CS 480

Introduction to Artificial Intelligence

March 28, 2024

Announcements / Reminders

- Please follow the Week 11 To Do List instructions (if you haven't already):
- Programming Assignment #02: due on Sunday (~~03/31~~ 04/07) at 11:59 PM CST
- Written Assignment #04 due on Sunday (03/31) at 11:59 PM CST
- Quiz #06 due on Sunday (03/31) at 11:59 PM CST

Plan for Today

- Decision Networks
- A Casual Introduction to Machine Learning (if time permits)

Value of Perfect Information

The value/utility of best action α without additional evidence (information) is :

$$\mathrm{MEU}(\alpha) = \max_{a} \sum_{s'} \mathrm{P}(Result(a) = s') * \mathrm{U}(s')$$

If we include new evidence/information ($\mathrm{E_j} = \mathrm{e_j}$) given by some variable Ej, value/utility of best action α becomes:

$$\mathrm{MEU}(a_{e_j} \mid e_j) = \max_{a} \sum_{s'} \mathrm{P}(Result(a) = s' \mid e_j) * \mathrm{U}(s')$$

The value of additional evidence/information from Ej is:

$$\mathrm{VPI}(E_j) = \left(\sum_{e_j} \mathrm{P}(E_j = e_j) * \mathrm{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

using our current beliefs about the world.

Value of Perfect Information

The value of additional evidence/information from Ej is:

$$\mathrm{VPI}(E_j) = \left(\sum_{e_j} \mathrm{P}(E_j = e_j) * \mathrm{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

Maximum expected utility based on everything we know so far:

a) **NO evidence whatsoever**

b) **OTHER (not value of E_j) evidence (could be more than one "piece" of evidence)**

Value of Perfect Information

The value of additional evidence/information from Ej is:

$$\mathrm{VPI}(E_j) = \left(\sum_{e_j} \mathrm{P}(E_j = e_j) * \mathrm{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

Value of perfect information for some NOT YET known (but we are considering obtaining / observing it) evidence (specific value of variable E_j) E_j

Value of Perfect Information

The value of additional evidence/information from Ej is:

$$\mathrm{VPI}(E_j) = \left(\sum_{e_j} \mathrm{P}(E_j = e_j) * \mathrm{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

Measure of how "much EXPECTED utility" we can gain by introduction evidence (specific value of variable E_j) E_j

Value of Perfect Information

The value of additional evidence/information from Ej is:

$$\mathrm{VPI}(E_j) = \left(\sum_{e_j} \mathrm{P}(E_j = e_j) * \mathrm{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

Possible value of variable E_j

Value of Perfect Information

The value of additional evidence/information from Ej is:

$$\mathrm{VPI}(E_j) = \left(\sum_{e_j} \mathrm{P}(E_j = e_j) * \mathrm{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

Probability of variable E_j having value e_j

Value of Perfect Information

The value of additional evidence/information from Ej is:

$$\mathrm{VPI}(E_j) = \left(\sum_{e_j} \mathrm{P}(E_j = e_j) * \mathrm{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

Calculate for every e_j and sum it

Value of Perfect Information

The value of additional evidence/information from Ej is:

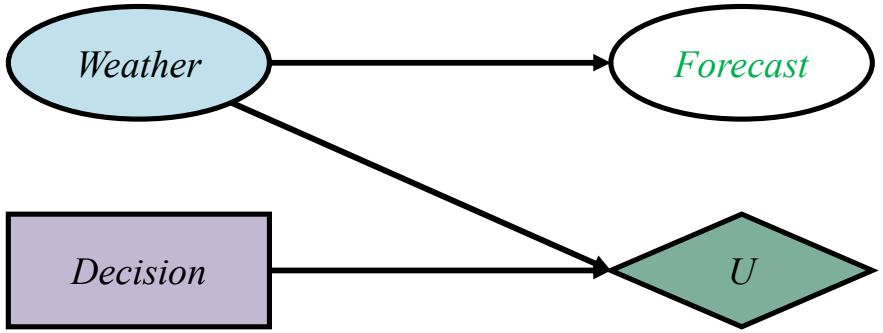
$$\mathrm{VPI}(E_j) = \left(\sum_{e_j} \mathrm{P}(E_j = e_j) * \mathrm{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

Maximum Expected Utility in the system under the assumption that is $E_j = e_j$ is given

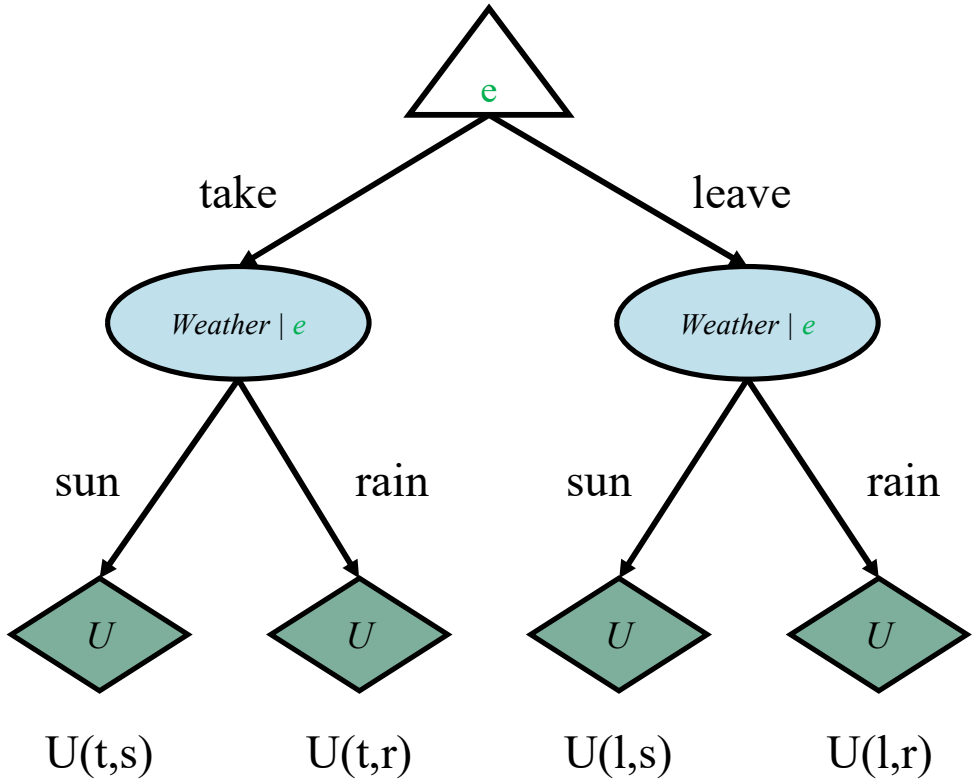
and everything else (could be nothing) that contributed to $MEU(a)$ is given

Decision Network: Example

Decision network



Outcome tree



The value of best action α without additional evidence

$$\text{MEU}(\alpha) = \text{MEU}(\text{leave}) = 70$$

With evidence information ($E_j = e_j$) given by Forecast:

$$\text{MEU}(a_{e_1} \mid e_1) = \text{MEU}(\text{take} \mid F = rain) = 53$$

$$\text{MEU}(a_{e_2} \mid e_2) = \text{MEU}(\text{leave} \mid F = sun) = 95$$

The value of additional evidence / information from F is:

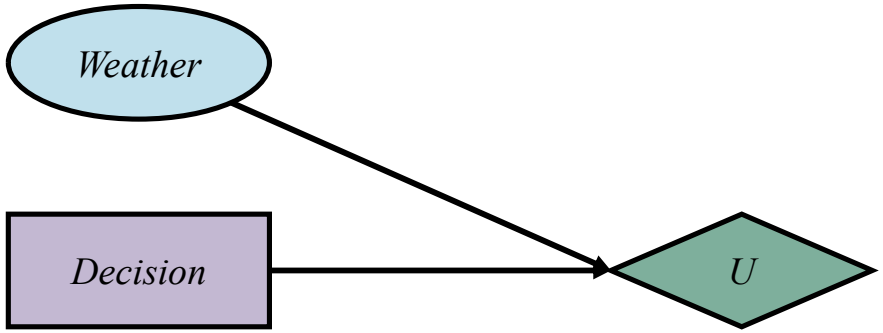
$$\text{VPI}(E_j) = \left(\sum_{e_j} \text{P}(E_j = e_j) * \text{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

$$\text{VPI}(F) = (\text{P}(F = rain) * \text{MEU}(take \mid F = rain) + \text{P}(F = sun) * \text{MEU}(\text{leave} \mid F = sun)) - MEU(leave) =$$

$$(0.41 * 53 + 0.59 * 95) - 70 = 7.78$$

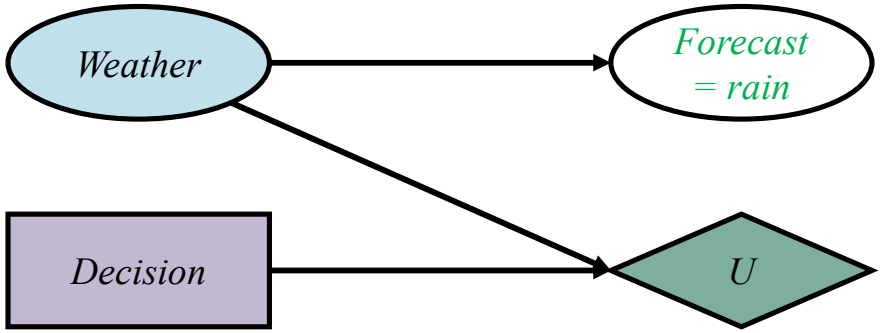
Decision Networks: Example

Decision:leave umbrella



EU(leave) = 70

Decision:take umbrella given rain



EU(take given rain forecast) = 53

The value of best action α without additional evidence

$\text{MEU}(\alpha) = \text{MEU}(\text{leave}) = 70$

With evidence information ($E_j = e_j$) given by Forecast:

$\text{MEU}(a_{e_1} \mid e_1) = \text{MEU}(\text{take} \mid F = rain) = 53$

$\text{MEU}(a_{e_2} \mid e_2) = \text{MEU}(\text{leave} \mid F = sun) = 95$

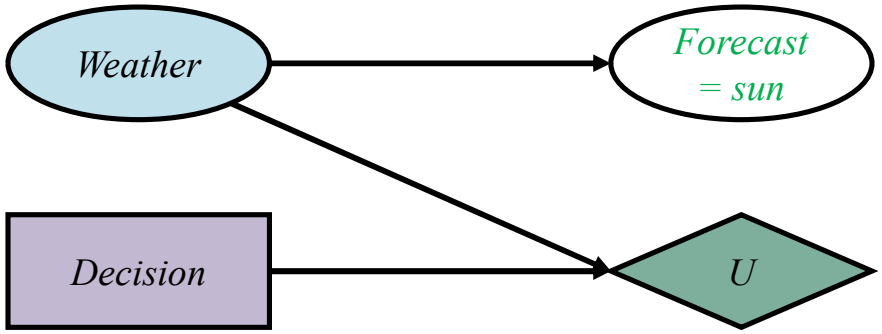
The value of additional evidence / information from F is:

$$\text{VPI}(E_j) = \left(\sum_{e_j} \text{P}(E_j = e_j) * \text{MEU}(a_{e_j} \mid E_j = e_j)\right) - MEU(a)$$

$$\text{VPI}(F) = (\text{P}(F = rain) * \text{MEU}(take \mid F = rain) + \text{P}(F = sun) * \text{MEU}(\text{leave} \mid F = sun)) - MEU(leave) =$$

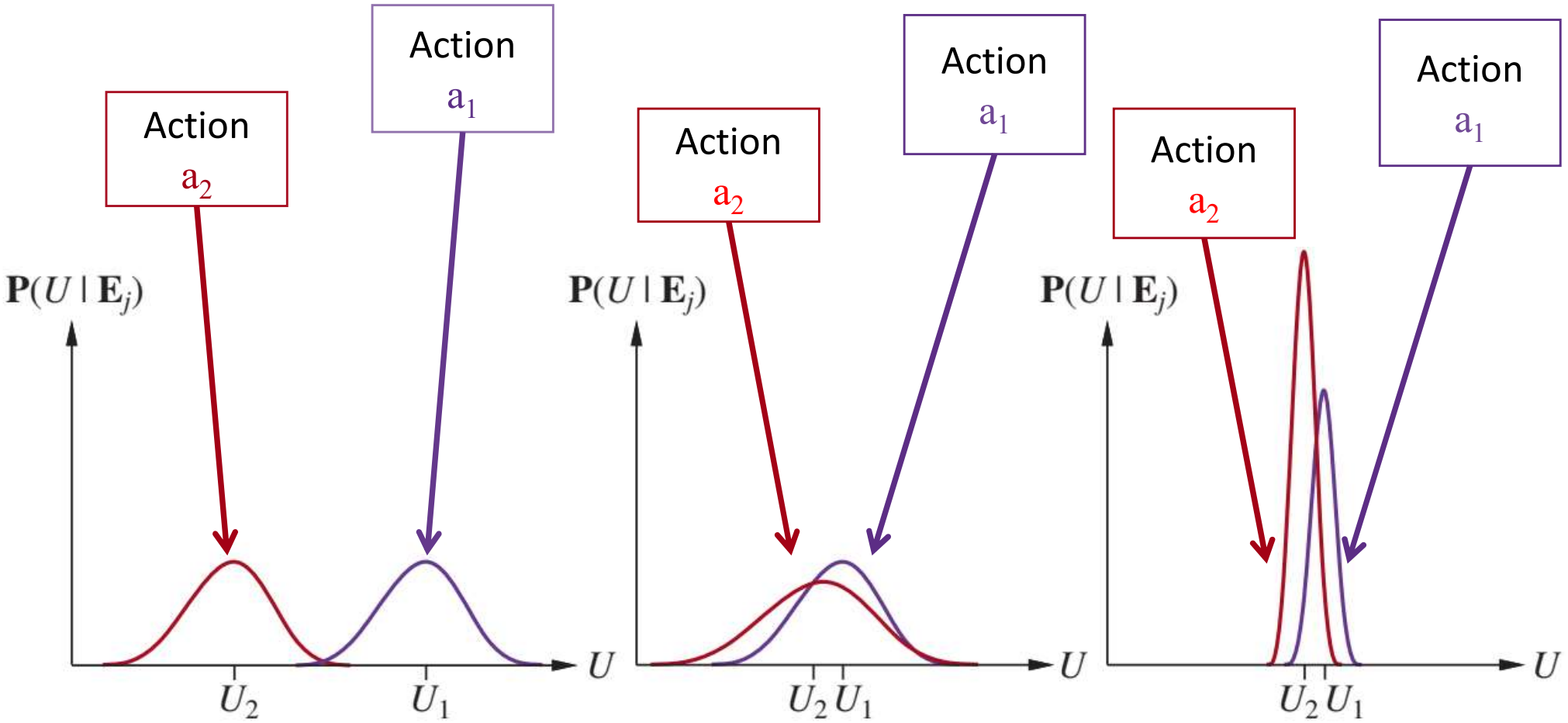
$$(0.41 * 53 + 0.59 * 95) - 70 = 7.78$$

Decision:leave umbrella given sun



EU(leave given sun forecast) = 95

Utility & Value of Perfect Information



New information will not help here.

New information may help a lot here.

New information may help a bit here.

VPI Properties

Given a decision network with possible observations E_j (sources of new information / evidence):

- **The expected value of information is nonnegative:**

$$\forall_j \, \mathrm{VPI}(E_j) \geq 0$$

- **VPI is not additive:**

$$\mathrm{VPI}(E_j, E_k) \neq \mathrm{VPI}(E_j) + \mathrm{VPI}(E_k)$$

- **VPI is order-independent:**

$$\mathrm{VPI}(E_j, E_k) = VPI(E_j) + VPI(E_k \mid E_j) = VPI(E_k) + VPI(E_j \mid E_k) = VPI(E_k, E_j)$$

Information Gathering Agent

function INFORMATION-GATHERING-AGENT(*percept*) **returns** an *action*
  persistent: D, a decision network

  integrate *percept* into D
  $j \leftarrow$ the value that maximizes $VPI(E_j) \,/\, C(E_j)$
  if $VPI(E_j) > C(E_j)$
    then return $Request(E_j)$
  else return the best action from D

Expected Action Utility

The expected utility of an action a given the evidence is the average utility value of all possible outcomes s' of action a, weighted by their probability (belief) of occurence:

$$\mathrm{EU}(a) = \sum_{s'} \sum_{s} \mathrm{P}(s) * \mathrm{P}(s' \mid s, a) * \mathrm{U}(s') = \sum_{s'} \mathrm{P}(Result(a) = s') * \mathrm{U}(s')$$

Rational agent should choose an action that maximizes the expected utility:

$$\text{chosen action} = \underset{a}{\operatorname{argmax}}\ \mathrm{EU}(a)$$

How Did We Get Here?

Let's start with relationships (and related notation) between agent's preferences:

- **agent prefers A over B:**

$$A \succ B$$

- **agent is indifferent between A and B:**

$$A \sim B$$

- **agent prefers A over B or is indifferent between A and B (weak preference):**

$$A \succcurlyeq B$$

The Concept of Lottery

Let's assume the following:

- **an action a is a lottery ticket**
- **the set of outcomes (resulting states) is a lottery**

A lottery L with possible outcomes S_1, ..., S_n that occur with probabilities p_1, ..., p_n is written as:

$$L = [p_1, S_1; p_2, S_2; \ldots ; p_n, S_n]$$

Lottery outcome S_i: atomic state or another lottery.

Lottery Constraints: Orderability

Given two lotteries A and B, a rational agent must either prefer one or else rate them as equally preferable:

Exactly one of $(\mathrm{A} \succ \mathrm{B})$, $(\mathrm{B} \succ \mathrm{A})$, or $(\mathrm{A} \sim \mathrm{B})$ holds

Lottery Constraints: Transitivity

Given three lotteries A, B, and C, if an agent prefers A to B AND prefers B to C, then the agent must prefer A to C:

$$(\mathrm{A} \succ \mathrm{B}) \wedge (\mathrm{B} \succ \mathrm{C}) \Rightarrow (\mathrm{A} \succ \mathrm{C})$$

Lottery Constraints: Continuity

If some lottery B is between A and C in preference, then there is some probability p for which the rational agent will be indifferent between getting B for sure or some other lottery that yields A with probability p and C with probability $1 - p$:

$$(A \succ B \succ C) \Rightarrow \exists p\ [p, A;\ 1- p, C] \sim B$$

Lottery Constraints: Substitutability

If an agent is indifferent between two lotteries A and B, then the agent is indifferent between two more complex lotteries that are the same, except that B is subsituted for A in one of them:

$$(A \sim B) \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$$

Lottery Constraints: Monotonicity

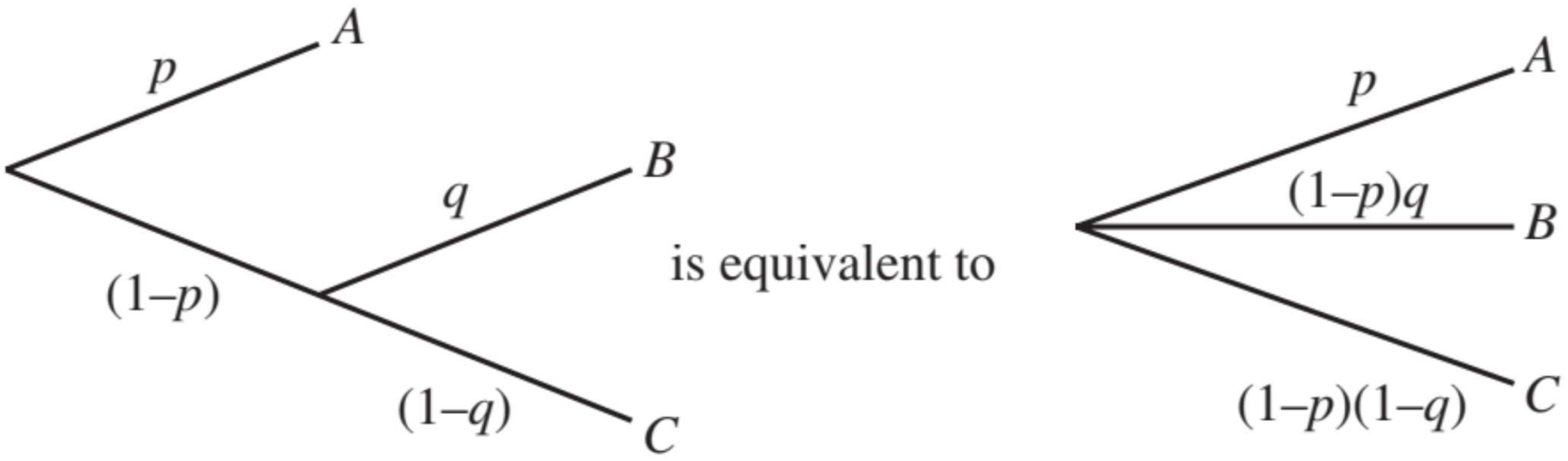
Suppose two lotteries have the same two possible outcomes, A and B. If an agent prefers A to B, then the agent must prefer the lottery that has a higher probability for A:

$$(A \succ B) \Rightarrow (p > q \Leftrightarrow [p, A;\ 1-p, B] \succ [q, A;\ 1-q, B])$$

Lottery Constraints: Decomposability

Compound lotteries can be reduced to smaller ones using the laws of probability:

[p, A; 1- p, [q, B; 1- q, C]] ~ [p, A; (1- p)*q, B; (1- p)*(1- q), C]



Preferences and Utility Function

An agent whose preferences between lotteries follow the set of axioms (of utility theory) below:

- **Orderability**
- **Transitivity**
- **Continuity**
- **Subsitutability**
- **Monotonicity**
- **Decomposability**

can be described as possesing a utility function and maximize it.

