CSCE 689- Deep Learning for Computer Graphics

Basics of deep learning I

Nima Kalantari

Recap

Learning categories

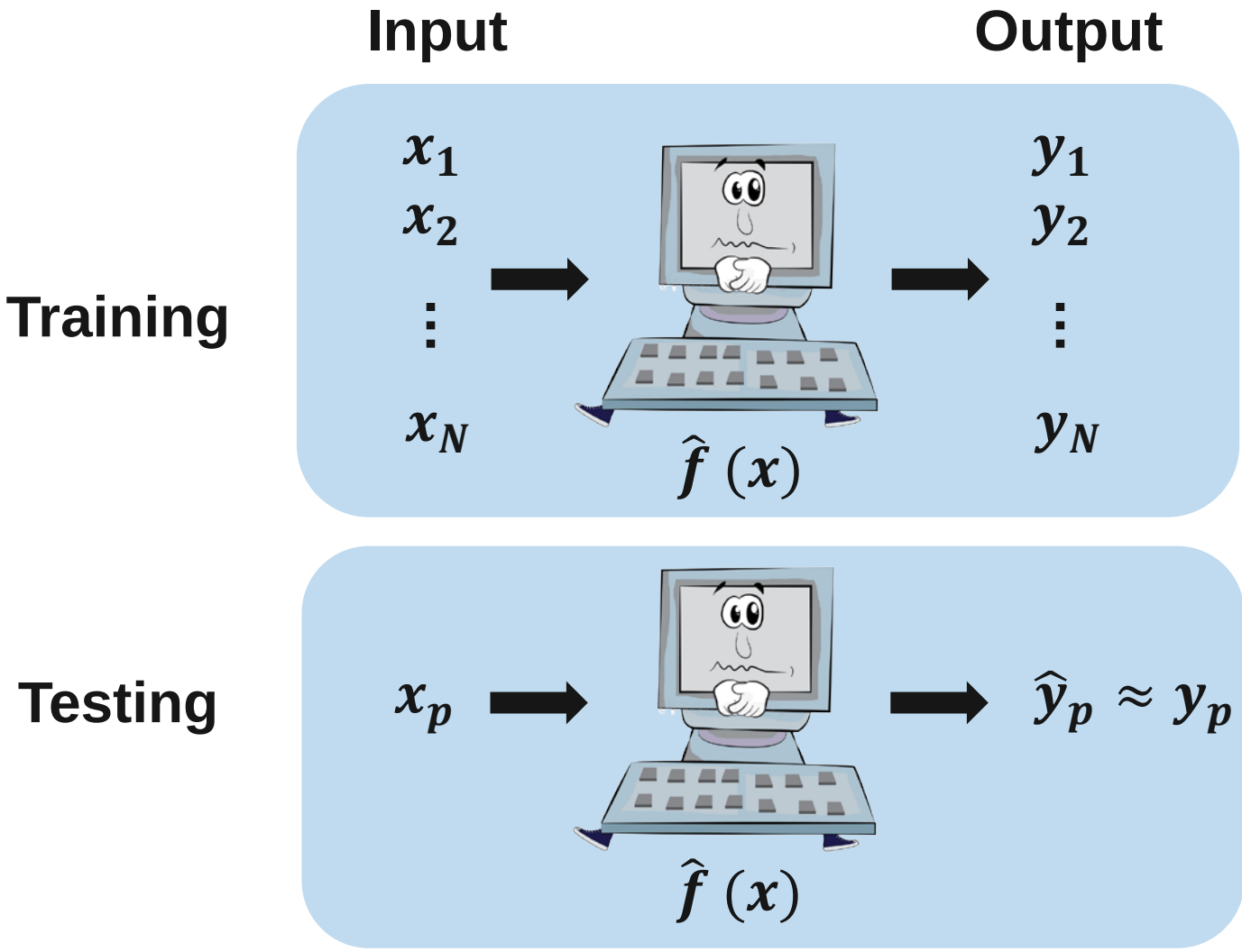
- **Supervised:** the inputs and desired outputs are provided for the machine
- **Unsupervised:** output is not provided and the machine needs to discover the structure in the inputs

We study supervised methods in this class!

Supervised learning

- **Classification:** the output is class label(s)
  - e.g., Image classification
- **Regression:** the output is real number(s)
  - e.g., crime data mining

Supervised learning



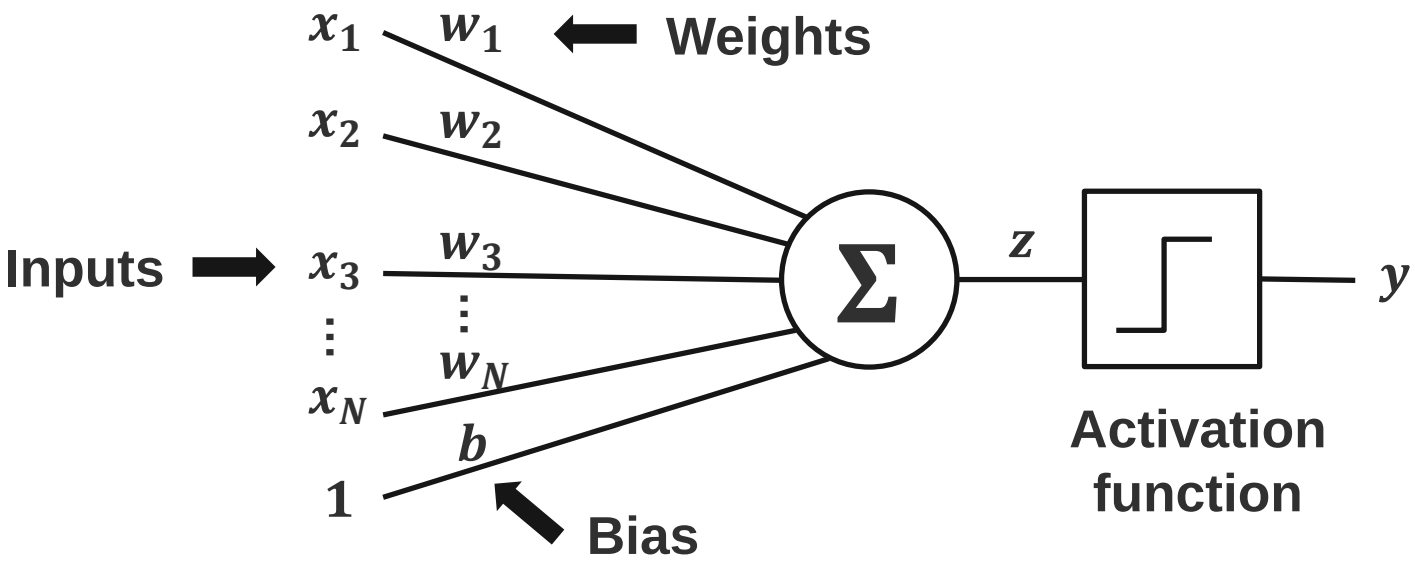
Important aspects

- Choice of model
  - Underfitting
  - Overfitting
- Loss
- Optimization strategy
- Training data
  - Large
  - Diverse

This class

Perceptron (1943)

