Universität Konstanz

02 | Introduction to Neural Networks

Giordano De Marzo

https://giordano-demarzo.github.io/

Deep Learning for Social Sciences

Outline

1. Basic Concepts and Notation
2. The Perceptron
3. Limits of the Perceptron
4. Shallow Neural Networks

Basic Concepts and Notation

Universität Konstanz

Learning from Data

Machine learning represents a paradigm shift in how computers solve problems.

Traditional programming: Humans write explicit rules for the computer to follow

Machine learning: Computers discover patterns and rules from examples in data

This data-driven approach can be categorized into **two main types**:

- **Unsupervised learning:** Finding patterns in unlabeled data
- **Supervised learning:** Learning from labeled examples to make predictions

Supervised learning further divides into:

- **Classification:** Predicting categories (spam/not spam, dog/cat)
- **Regression:** Predicting numerical values (price, temperature)

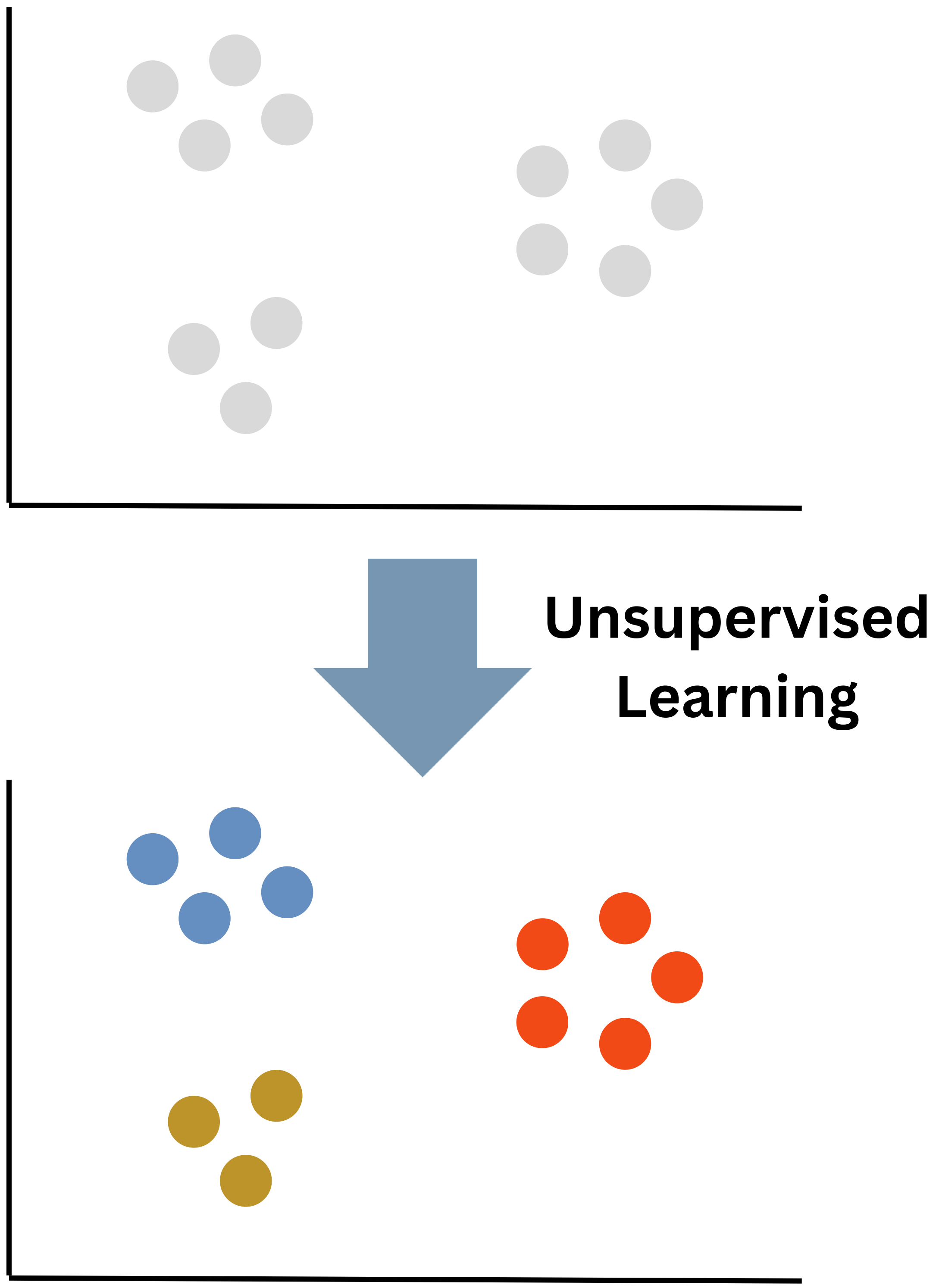
Unsupervised Learning

Unsupervised learning algorithm find structure in data without explicit guidance

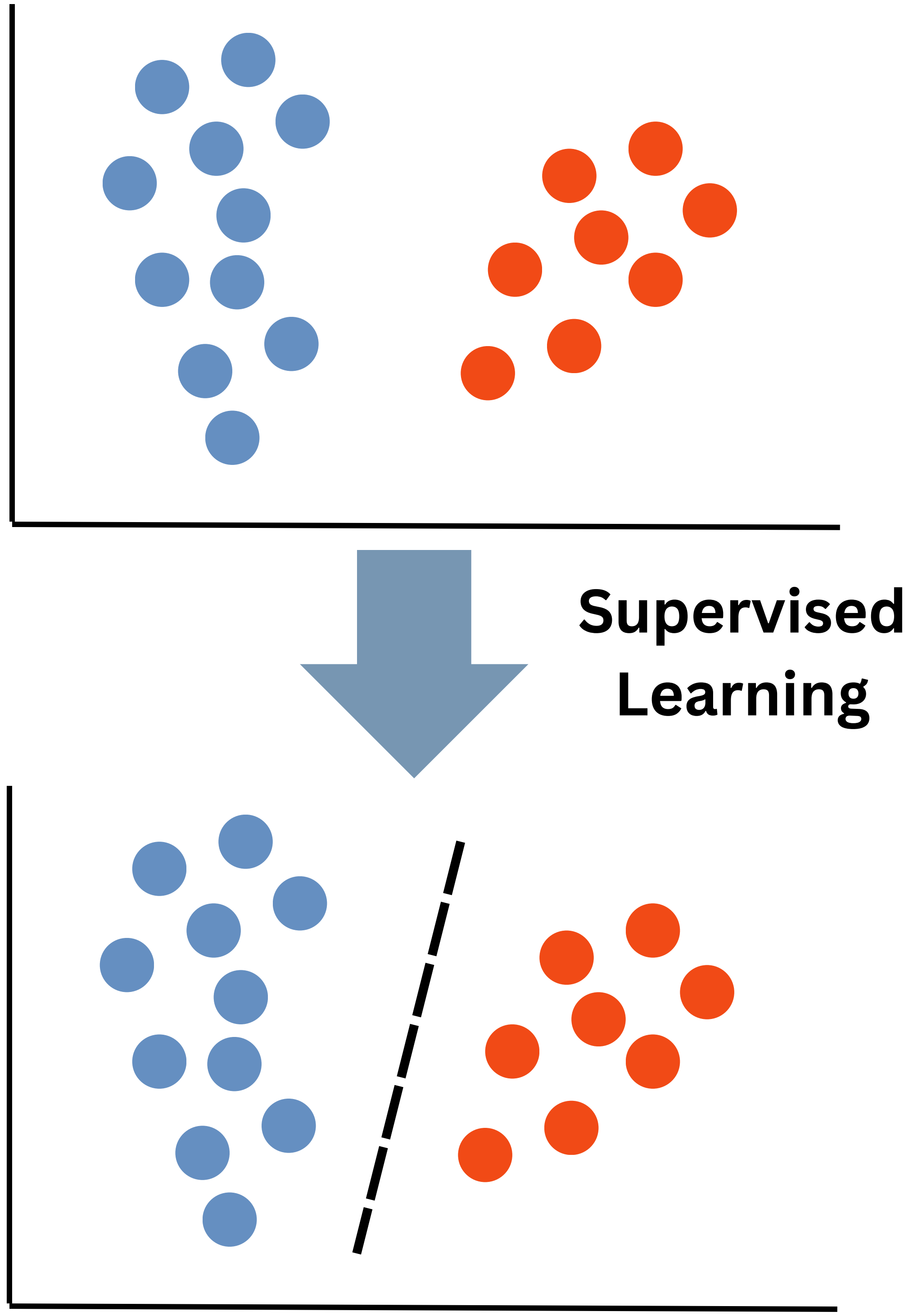
- No "correct answers" or labels are provided in training
- The algorithm must discover meaningful patterns
- Unlabeled data is typically more abundant

Several distinct tasks fall under the unsupervised learning umbrella.

- Clustering algorithms
- Dimensionality reduction techniques
- Anomaly detection methods



Supervised Learning



Supervised learning relies on examples where the correct answer is already known.

- The fundamental idea is to learn from input-output pairs
- These paired examples serve as a teacher, showing the model what output should result from each input
- Creating these teaching examples typically requires human effort

The essential goal is to generalize beyond the training examples.

Regression and Classification

Supervised learning problems generally fall into two categories

Regression tasks

- The output is a continuous numerical value representing a quantity or magnitude.
- Common applications include predicting house prices based on features, forecasting temperatures, or estimating a person's age from a photograph.

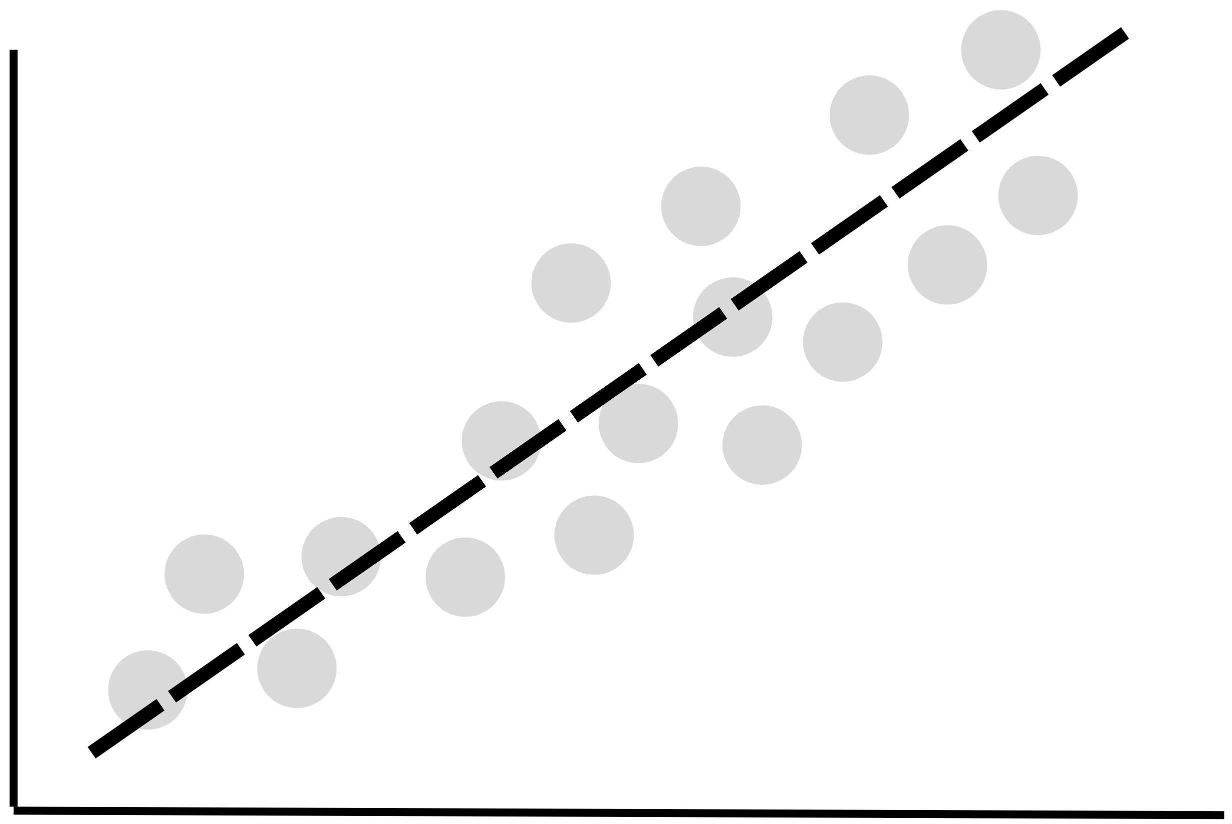
Classification tasks

- The goal is to assign inputs to distinct categories. The output is either a discrete class label or a probability distribution across possible classes.
- Examples include filtering spam emails, diagnosing diseases from symptoms, or recognizing handwritten digits.