Preferences and Utility Function

If an agent's preferences obey the axioms of utility theory, then there exist a function U such that:

$$U(A) = U(B) \text{ if and only if } (A \sim B)$$

and

$$U(A) > U(B) \text{ if and only if } (A \succ B)$$

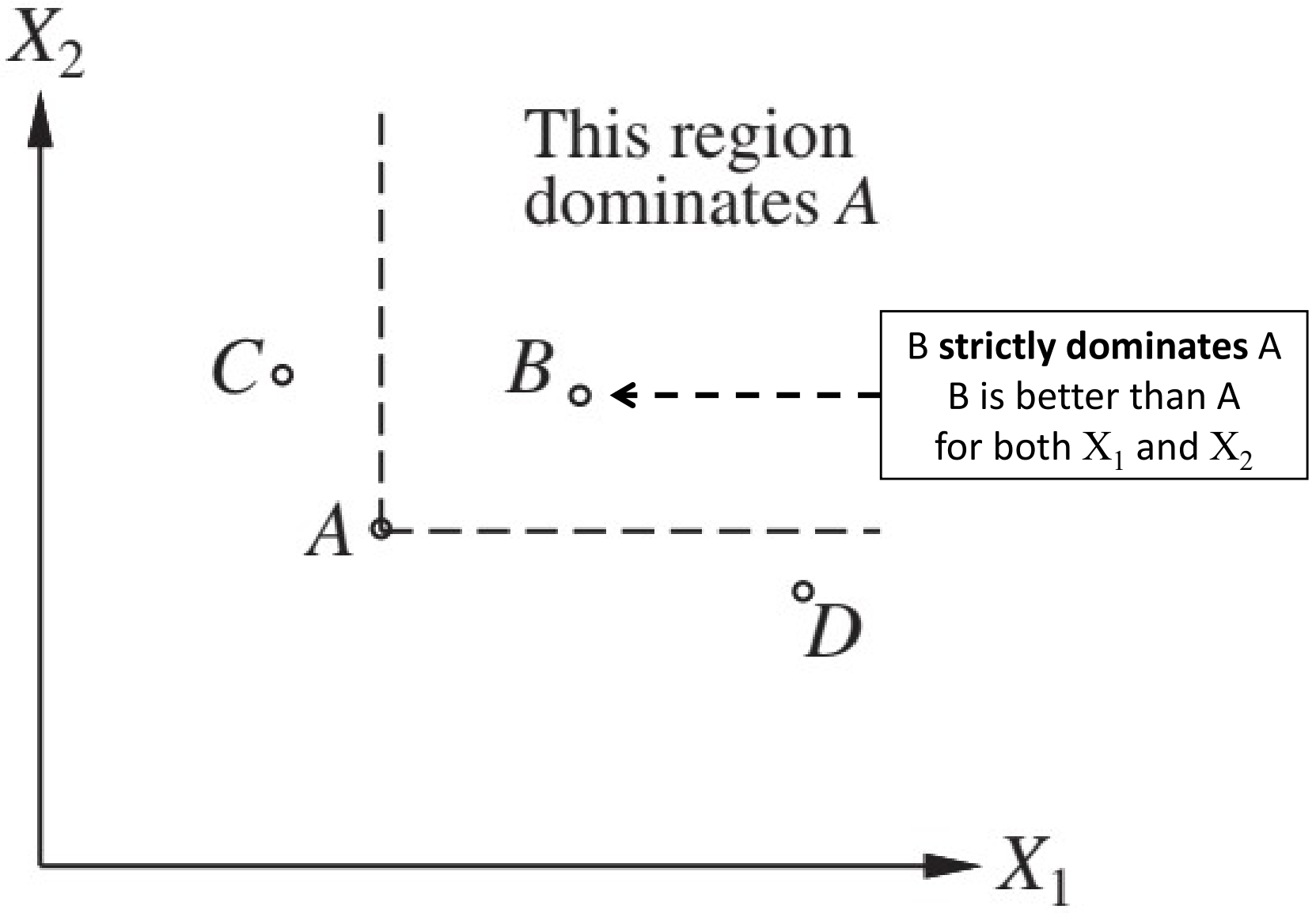
Multiattribute Outcomes

Outcomes can be characterized by more than one attribute. Decisions in such cases are handled by Multiattribute Utility Theory.

Attributes: $\mathbf{X} = X_1, \ldots, X_n$

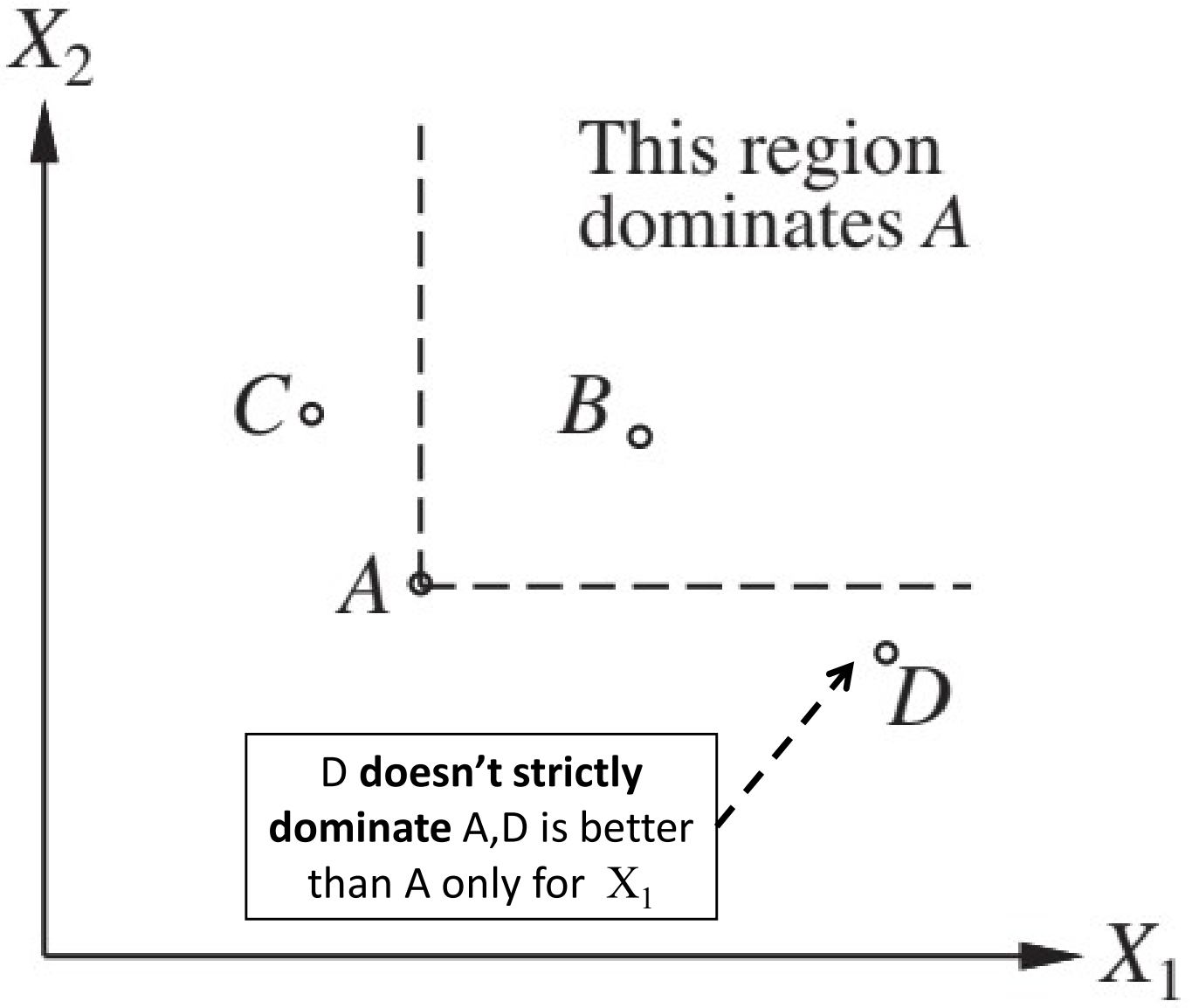
Assigned values: $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$

Strict Dominance: Deterministic

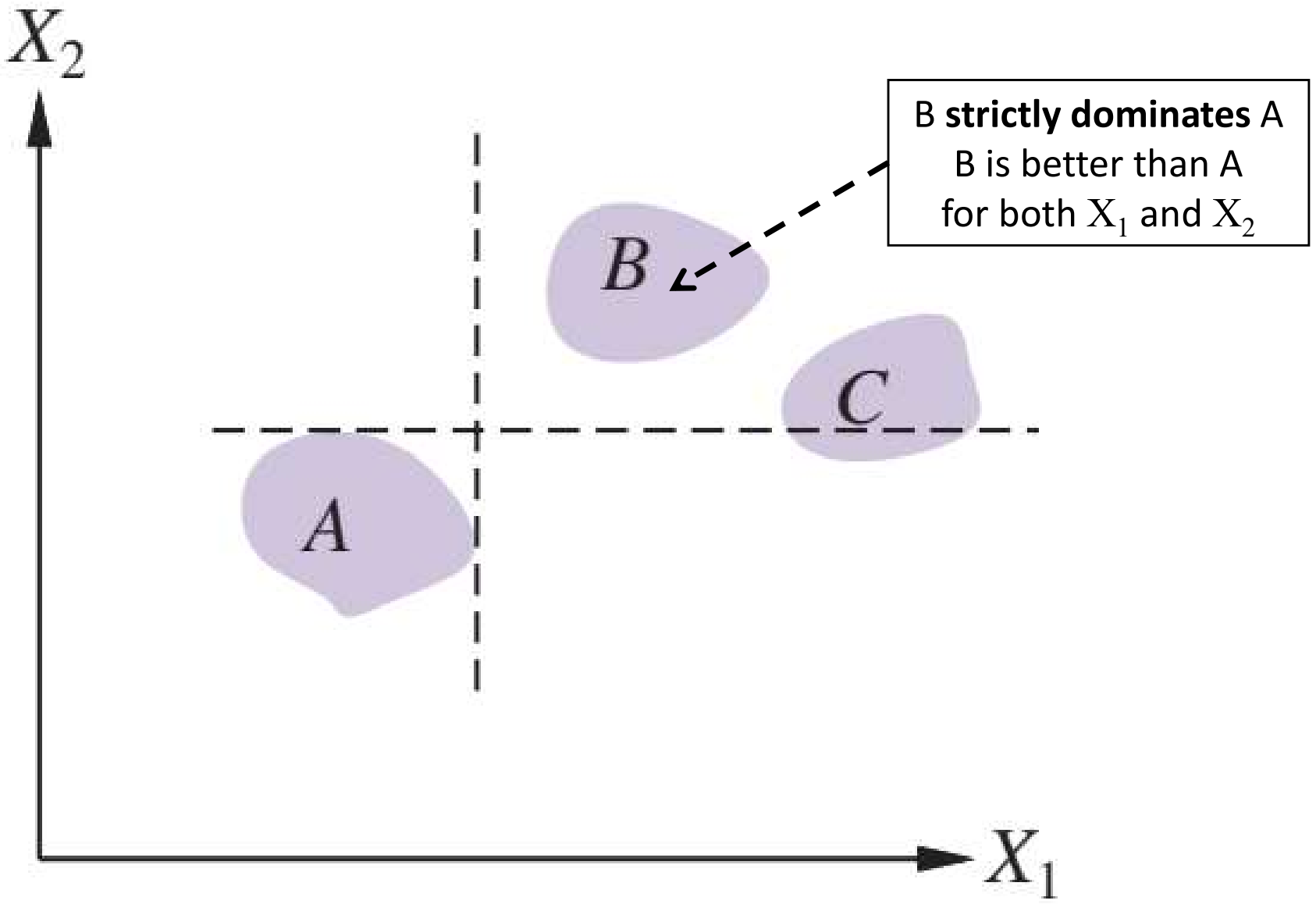


B **strictly dominates** A
B is better than A
for both X_1 and X_2

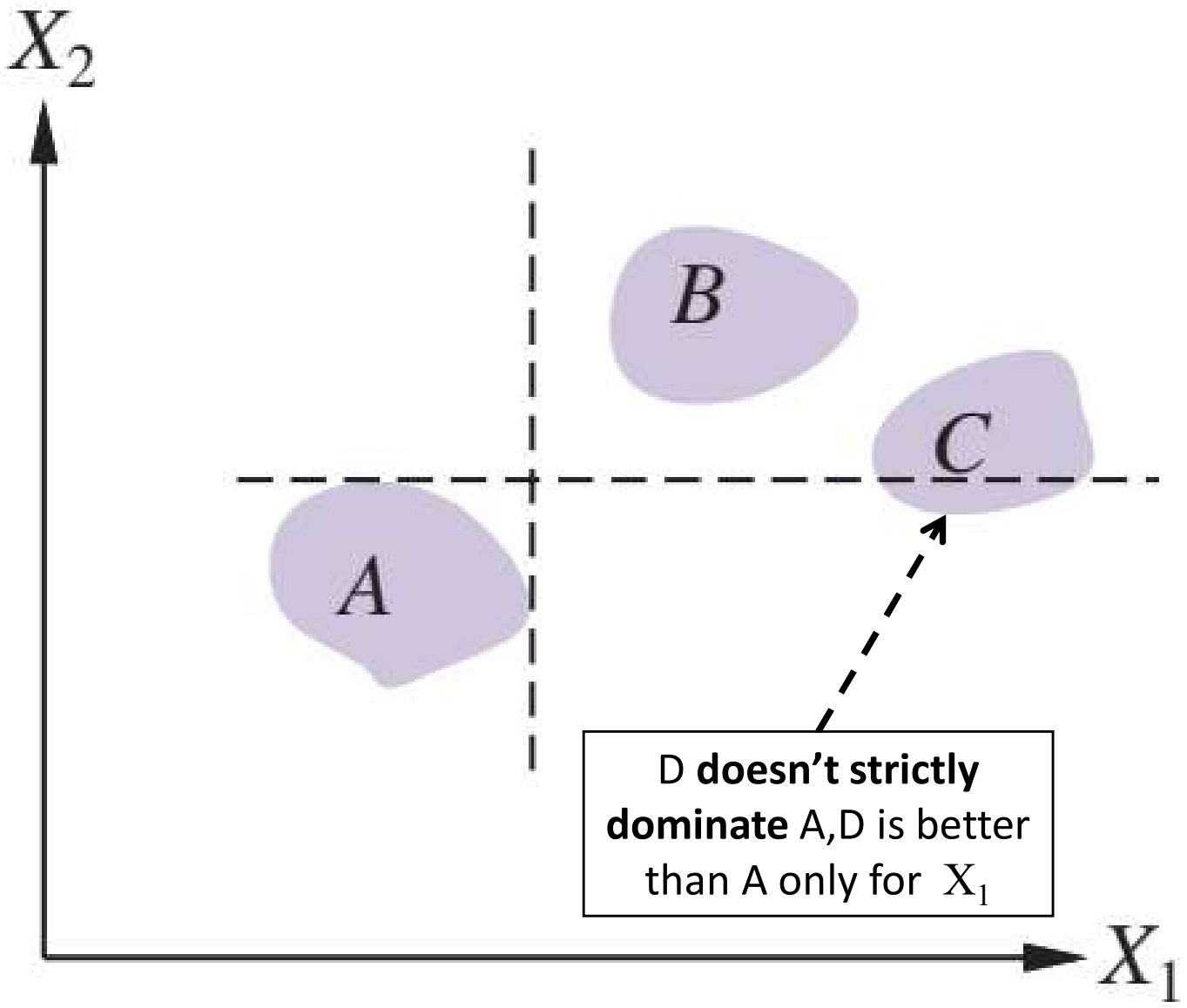
Strict Dominance: Deterministic



Strict Dominance: Uncertain



Strict Dominance: Uncertain

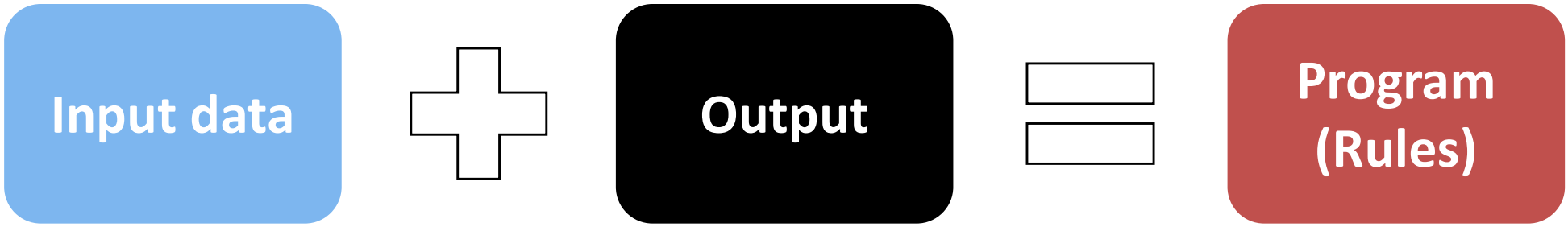


What is Machine Learning?