- Simplest form of neural networks



$$z = b + \sum_{i=1}^{N} w_i x_i$$

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

Example

- Determine if picture of a cat or dog



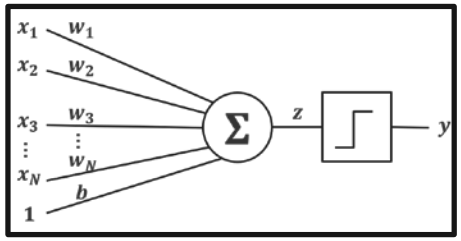



Example

Extract features

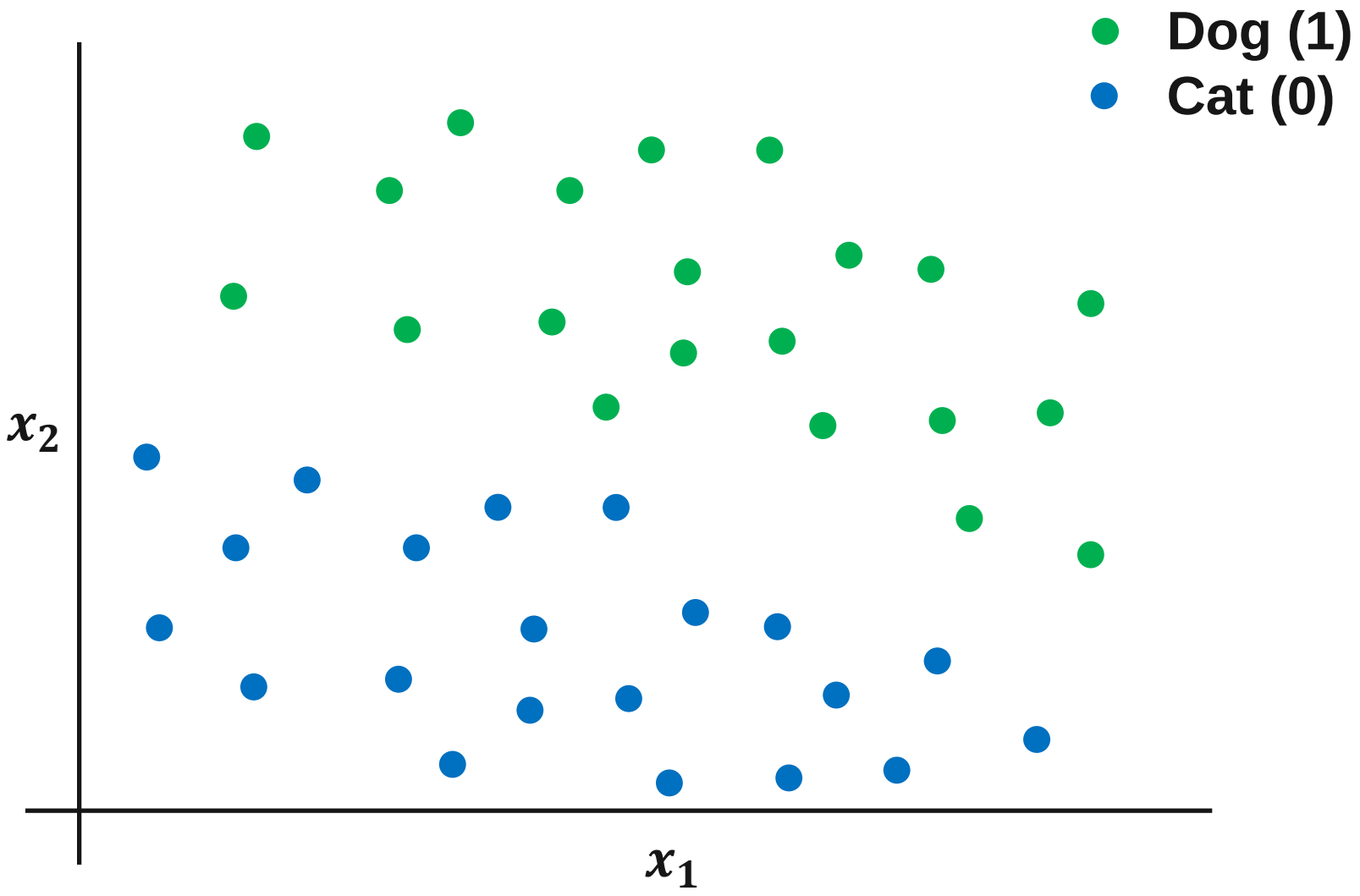
$$X = \{x_1, x_2, \dots, x_N\}$$

X

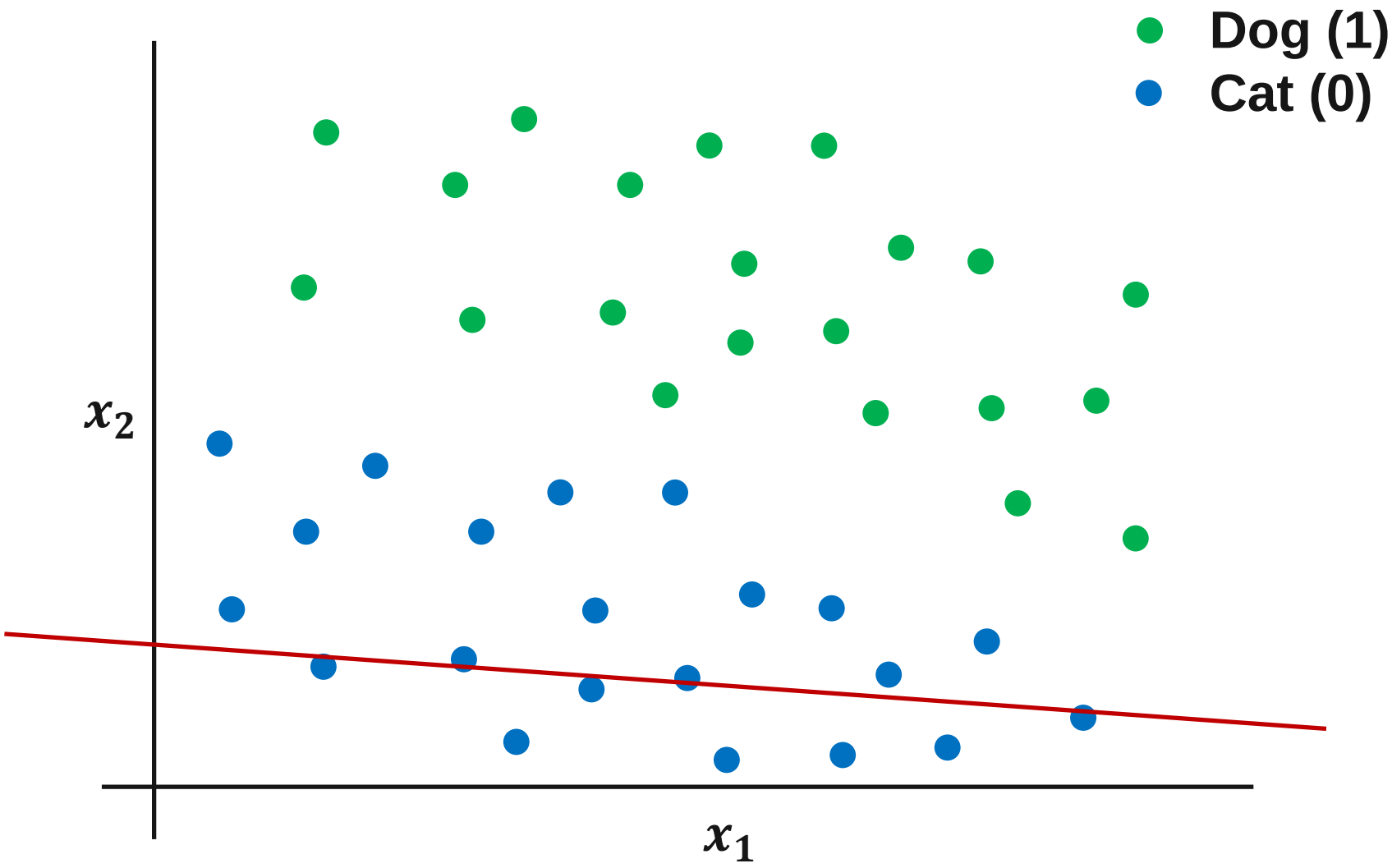