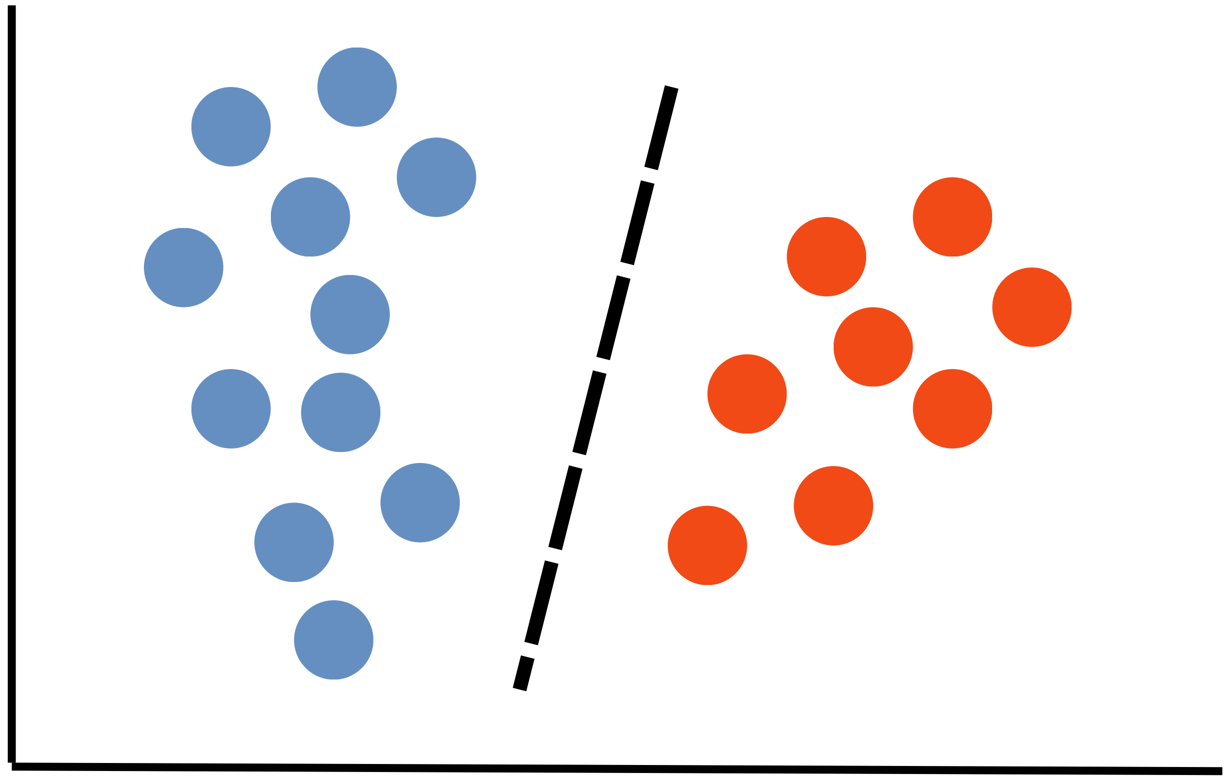
Regression and Classification

Supervised learning problems generally fall into two categories

Regression



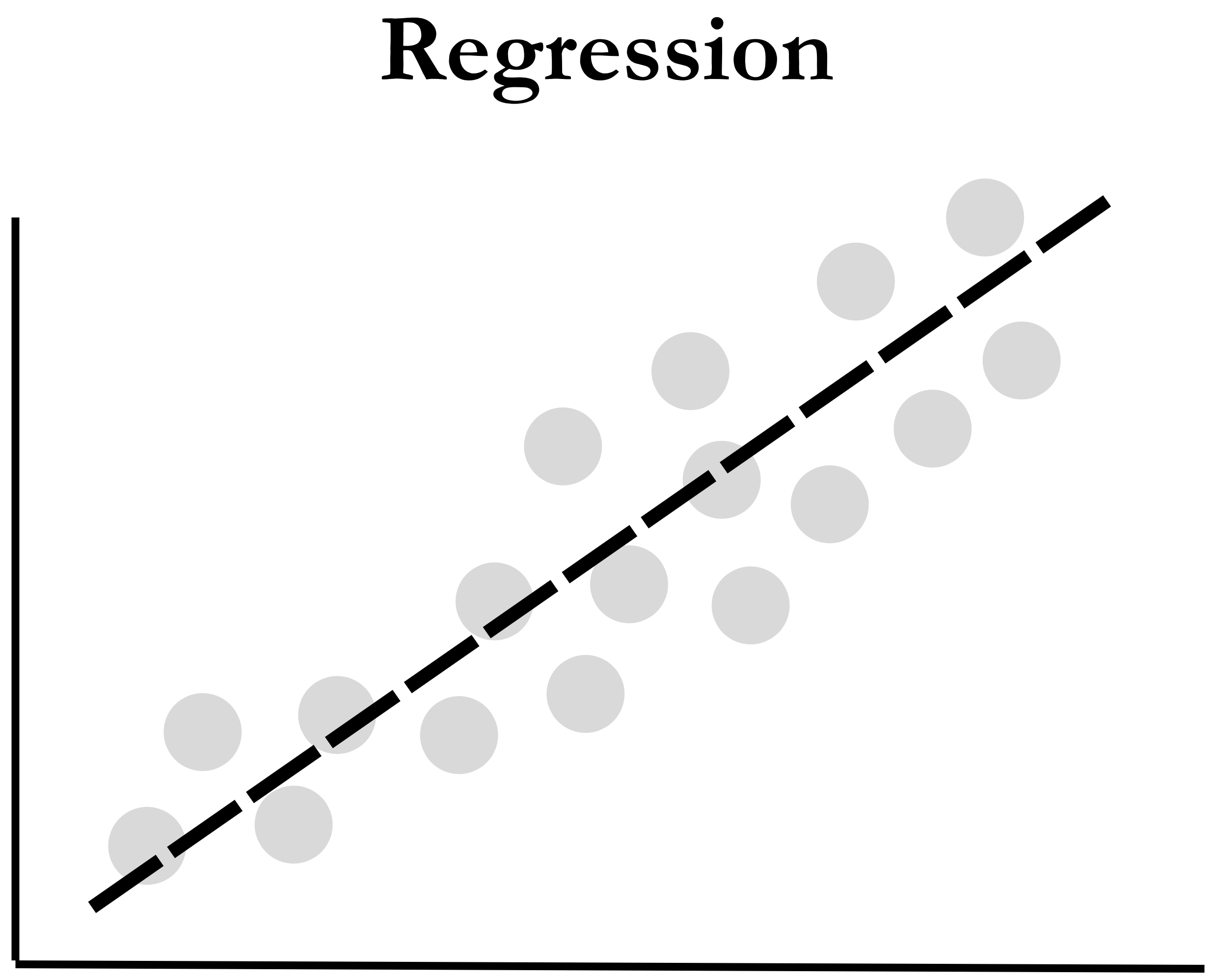
Classification



Example: Linear Regression

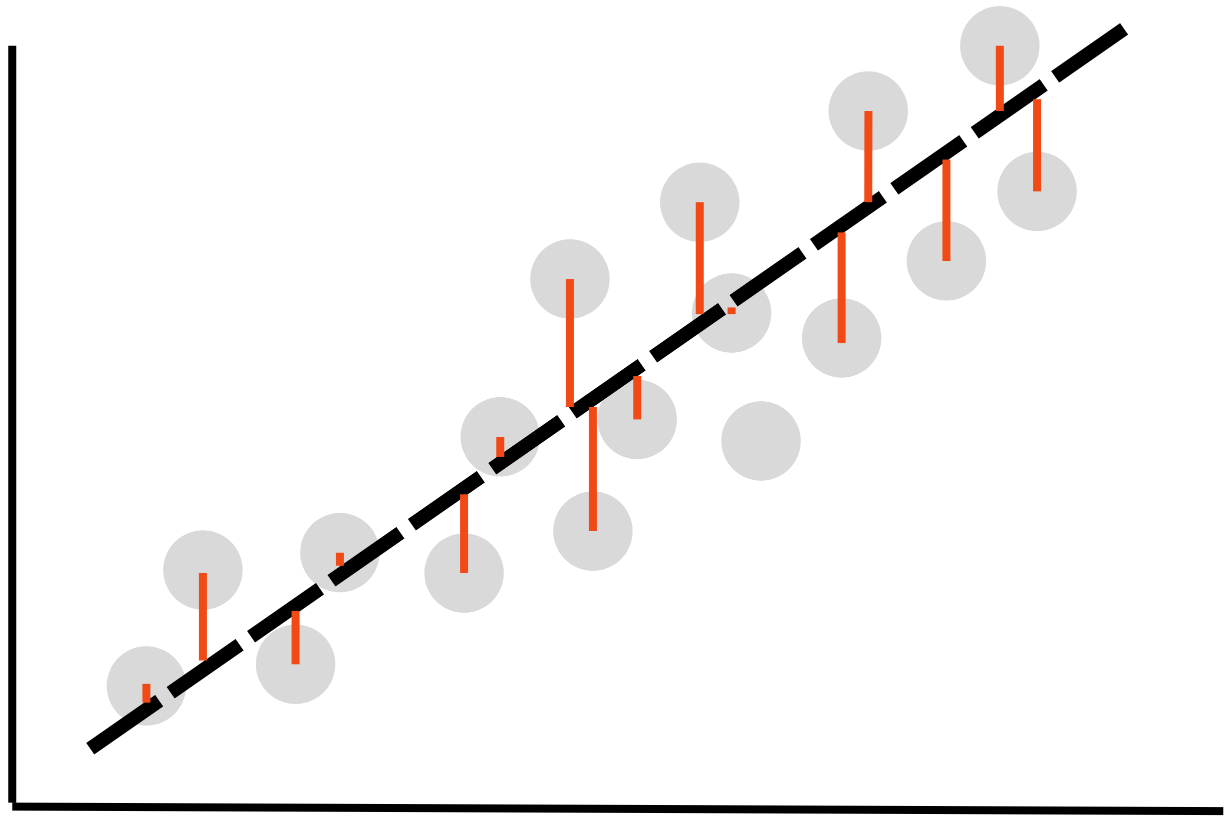
Linear regression is the most simple example of supervised learning

- The model is simply: $y = wx + b$
  - y is the output (prediction)
  - x is the input feature
  - w is the weight, and b is the bias.
- We provide training examples as pairs of (x, y) values, such as (house size, house price).
- The model learns to predict y from x by adjusting its parameters (w and b).
- After training, we can predict y values for new x inputs the model hasn't seen before.



Loss Function

Mean Absolute Error



The Loss Function provides a quantitative measure of how well model predictions align with actual targets.