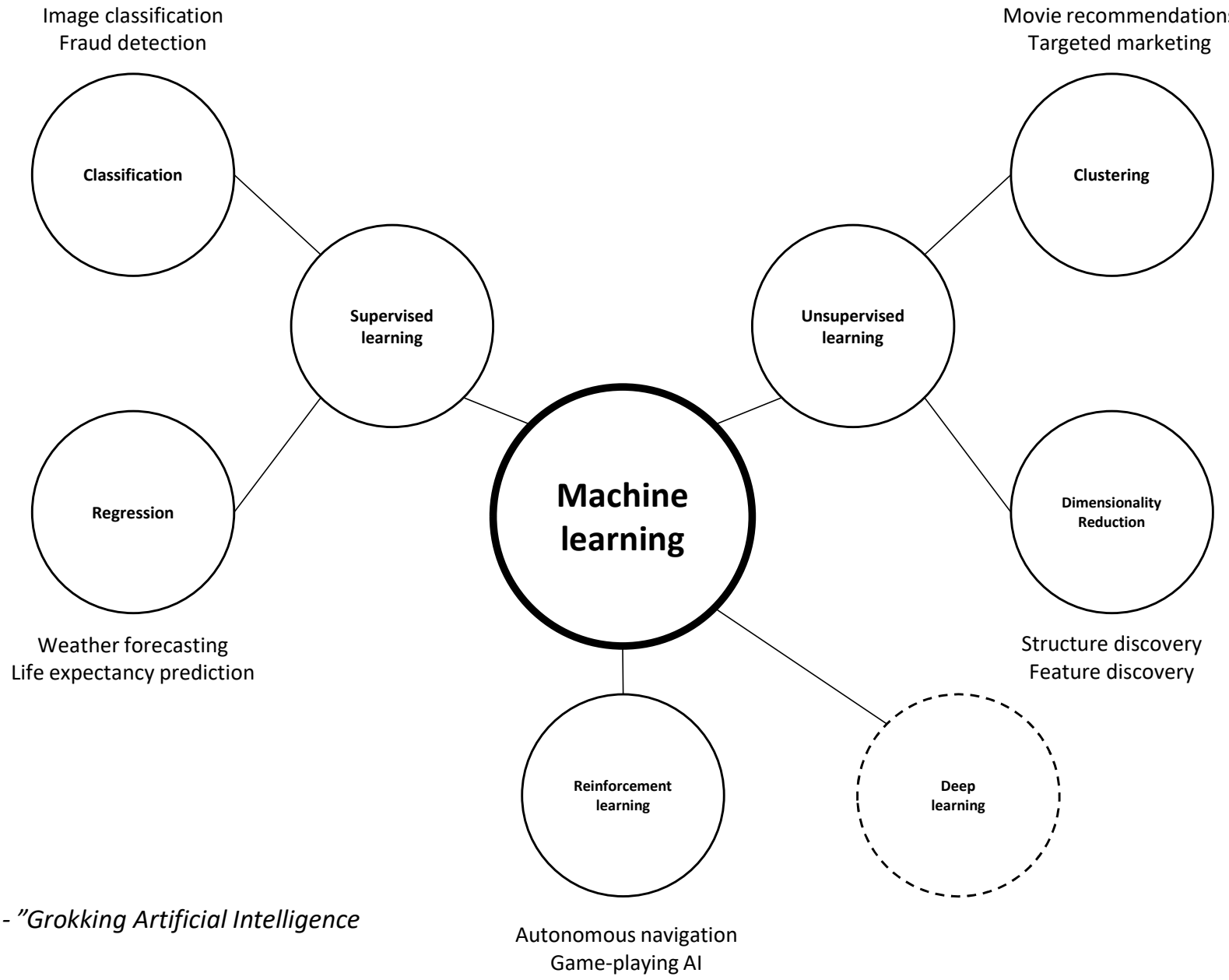
Traditional Programming vs ML

Traditional programming:

Machine learning:



Machine Learning Categories

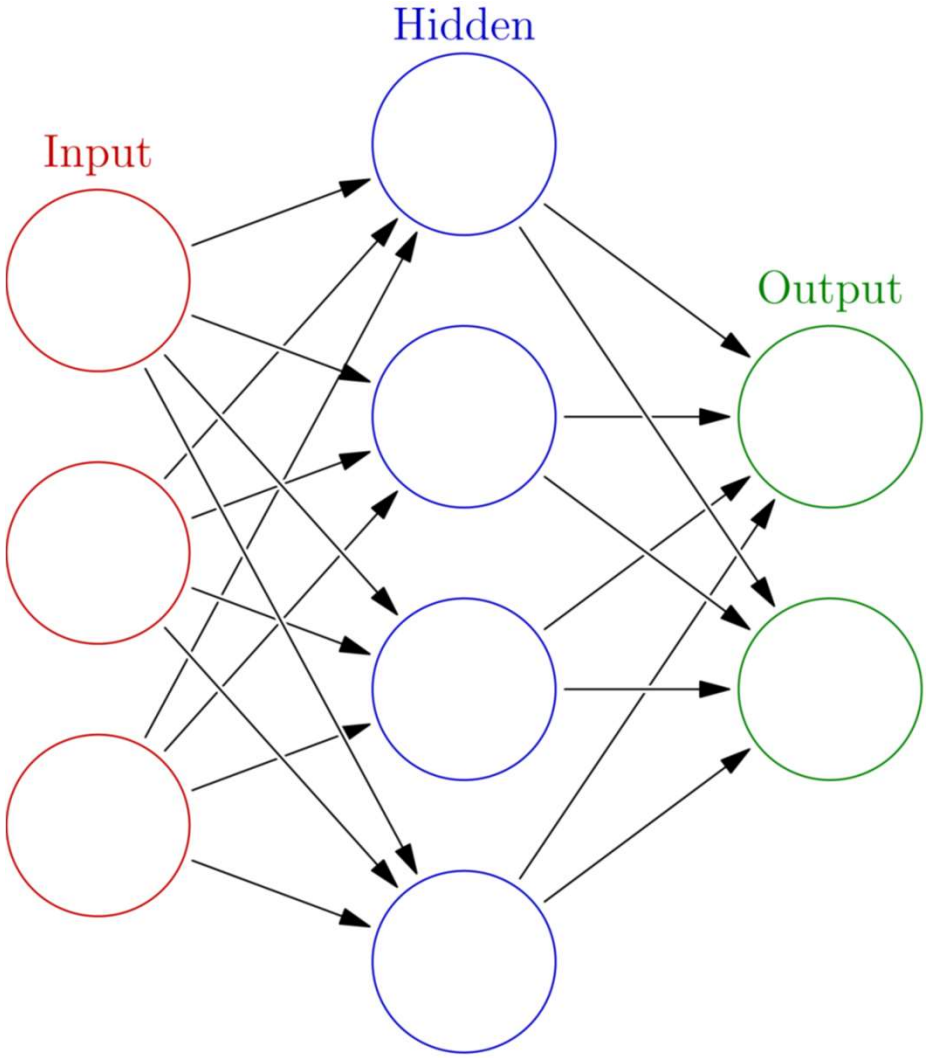


Based on:
Rishal Hurbans - "Grokking Artificial Intelligence Algorithms"

What Kind of Questions ML Answers?

Question	ML Category	Example
Is this A or B?	Classification	Will this car fail in the next two months? Yes or no?
Is this weird?	Anomaly detection	Is this credit card charge normal?
How much / many?	Regression	What will the temperature be tomorrow?
How is this organized?	Clustering	Which car models have the most brake problems?
What should I do next?	Reinforcement learning	Adjust room humidity or leave as is?

Artificial Neural Network (1943)



Source: https://en.wikipedia.org/wiki/Artificial_neural_network

First computational models of an **Artificial Neural Network** (loosely inspired by biological neural networks) were proposed by Warren McCulloch and Walter Pitts in 1943. Their ideas are a **key component of modern day machine and deep learning**.

Main Machine Learning Categories

Supervised learning	Unsupervised learning	Reinforcement learning
Supervised learning is one of the most common techniques in machine learning. It is based on **known relationship(s) and patterns within data** (for example: relationship between inputs and outputs). Frequently used types: **regression**, and **classification**.	**Unsupervised learning** involves finding underlying patterns within data. Typically used in **clustering** data points (similar customers, etc.)	Reinforcement learning is inspired by behavioral psychology. It is **based on a rewarding / punishing an algorithm**. Rewards and punishments are based on algorithm's action within its environment.

Choosing Hypothesis / Model

Given a training set of N example input-output (feature-label) pairs

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

where each pair was generated by

$$y = f(x)$$

Ideally, we would like our model $h(x)$ (hypothesis) that approximates the true function $f(x)$ to be:

$$h(x) = y = f(x)$$ **(consistent hypothesis)**

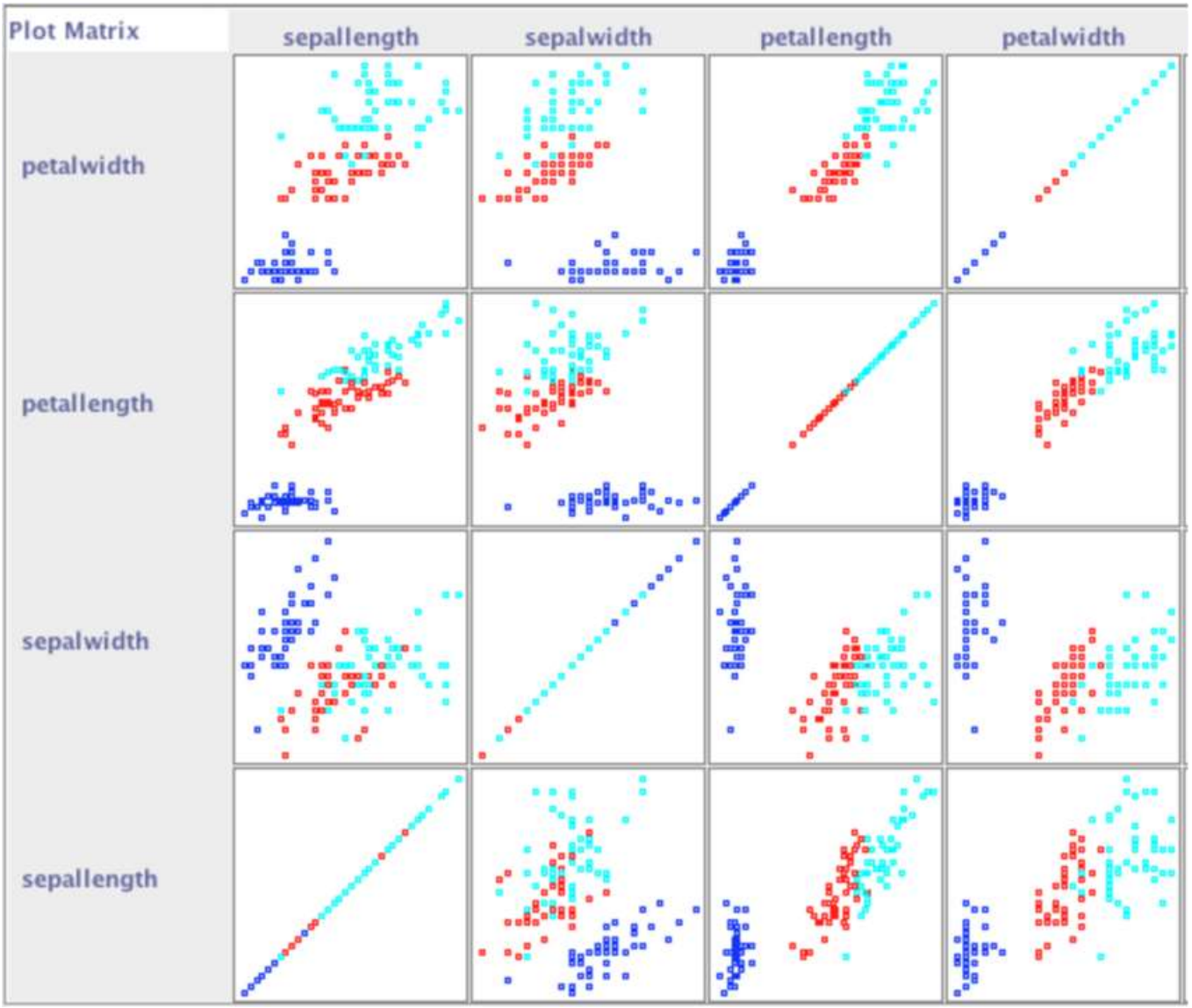
Choosing Hypothesis / Model

Typically consistent hypothesis is impossible or difficult to achieve:

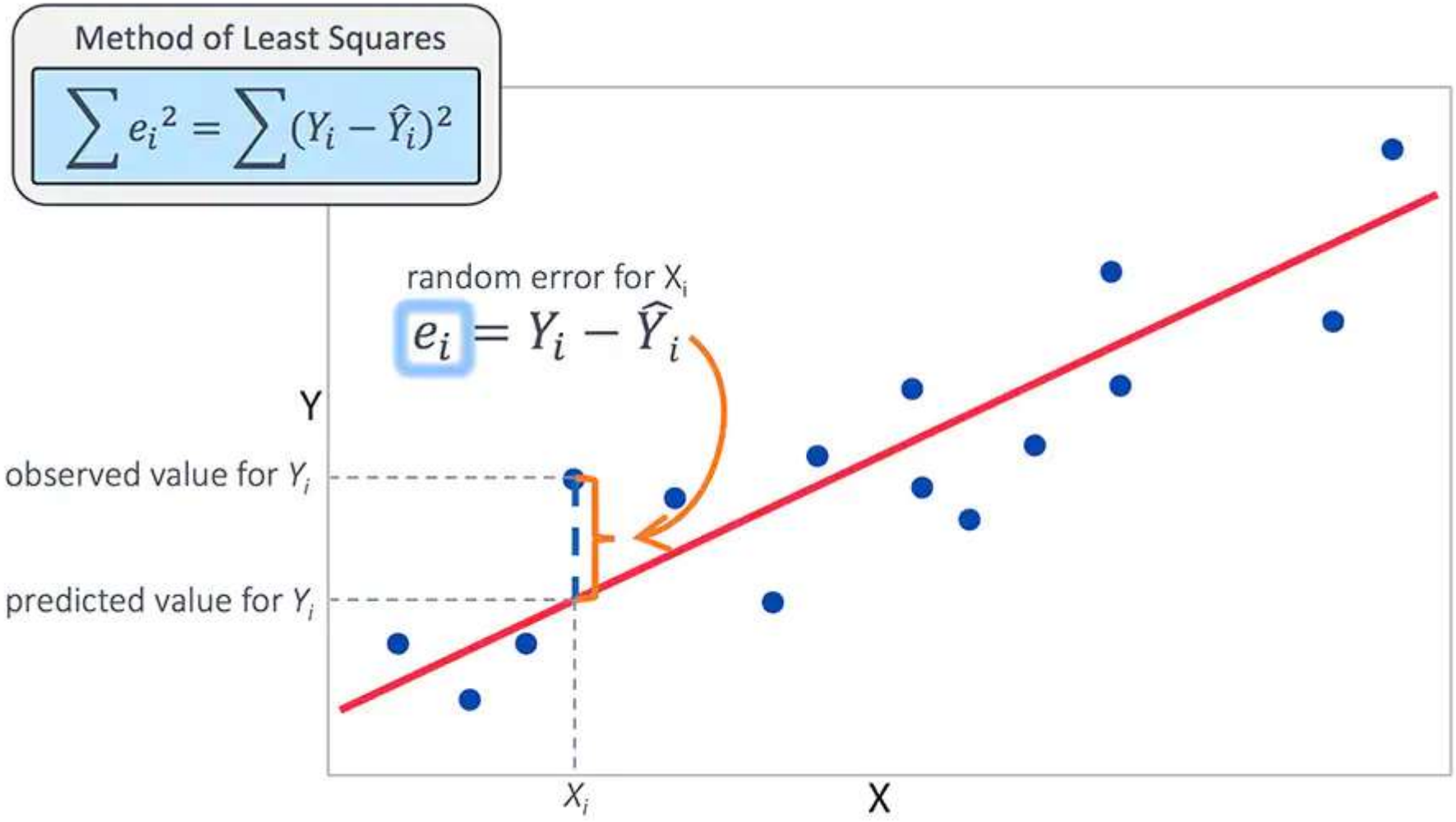
- **use best-fit model / hypothesis**

Our model needs to be tested on the test set inputs (data the model has not "seen" yet) to see how well it generalizes (how accurately it predicts the outputs of the test set).

Exploratory Data Analysis



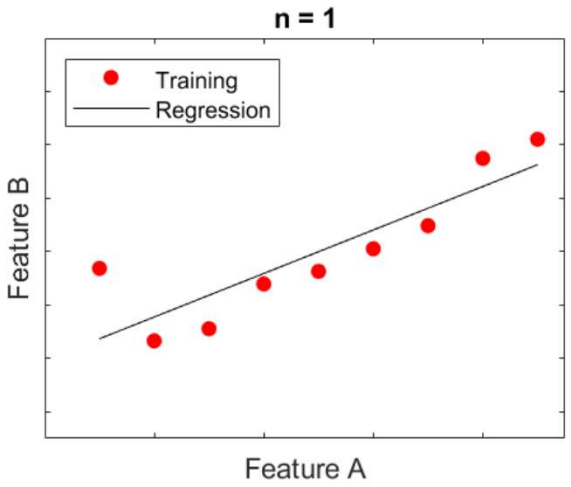
Linear Regression Using Least-Squares

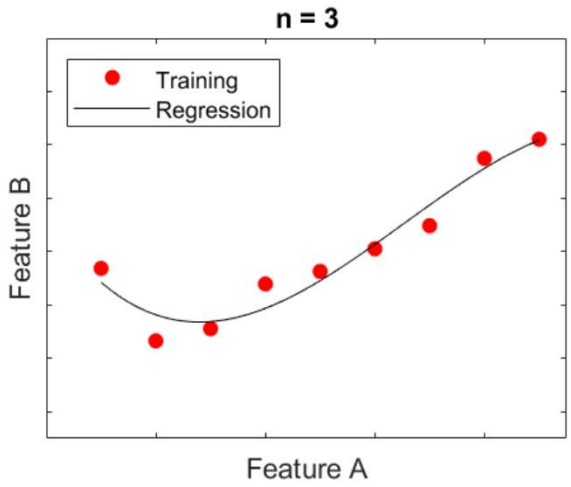


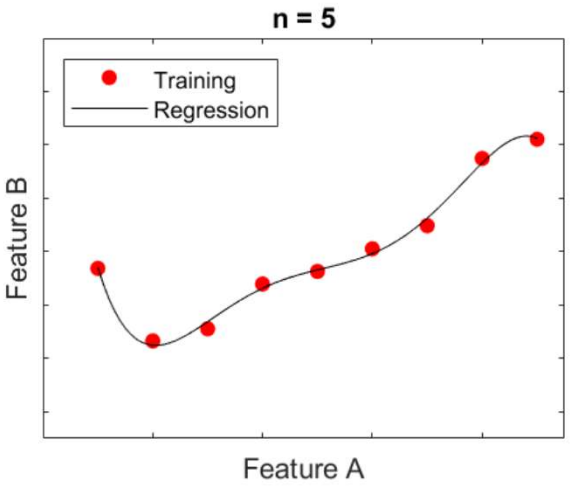
The goal is to find the line y = ax + b that minimizes the amount of error.

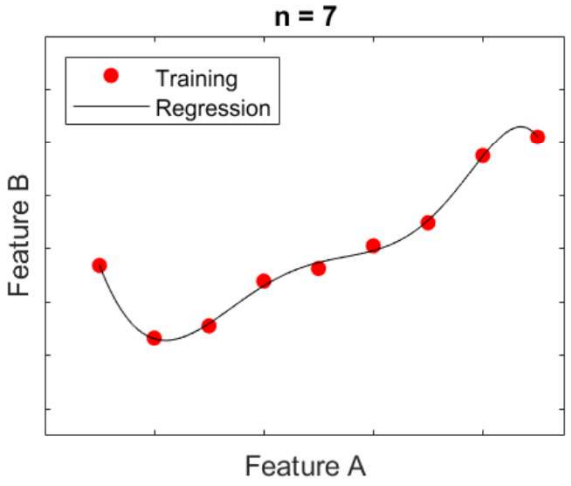
Source: https://www.jmp.com/en_us/statistics-knowledge-portal/what-is-multiple-regression/fitting-multiple-regression-model.html