$$\begin{cases} 1 & \text{Dog} \\ 0 & \text{Cat} \end{cases}$$

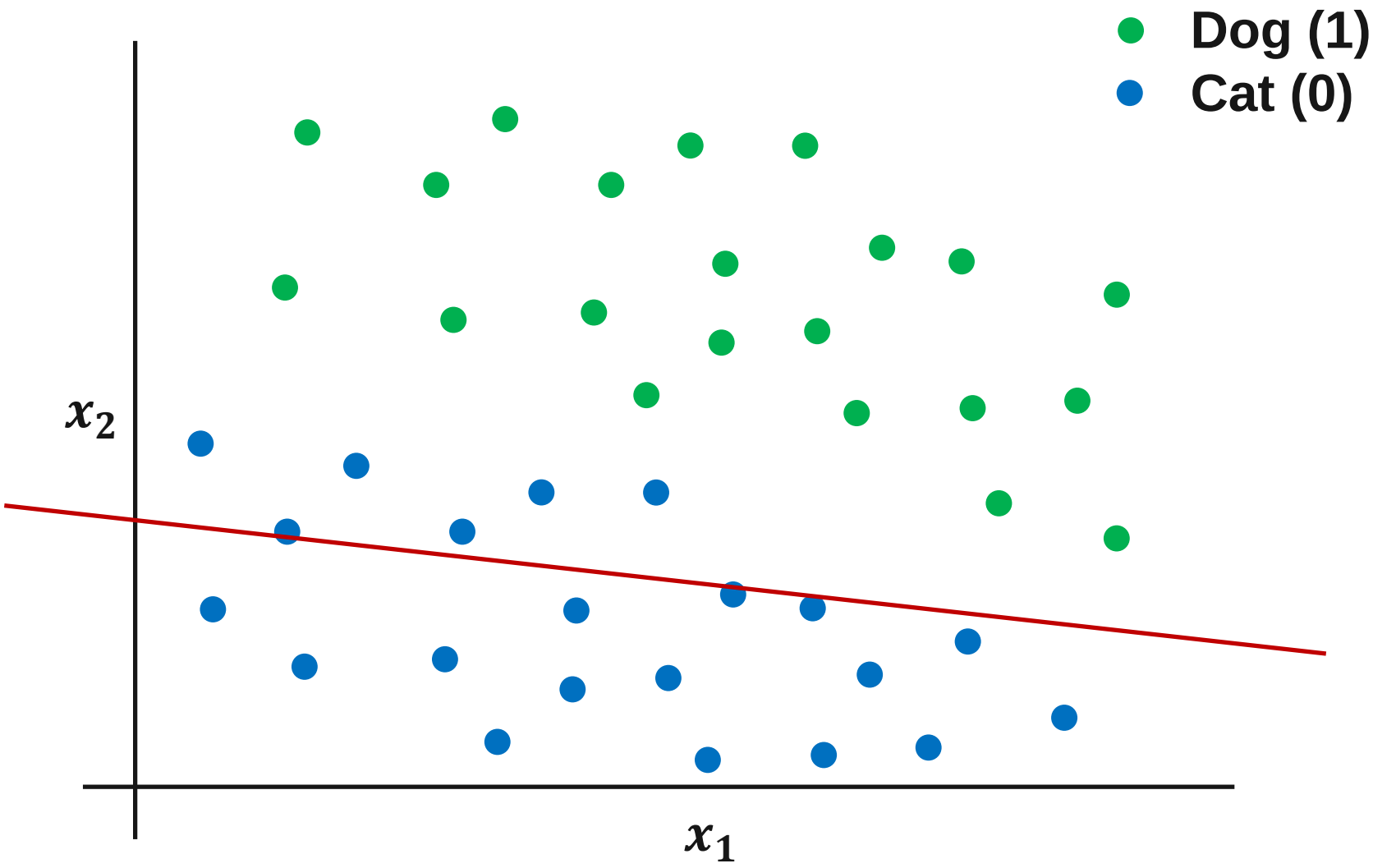
Example



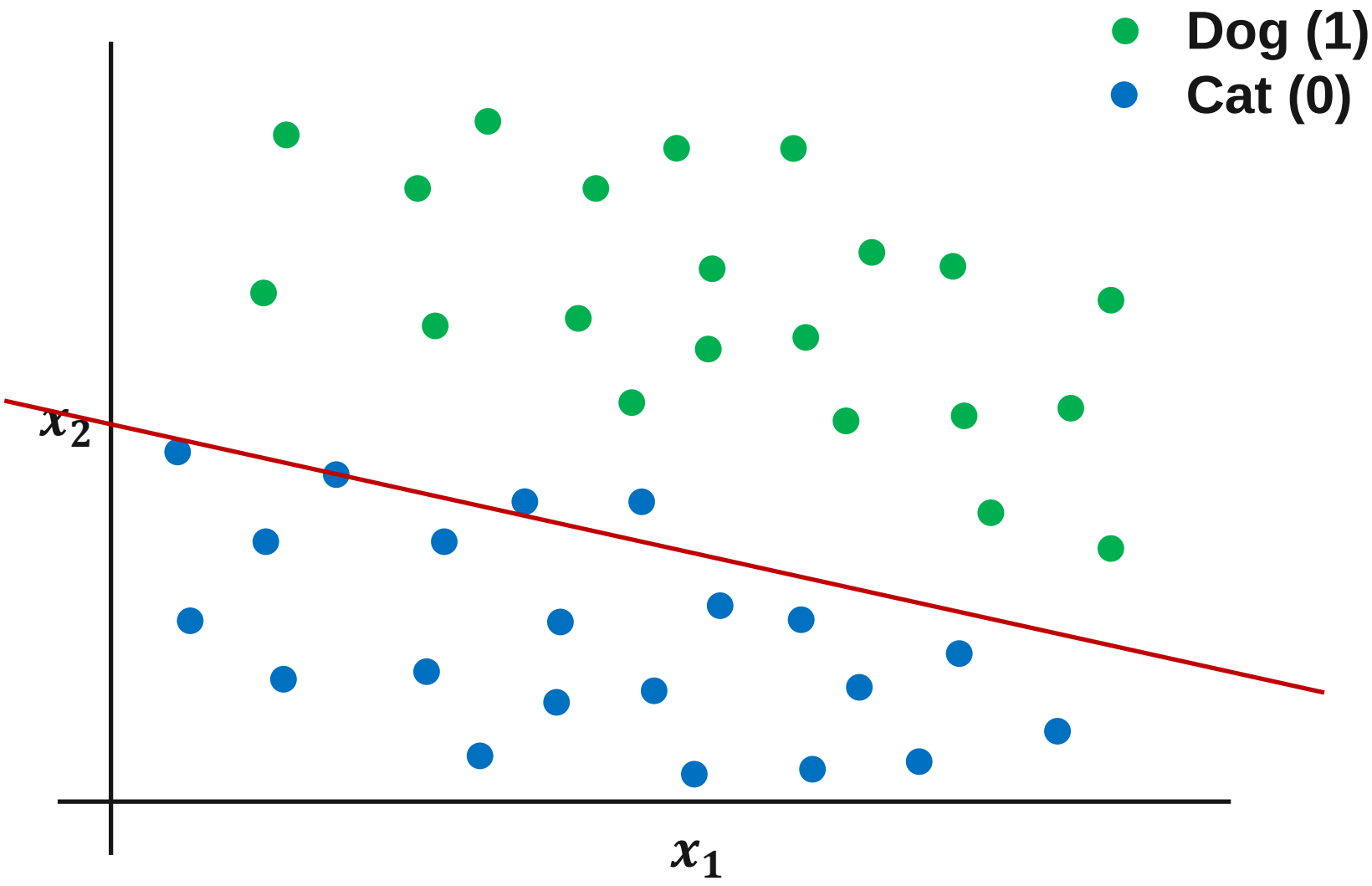
Example



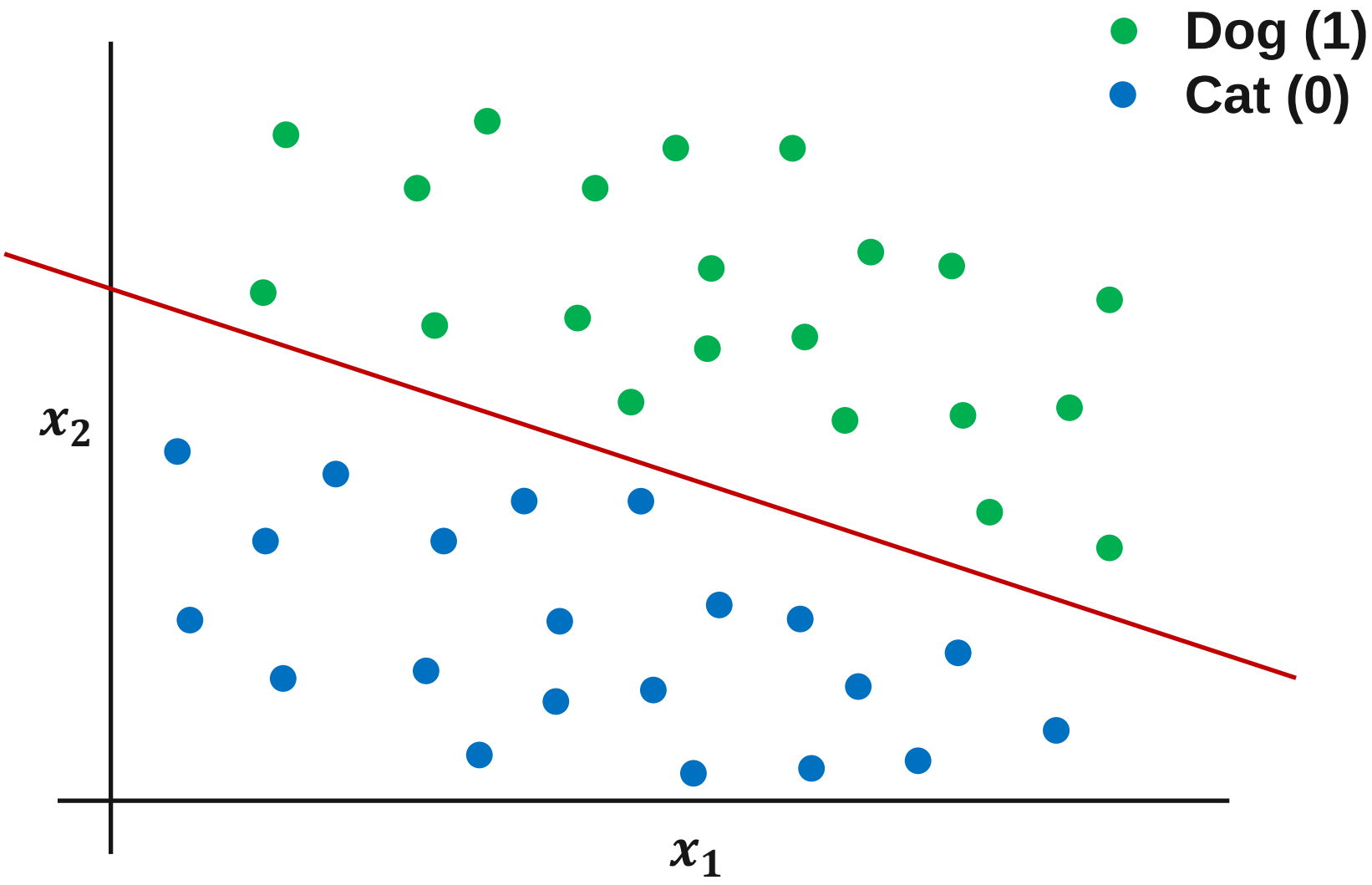
Example



Example



Example



Training

- Need a set of training examples

→ Extract features →

X_1 y_1

X_2 y_2

X_3 y_3 ➡ Dog (1)