- They are a clear optimization target.
- We aim to find the combination of model parameters that minimizes the chosen loss function.
- Different problems call for different types of loss functions.
  - Mean Squared Error (MSE) or Mean Absolute Error (MAE) are typical choices for regression
  - Cross Entropy Loss is the typical choice for classification

Notation

I will try to be consistent with the notation in the slides

- x will always be the input
- y the output
- bold letters denote vectors
  - for instance $\mathbf{x}$ is an input with multiple dimensions
- bold capital letters denote matrices
  - for instance $\mathbf{W}$ can be the weight matrix of a Neural Network
  - $\mathbf{Wx}$ is the product between the matrix $\mathbf{W}$ and the vector $\mathbf{x}$
- the parameters of the models are denoted by Greek letters or with a $\mathbf{W}$
- the loss function is denoted with an L
  - it is a function of the model parameters $L[\mathbf{W}]$

Math Recap

Matrix Multiplication

Matrix Multiplication consists in row by column multiplications:

- the elements (i, j) of the product matrix is obtained starting from the row i of the first matrix and the column j of the second matrix
- in general matrix multiplication is non commutative **A** x **B** ≠ **B** x **A**
- the number of columns of the first matrix must be equal to the number of rows of the second matrix

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \times \begin{bmatrix}10 & 11\\ 20 & 21\\ 30 & 31\end{bmatrix}$$

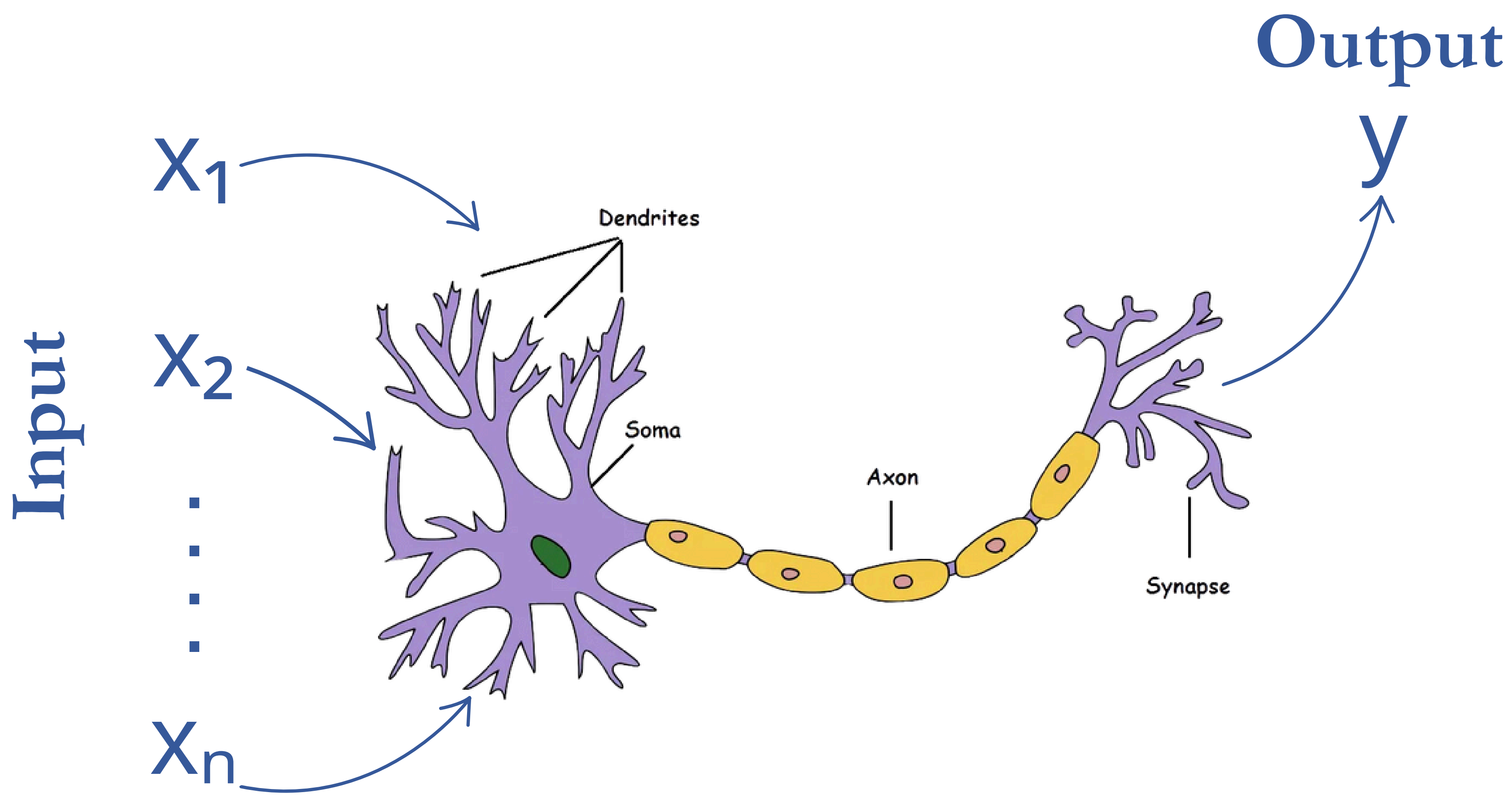
$$= \begin{bmatrix}1\times10 + 2\times20 + 3\times30 & 1\times11 + 2\times21 + 3\times31\\ 4\times10 + 5\times20 + 6\times30 & 4\times11 + 5\times21 + 6\times31\end{bmatrix}$$

$$= \begin{bmatrix}10+40+90 & 11+42+93\\ 40+100+180 & 44+105+186\end{bmatrix} = \begin{bmatrix}140 & 146\\ 320 & 335\end{bmatrix}$$

Universität Konstanz

The Perceptron

Biological Neurons



Neural networks draw inspiration from the brain's biological structure.

- **Input:** Neurons receive signals from other neurons through dendrites.
- **Processing:** Incoming signals are combined and processed in the cell body.
- **Activation Function:** If the processed signal is strong enough, it triggers firing.
- **Output:** When activated, the neuron sends a signal through its axon.

The First Neural Network

The Perceptron was the first artificial neural network model.

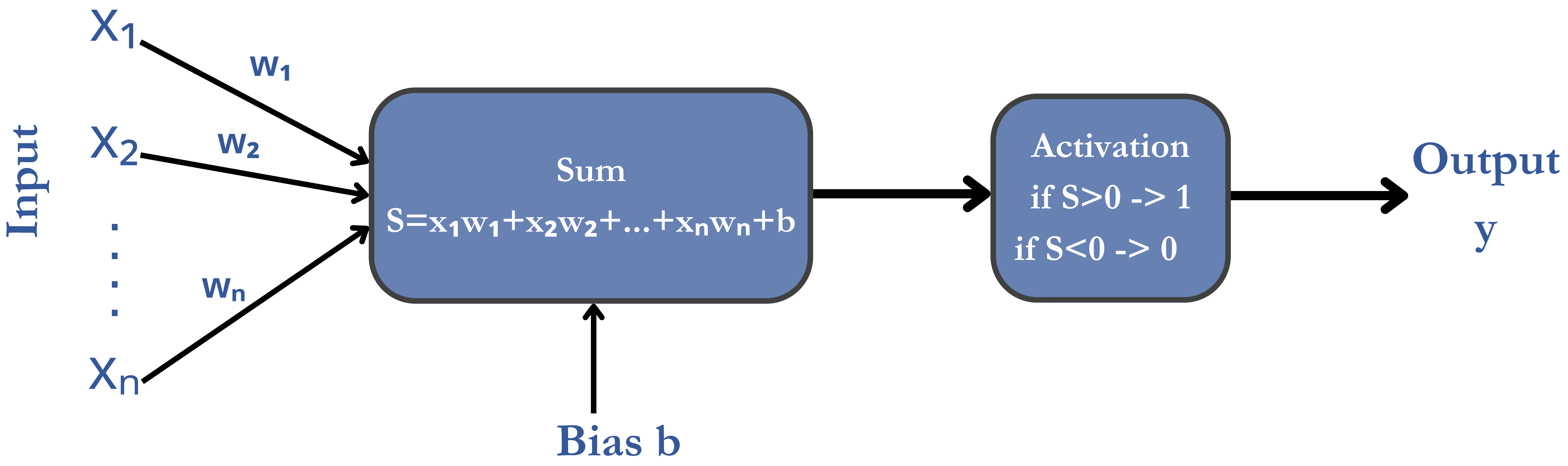
- Created in the 1950s by Frank Rosenblatt, a pioneer in artificial intelligence
- Simulates how neurons in the human brain process information

The Perceptron follows a simple operational principle:

- It receives multiple inputs, each with an associated weight
- These inputs are combined and produce an output if they exceed a threshold

Structure of the Perceptron

The Perceptron combines weighted inputs and activation:



Mathematical Notation

Mathematically we can represent the perceptron using a vector product

$y=a[\mathbf{wx}+b]$

Here

- $\mathbf{w}=(w_1, w_2, ..., w_n)$ is the vector containing the n weights
- $\mathbf{x}=(x_1, x_2, ..., x_n)$ is the vector containing the n-dimensional input
- a is the activation function. In our case
  - $a(x)=1$ if $x>0$
  - $a(x)=0$ if $x<0$
- b is the bias

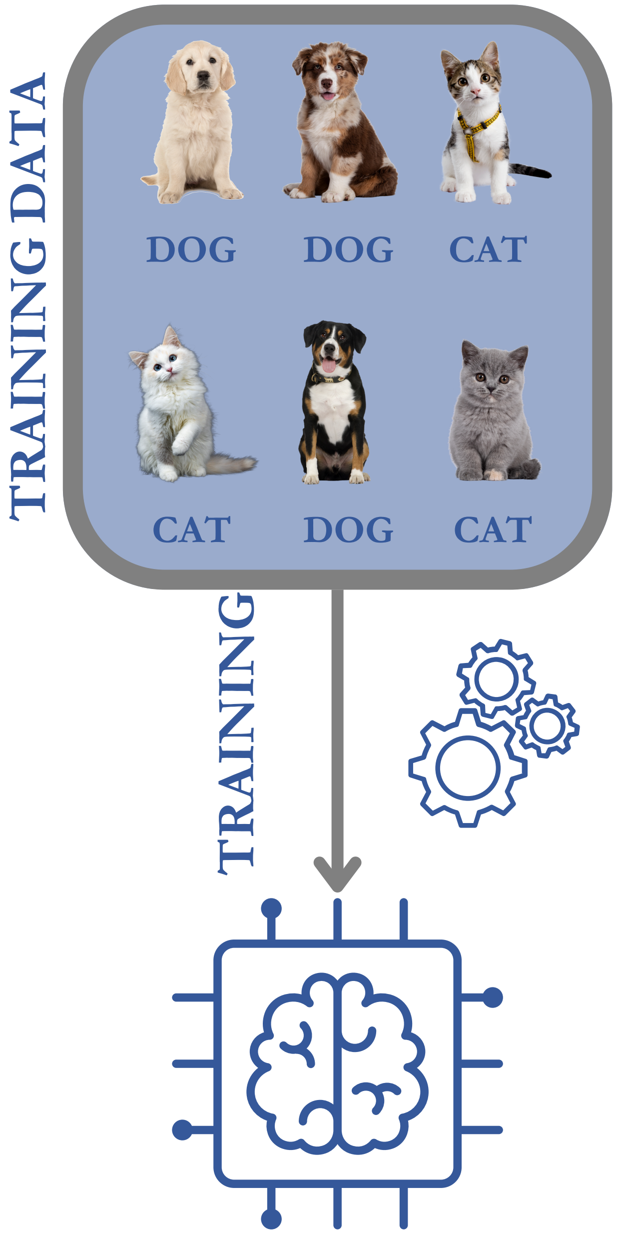
With **wx** we denote the scalar product of the two vectors

$\mathbf{wx}=x_1w_1+x_2w_2+...+x_nw_n$

Supervised Classification

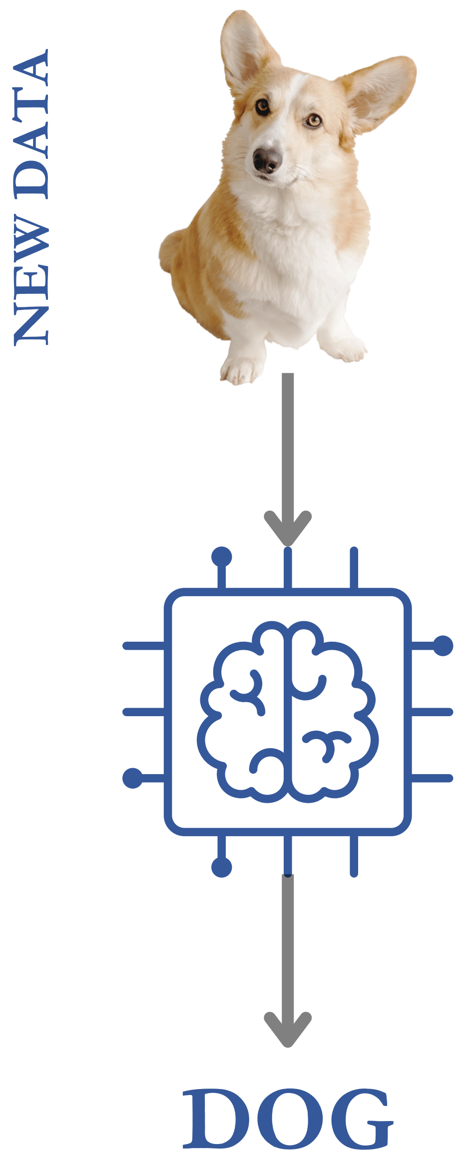
The computer learns from classified examples to predict categories for new data.

