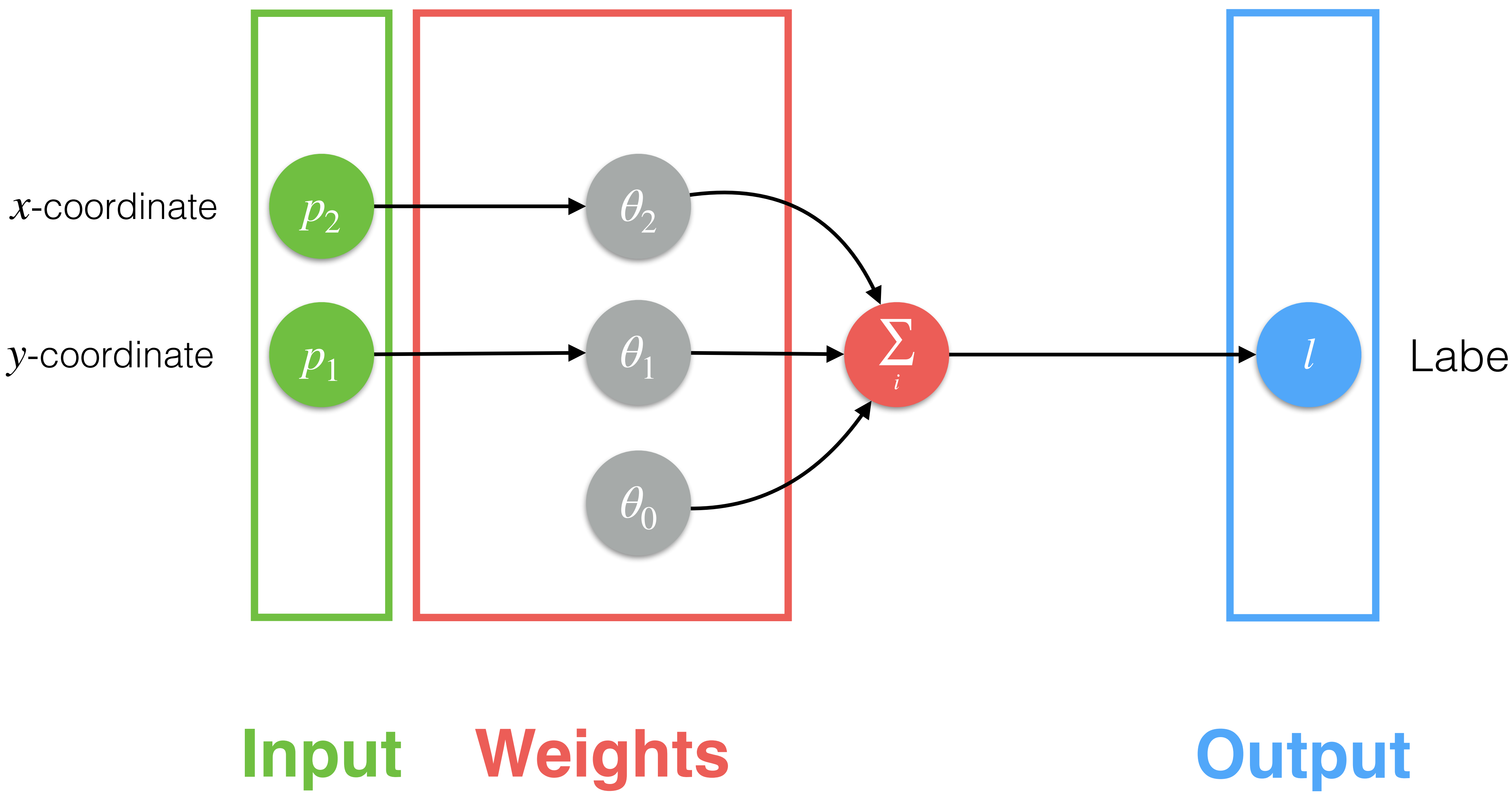
- If it belongs to C_2 we have:

$$ax^i + by^i - c < 0$$

Binary Classification: Our Model h



Binary Classification: Our Model h



Binary Classification: Our Model h

- Our model can be so defined as:

$$h(\mathbf{p}, \theta) = [\mathbf{p}, 1]^{\mathsf{T}} \cdot \theta$$

Neural Networks: Supervised Learning

- We need to collect m couples $(\mathbf{p}^j, l^j)$.
- We need to minimize an error function:

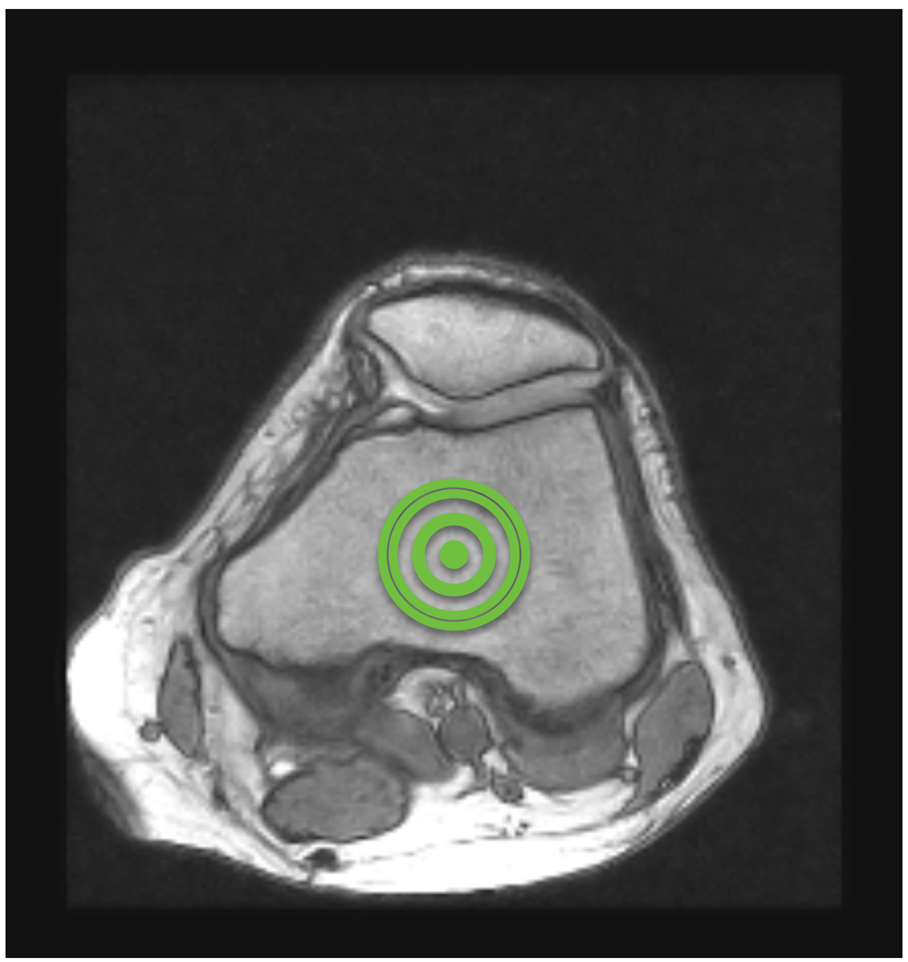
$$J(\theta) = \frac{1}{2}\sum_{j=1}^{m}\left(h(\mathbf{p}^j, \theta) - l^j\right)^2$$

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

A Segmentation Example

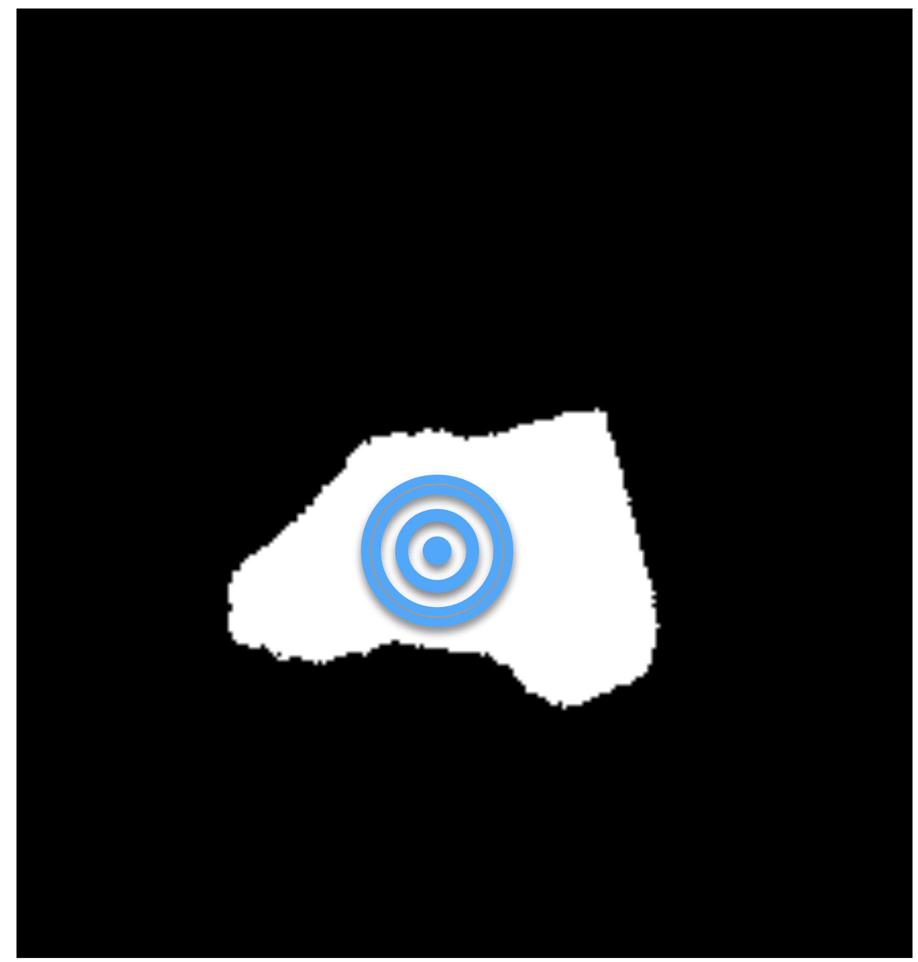
Segmentation: Dataset Set (1)