⋮ ⋮

X_N y_N

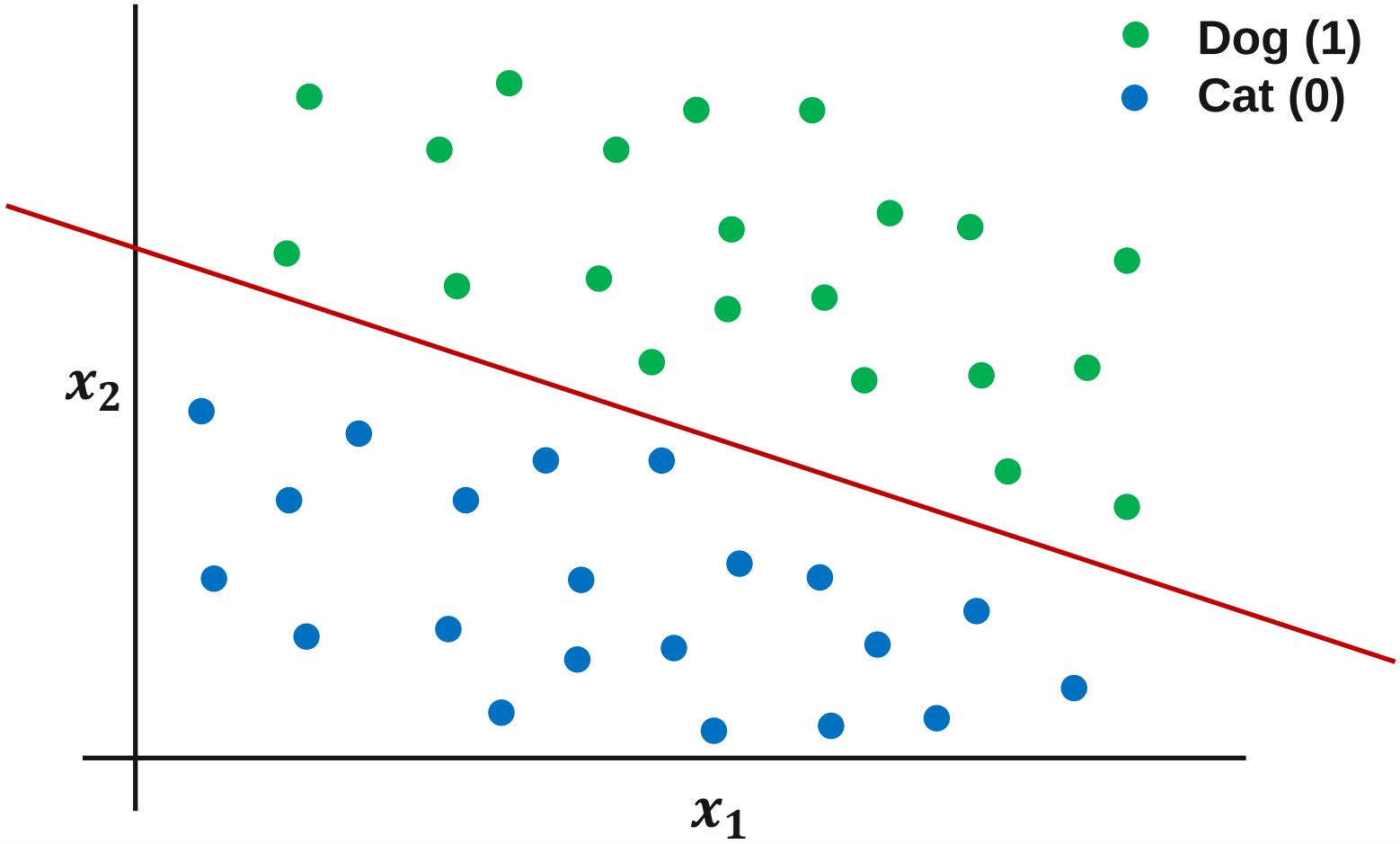
Training

- Initialize the weights and the bias randomly
- For correct output, leave the weights as is
- If the output is zero incorrectly, add input to the weight
- If the output is one incorrectly, subtract the input from the weight

Guaranteed to work if such a solution exists.