LEARNING



The model analyzes pre-classified examples and learns how to distinguish between categories

CLASSIFICATION



The model applies what it learned to categorizes new, previously unseen data

Classifying Dogs and Cats

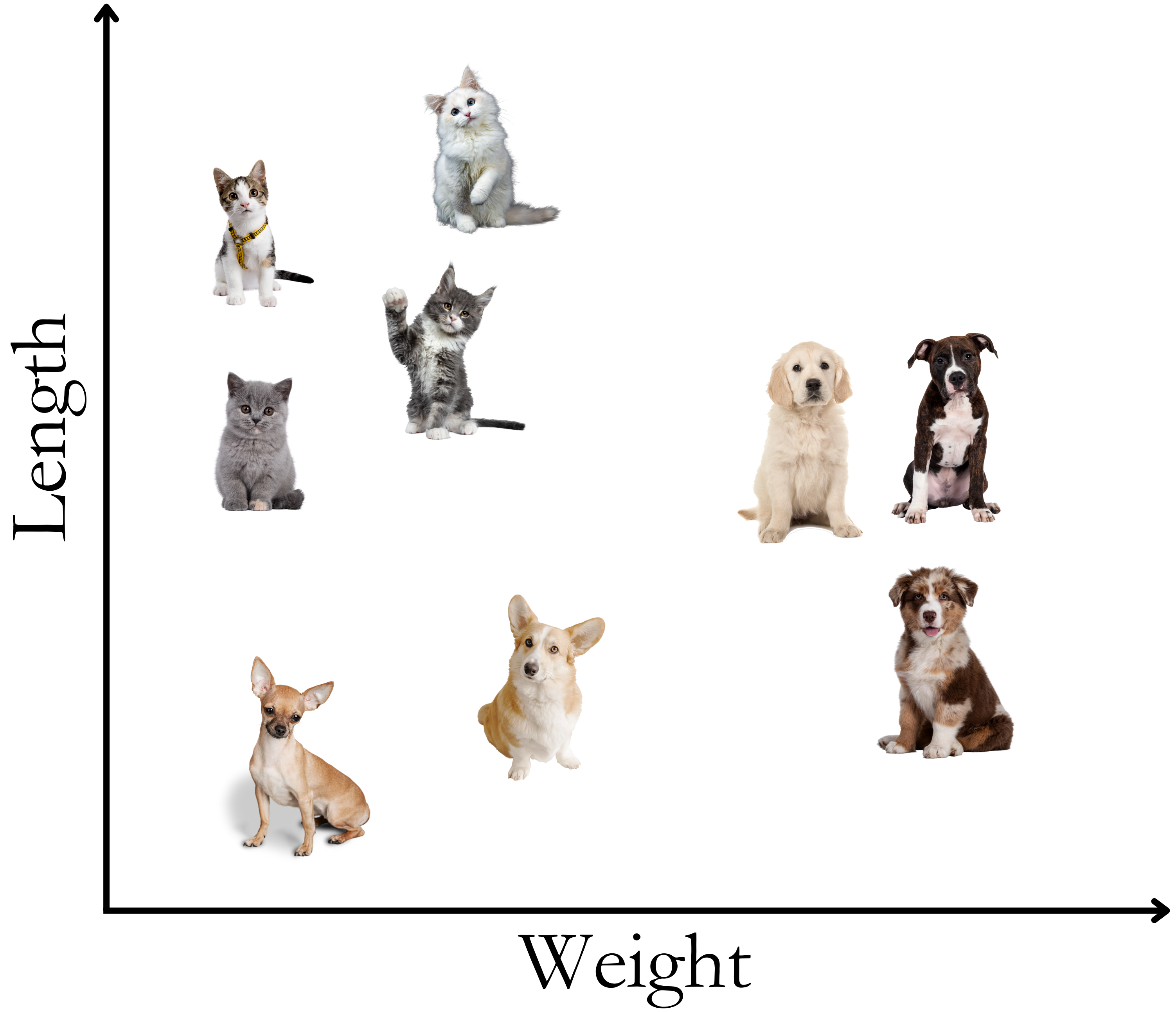
Let's consider a simplified version of the classification problem

- Dogs and cats can be characterized by various features
- We focus on just two: weight and length

When can plot different animals on a graph using weight and length

- Each animal is represented by a point

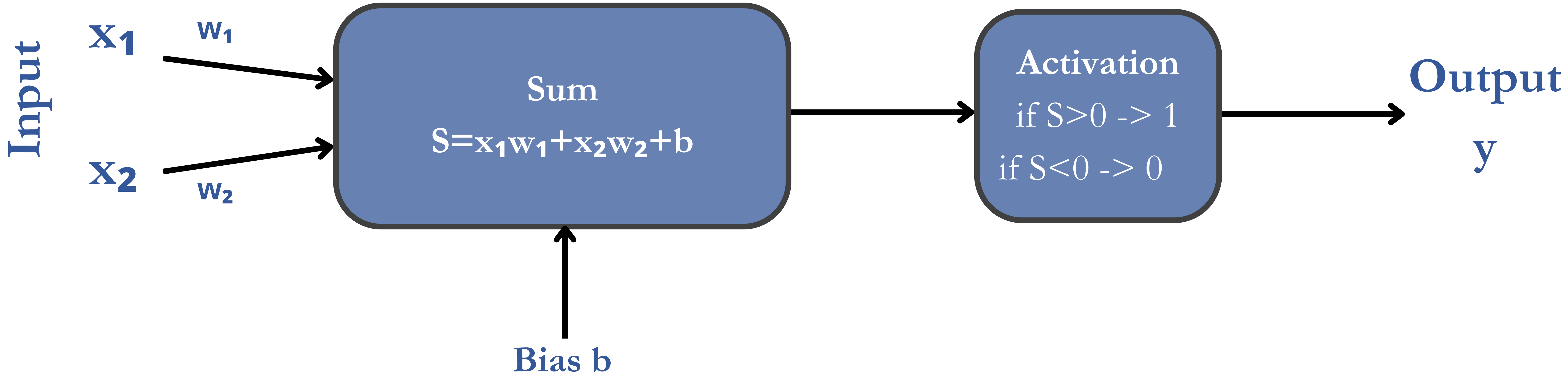
The Perceptron's job is to determine which group (dog or cat) a new animal belongs to



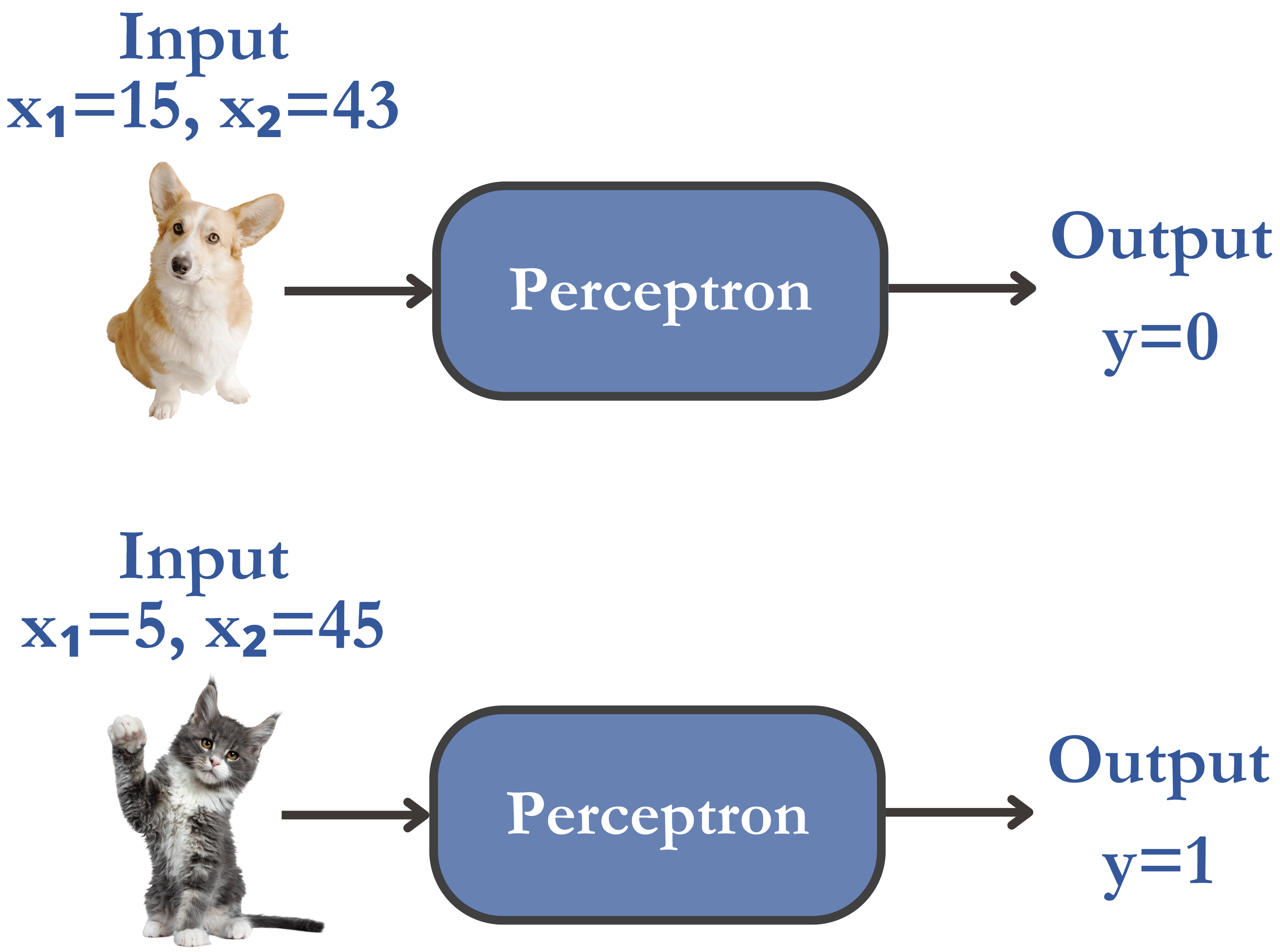
Perceptron with 2 Inputs

In this simple case we have just 2 inputs $\mathbf{x}=(x_1, x_2)$ and 3 parameters

- x_1 is the weight of the animal, x_2 is length
- w_1 and w_2 are the weights
- the bias is denoted by b



Automatic Classification



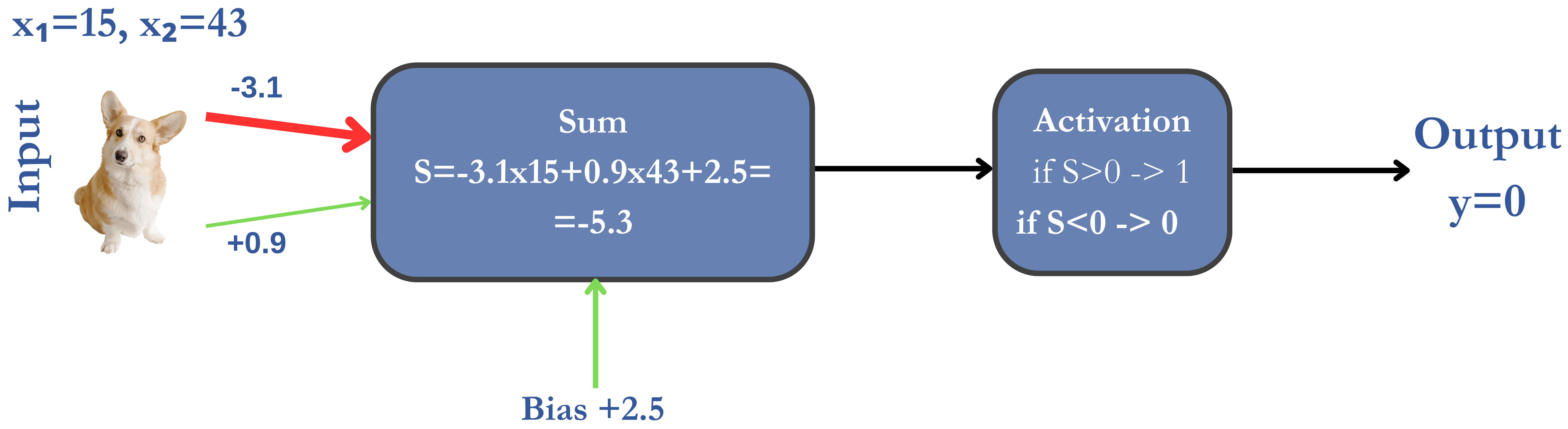
The Perceptron distinguishes between two categories:

- **Input:** Animal characteristics (e.g., weight and length)
- **Processing:** The Perceptron applies weights and calculates the output
- **Output:** Classification result
  - 0 = dog
  - 1 = cat

Perceptron with 2 Inputs

Let's see a practical example plugging some numbers in the perceptron

- we use as parameters w_1=-3.1, w_2=0.9 and b=2.5
- the input is **x**=(15, 43)



Universität Konstanz

Limits of the Perceptron

Understanding the Perceptron

The Perceptron creates a linear decision boundary to separate categories.