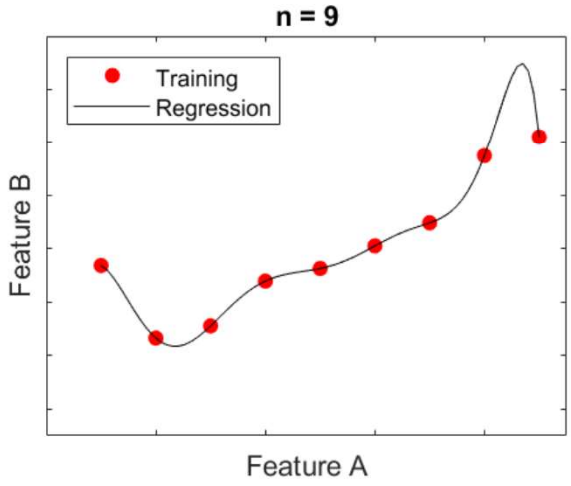
Regression

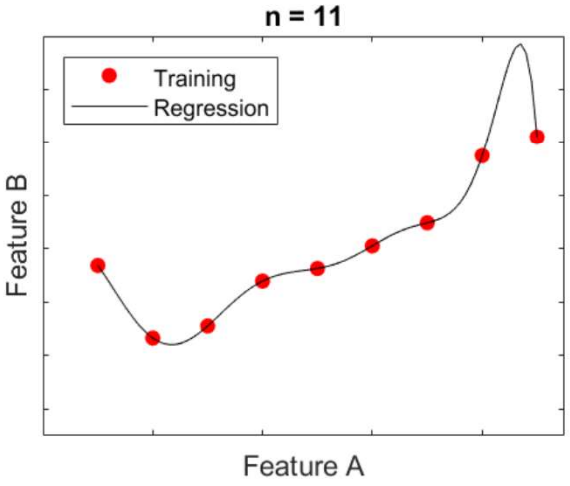

n = 1
Training
Regression
Feature B
Feature A


n = 3
Training
Regression
Feature B
Feature A

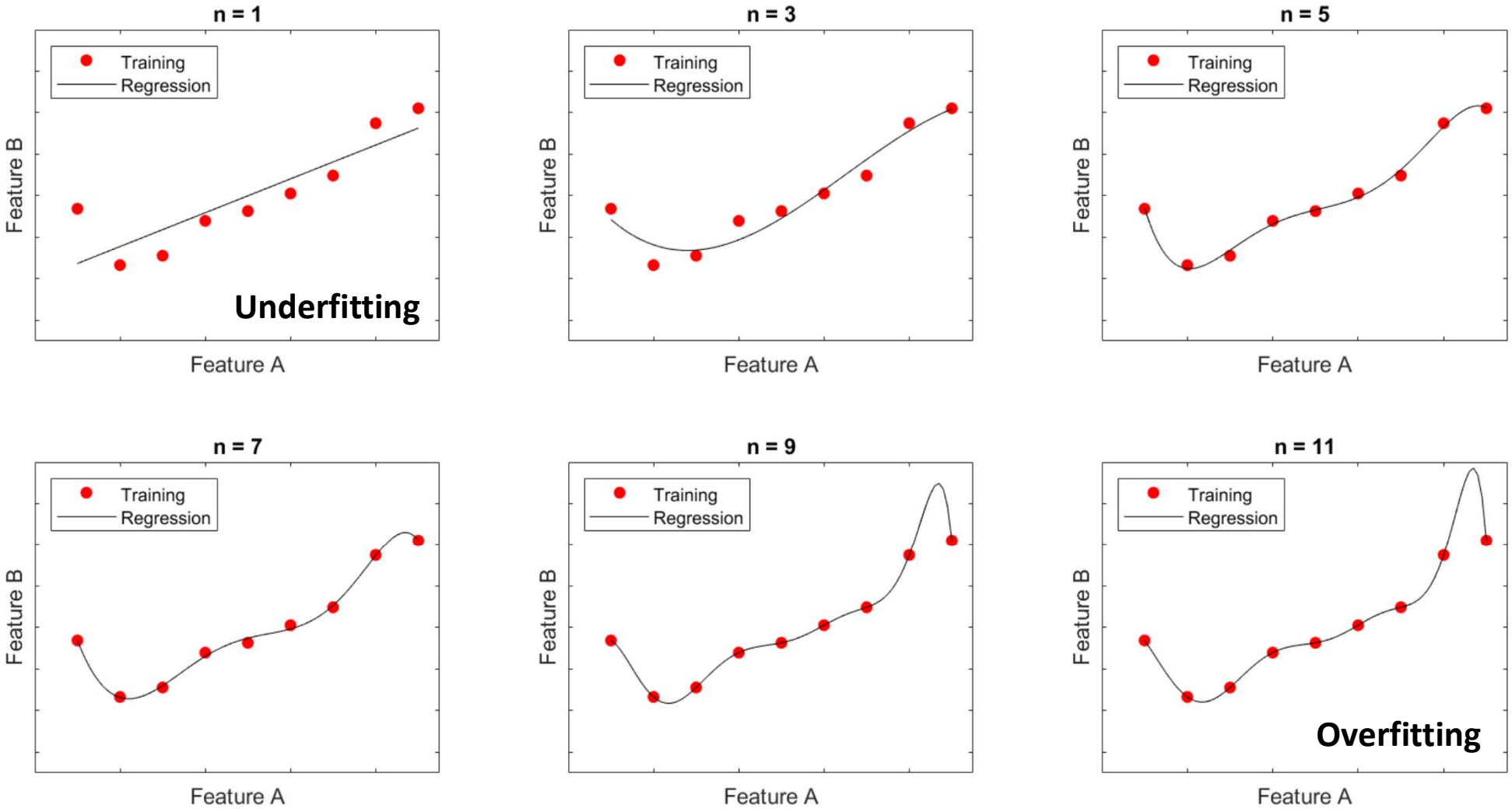

n = 5
Training
Regression
Feature B
Feature A


n = 7
Training
Regression
Feature B
Feature A


n = 9
Training
Regression
Feature B
Feature A

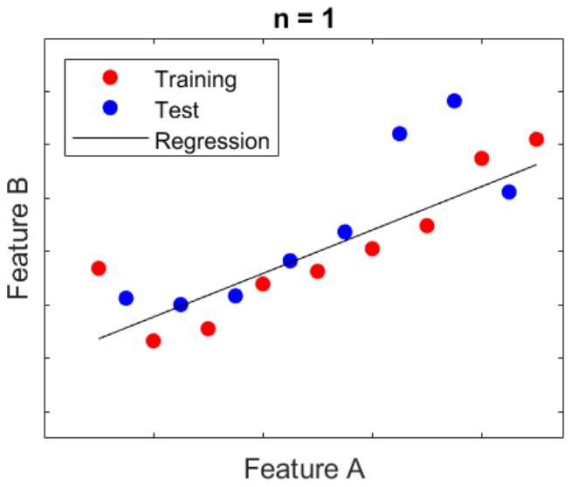

n = 11
Training
Regression
Feature B
Feature A

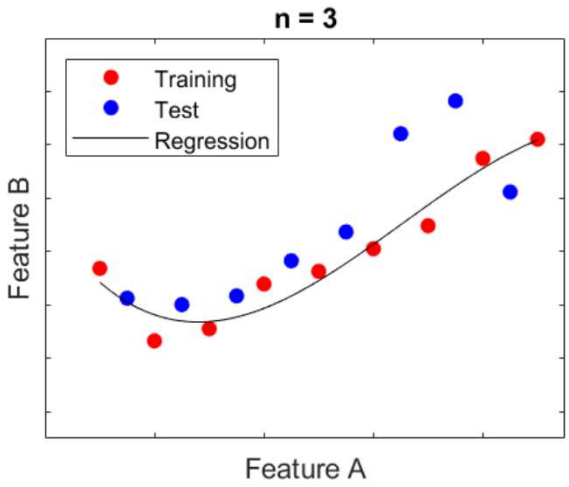
Regression: Underfitting / Overfitting

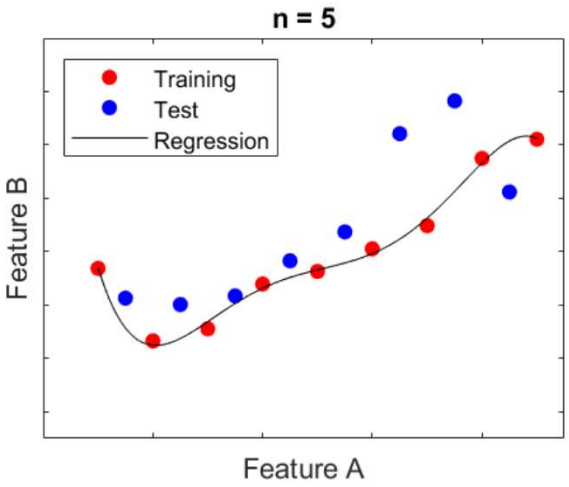


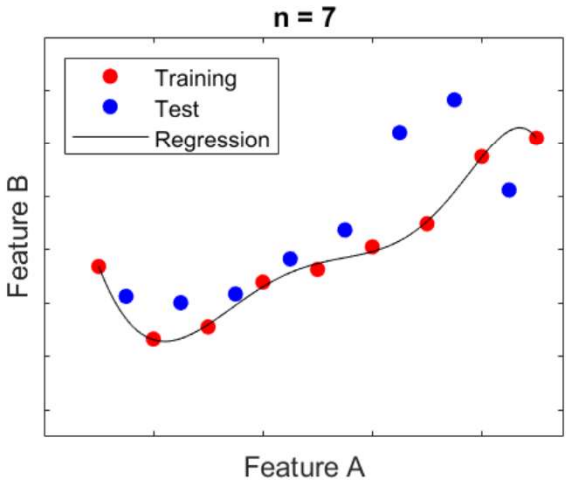
Underfitting: "failing" to find pattern in the data.

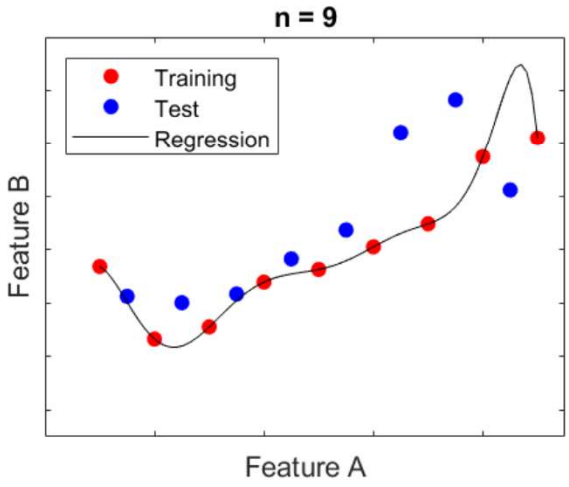
Regression

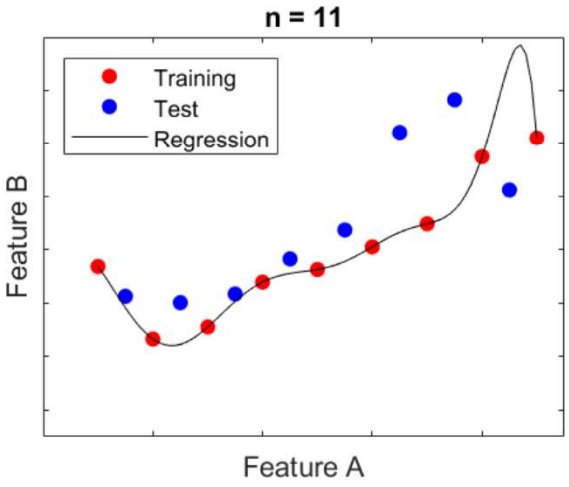

n = 1
Training
Test
Regression
Feature B
Feature A


n = 3
Training
Test
Regression
Feature B
Feature A

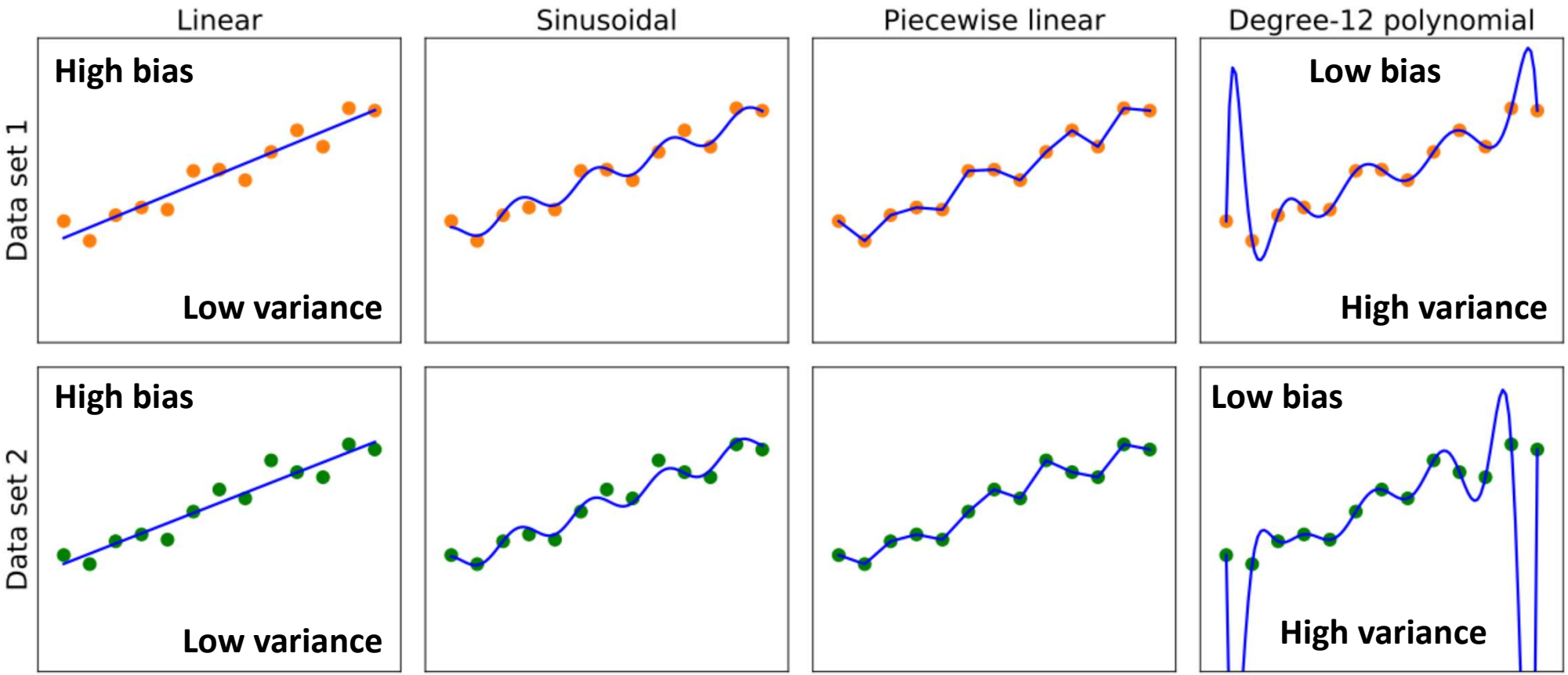

n = 5
Training
Test
Regression
Feature B
Feature A


n = 7
Training
Test
Regression
Feature B
Feature A


n = 9
Training
Test
Regression
Feature B
Feature A


n = 11
Training
Test
Regression
Feature B
Feature A

Bias vs. Variance



Bias: the tendency of a predictive hypothesis to deviate from the expected value when averaged over different training sets

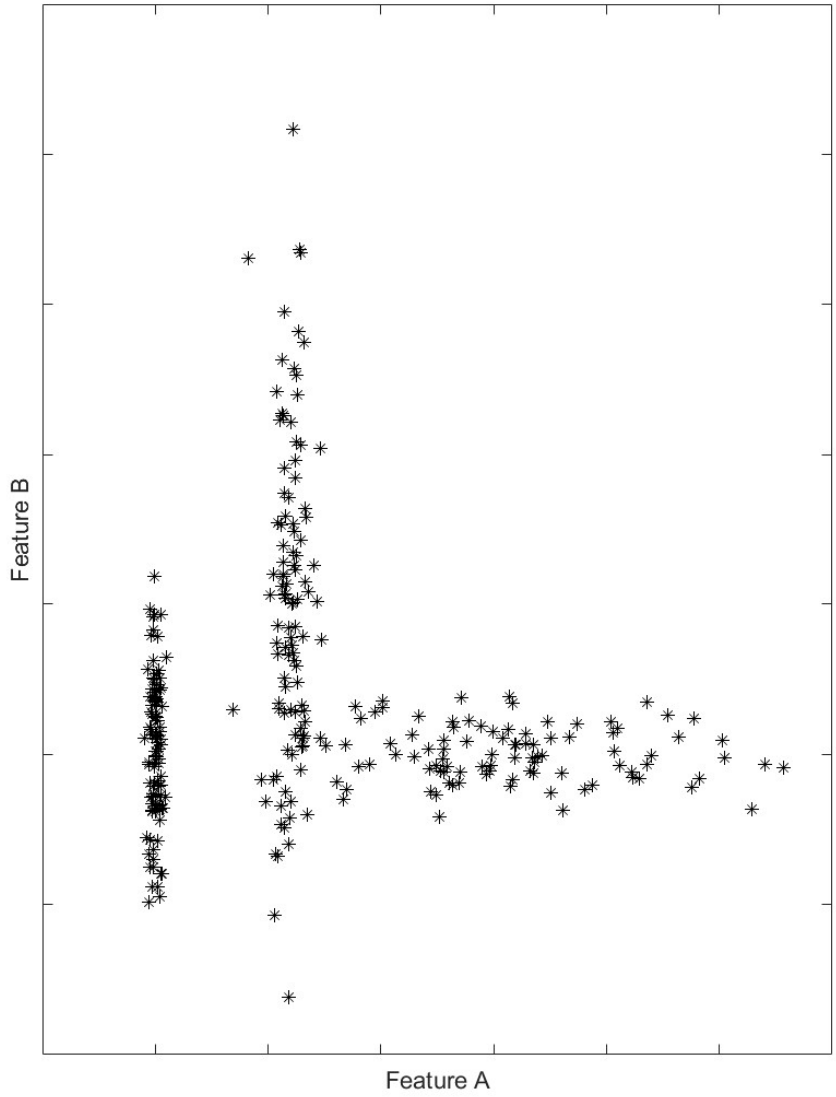
Variance: the amount of change in the hypothesis (model) due to fluctuations in training data

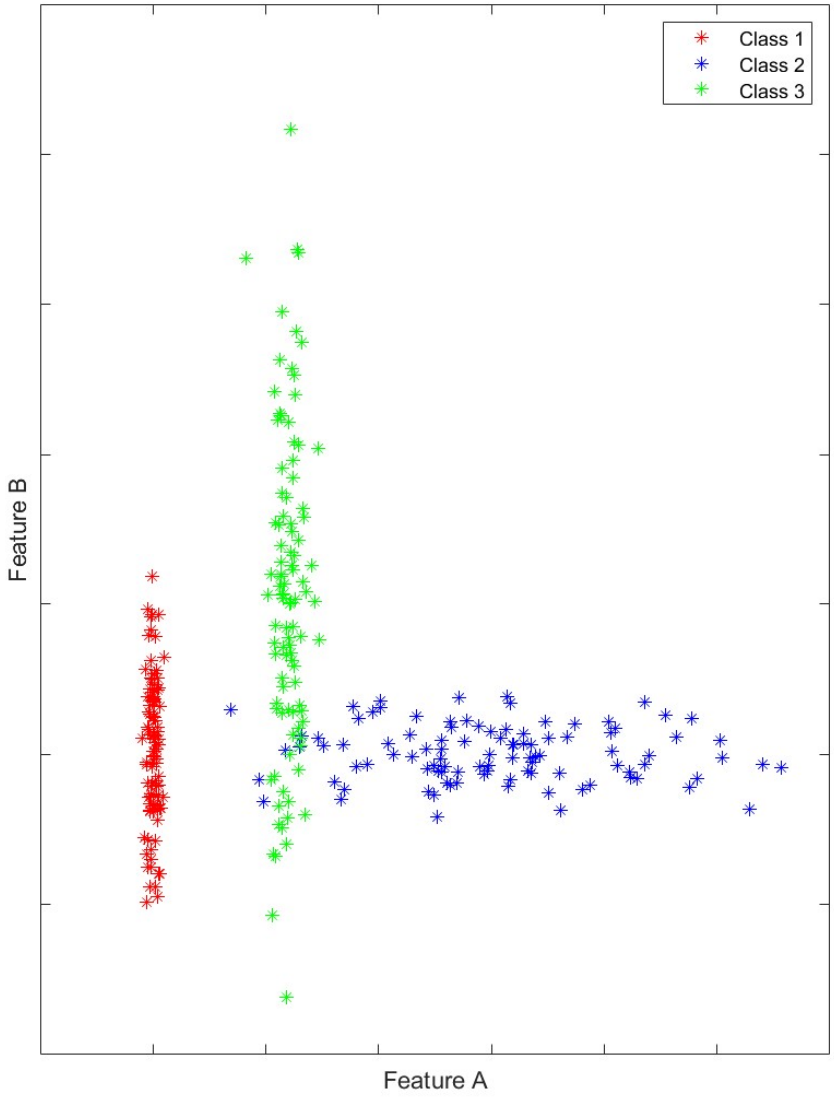
Training / Validation / Test Sets

In order to create the best model possible, given some (relatively large) data set, we should divide it into:

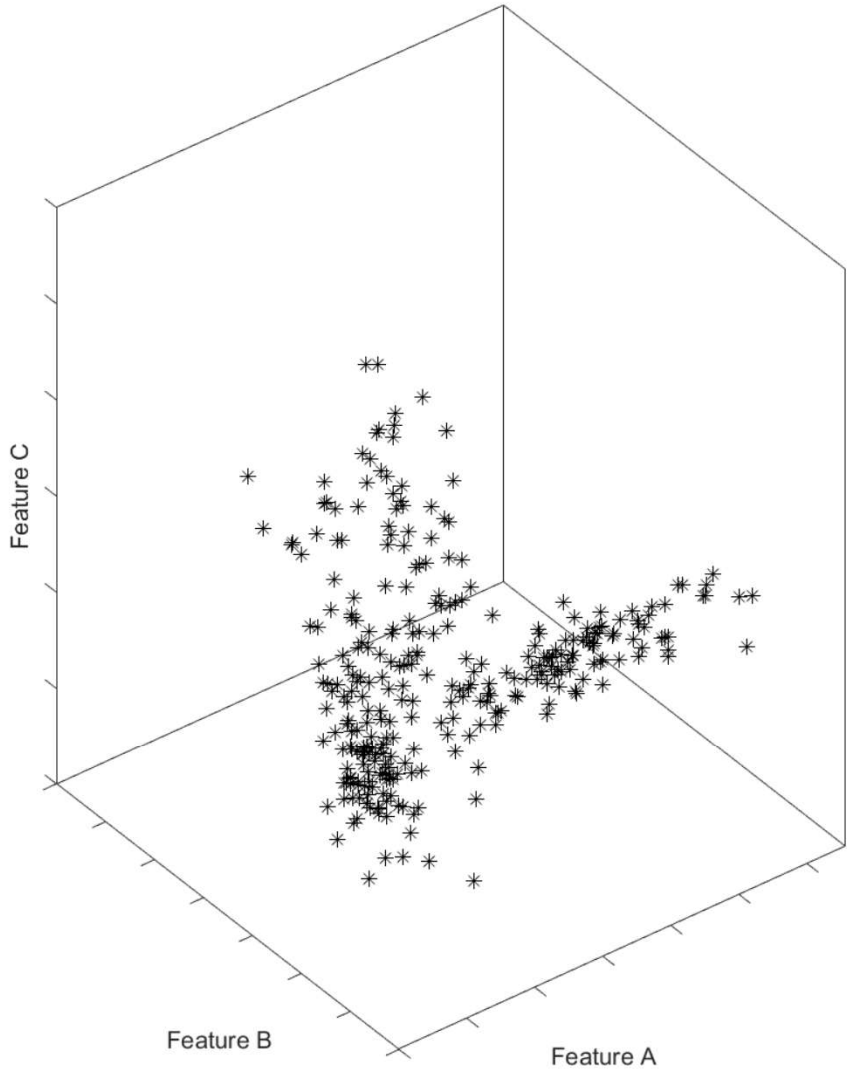
- training set: to train candidate models
- validation set: to evaluate candidate models and pick the best one
- test set: to do the final evaluation of the model