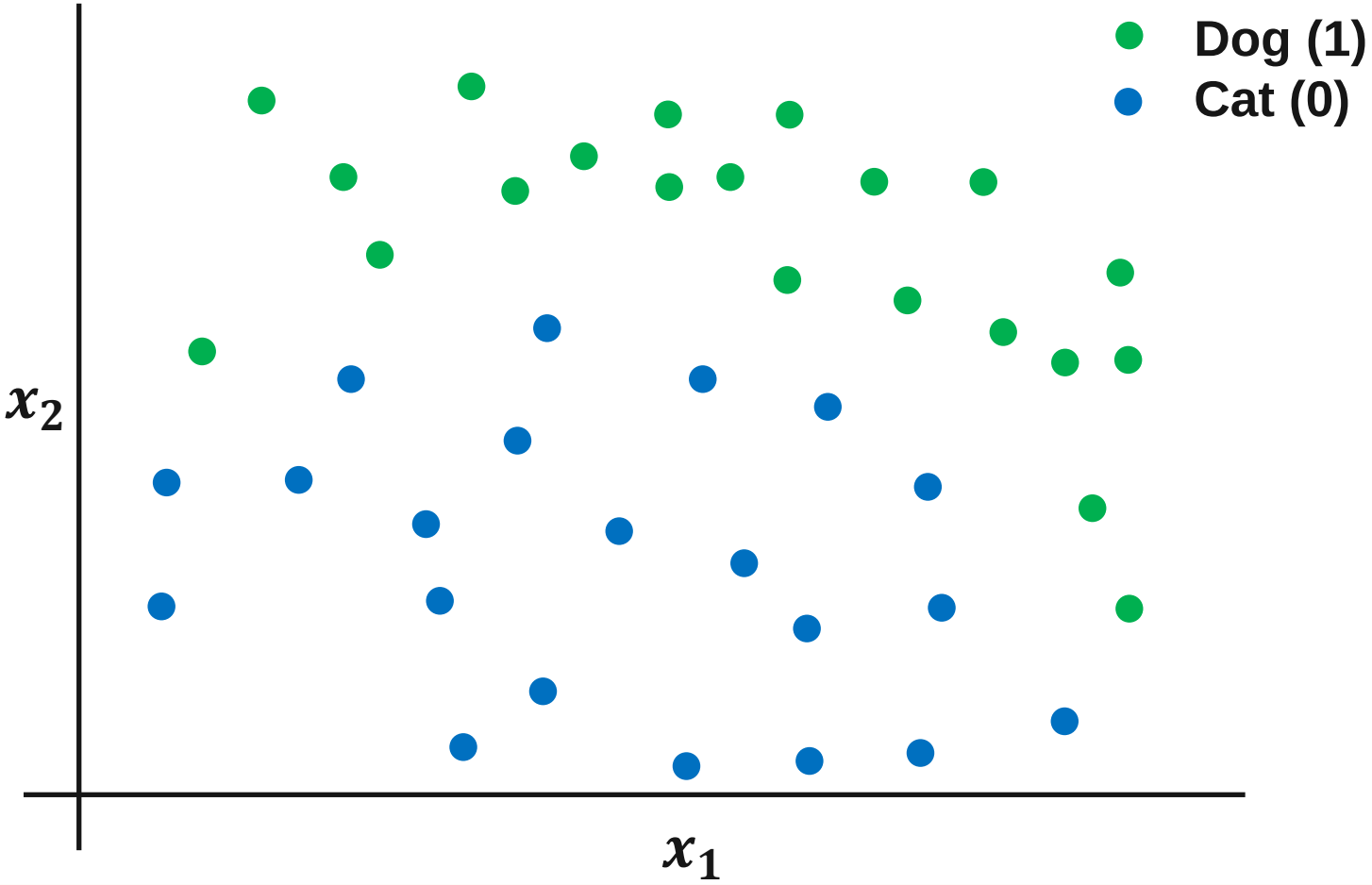
However…

- A solution does not always exist

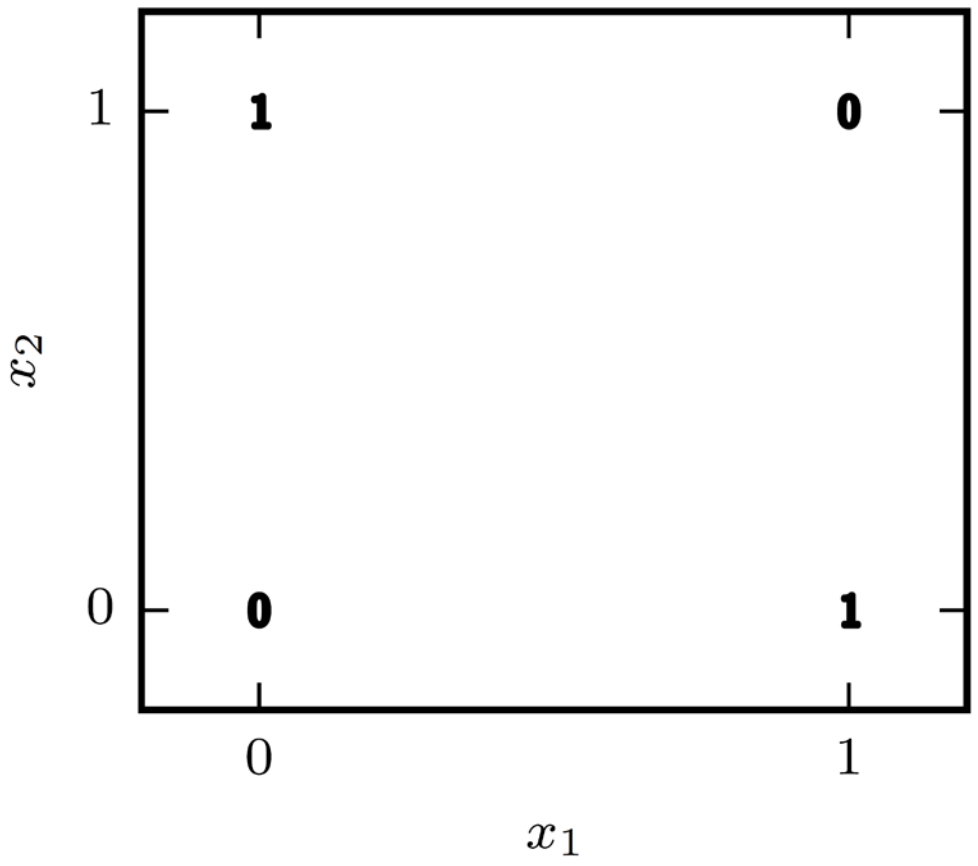


However…

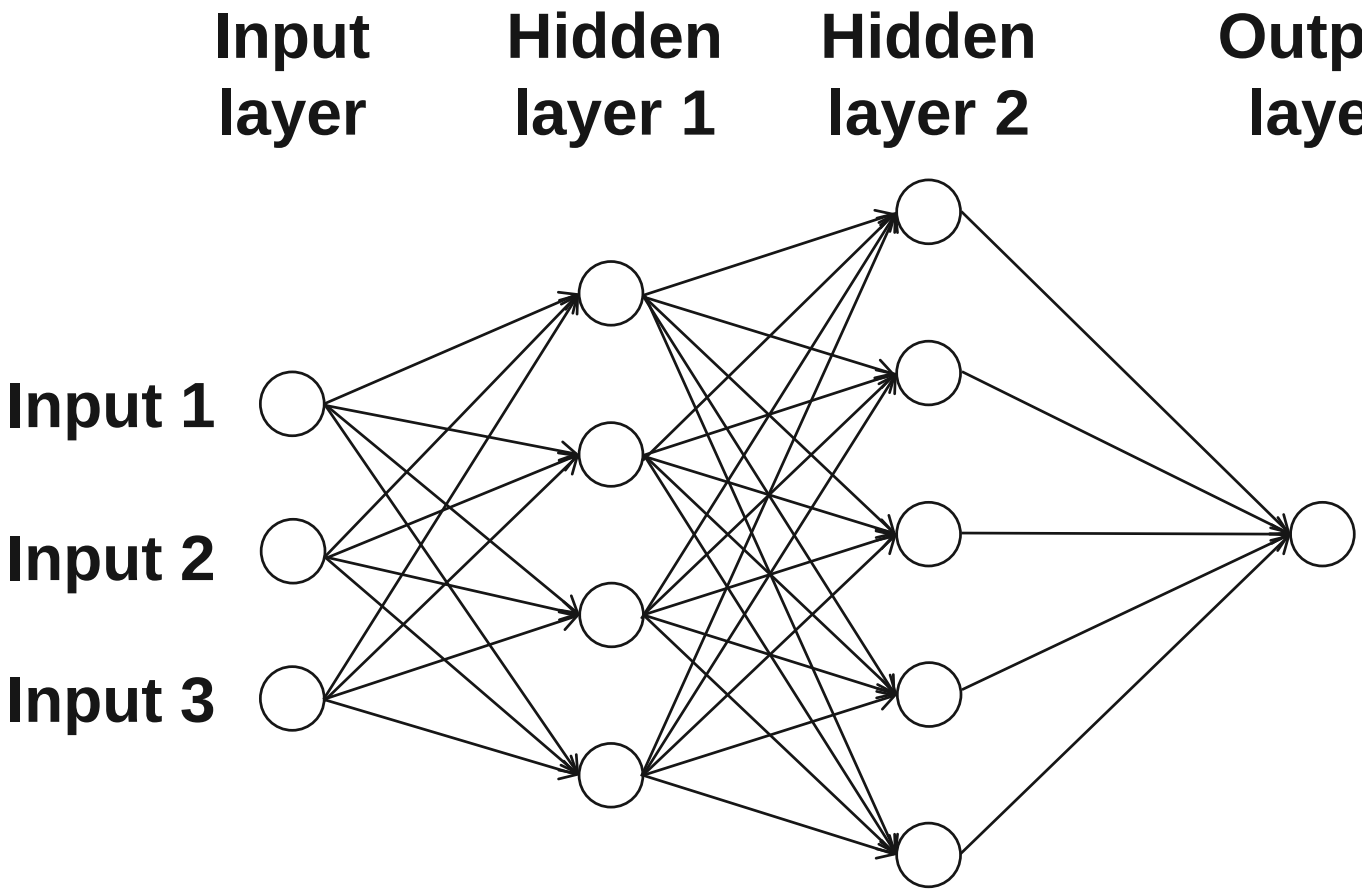
- A solution does not always exist



XOR problem

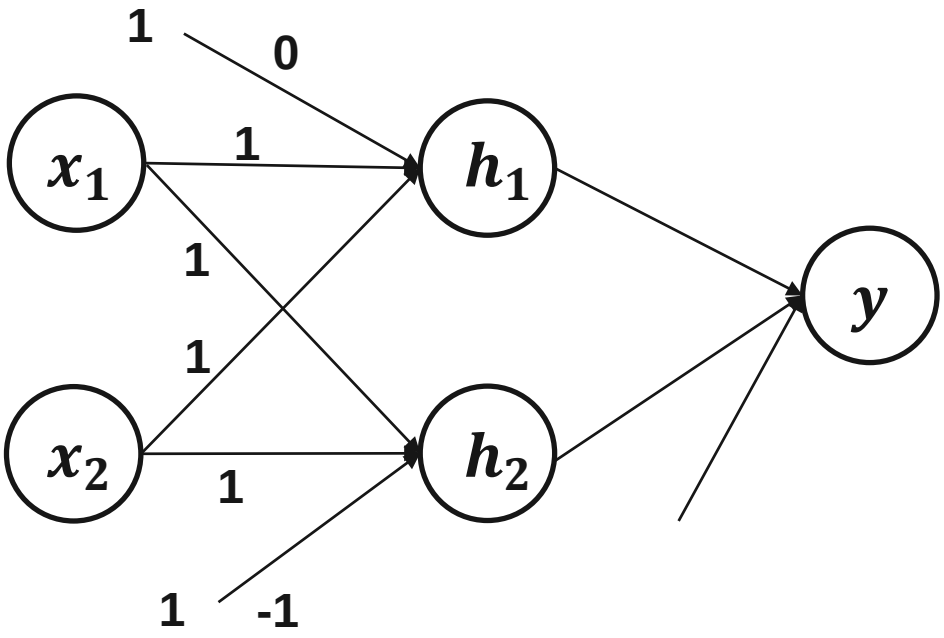


Multilayer perceptron



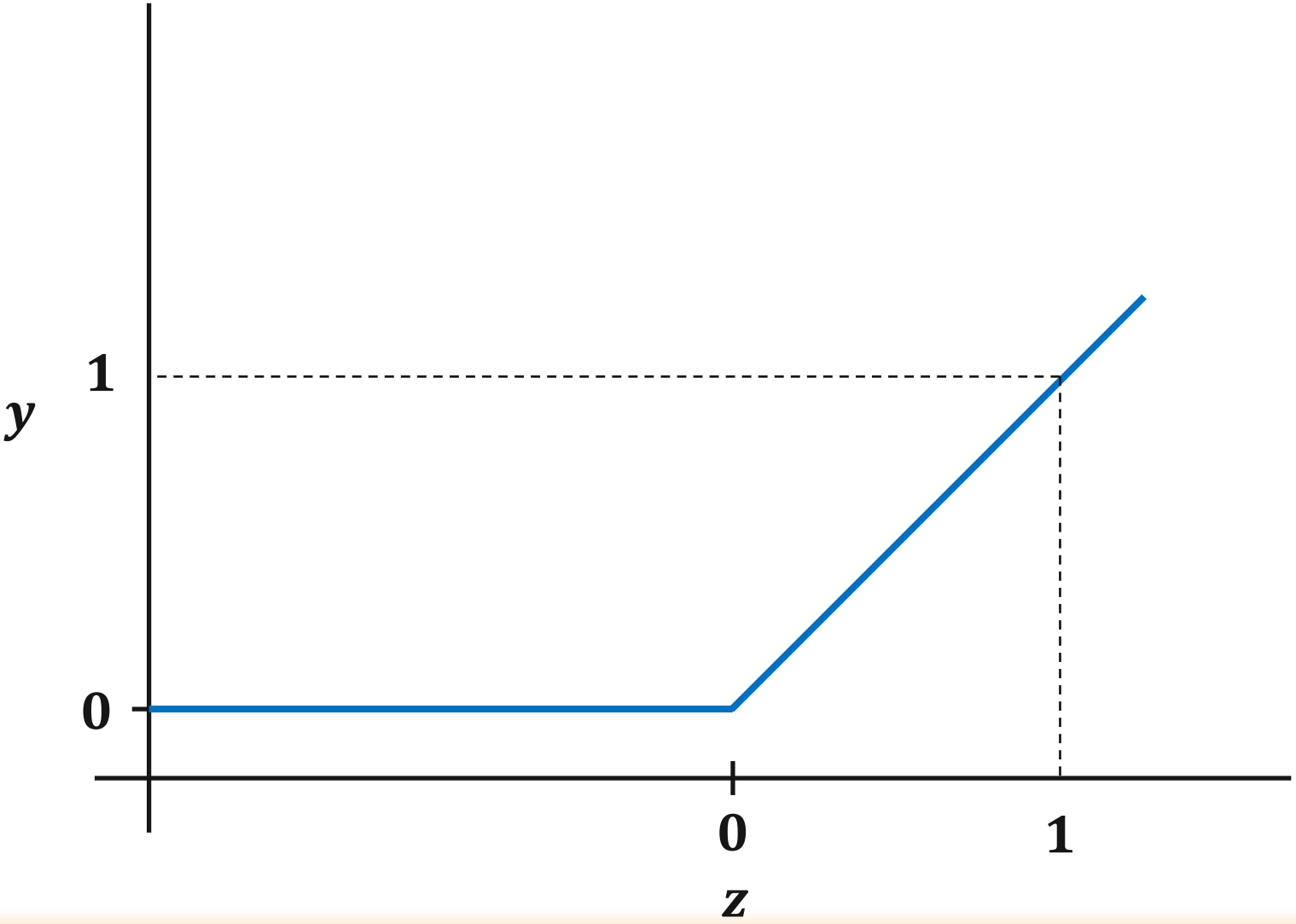
Hidden layers should have non-linear activation functions!

XOR problem

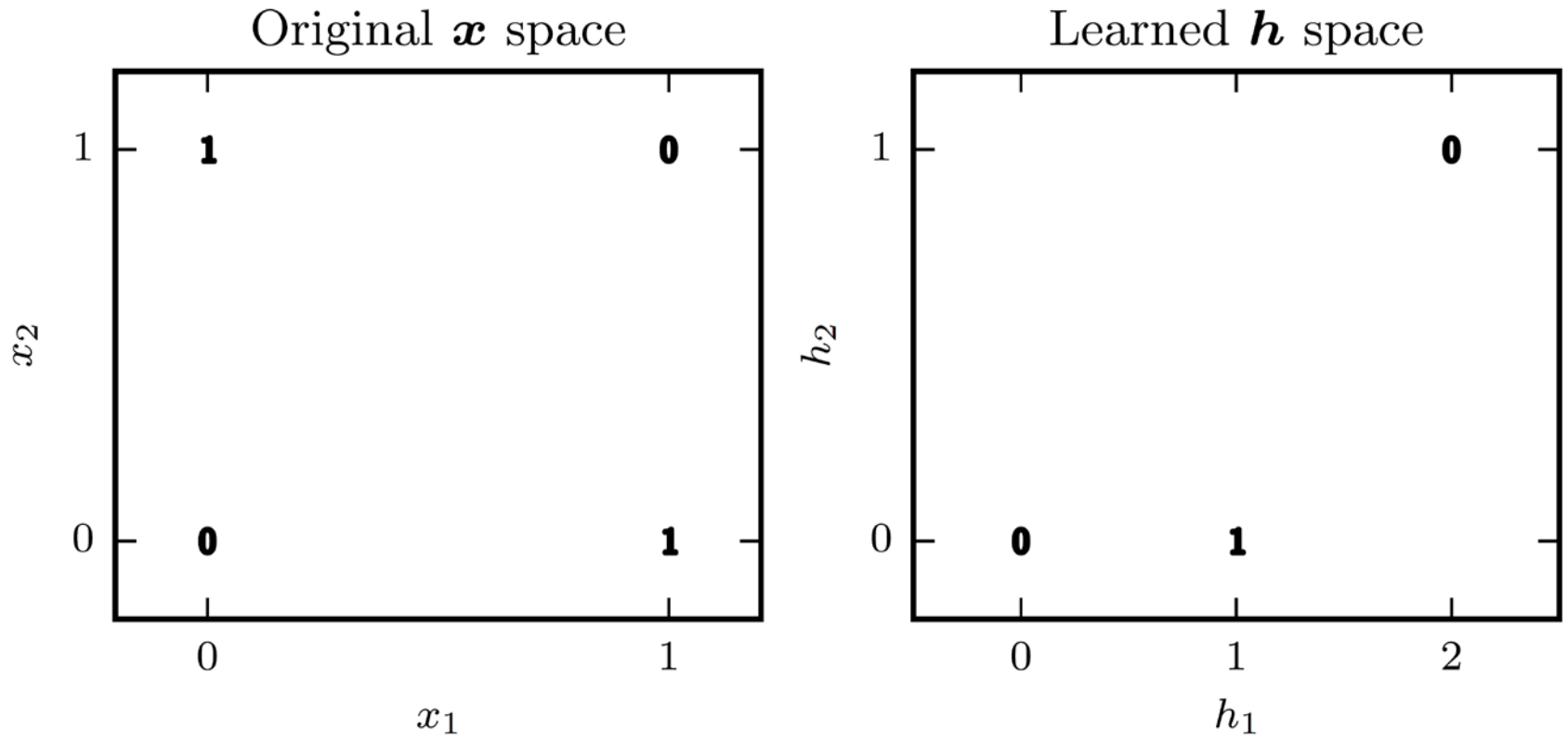