Cat: $w_1x_1 + w_2x_2 + b > 0$

Dog: $w_1x_1 + w_2x_2 + b < 0$

The decision boundary itself is defined by:

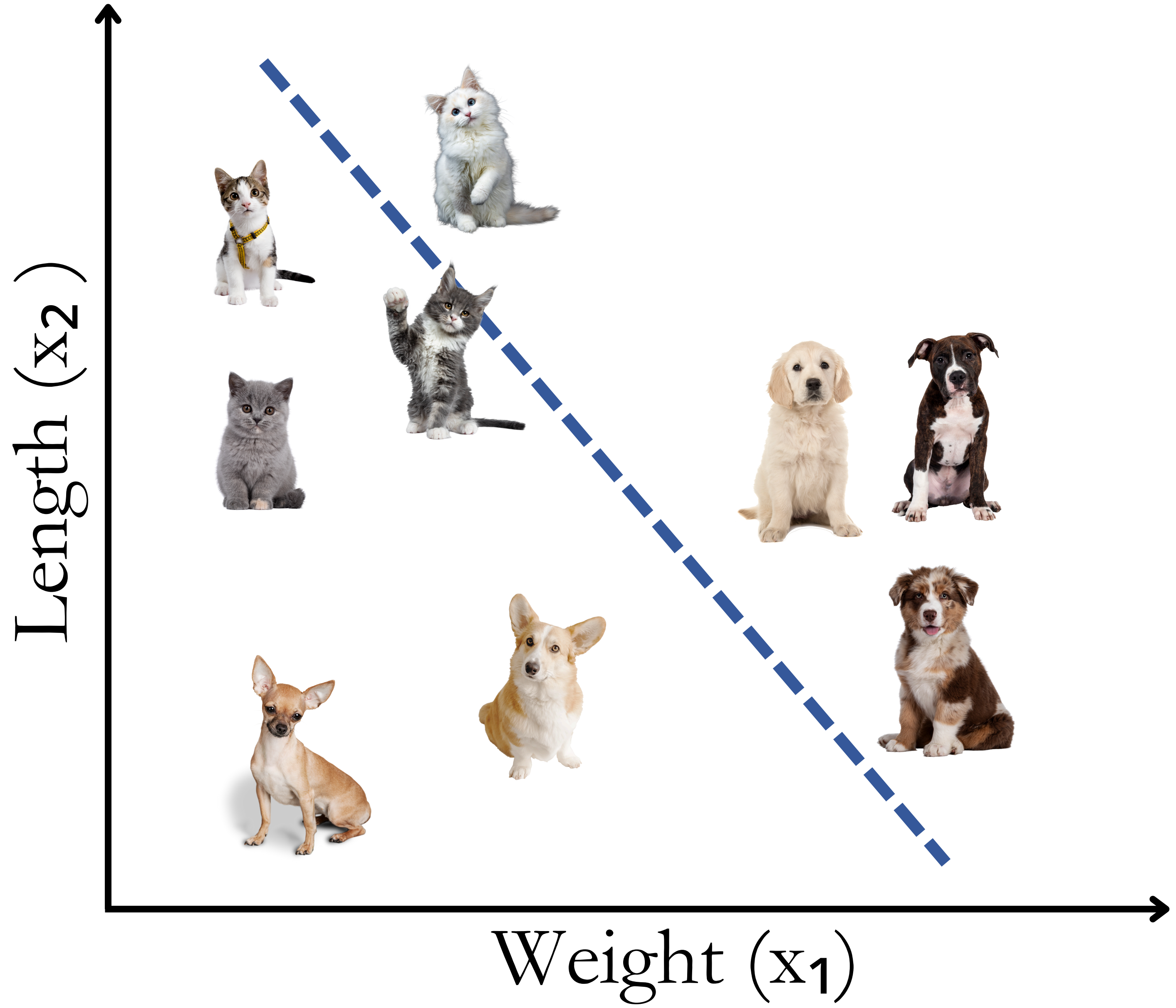
$w_1x_1 + w_2x_2 + b = 0$

Which can be rewritten as:

$x_2 = -(w_1/w_2)x_1 - (b/w_2)$

This demonstrates that the Perceptron:

- Draws a straight line in a 2D feature space
- Classifies points above the line as cats
- Classifies points below the line as dogs



https://giordano-demarzo.github.io/teaching/deep-learning-25/perceptron/

Universität Konstanz

Training Rule

During training the perceptron is shown labelled data and its weights are adjusted when it produces wrong classifications

Input

→ **Perceptron** →

Error!

Output

1

$w_1 = w_1 - x_1$

$w_2 = w_2 - x_2$

$b = b + 1$

Input

→ **Perceptron** →

Error!

Output

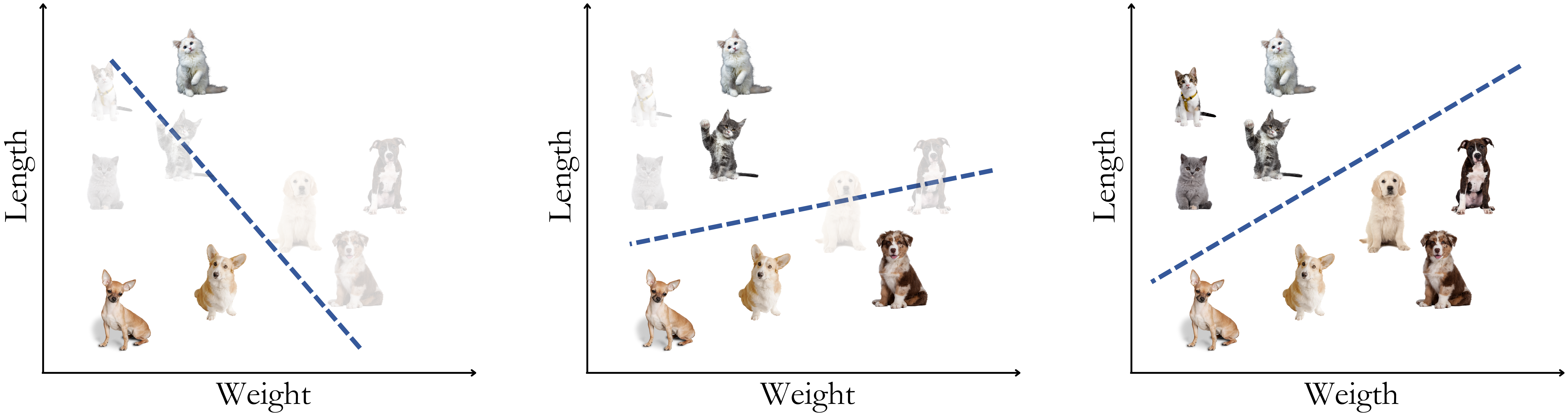
0

$w_1 = w_1 + x_1$

$w_2 = w_2 + x_2$

$b = b - 1$

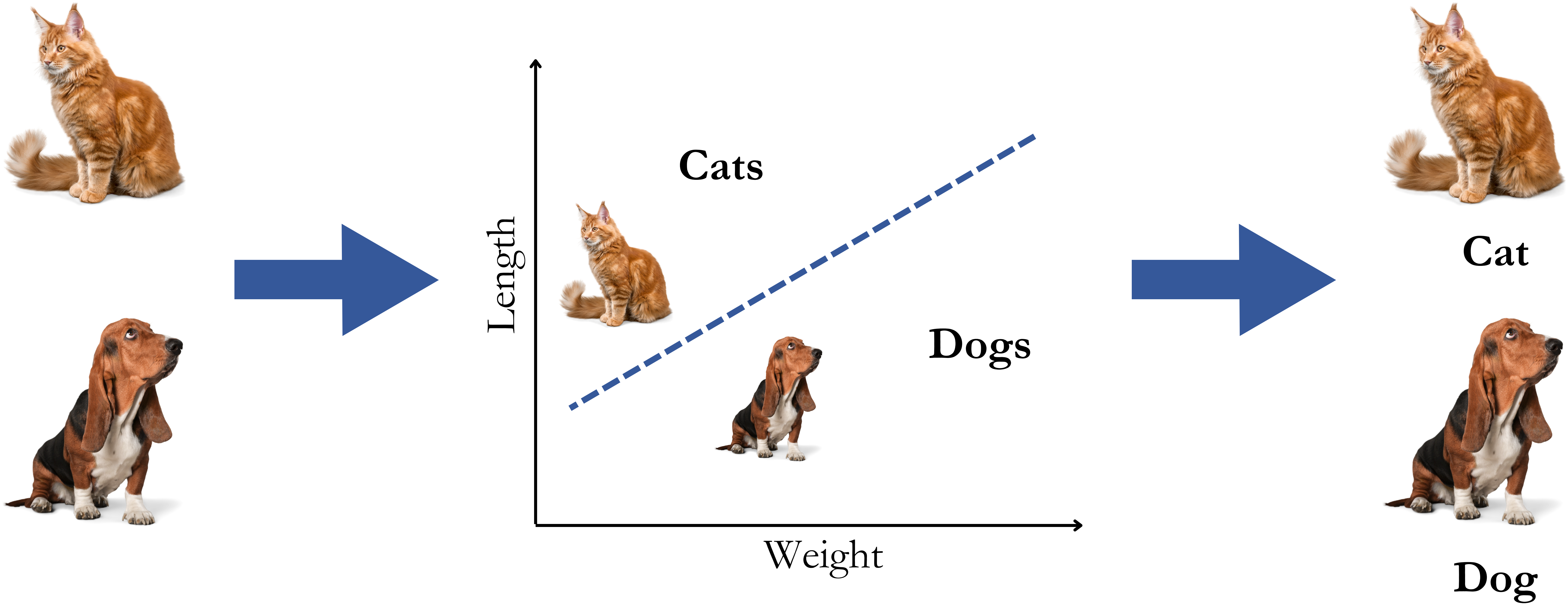
Visualizing Training



As training progresses, the decision boundary moves to better separate the classes.