Input



$\mathbf{p}^1 = (100,100,0.67)$

Output



$l^1 = 1$

Segmentation: Dataset Set (2)

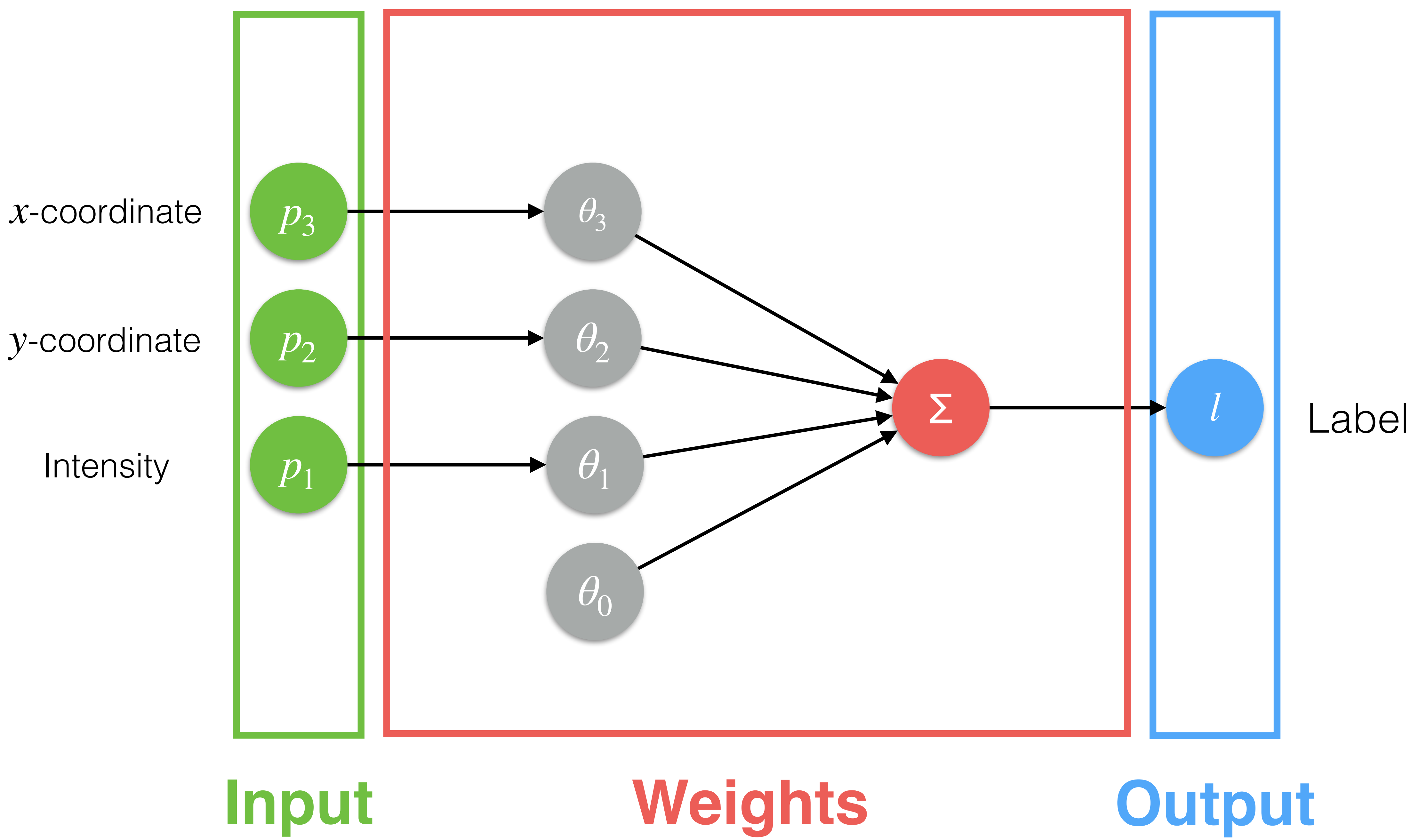
Input

$\mathbf{p}^2 = (20,20,0.01)$

Output

$l^2 = 0$

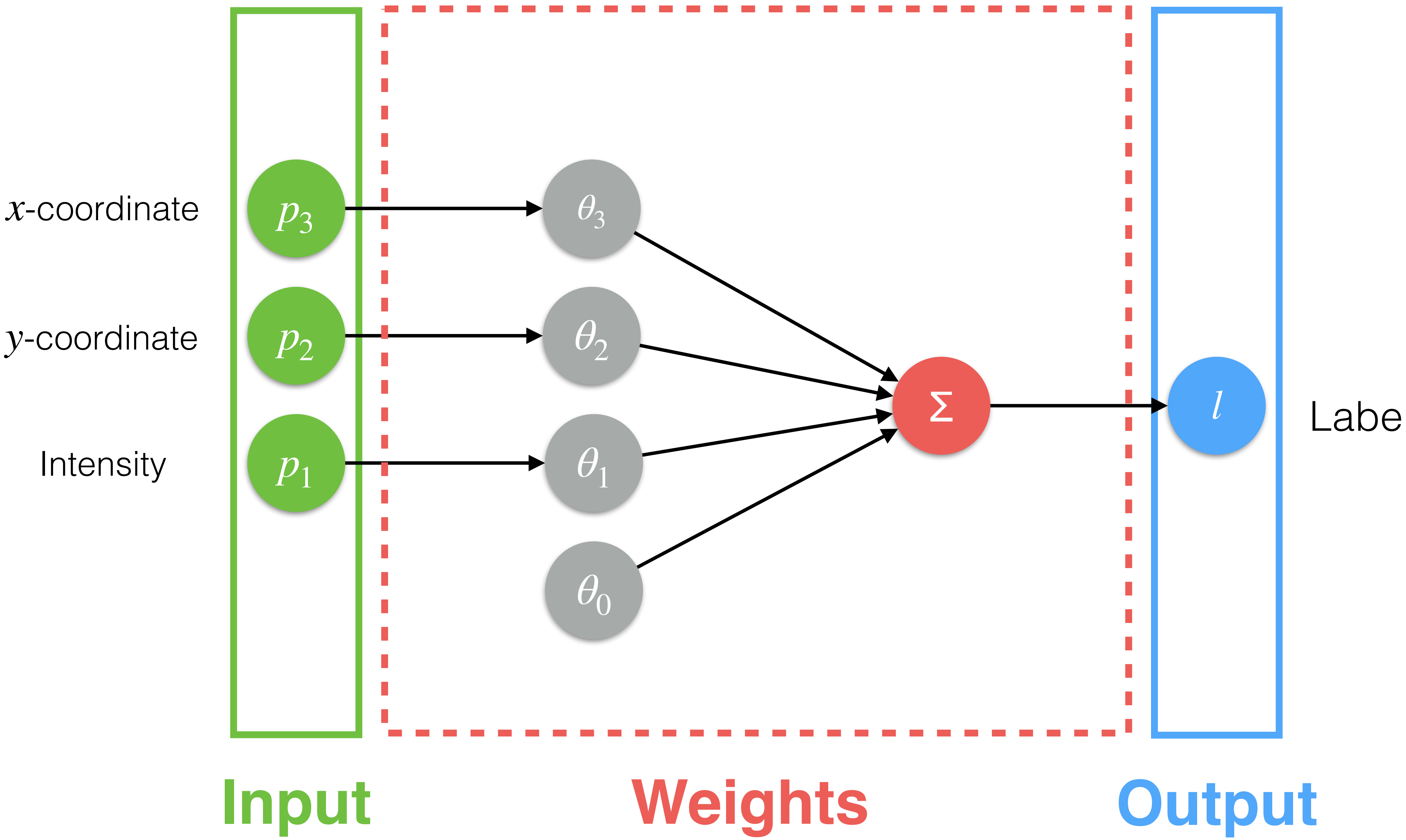
Segmentation: The Model



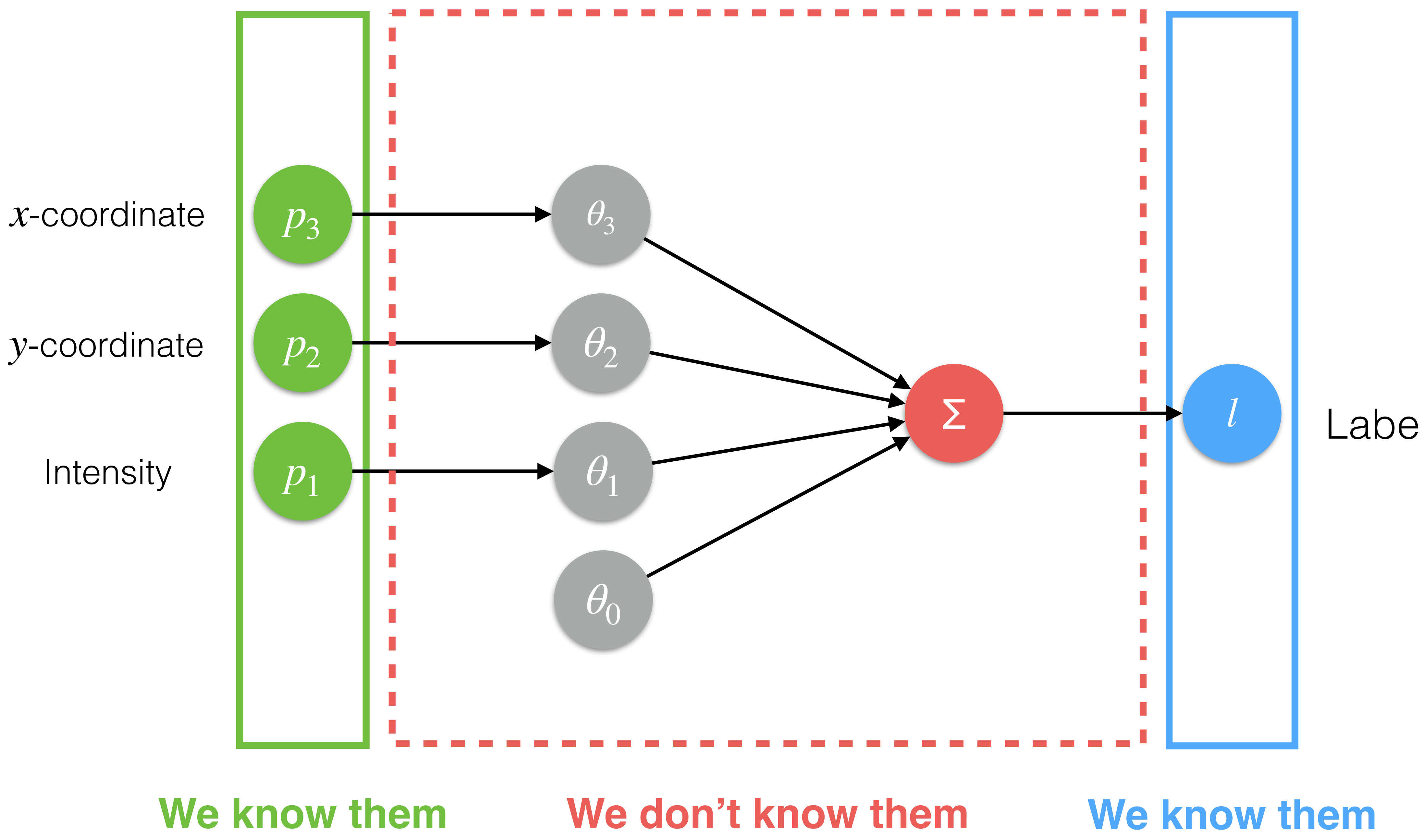
Segmentation: Dataset Set (3)