Slide based on Goodfellow, Bengio, and Courville

Rectified linear unit



XOR problem



Slide based on Goodfellow, Bengio, and Courville

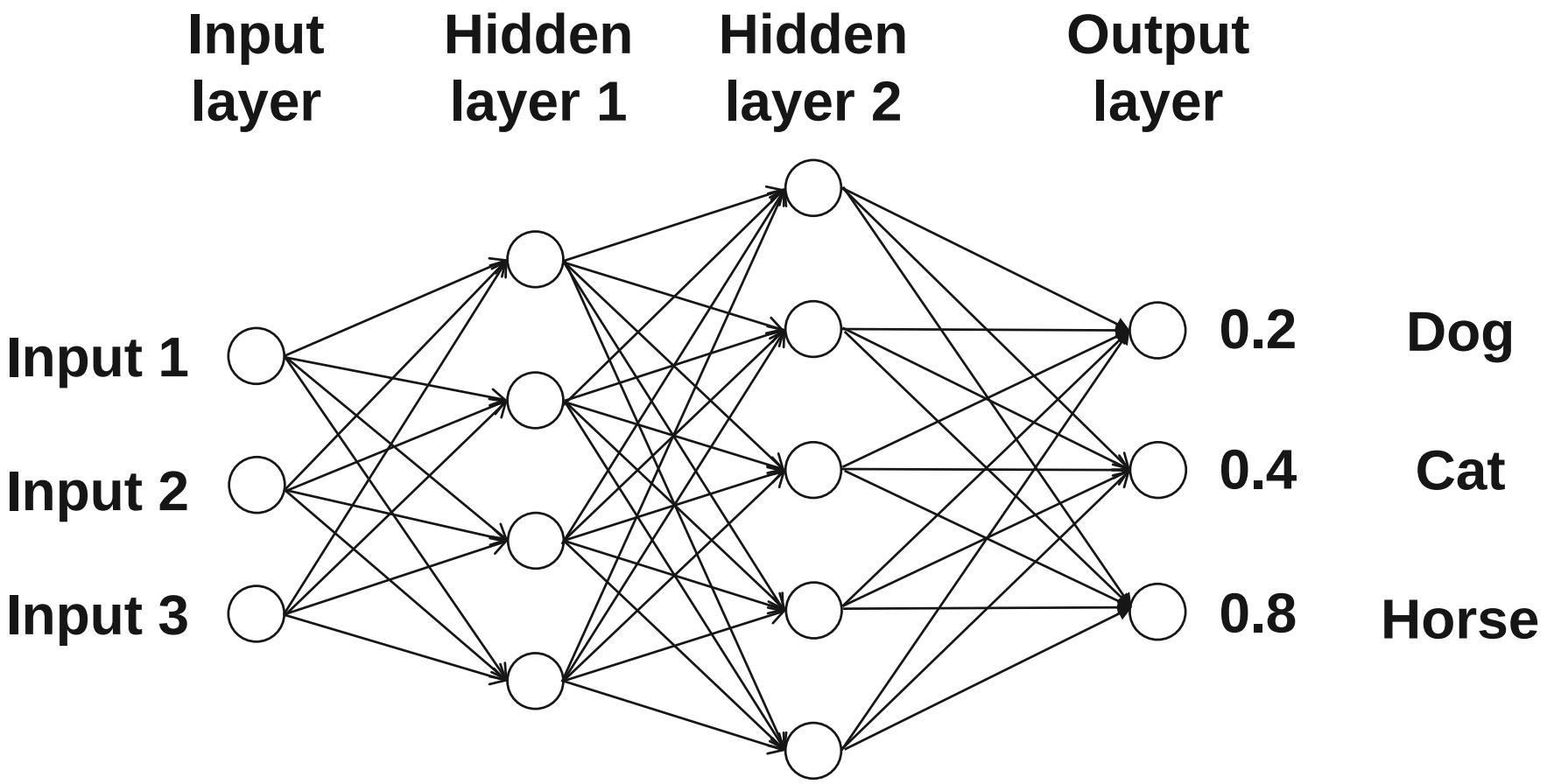
Universal approximation theorem

- A multi layer perceptron with a single hidden layer containing a finite number of hidden neurons can represent any bounded continuous function
- Why not use one hidden layer all the time?
  - Need large number of hidden neurons for complex processes
  - Optimization is difficult