Classification

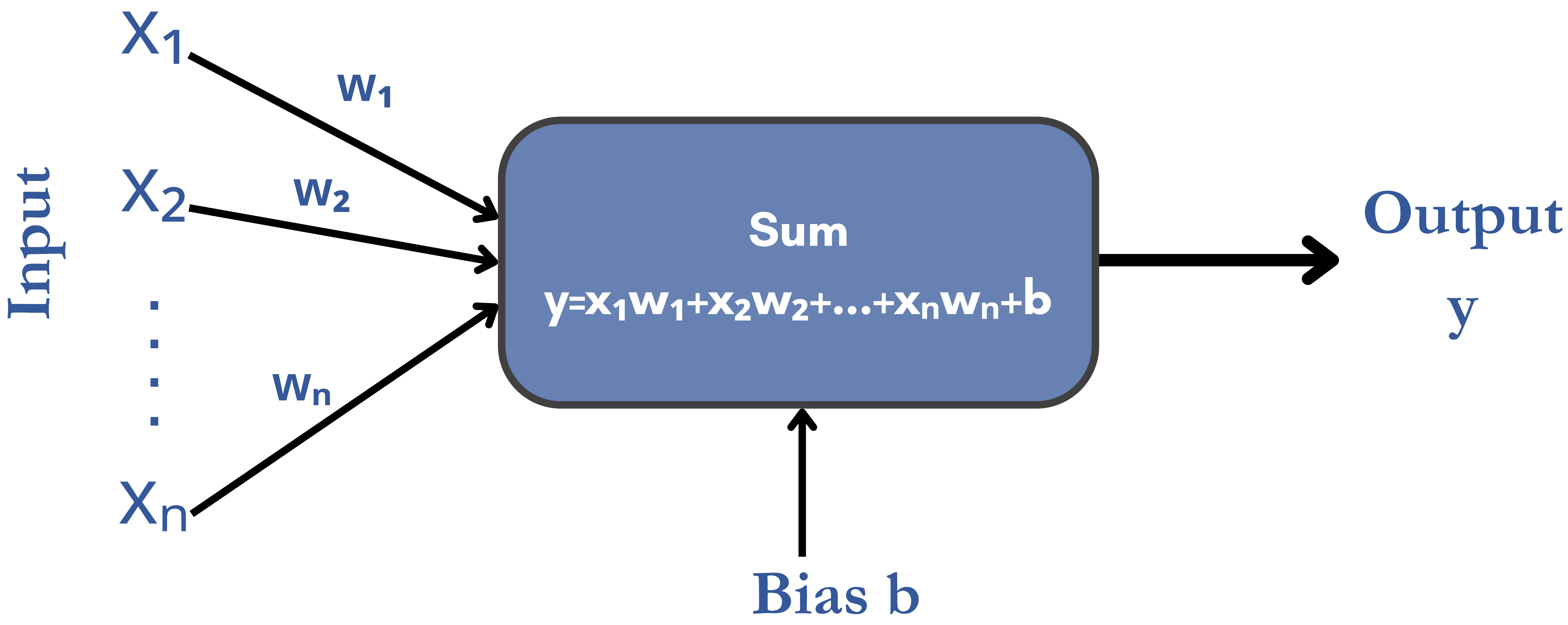
Once trained, the Perceptron can classify new animals from their weight and length



Perceptron and Regression

The Perceptron can also perform regression

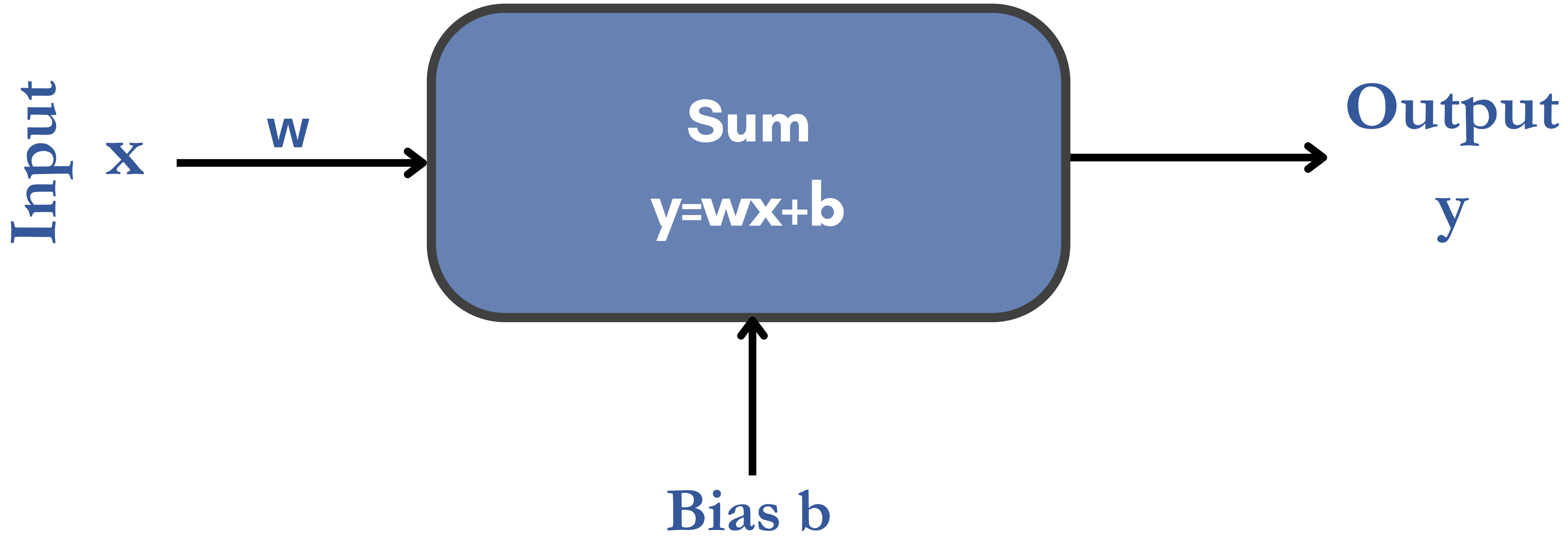
- Predicts a continuous numerical value instead of a category
- Skips the thresholding step in the activation function

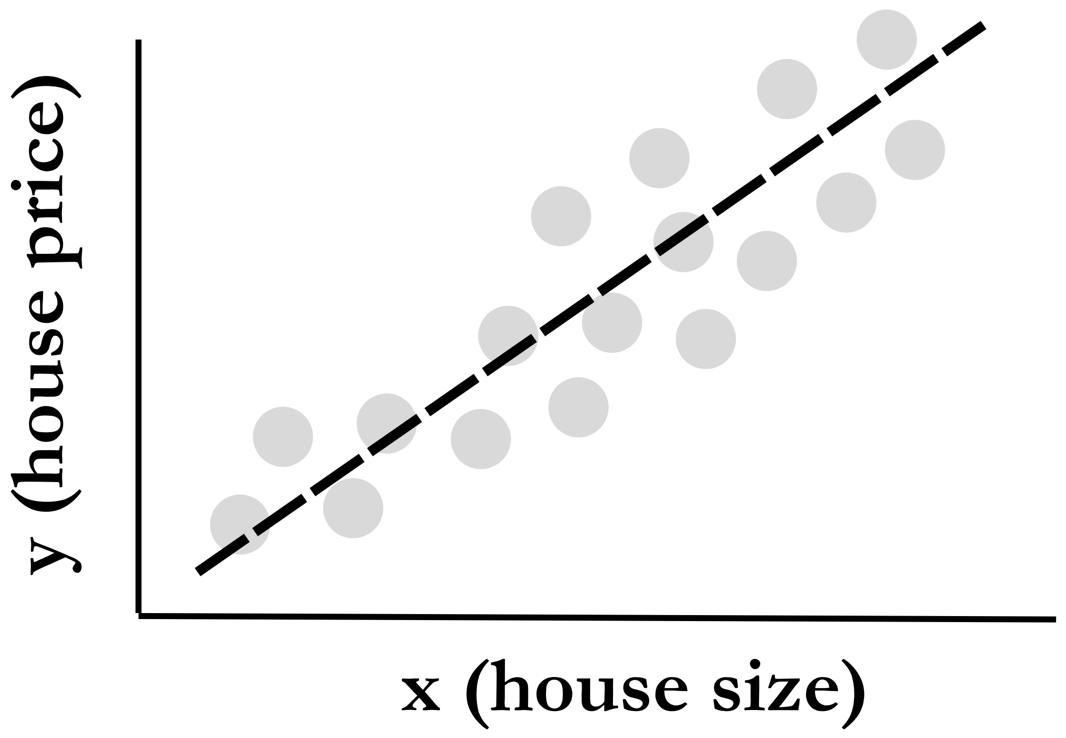


Single Input Case

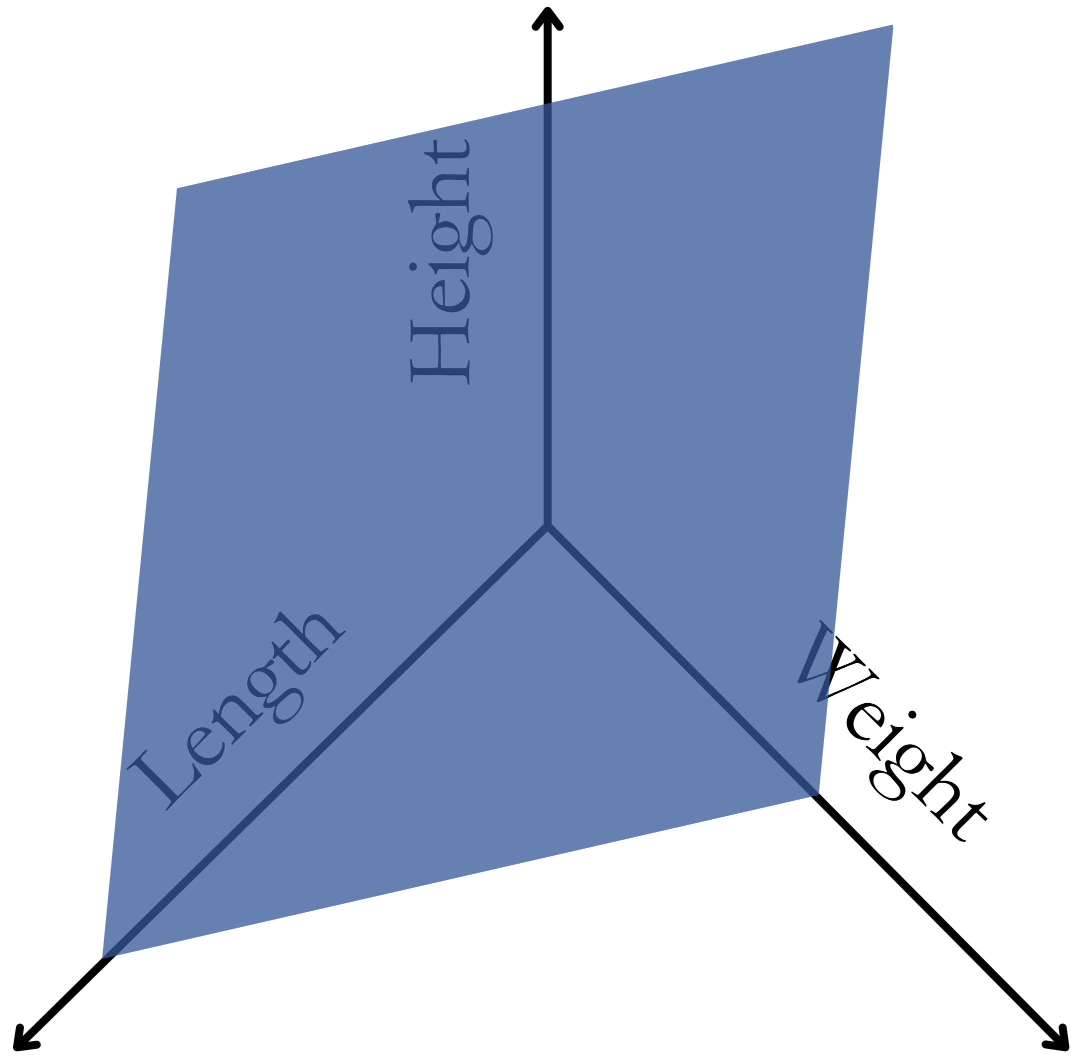
In the most simple case we have a single input

- the model output is y=wx+b
- during training the model learns w and b to fit the data
- this is equivalent to a linear regression





Higher Dimensions



The Perceptron's principles extend to higher dimensions:

- 2 Dimensions: The Perceptron uses a line to separate categories (e.g., dogs and cats based on weight and length)
- 3 Dimensions: Adding another feature (e.g., height) creates a 3D space where the Perceptron uses a plane as separator

What happens in higher dimensions?

Limits of the Perceptron

The Perceptron works well when data categories can be separated by a line (in 2D), a plane (in 3D), or a hyperplane (in higher dimensions).