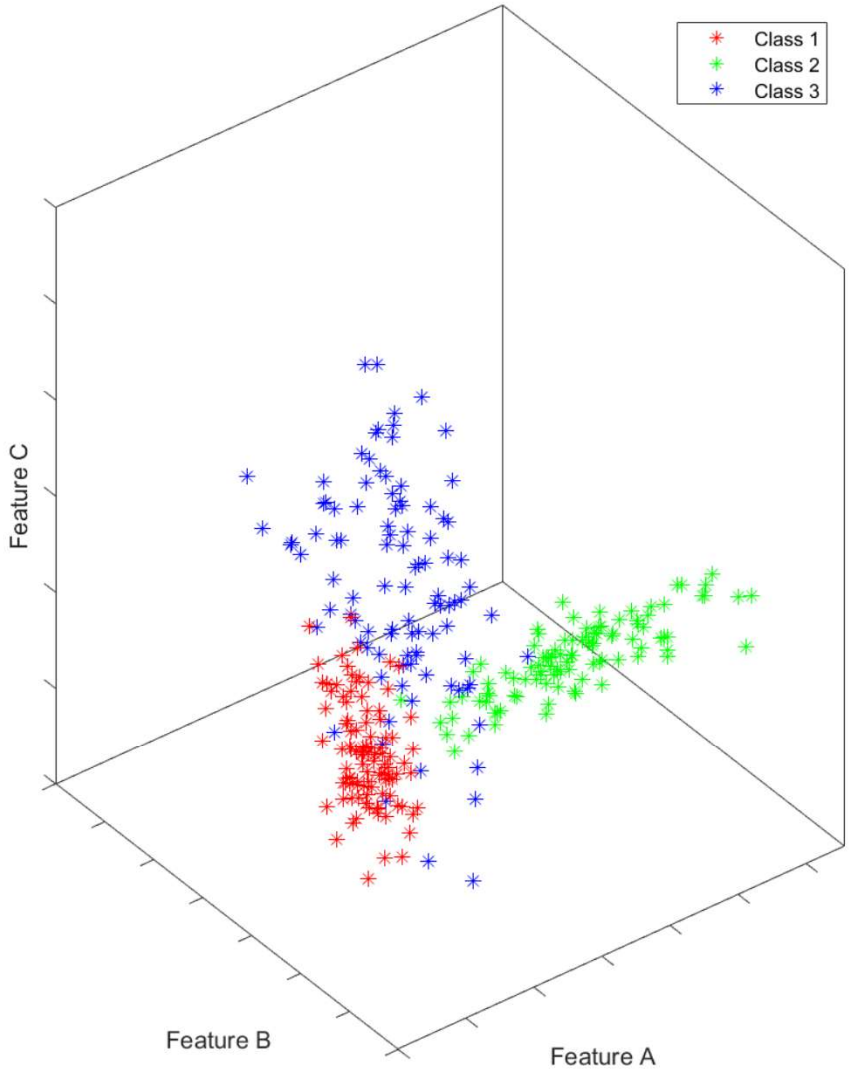
Supervised Learning: Classification



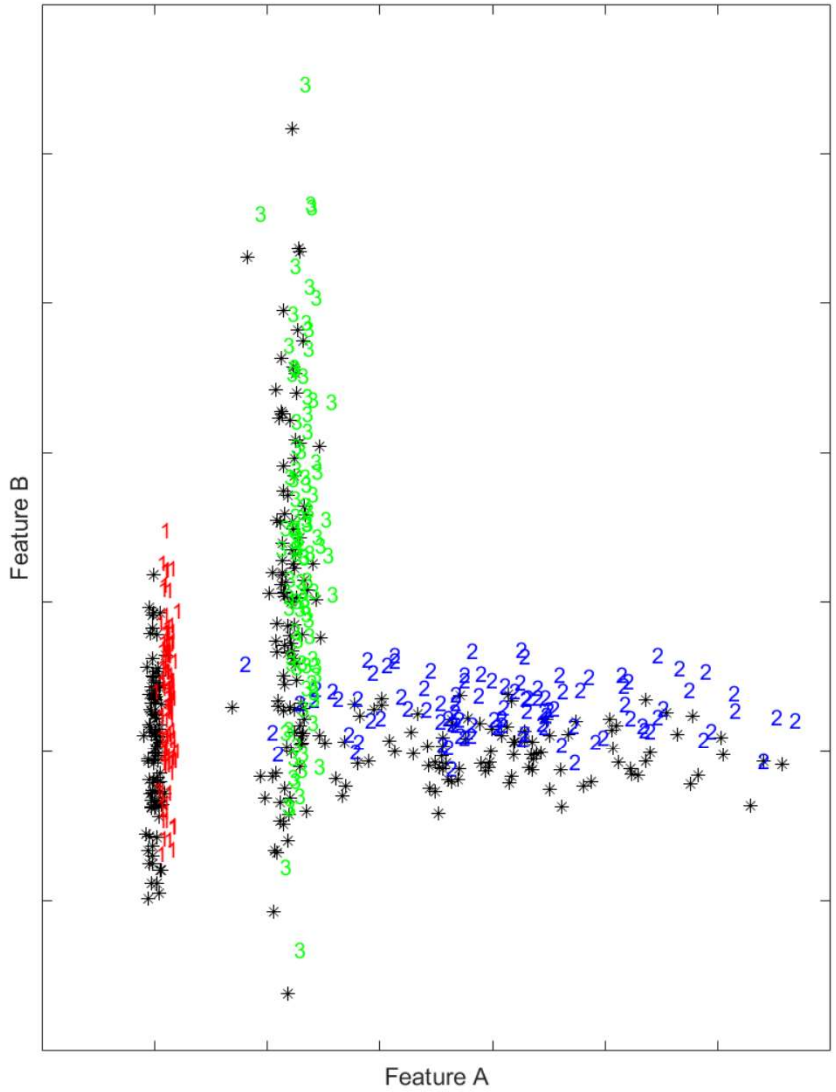


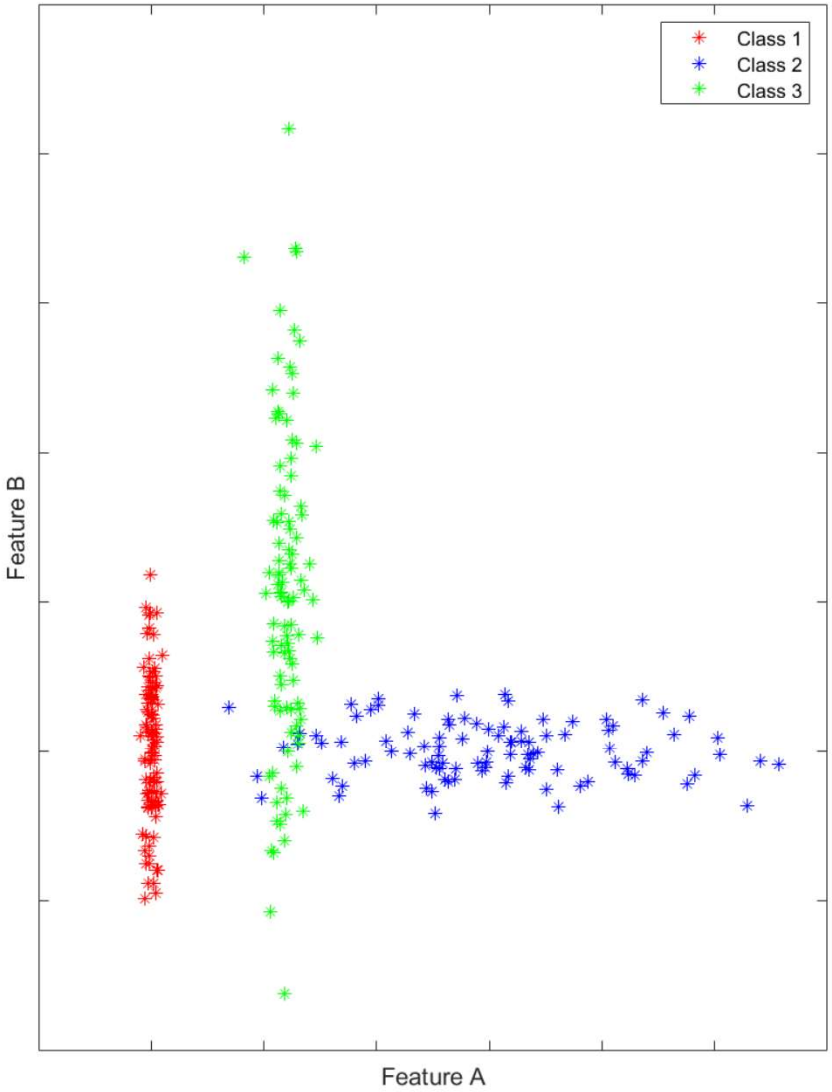
Supervised Learning: Classification



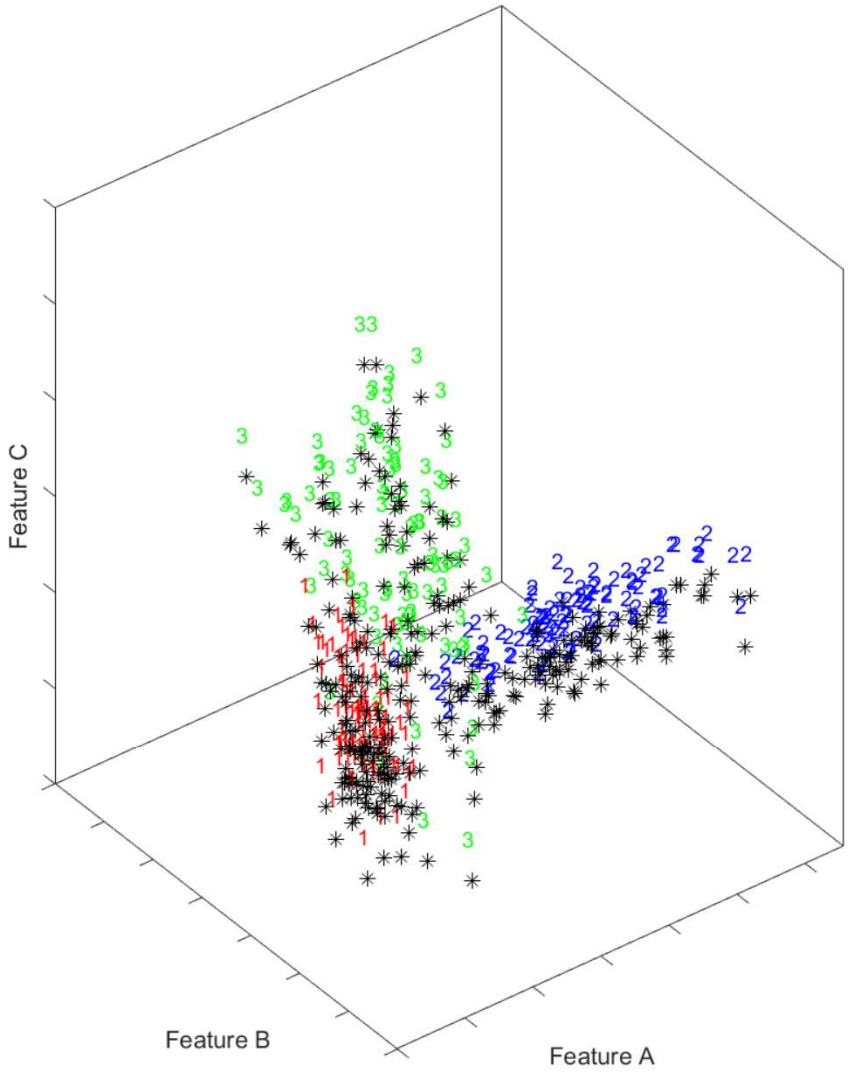


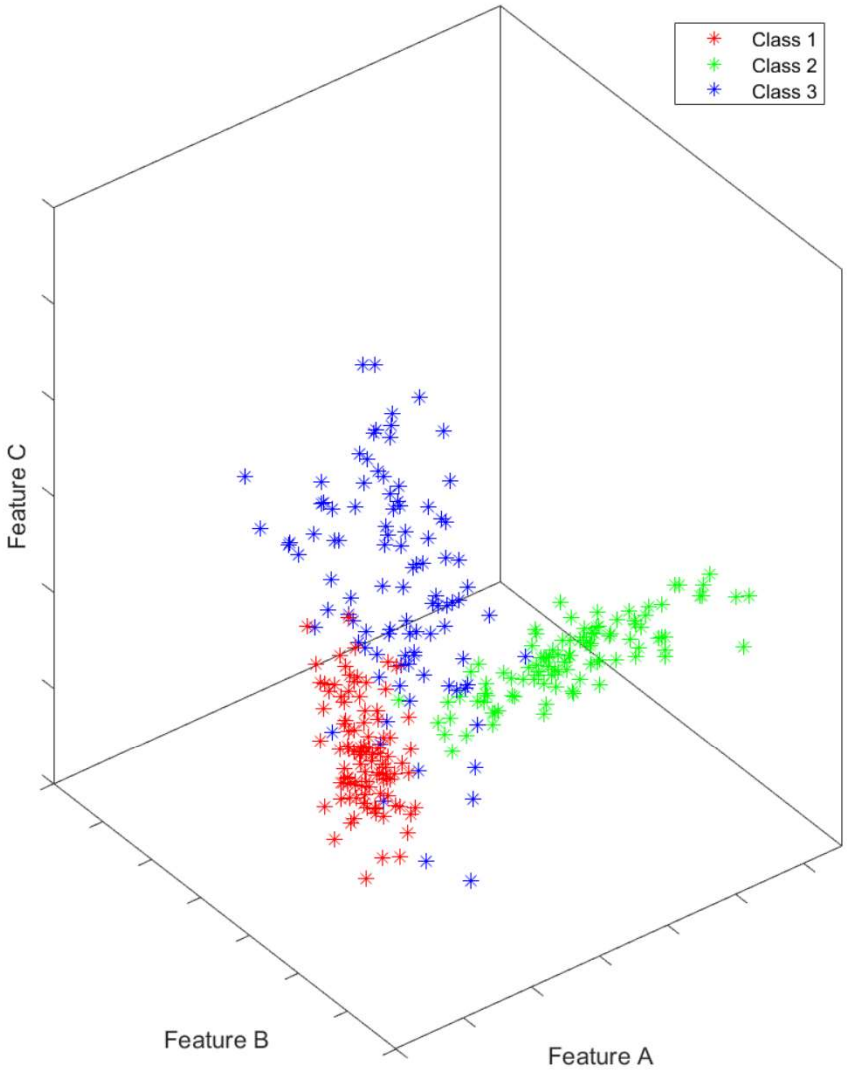
Data Set: Labeled Data





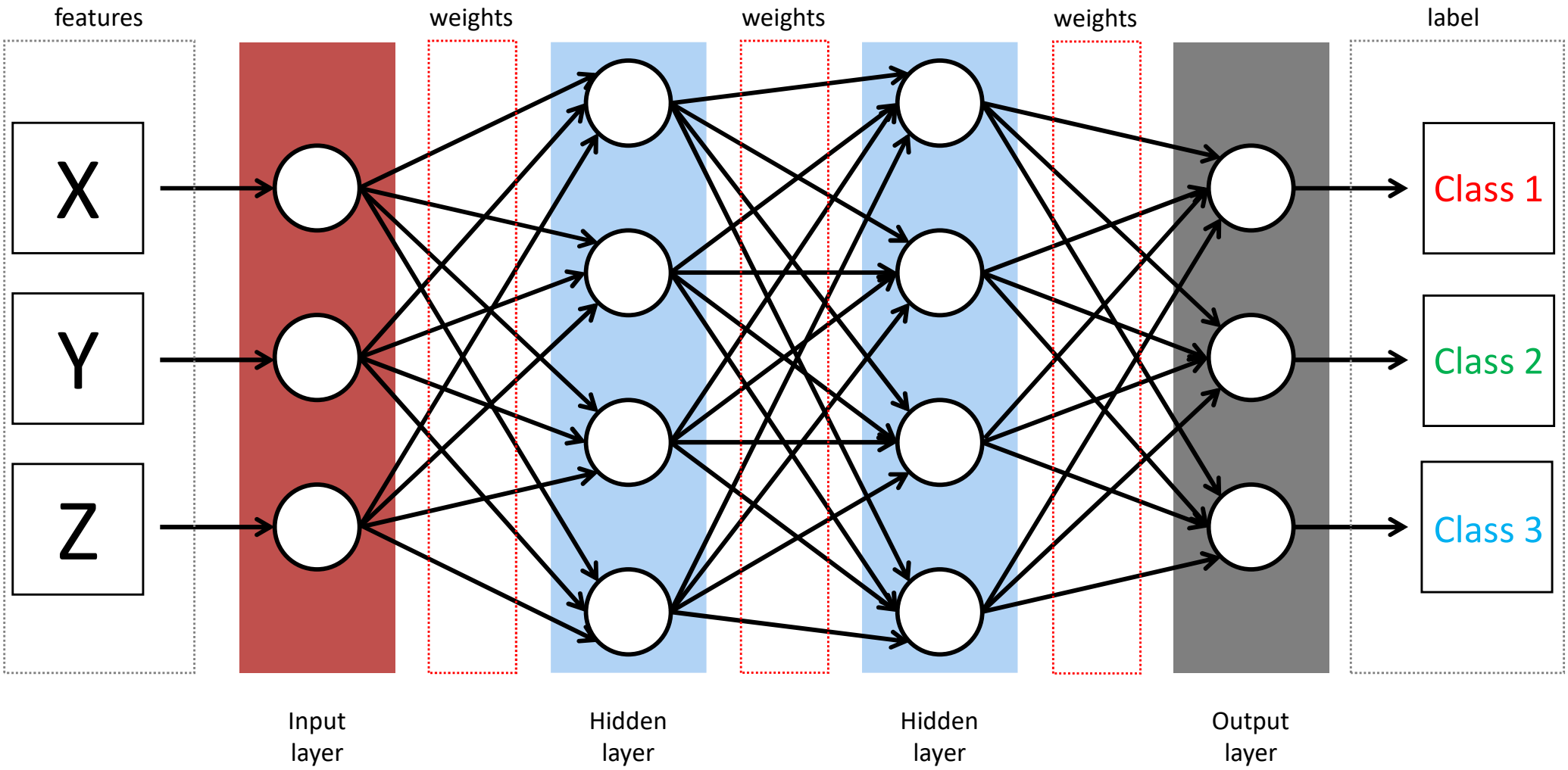
Data Set: Labeled Data





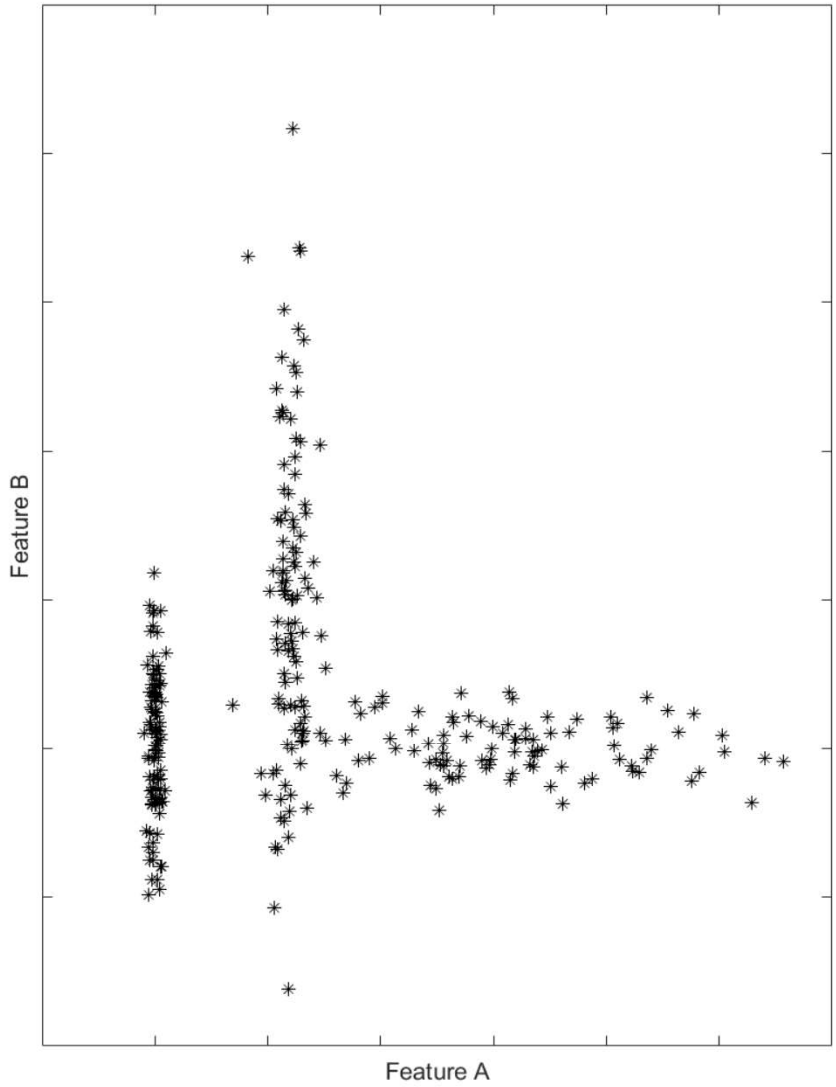
ANN: Supervised Learning

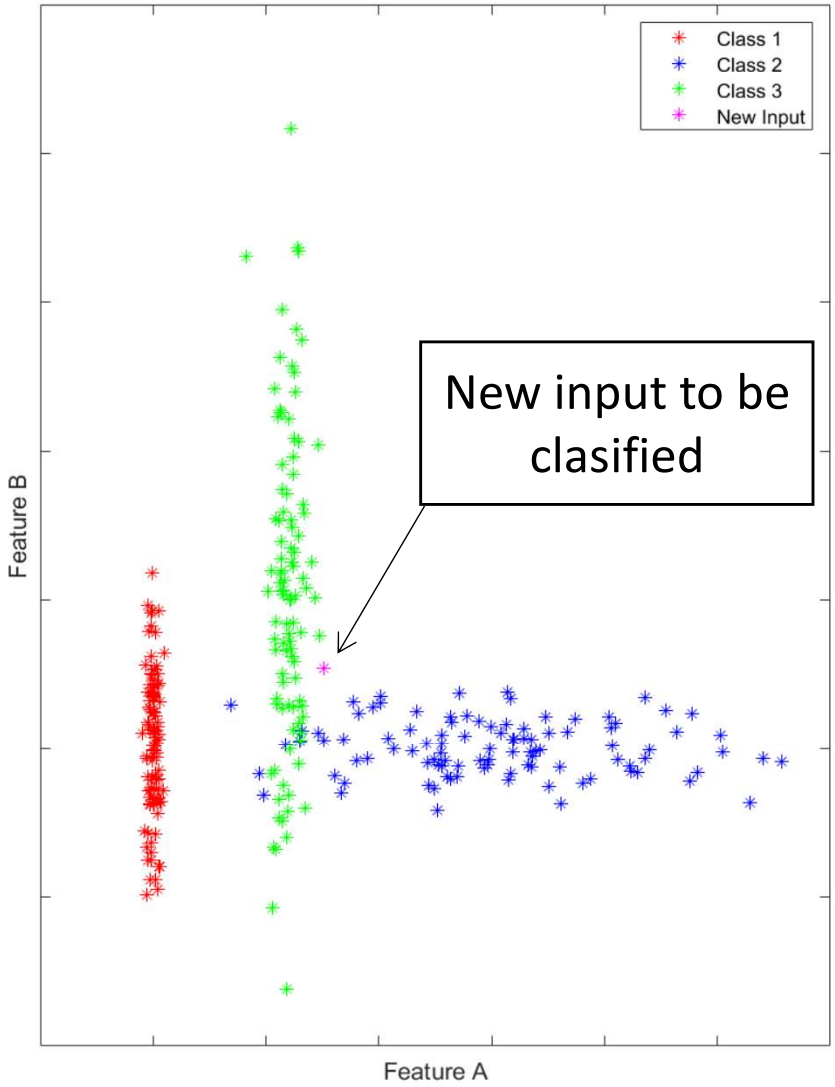
In order to work properly a classifier **needs to be trained** first with **labeled data**.



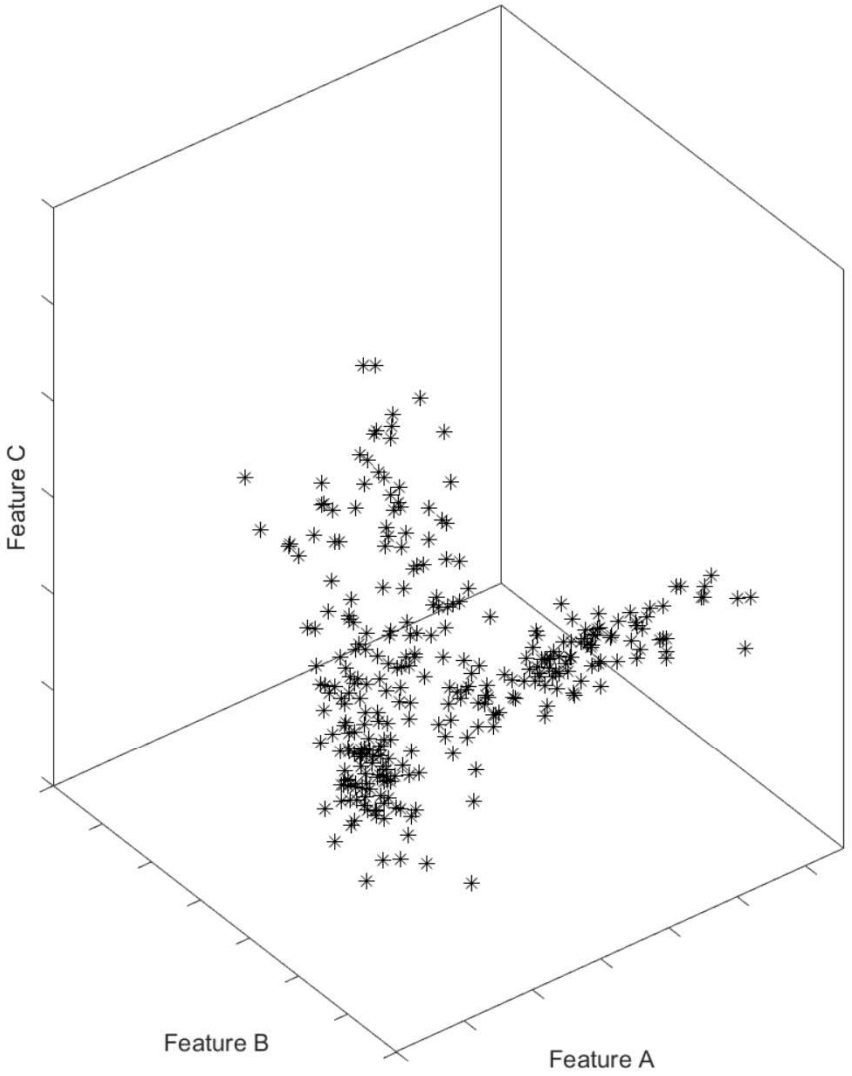
Training will **adjust all the weights** within this artificial neural network.