So far…

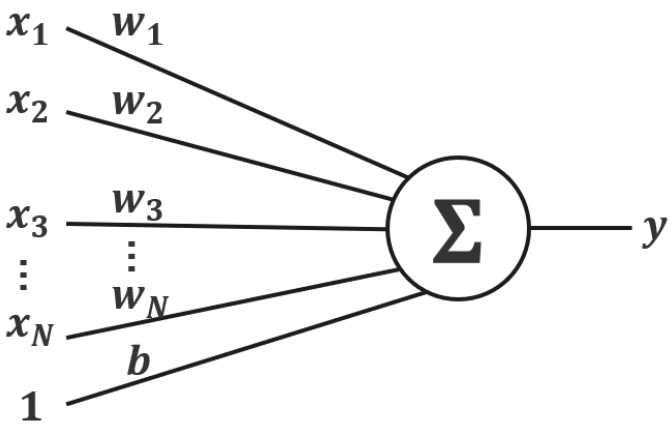
- Image classification (Dogs and Cats)
- XOR problem (0, 1)
- What if we have more than two classes?

Multilayer perceptron



Training

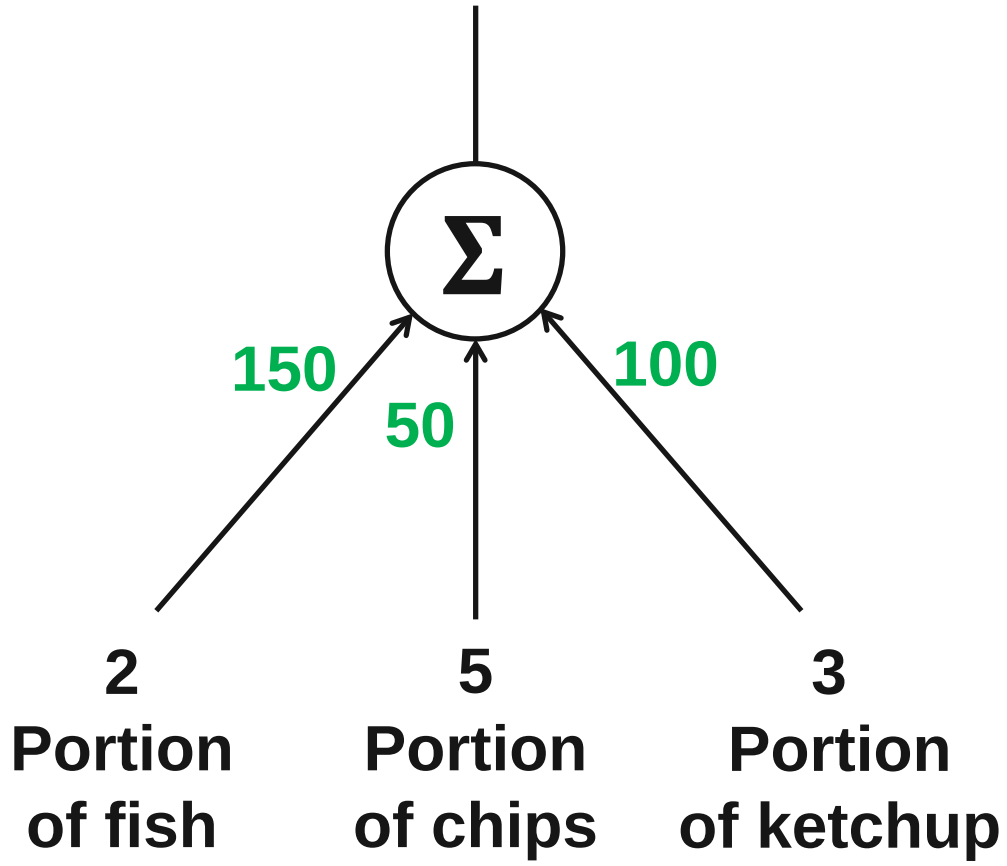
- Linear neuron
- Neuron has a real valued output, which is a weighted sum of its inputs
- Goal is to minimize the error summed over all training cases
  - Error is squared difference between the desired and actual outputs



$$y = \sum_{i=1}^{N} w_i x_i$$

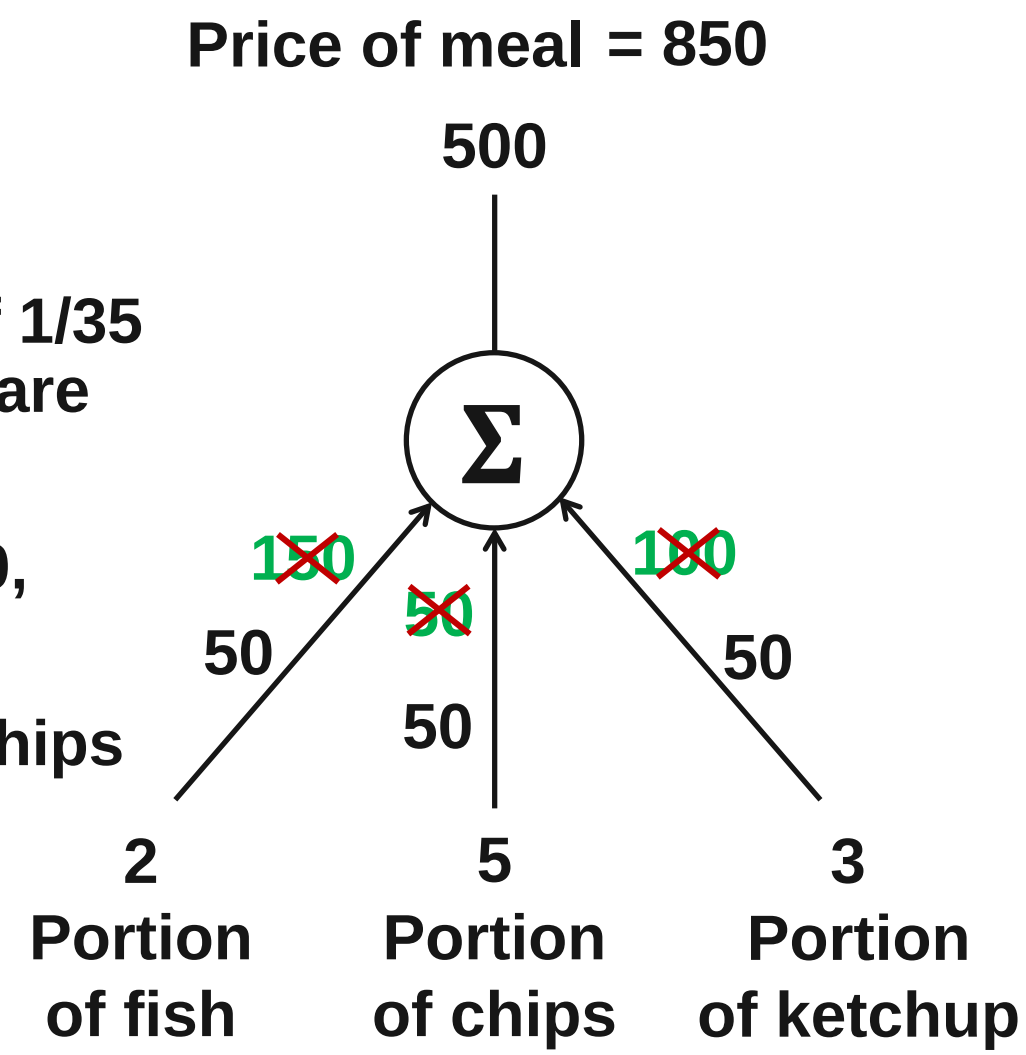
Toy example

Price of meal = 850



Toy example

- Residual error = 350
- The "delta rule" is:
  $\Delta w_i = \varepsilon x_i (t - y)$
- With learning rate of 1/35 the weight changes are +20, +50, and +30
- New weights are +70, +100, and +80
  - The weight for chips got worse