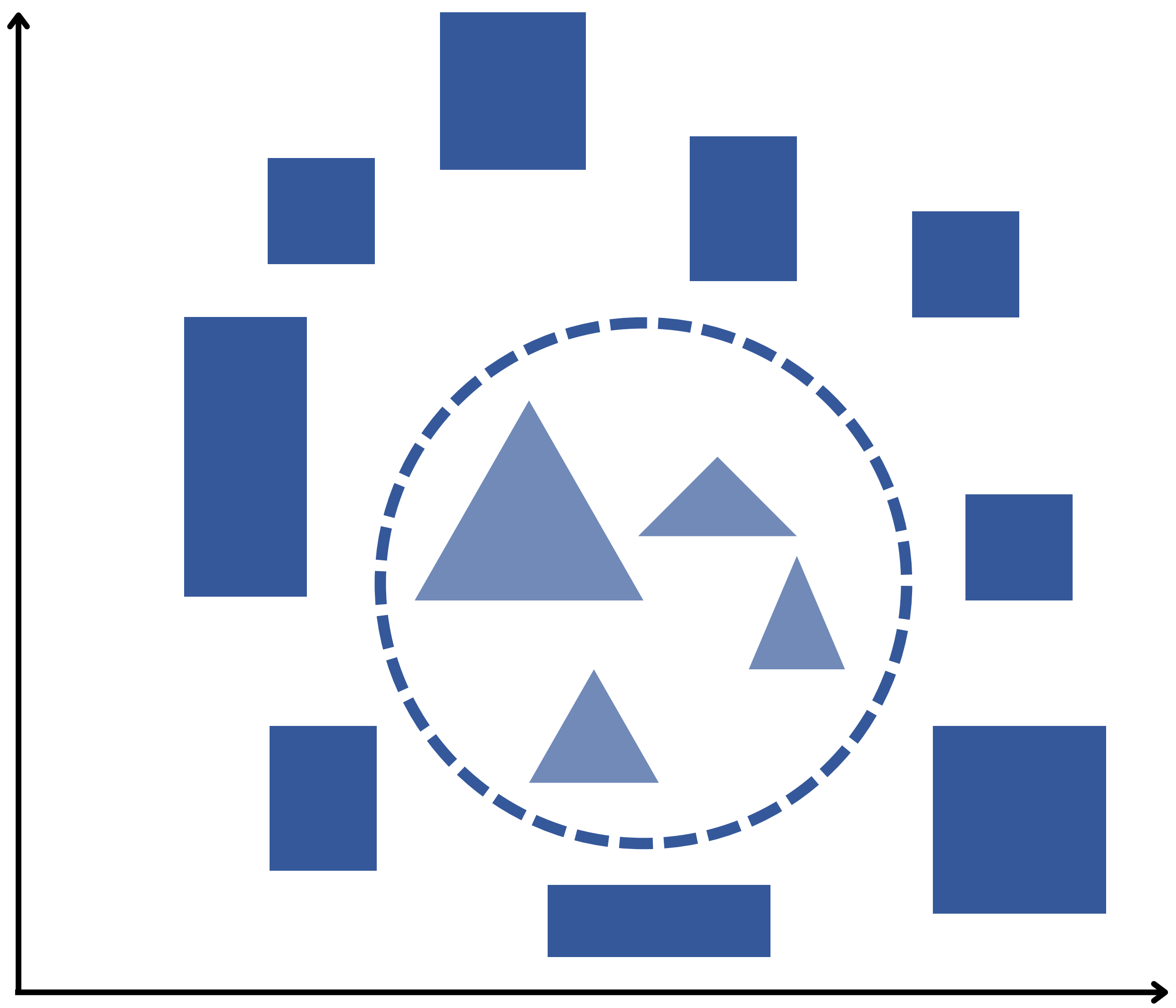
However, many real-world problems aren't linearly separable:

- If data forms patterns like circles or spirals
- If categories are intermingled in complex ways

In these cases, a single Perceptron is insufficient!

- Problems like XOR cannot be solved by a single Perceptron

This limitation led to the so called **AI Winter**



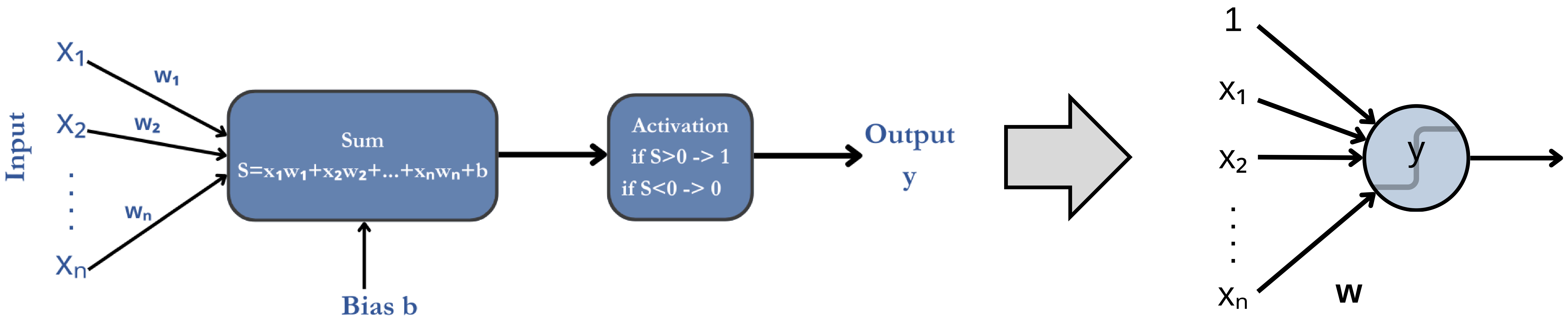
Universität
Konstanz

Shallow Neural Networks

Another Representation

In the following we will use a more simple representation for the perceptron

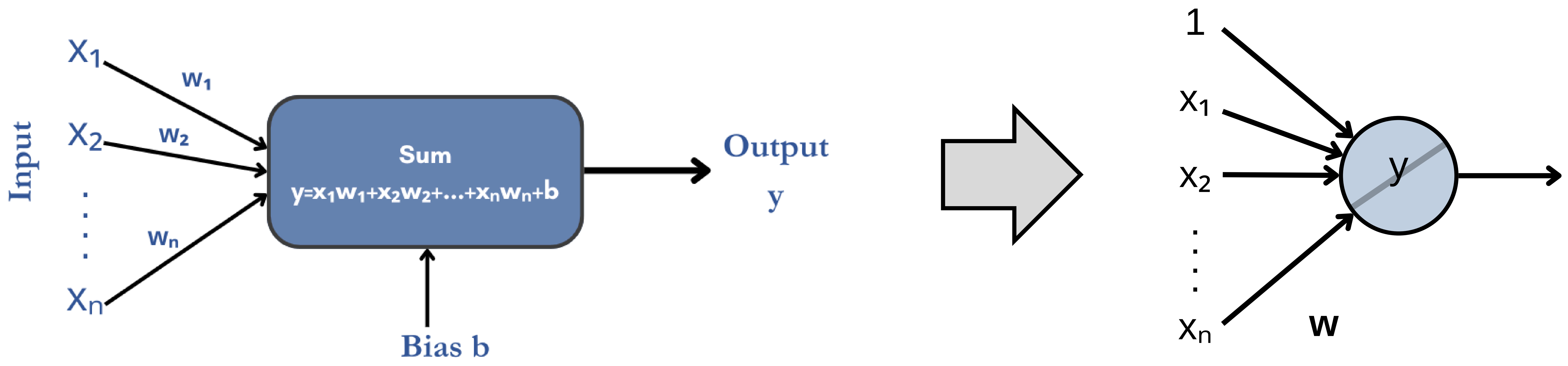
- we combine the weights and the bias in the same vector
  - $w=(b, w_1, w_2, ..., w_n)$
- we add a dummy input that is always 1 and that gets multiplied by the bias
- we write on the neuron the name of its output and we plot its activation on it



Another Representation

In the following we will use a more simple representation for the perceptron

- we combine the weights and the bias in the same vector
  - $w=(b, w_1, w_2, ..., w_n)$
- we add a dummy input that is always 1 and that gets multiplied by the bias
- we write on the neuron the name of its output and we plot its activation on it



Activation Functions

Step Function

- Used as output for the classification task
  - a(x)=1 if x>0
  - a(x)=0 if x<0

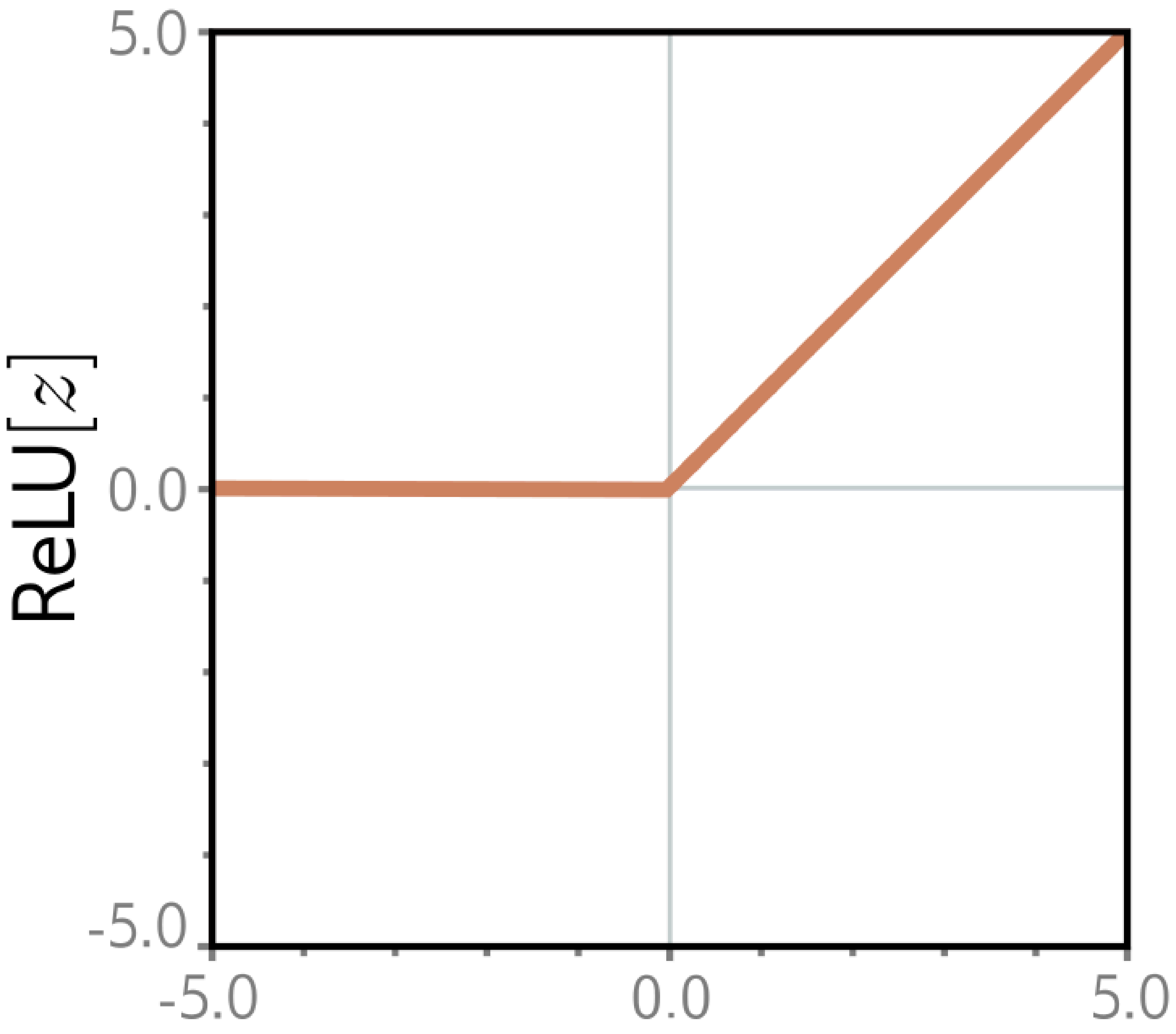
Linear Activation

- Used as output for the regression task
  - a(x)=x

The Rectified Linear Unit (ReLU) is another example

ReLU

- Used in hidden layers of deep neural networks
  - a(x)=x if x>0
  - a(x)=0 if x<0