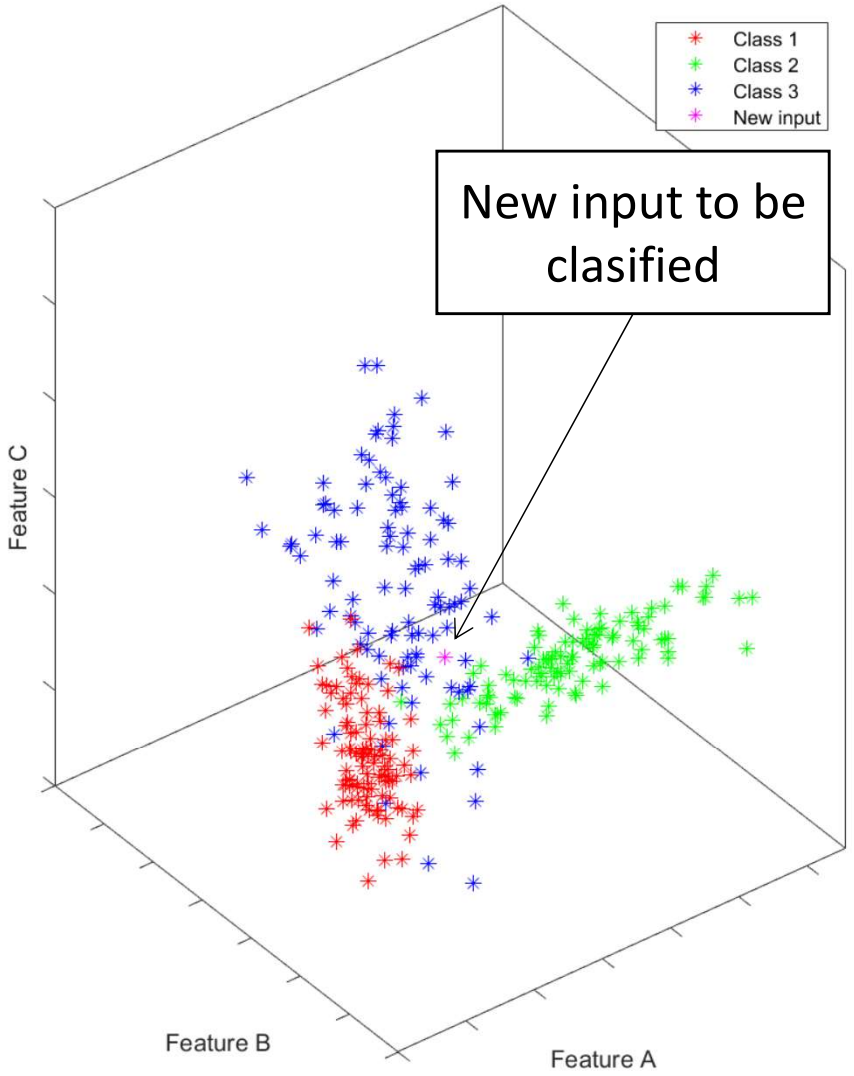
Supervised Learning: New Input



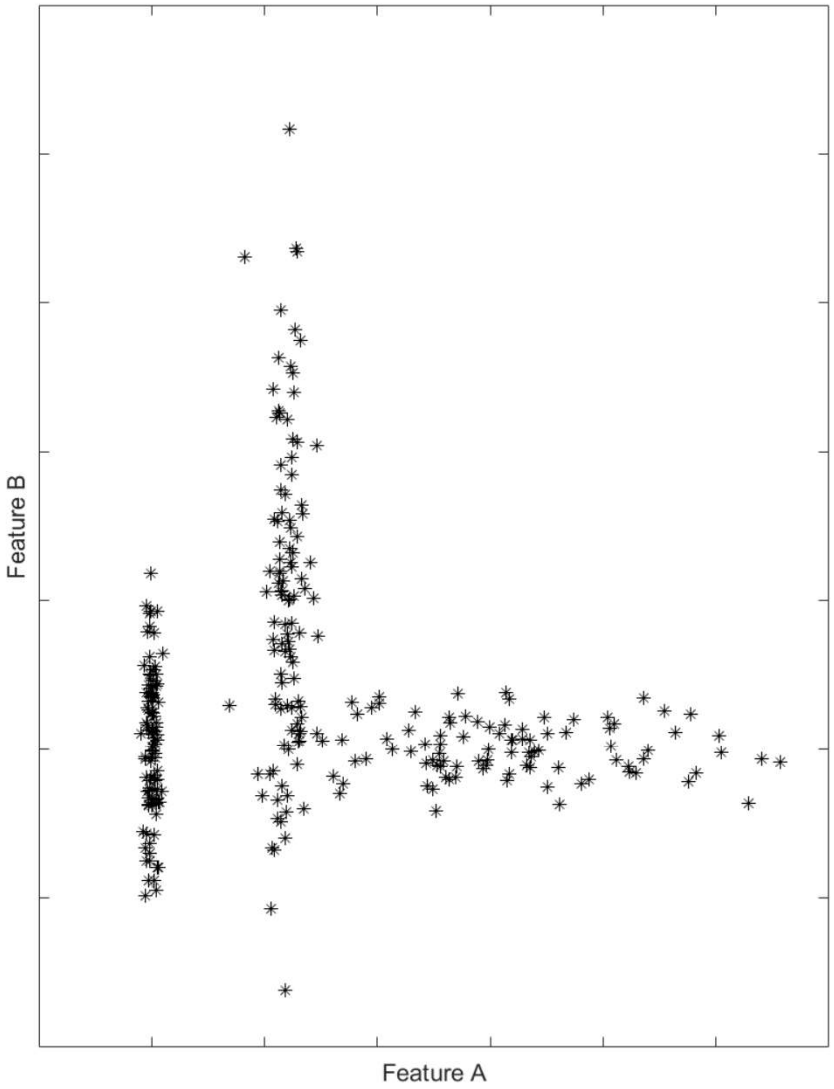


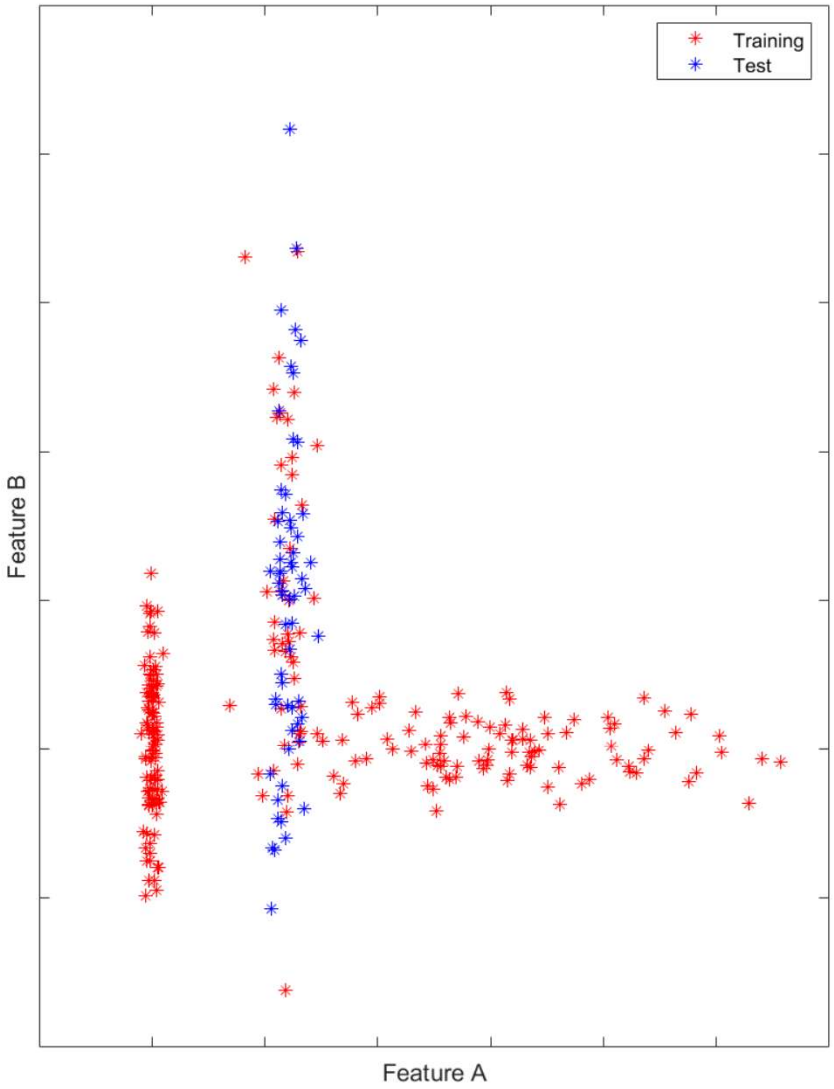
Supervised Learning: New Input



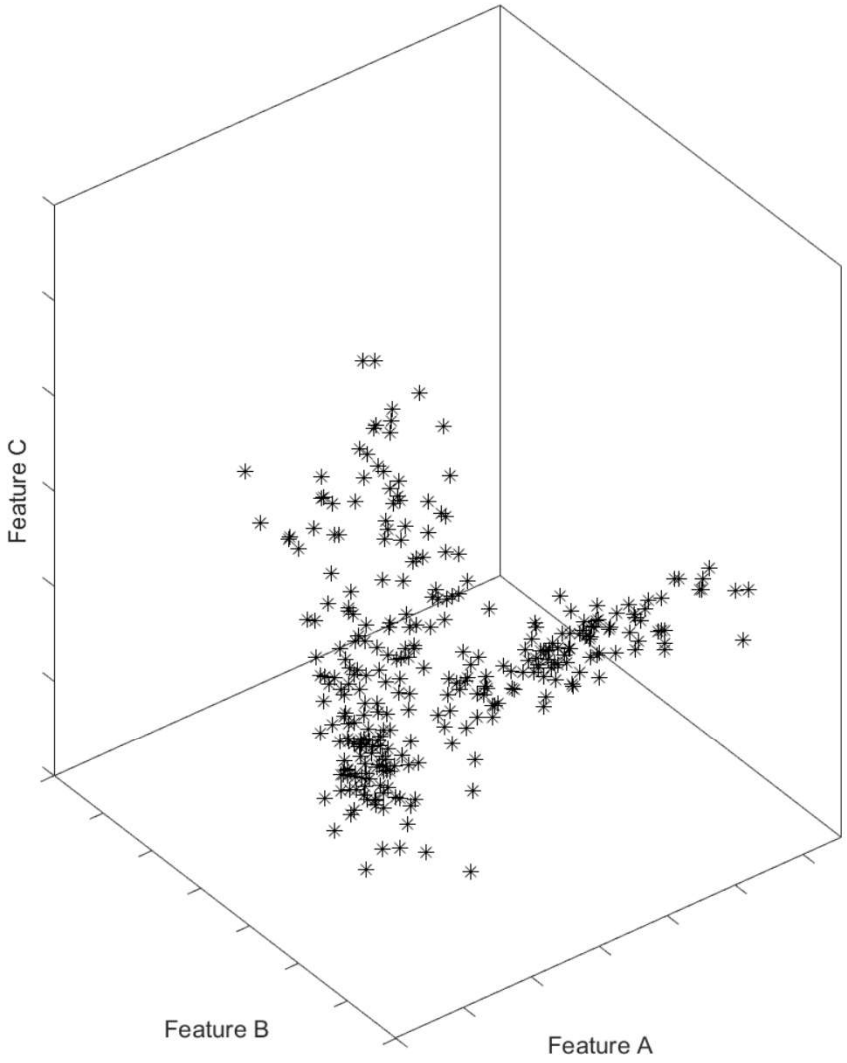


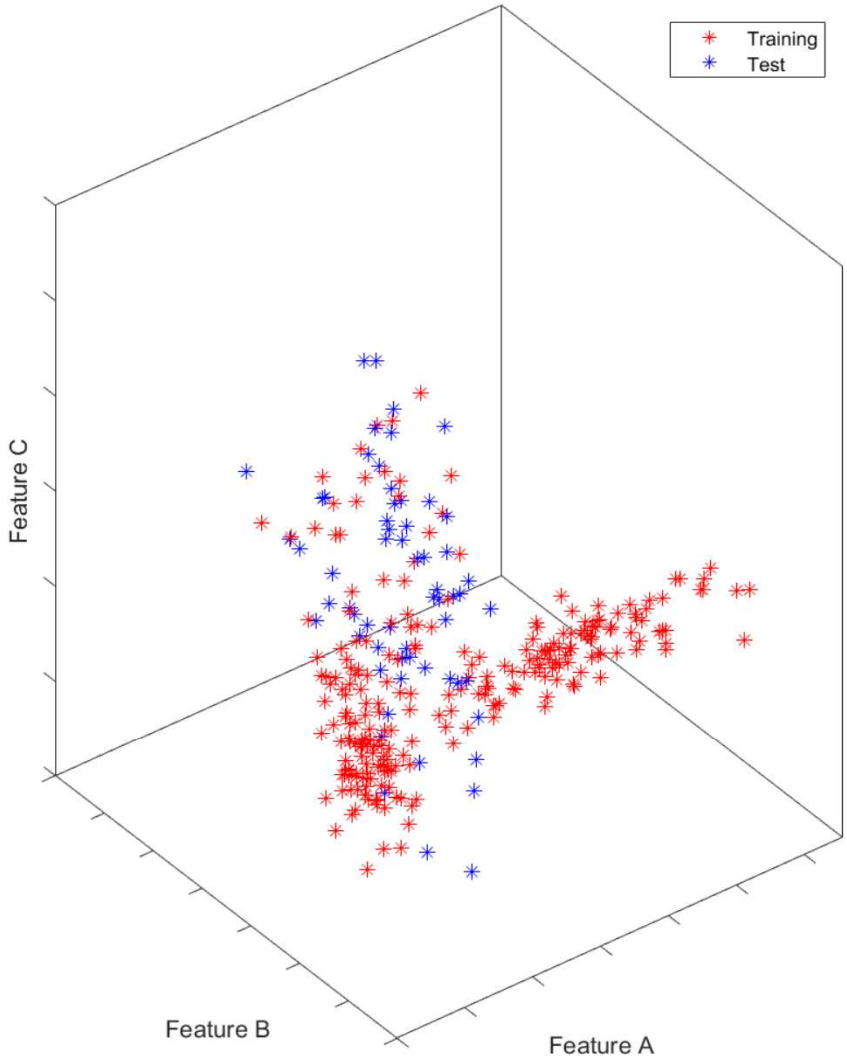
Training / Test Sets: Poor Split



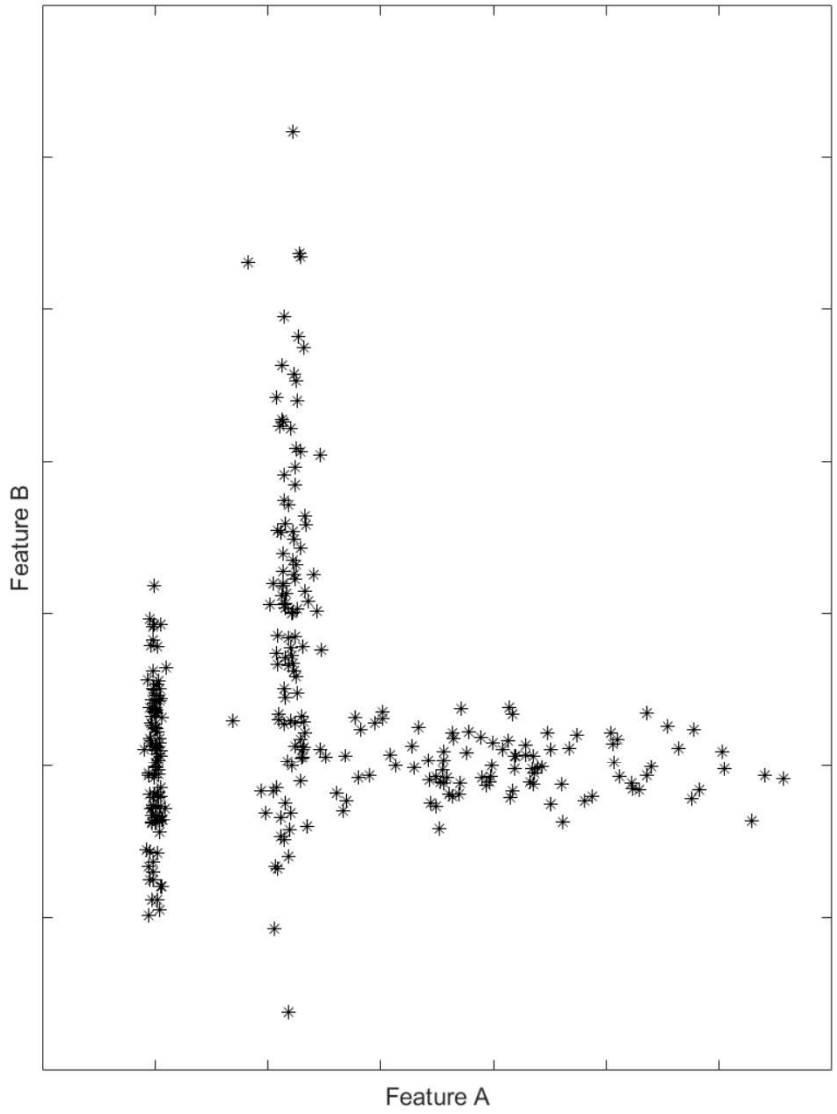


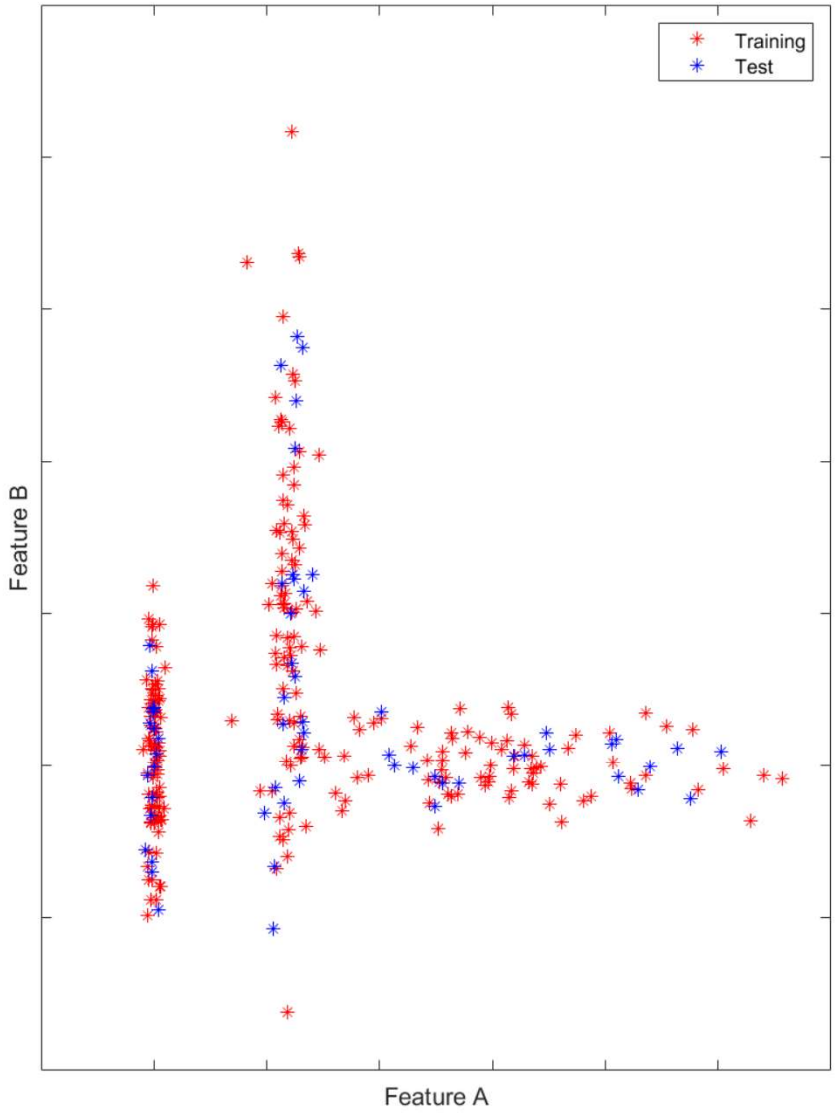
Training / Test Sets: Poor Split



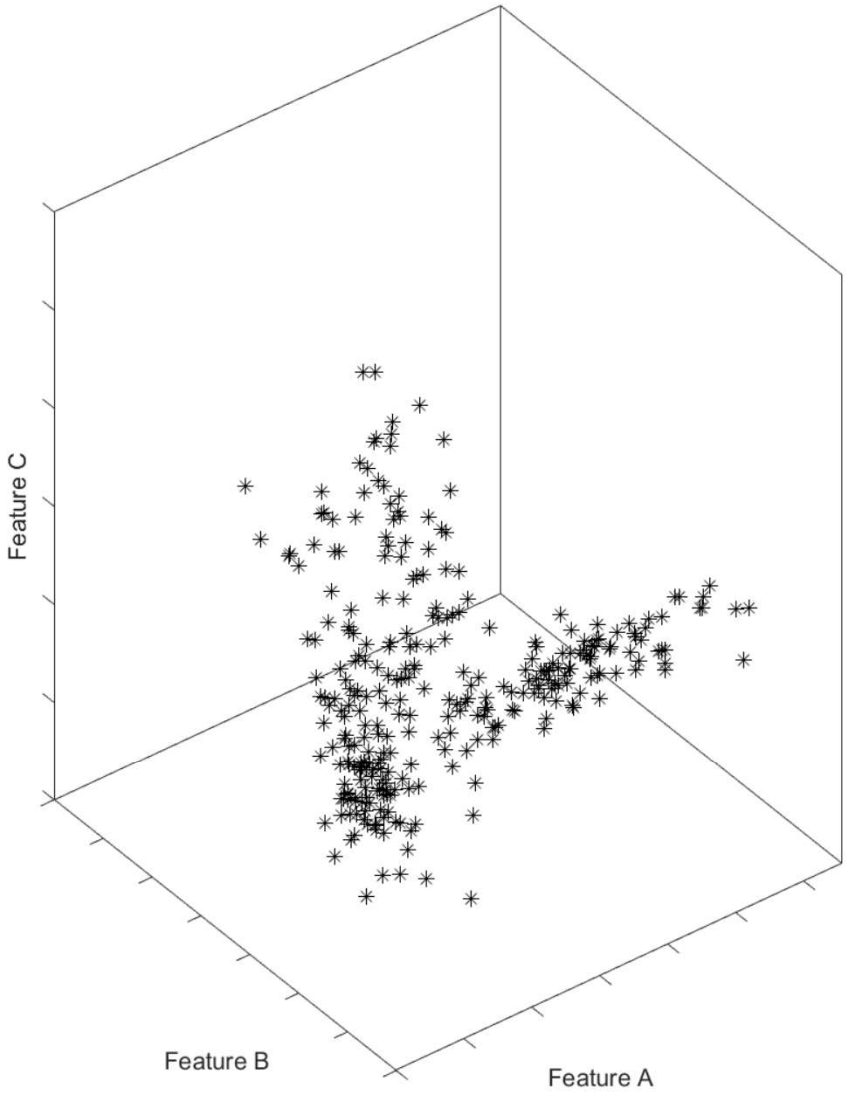


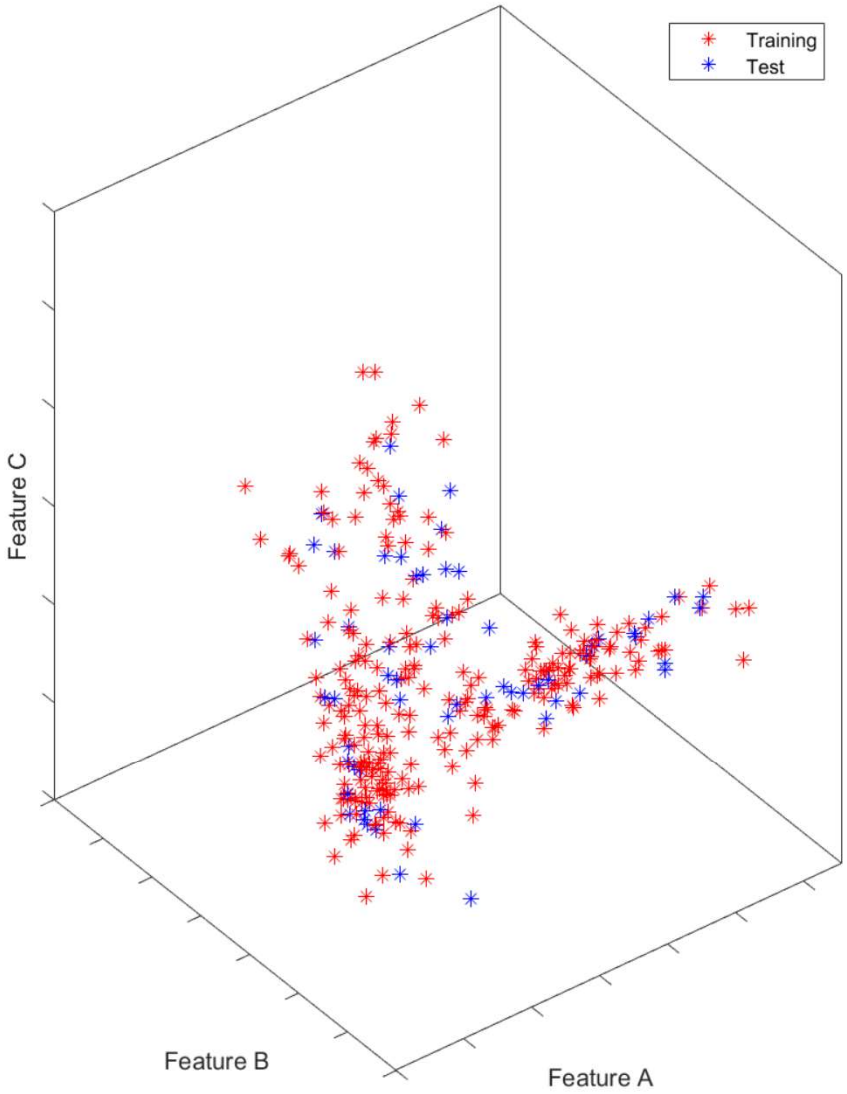
Training / Test Sets: Better Split



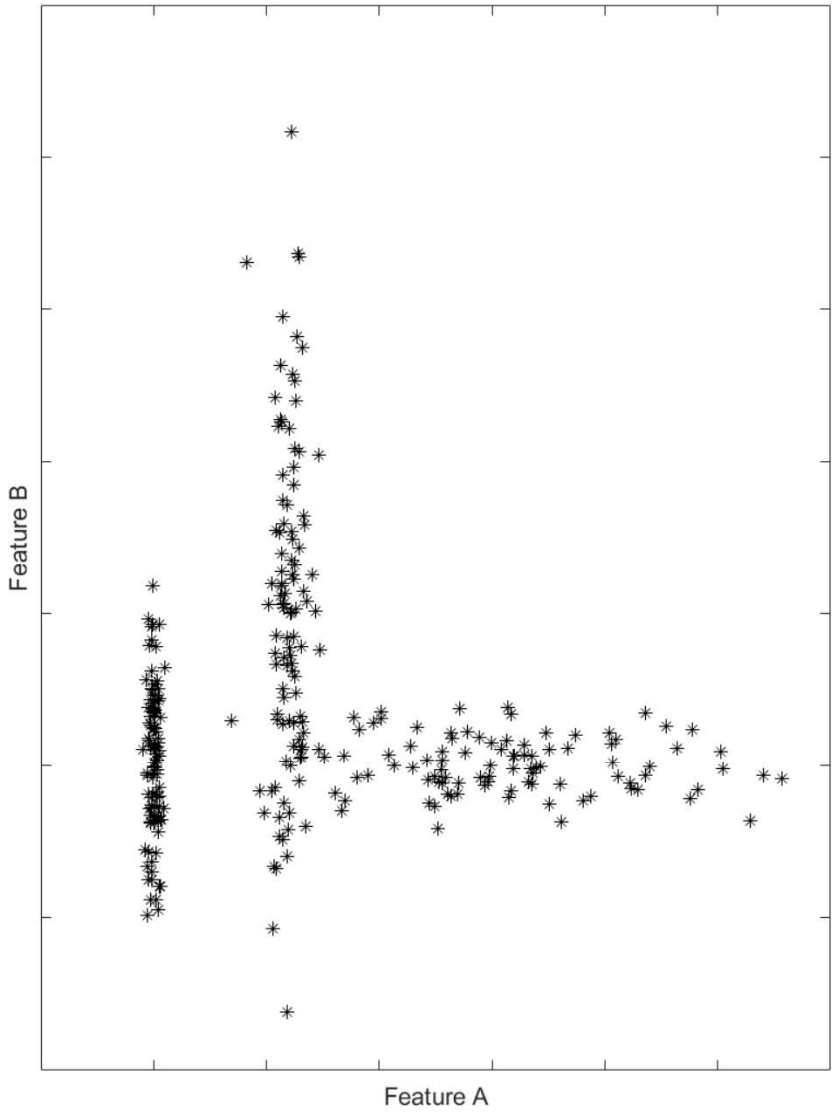


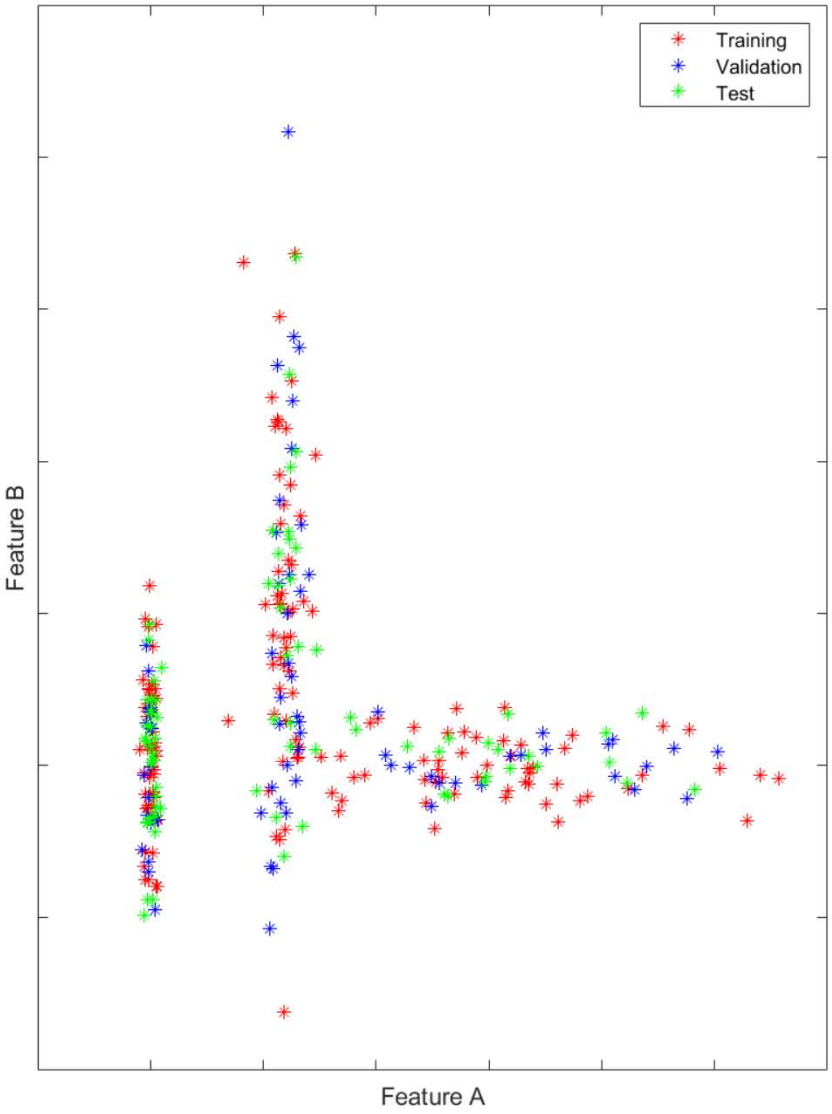
Training / Test Sets: Better Split



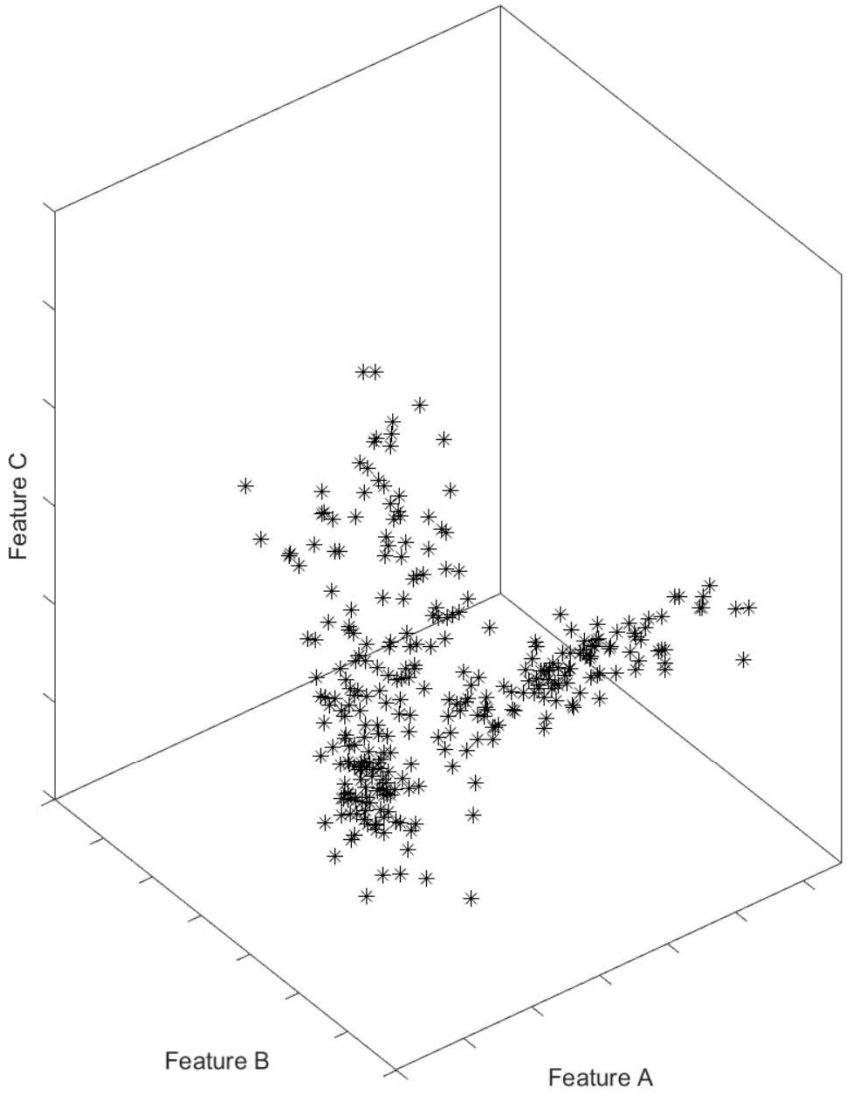


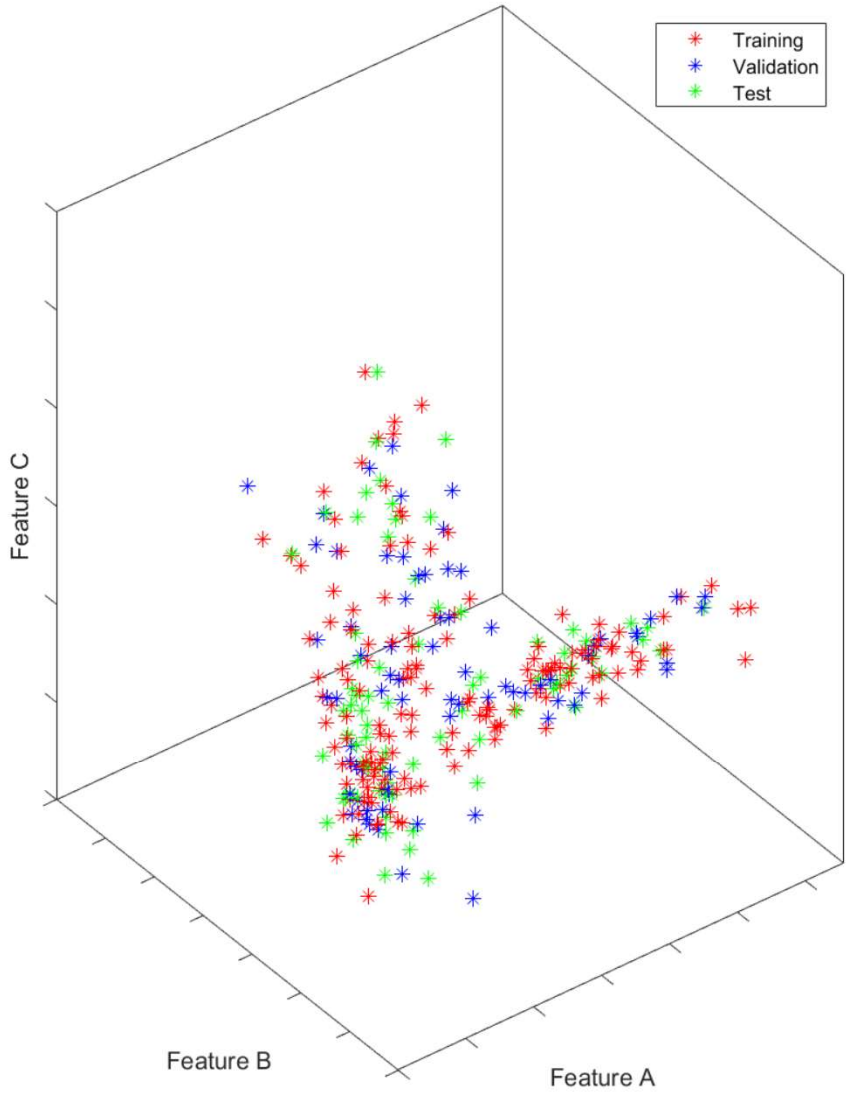
Training / Validation / Test Sets