- 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.

Segmentation: Training Phase

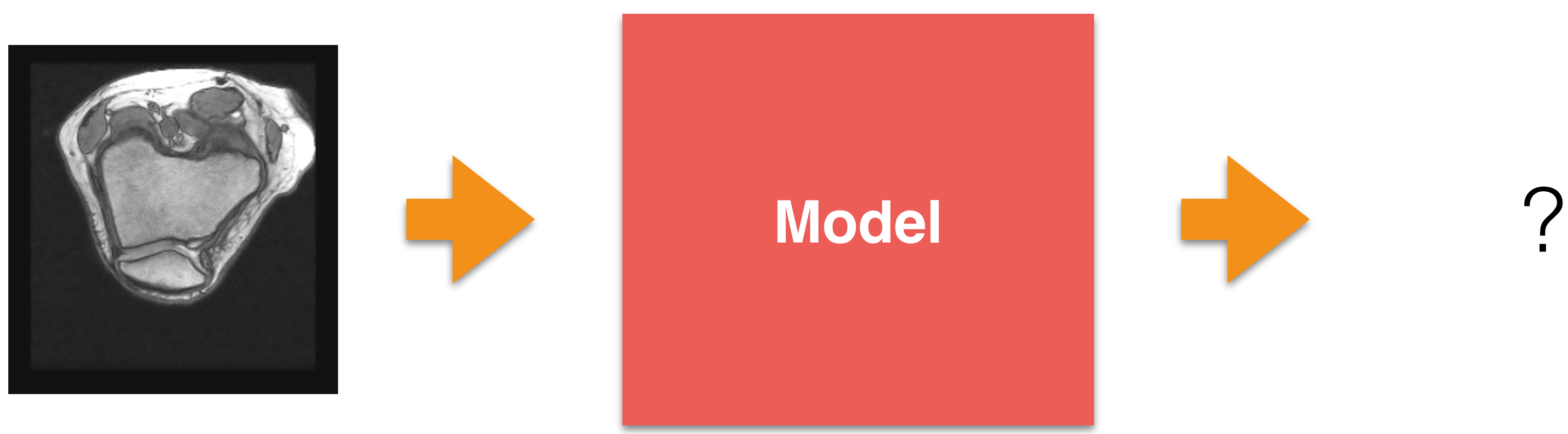


Segmentation: Training Phase



Segmentation: Prediction/Inference Phase

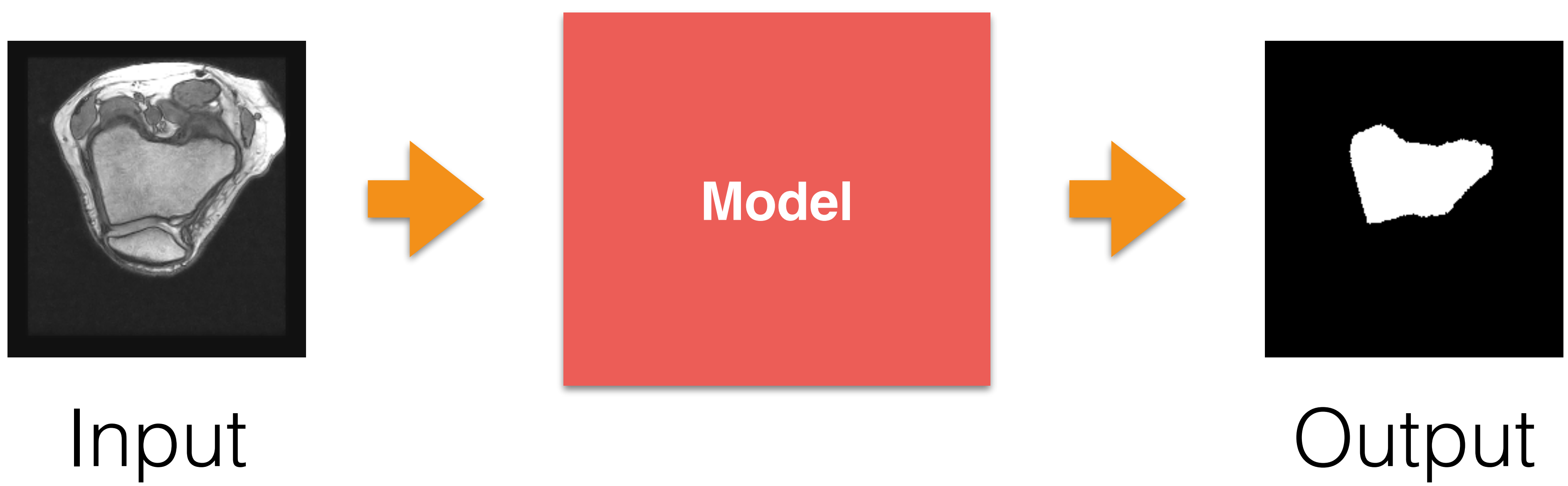
- After learning, we can use our network on new images to segment the image:



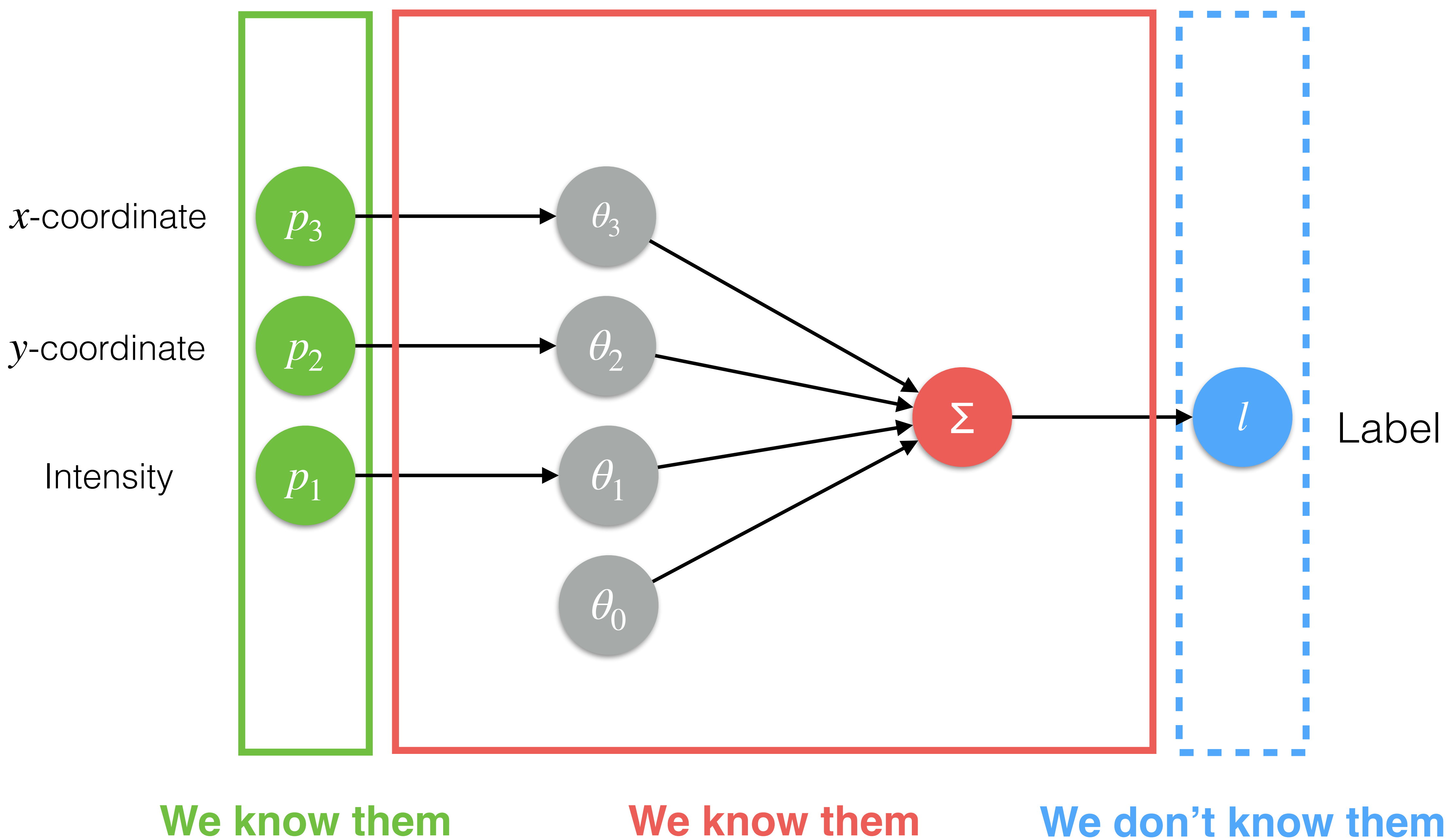
Input

Segmentation: Prediction/Inference Phase

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



Segmentation: Prediction Phase



Going Non-Linear

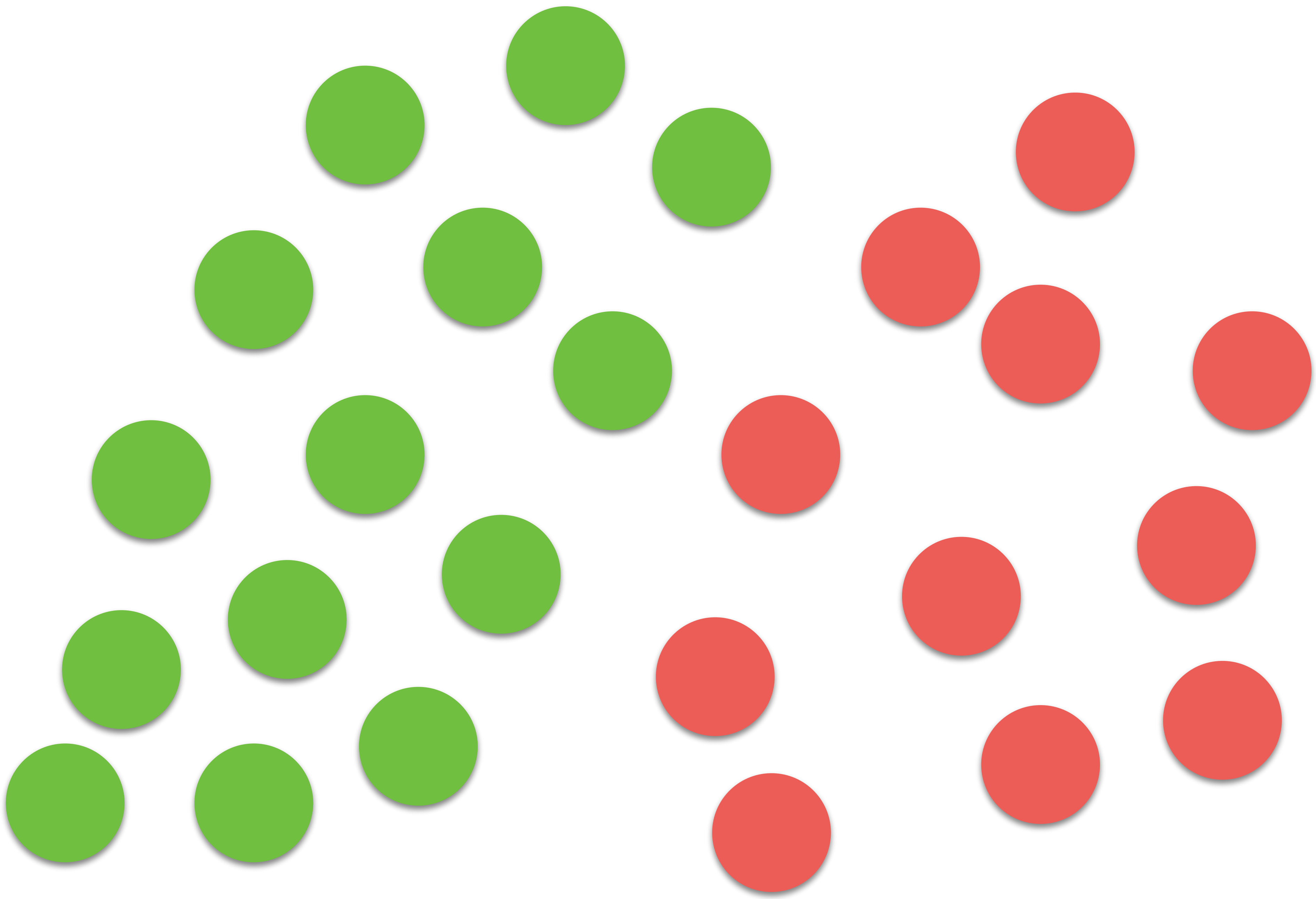
Why Non-Linear

- The model that we have seen so far is:

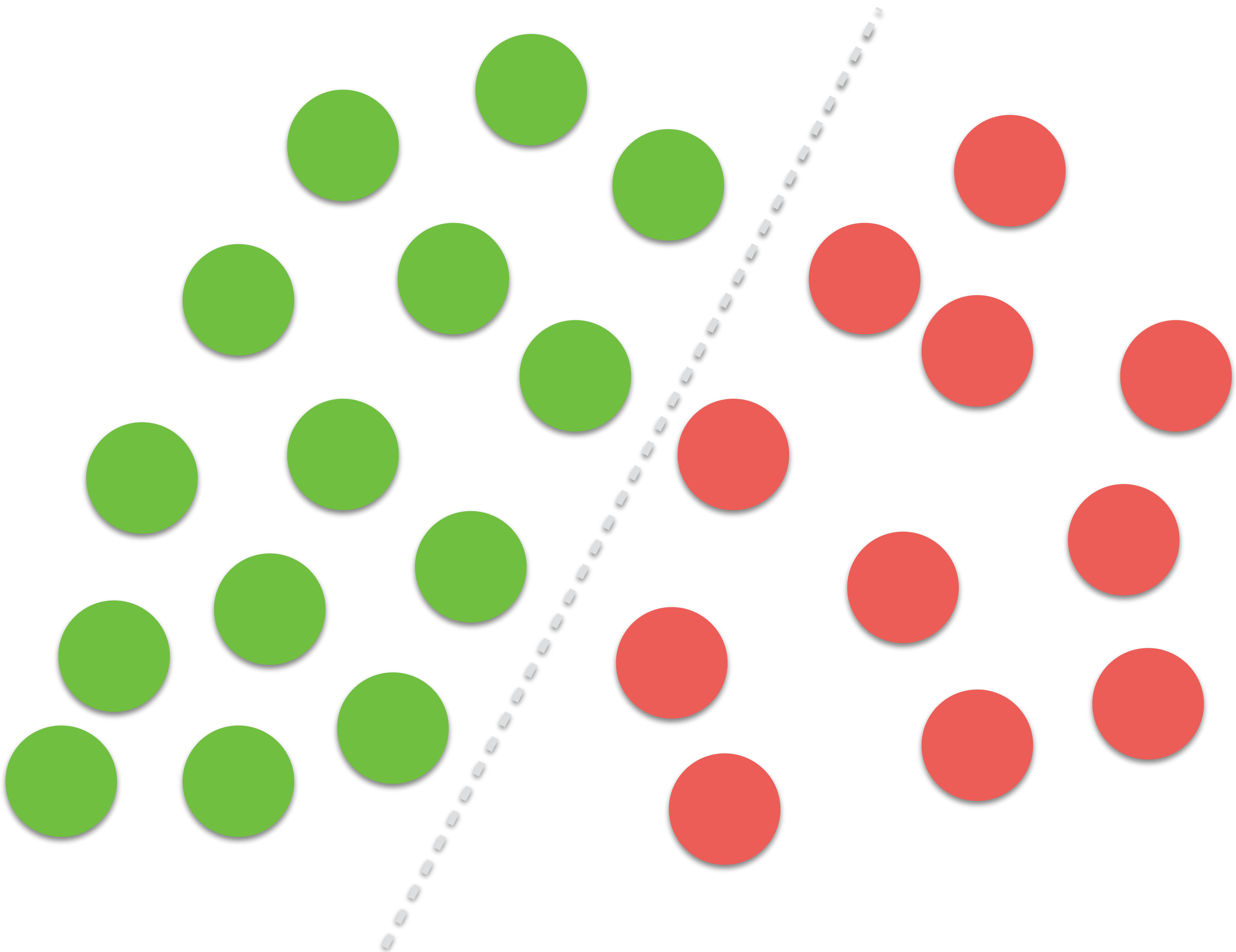
$$h(\mathbf{p}, \theta) = [\mathbf{p}, 1]^{\mathsf{T}} \cdot \theta$$

- This model can only capture linear models; i.e., we can classify/segment by separating data using a plane/hyper-plane.

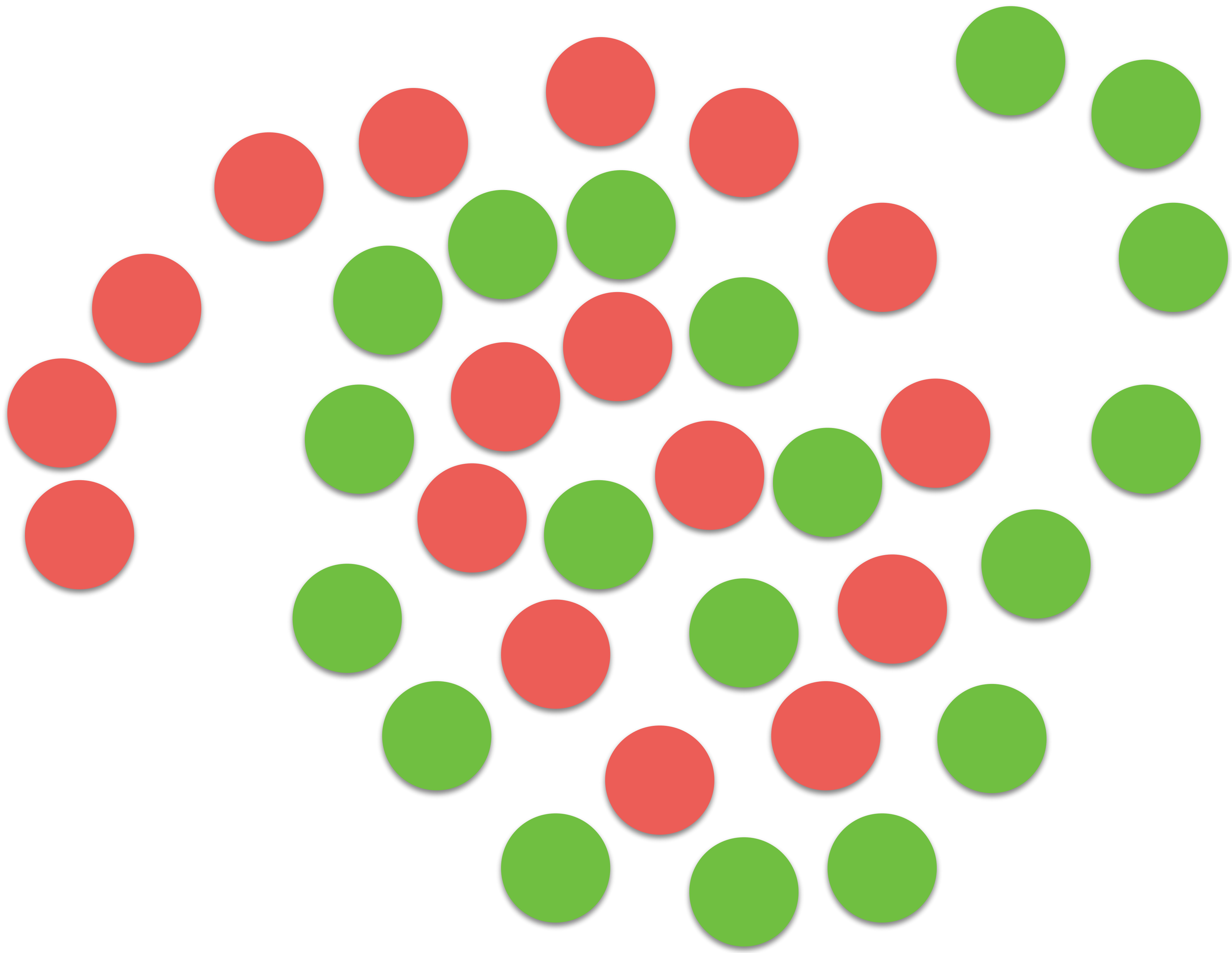
Why Non-Linear



Why Non-Linear



Why Non-Linear



Why Non-Linear

- We can extend the model to

$$h(\mathbf{p}, \theta) = g\left([\mathbf{p}, 1]^{\mathsf{T}} \cdot \theta\right)$$

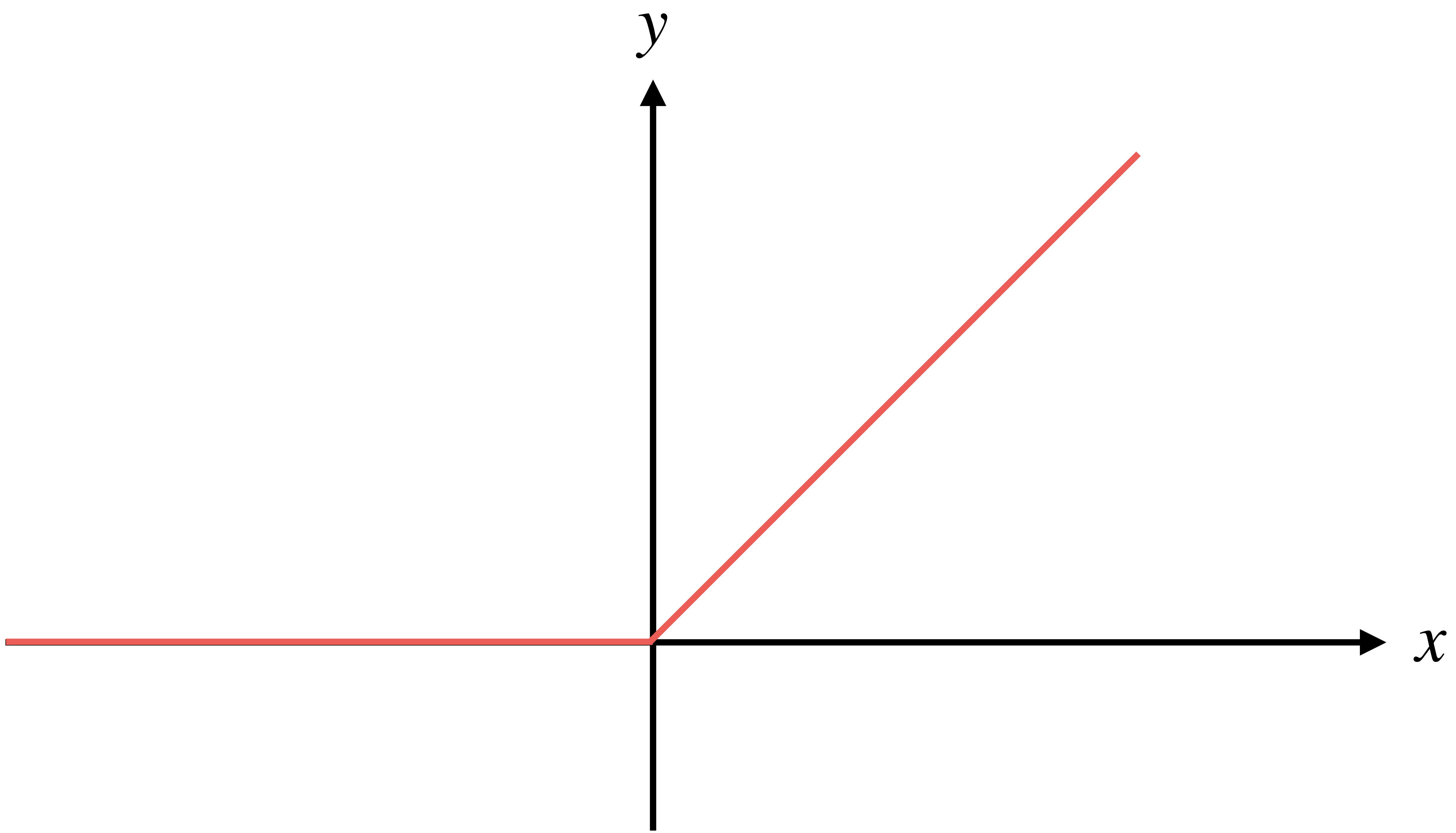
where $g(\,\cdot\,)$ is a non linear function.