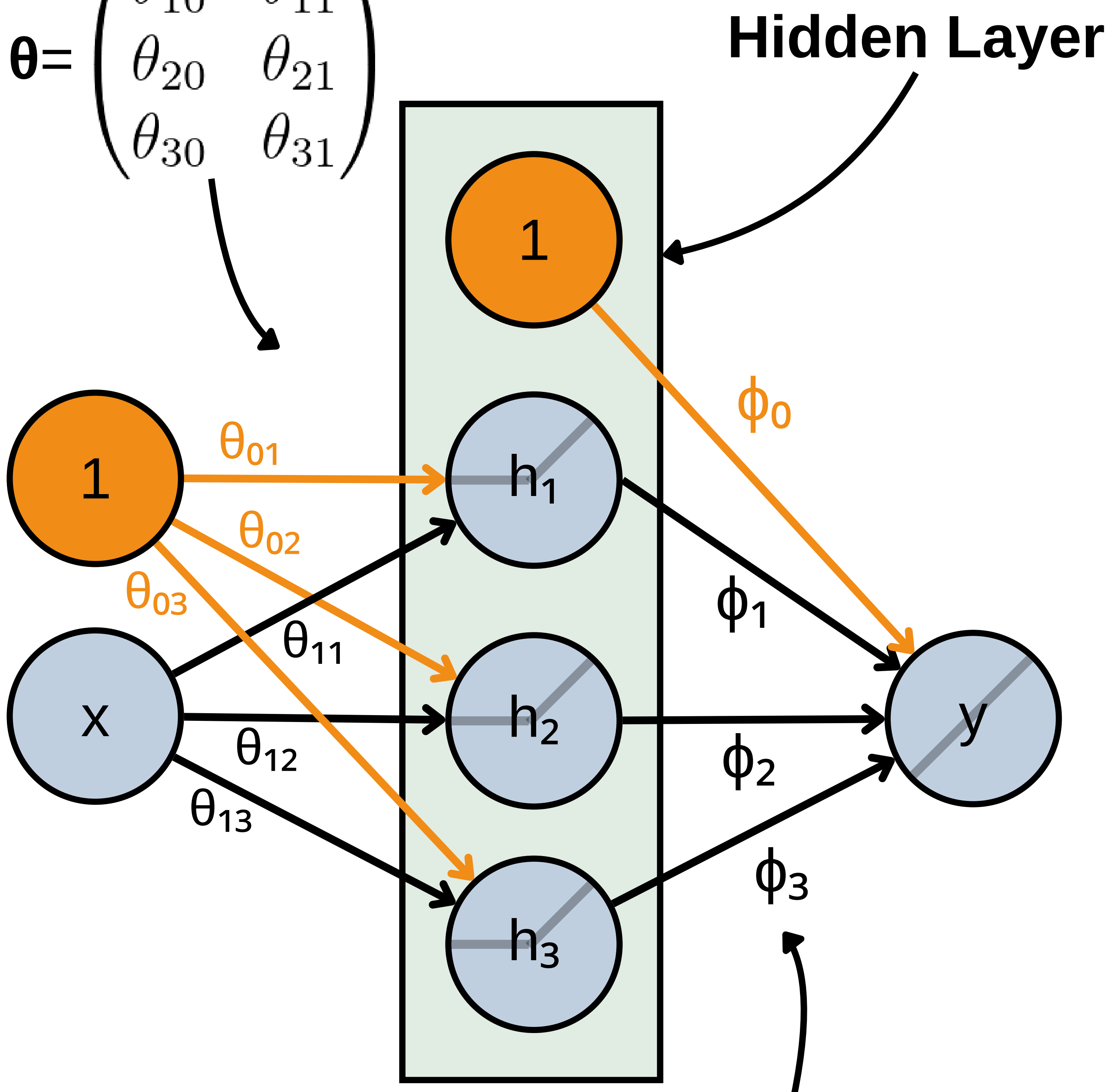
Combining Perceptrons

We consider again the simple regression problem with a single input x and output y

- we can apply more than one single perceptron to the input (and dummy)
- each of these perceptrons will produce a different output h_i
- we can then use these outputs as input for another perceptron that produce the output y

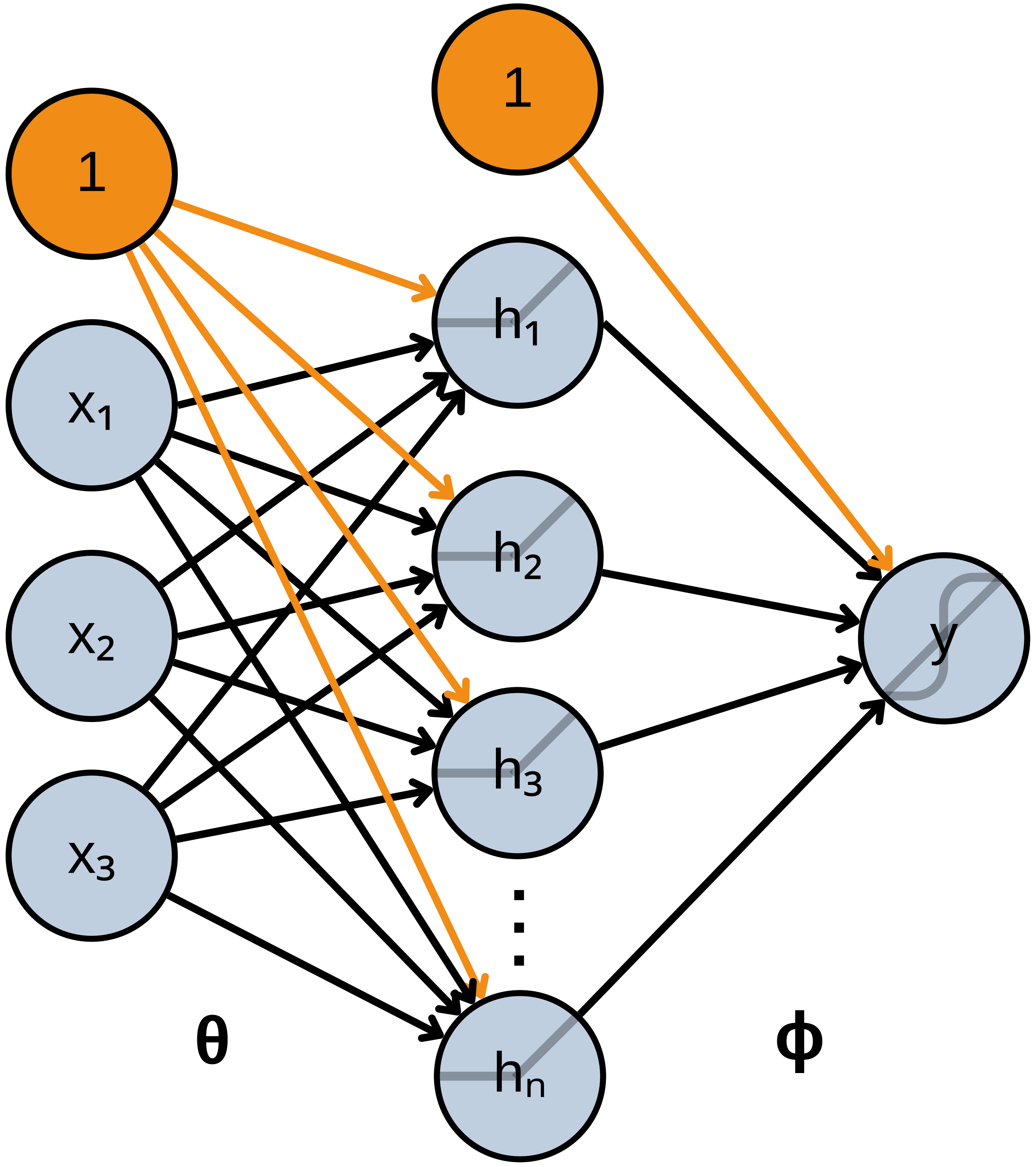
In this way we are adding an **hidden layer** to the neural network

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{10} & \theta_{11} \\ \theta_{20} & \theta_{21} \\ \theta_{30} & \theta_{31} \end{pmatrix}$$



$$\boldsymbol{\phi} = \begin{pmatrix} \phi_0 & \phi_1 & \phi_2 & \phi_3 \end{pmatrix}$$

Shallow Neural Networks



More generally we can have neural networks with

- as many inputs as we want
- as many hidden neurons as we want

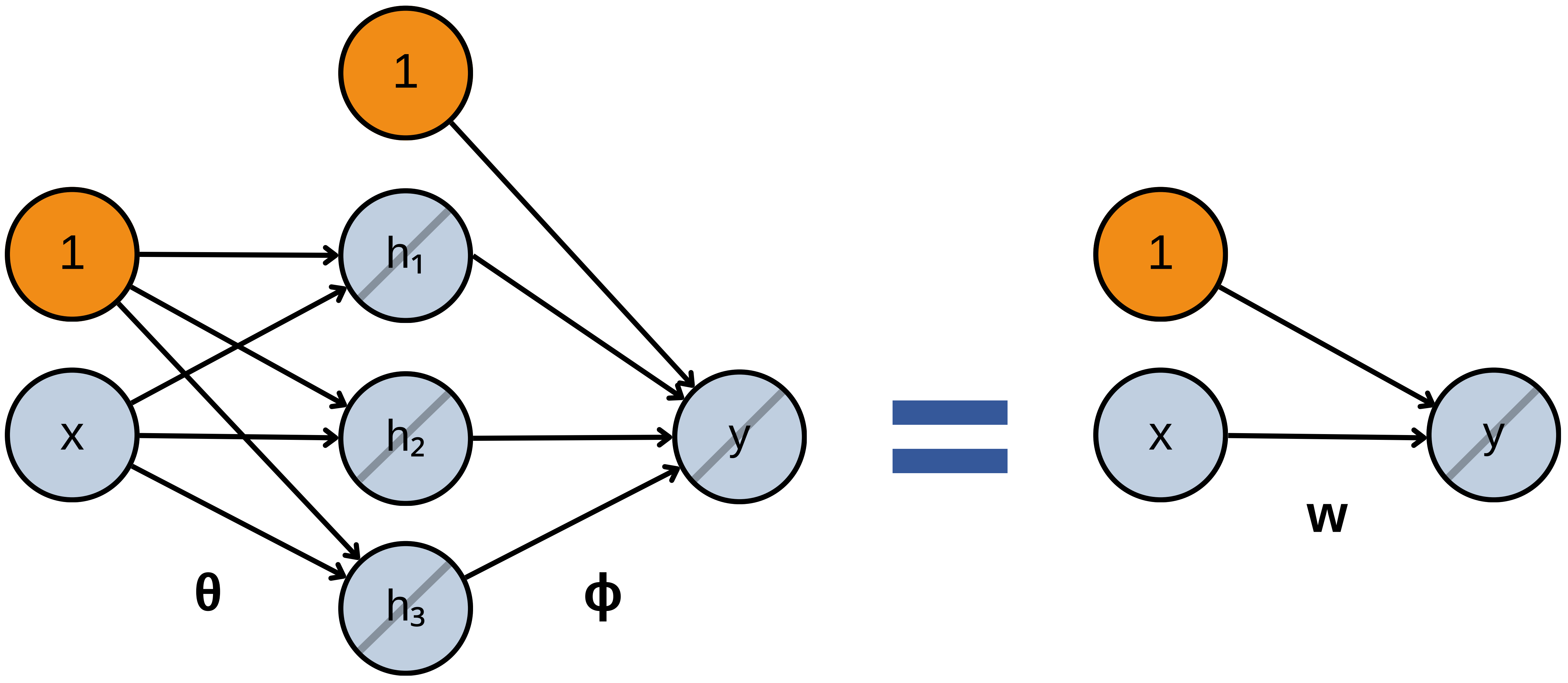
The parameters of this neural network will be contained in two weight matrices

- **θ** connecting the input to the hidden layer
- **ϕ** connecting the hidden layer to the output

This type of neural network with a single hidden layer is called **Shallow Neural Network**

Why is the ReLU Important?

Non-linear activation functions like the ReLU are crucial in Deep Learning. A shallow neural network with linear activation functions is equivalent to a simple perceptron



What's Next?

The perceptron can only solve liner problem

- most real life problems are much more complex
- this lead to the AI Winter

We have just introduced shallow neural networks

- if we use non-linear activation functions like the ReLU this is different with respect to a simple perceptron

We still have to understand some things

- can shallow neural networks solve non-linear problems?
- how can we train a shallow neural network?

What's Next?

Tomorrow room G420

- introduction to google colab and GPU server
- basic machine learning concepts
- fill the form to get access to the GitHub of the course

https://docs.google.com/forms/d/e/1FAIpQLSc9bKHplUFxv_jxfY20OYmA0OrjilCcAaCMEICREQtA0t9Q2w/viewform?usp=sharing

Next week

- on Wednesday we answer to the open questions and we introduce the first Deep Neural network, the Multilayer Perceptron
- on Thursday we will implement our first neural network using PyTorch

Summary

Basic Concepts and Notation

We introduced the main machine learning concepts like supervised vs unsupervised learning, classification and regression, the loss function

The Perceptron

The perceptron is the first artificial neural network. It consists in a weighted sum and an activation and allows to perform automatic classification/regression

Limits of the Perceptron

The perceptron can only solve linear problems: in the case of classification it draws a linear decision boundary, while in the case of regression it can only perform a linear regression

Shallow Neural Networks

The output of a perceptron can be fed into another perceptron, leading to a shallow neural network. The hidden layer must have non-linear activation functions like the ReLU