Deriving the delta rule

- Error is squared diff. summed over all training cases

$$E = \frac{1}{2}\sum_{n \in N}(t^n - y^n)^2$$

- Differentiate to get derivative of error wrt. weights

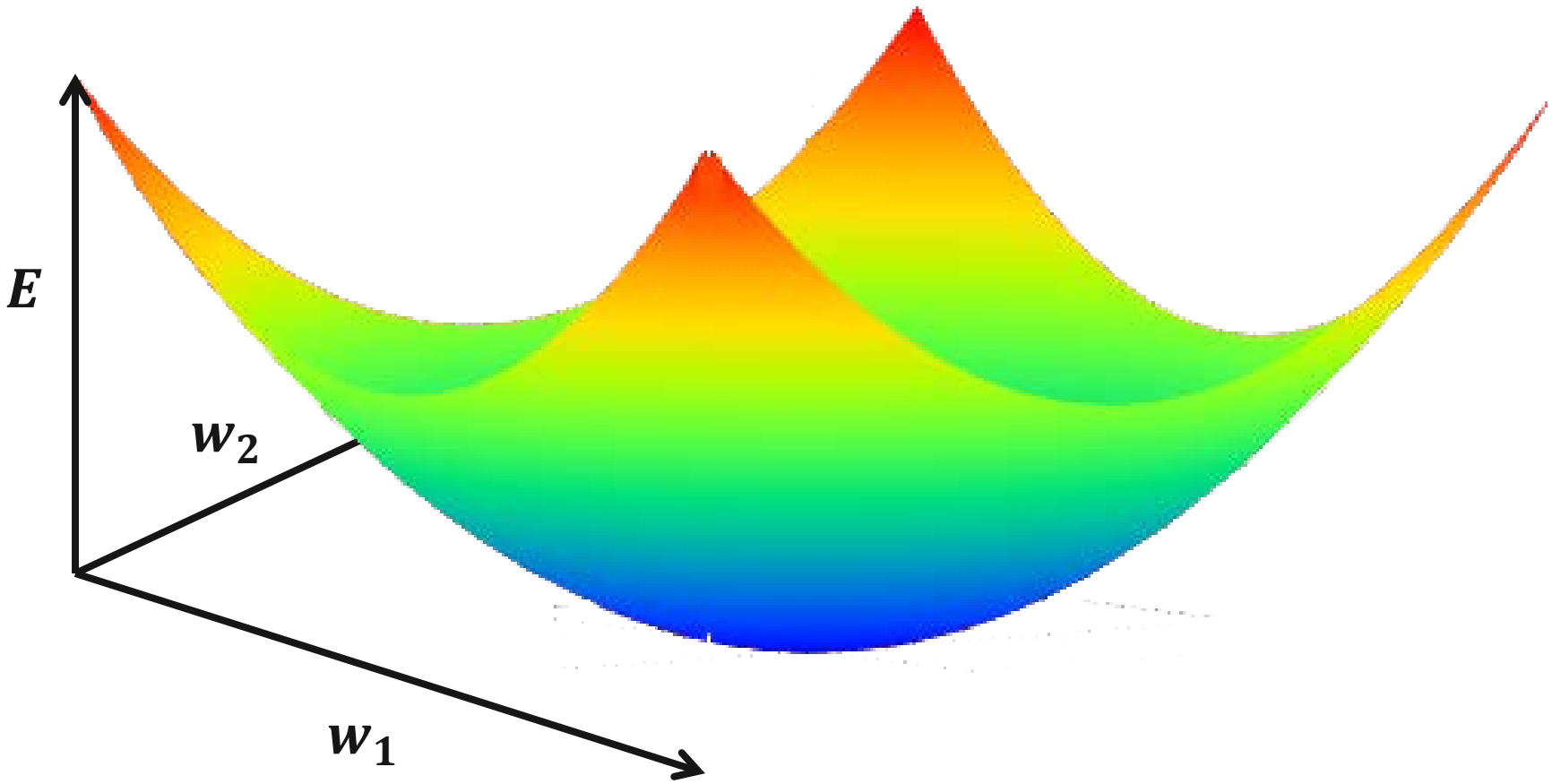
$$\frac{\partial E}{\partial w_i} = \frac{1}{2}\sum_{n \in N}\frac{\partial E}{\partial y^n}\frac{\partial y^n}{\partial w_i}$$

$$= -\sum_{n \in N}(t^n - y^n)\, x_i^n$$

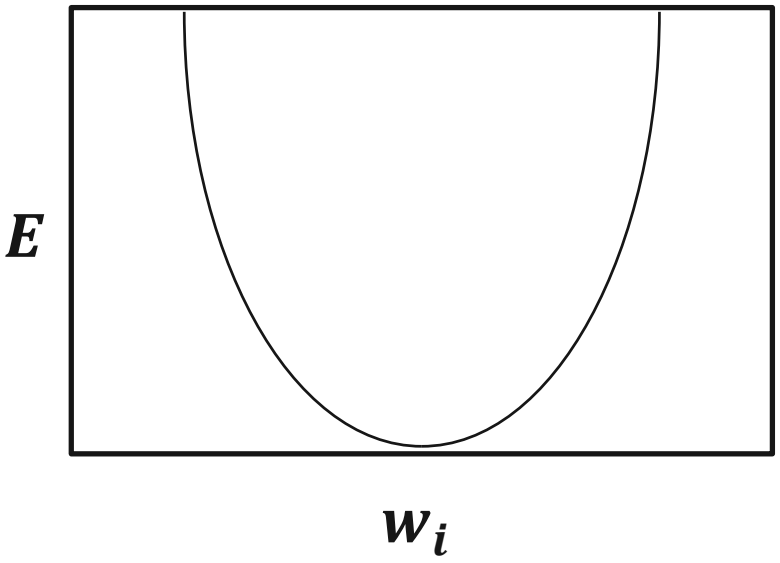
- Learning rule is to change the weight proportional to their error derivative summed over all training cases

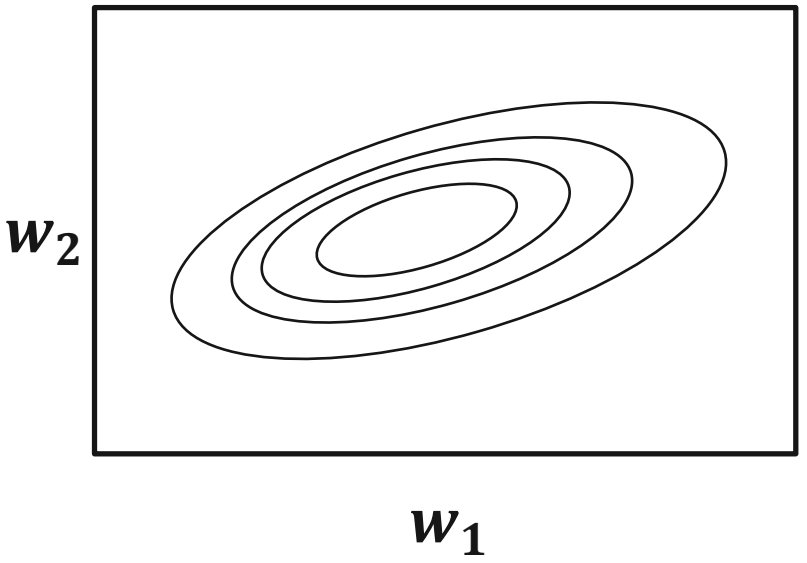
$$\Delta w_i = -\varepsilon\frac{\partial E}{\partial w_i} = \sum_{n \in N}\varepsilon\,(t^n - y^n)x_i^n$$

Error surface



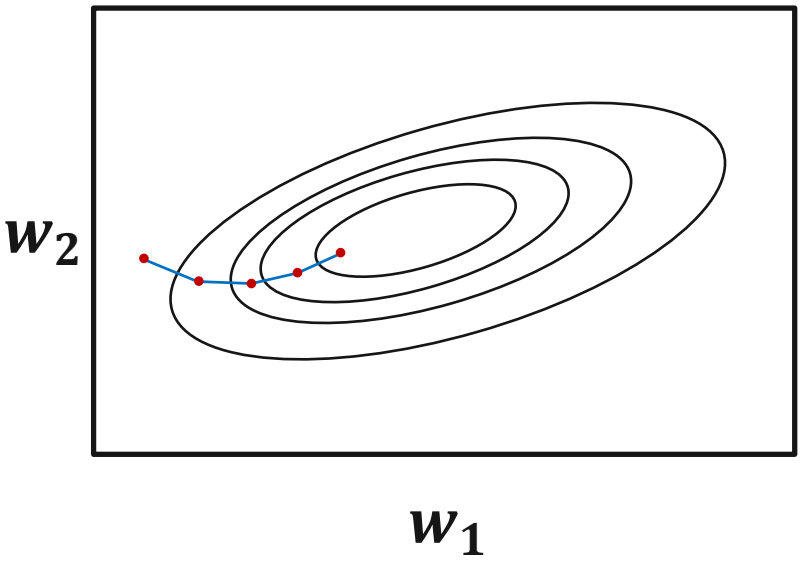
Error surface





Online vs. batch learning

- Batch learning does steepest descent on the error surface
  - perpendicular to the contour lines



- Online learning zig zags around the direction of steepest descent