Why Non-Linear: Rectified Linear Unit (ReLU)

- A simple and effective function is the rectified linear unit or ReLU:

$$g(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Why Non-Linear:
Rectified Linear Unit (ReLU)



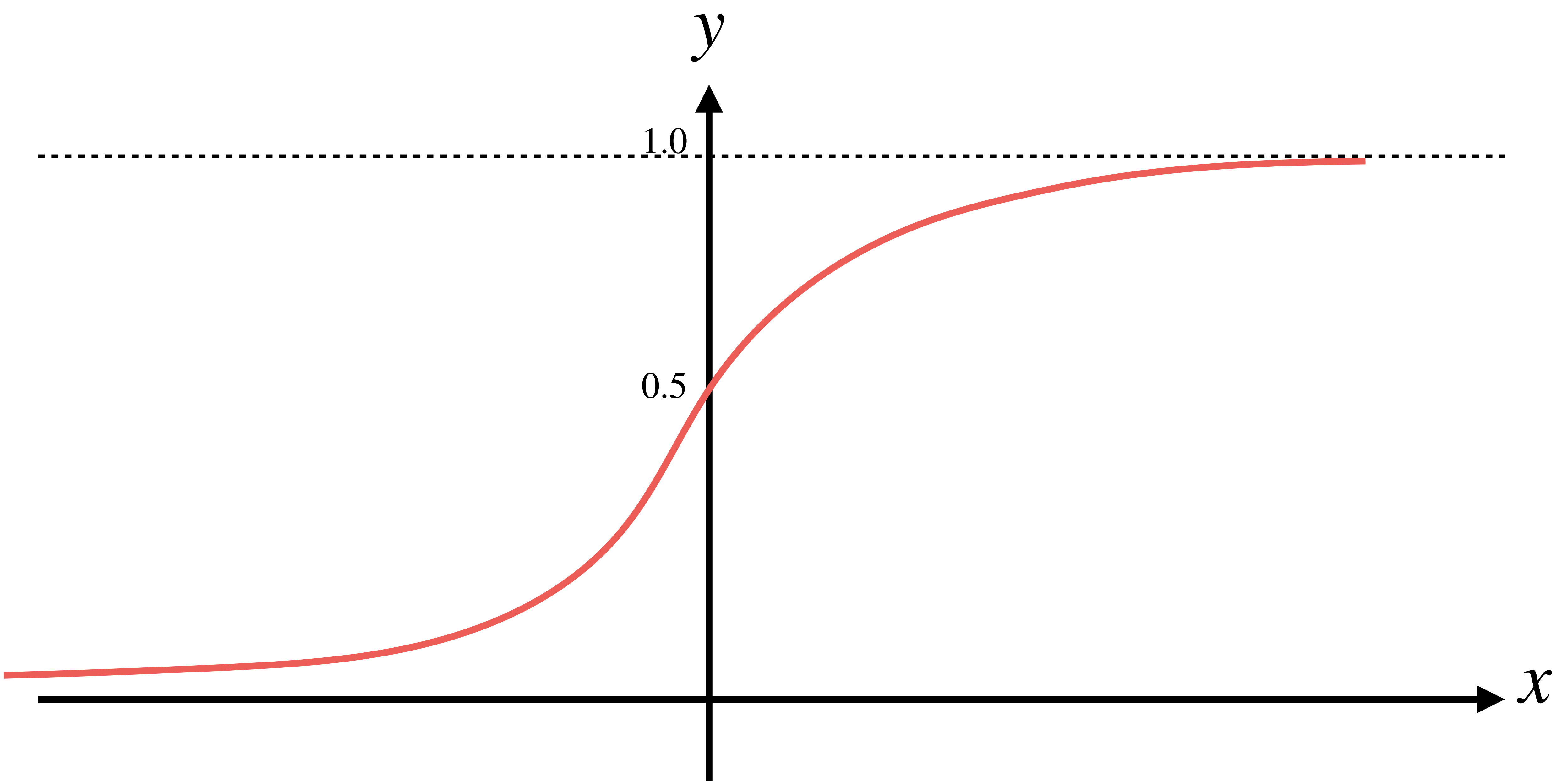
Why Non-Linear: Sigmoid

- For classification/segmentation, a more effective function is the sigmoid:

$$g(x) = \frac{1}{1 + e^{-x}}$$

- When $g(x) \geq 0.5$ —> we label as 1 our sample
- When $g(x) < 0.5$ —> we label as 0 our sample

Why Non-Linear: Sigmoid



Why Non-Linear: Sigmoid + Error Function

- We have our error function:

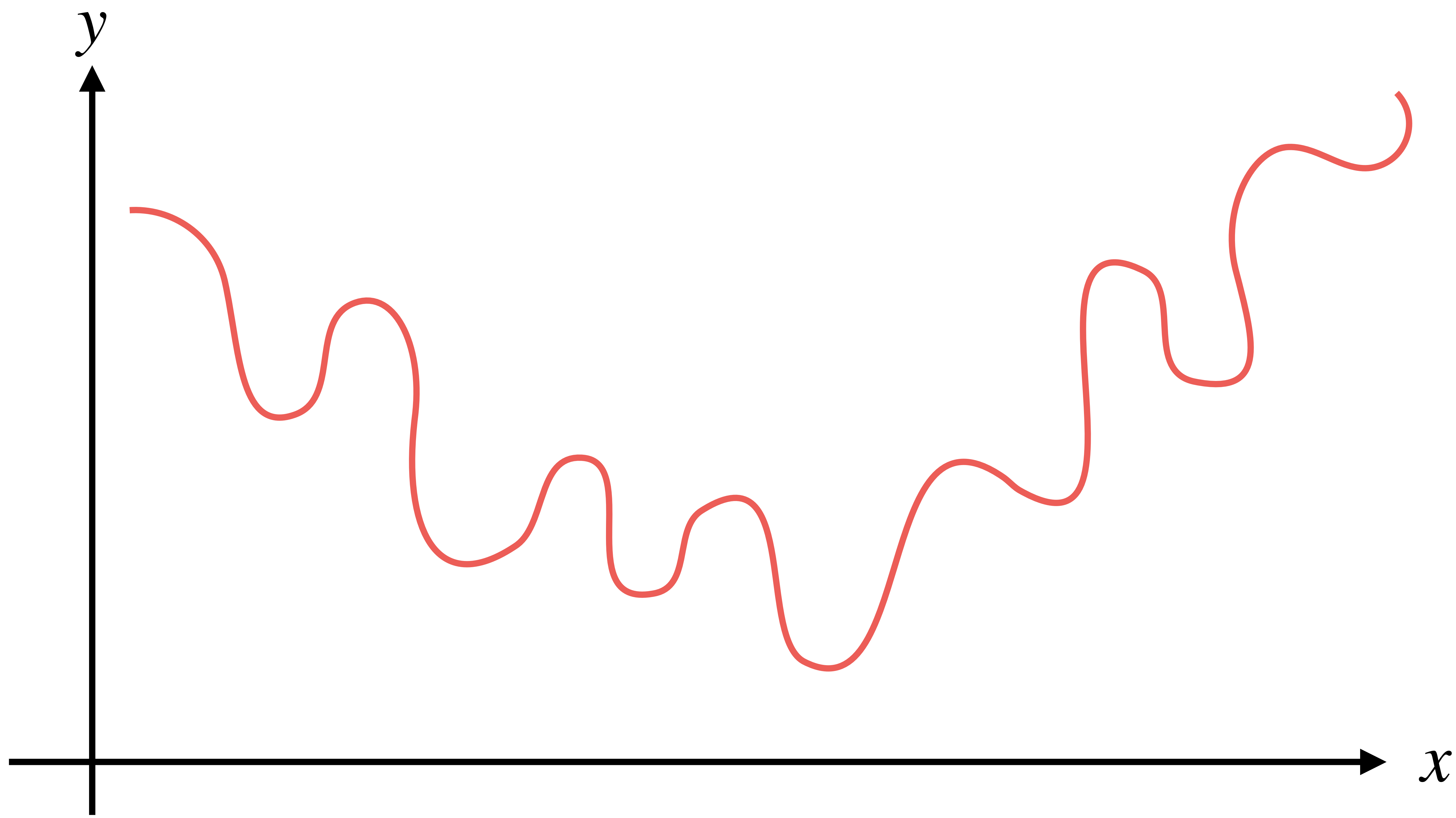
$$J(\theta) = \frac{1}{2}\sum_{j=1}^{m}\left(h(\mathbf{p}^j, \theta) - l^j\right)^2$$

which is:

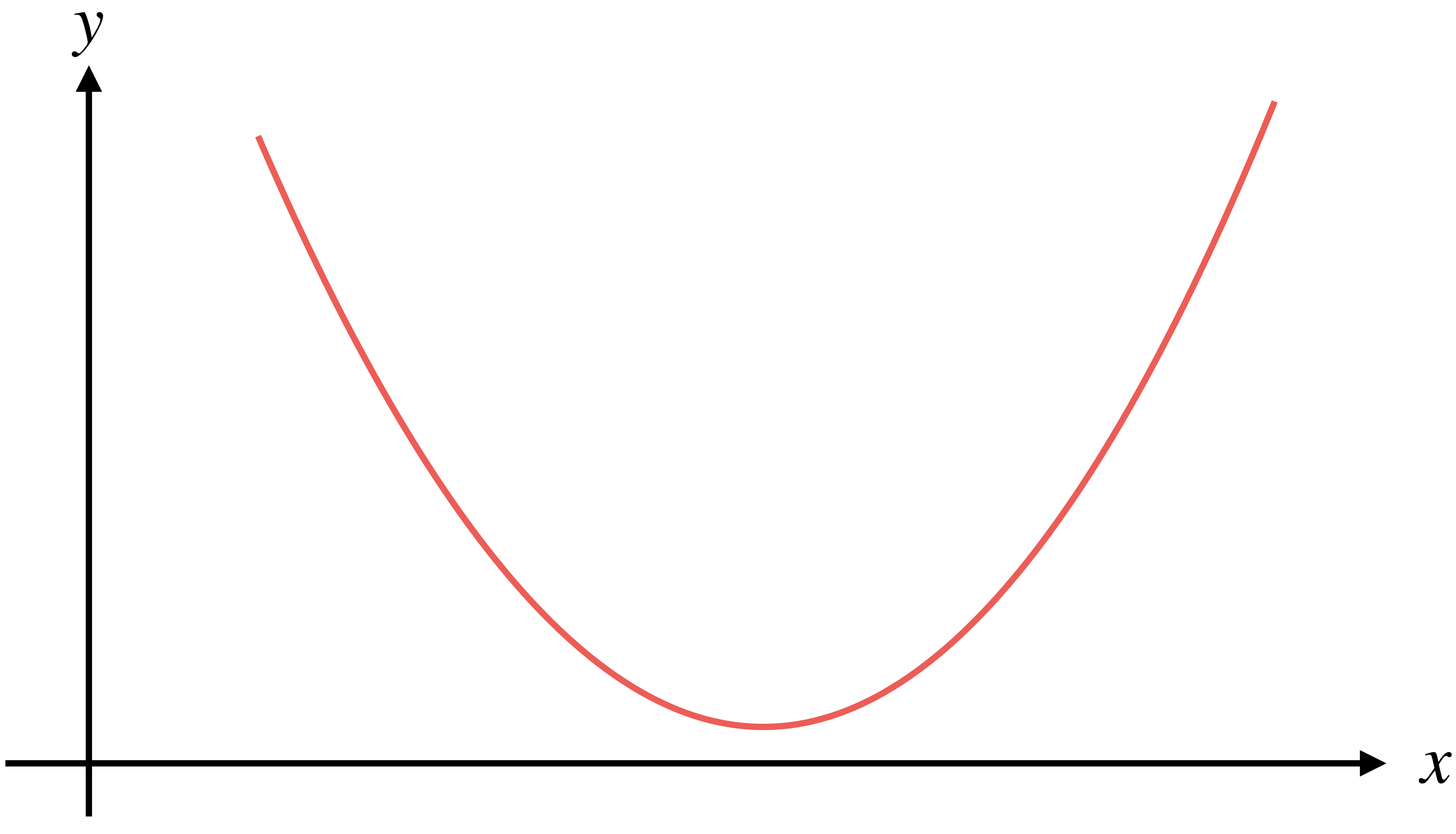
$$J(\theta) = \frac{1}{2}\sum_{j=1}^{m}\left(g\left([\mathbf{p},1]^\top \cdot \theta\right) - l^j\right)^2$$

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{m}\left(\frac{1}{1 + \exp(-[\mathbf{p},1]^\top \cdot \theta)} - l^j\right)^2$$

Why Non-Linear:
Error Function is Non-Convex



Why Non-Linear: We Want a Convex Error Function



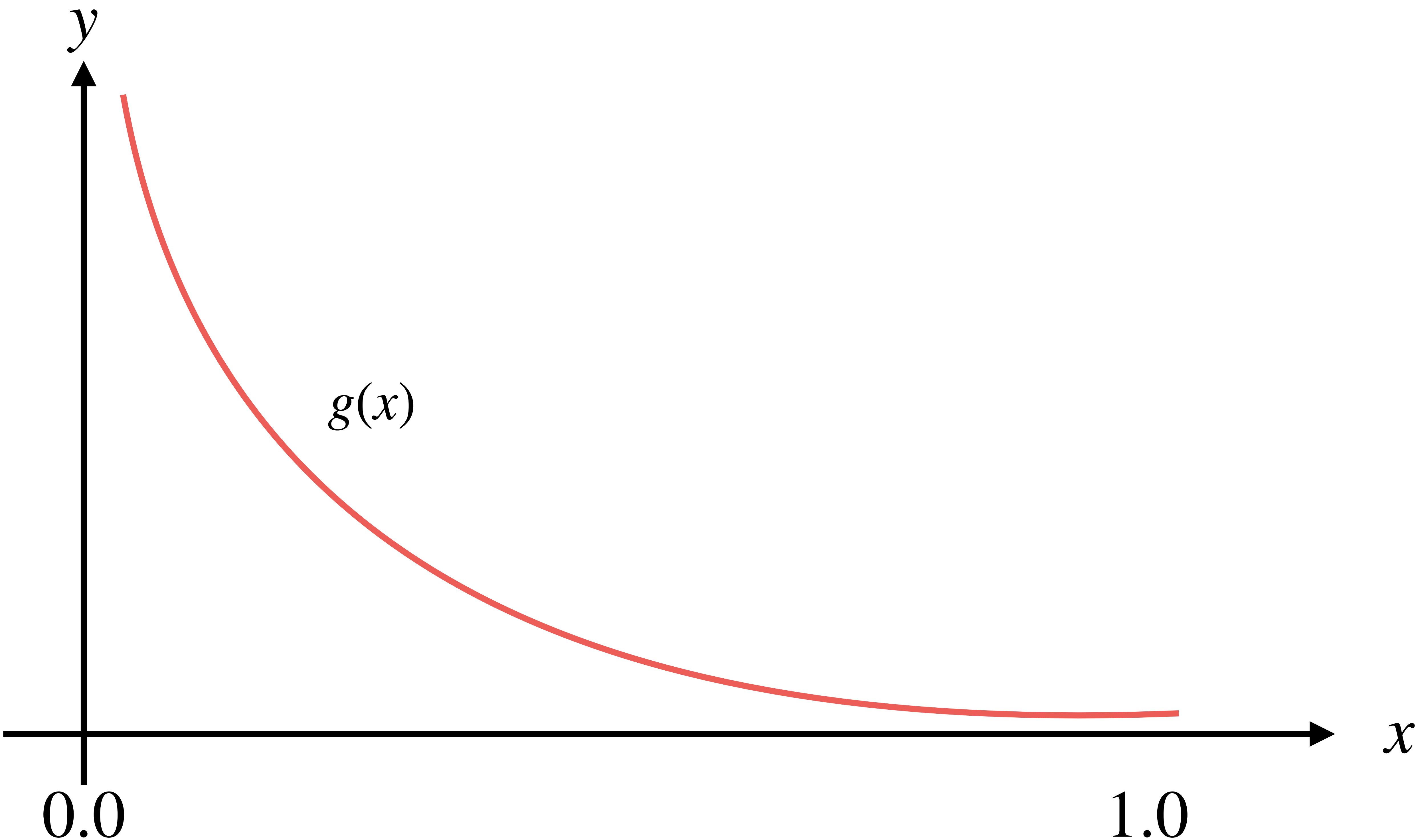
Why Non-Linear: Sigmoid + Error Function

- We define a cost function:

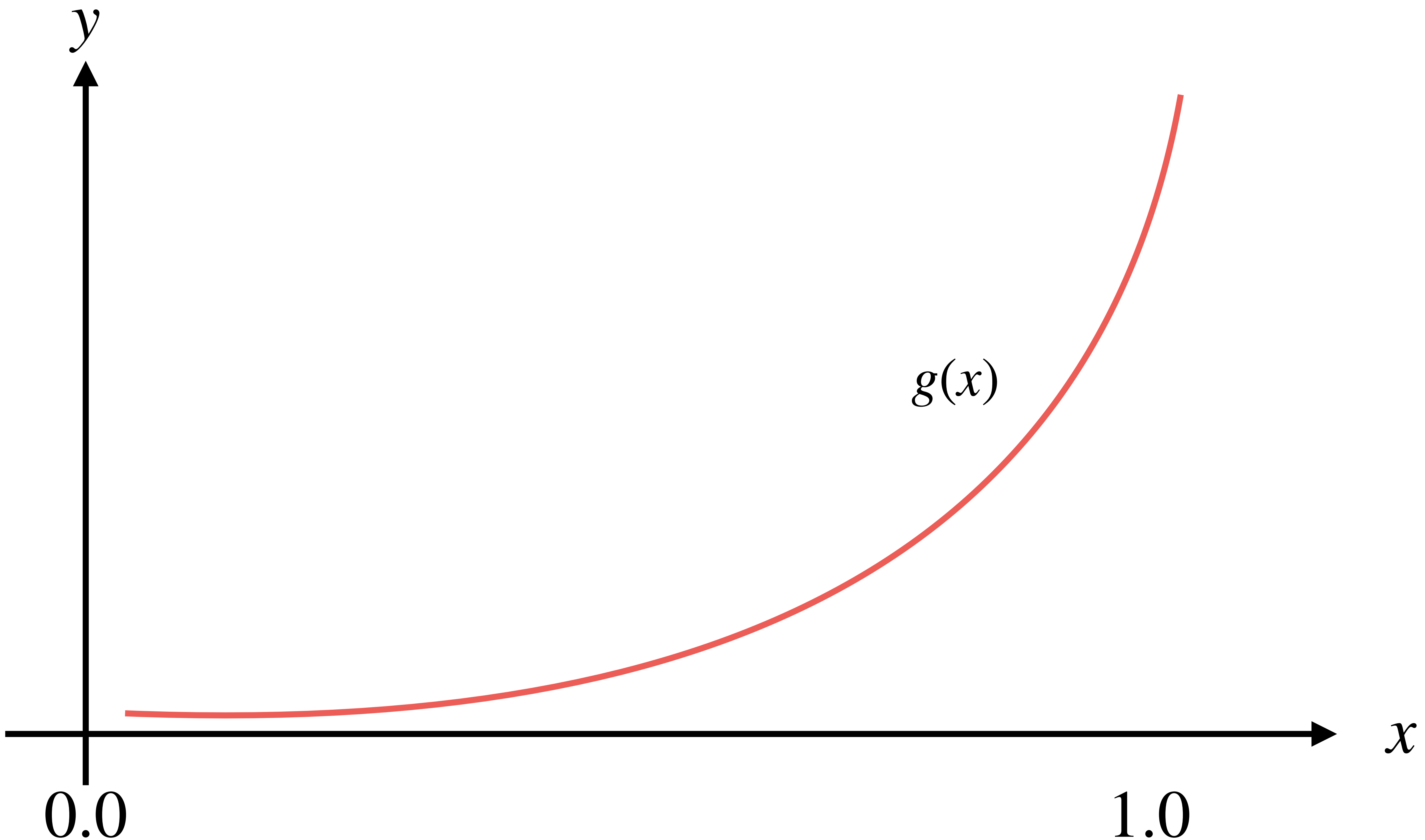
$$C(x, l) = \begin{cases} -\log(g(x)) & \text{if } l = 1 \\ -\log(1 - g(x)) & \text{if } l = 0 \end{cases}$$

where: $g(x) = \dfrac{1}{1 + e^{-x}}$.

Why Non-Linear:
If $l = 1$



Why Non-Linear:
If $l = 0$



Why Non-Linear: Sigmoid + Error Function

- Using the cost, we have our new error function:

$$J(\theta) = \frac{1}{2} \sum_{j=1}^{m} C\Big(g(x^j), l^j\Big)$$

where $x^j = [\mathbf{p}^j, 1]^\top \cdot \theta$.

- This can be simplified into:

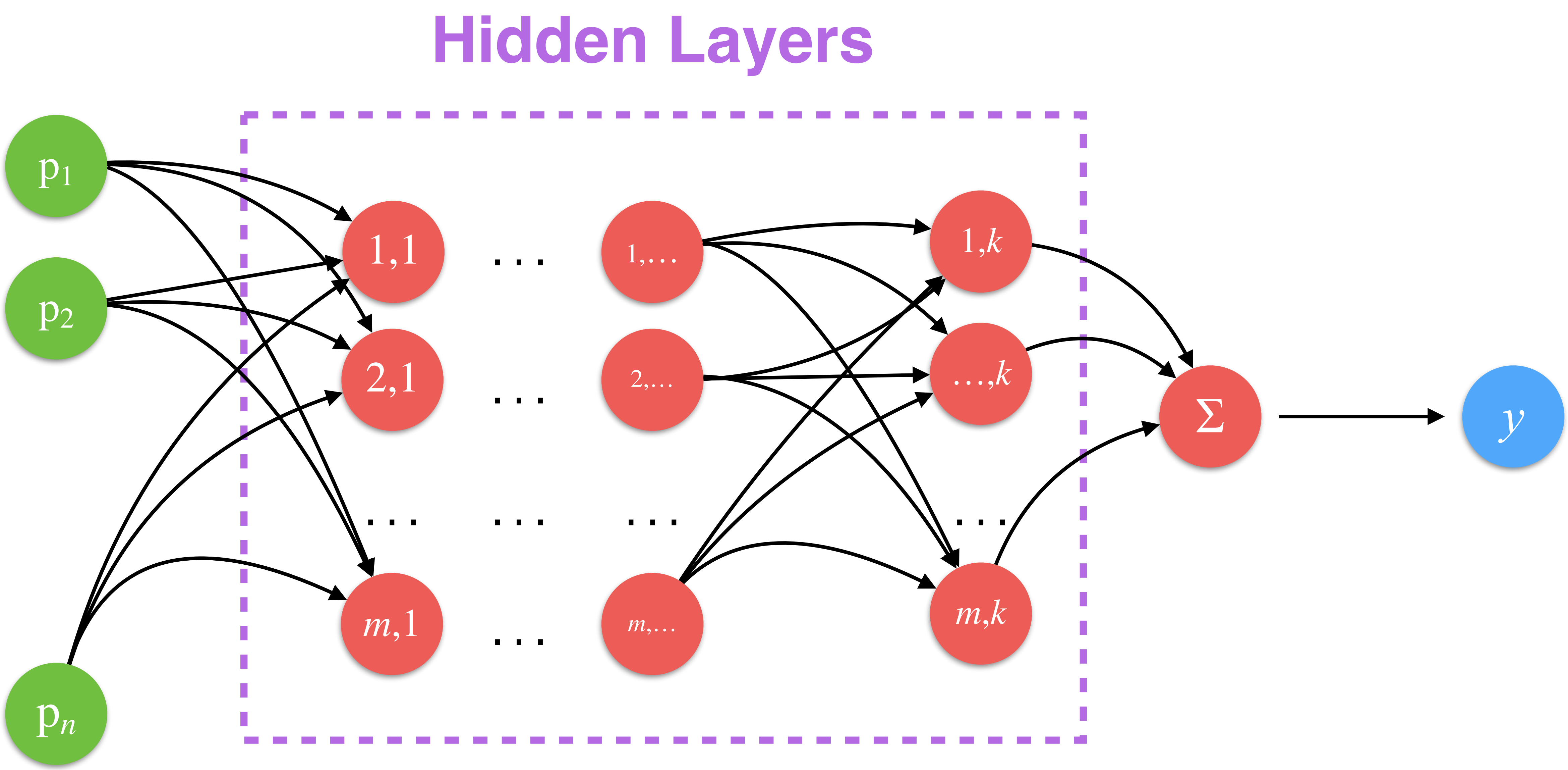
$$J(\theta) = \frac{1}{2} \sum_{j=1}^{m} y^j \log g(x^j) + (1 - y^j)\log(1 - g(x^j))$$

More Complex Examples

More Complex Nets

- To achieve high-quality results, a model 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



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

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!