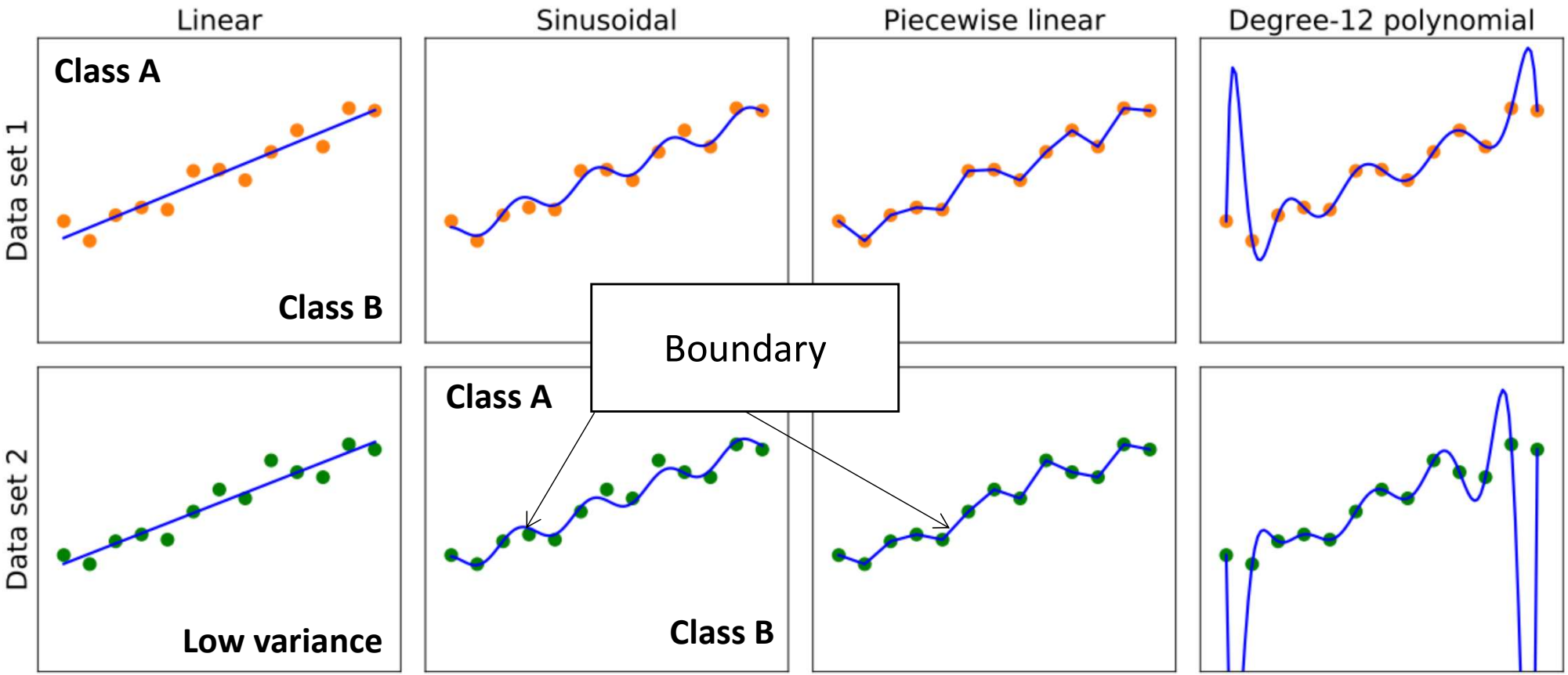


Training / Validation / Test Sets





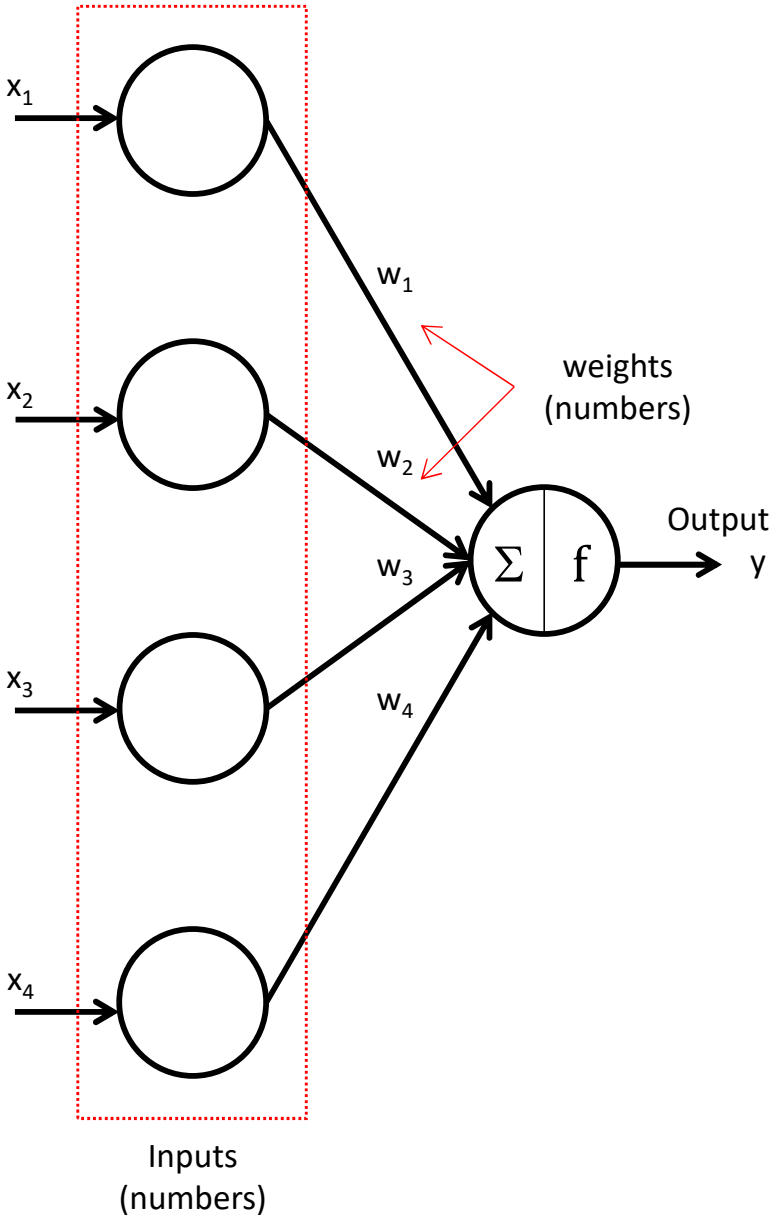
Hypothesis: Decision "Boundary"



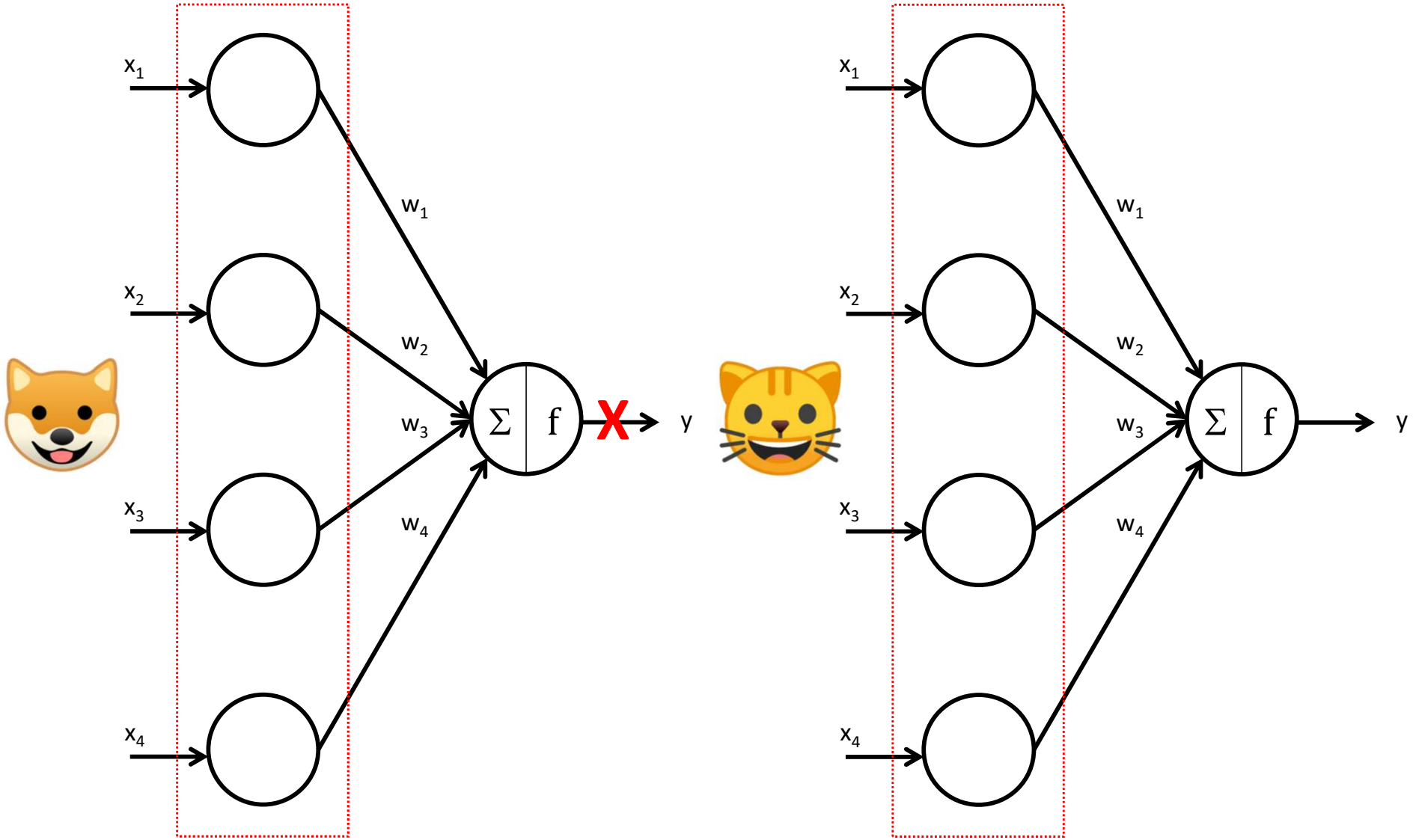
Artificial Neuron (Perceptron)

A (**single-layer**) **perceptron** is a model of a biological neuron. It is made of the following components:

- inputs x_i - numerical values representing information
- weights w_i - numerical values representing how "important" corresponding input is
- weighted sum: $\Sigma\ w_i * x_i$
- activation function f that decides if the neuron "fires"



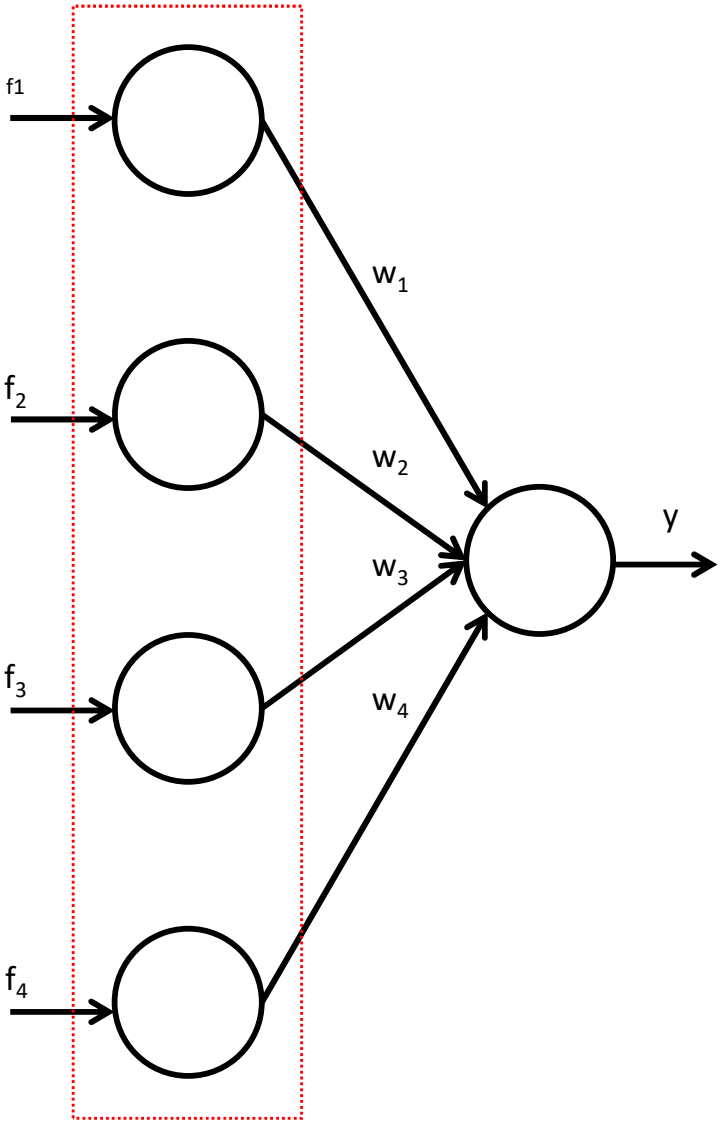
Single-layer Perceptron as a Classifier

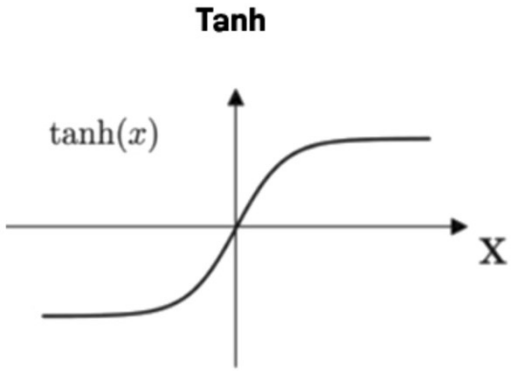


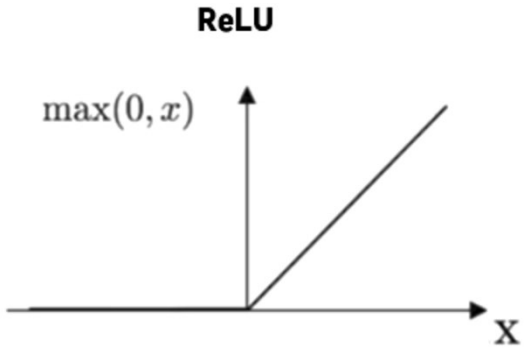
$\Sigma\, w_i * x_i < 0 \rightarrow f = 0$ → **NOT CAT**

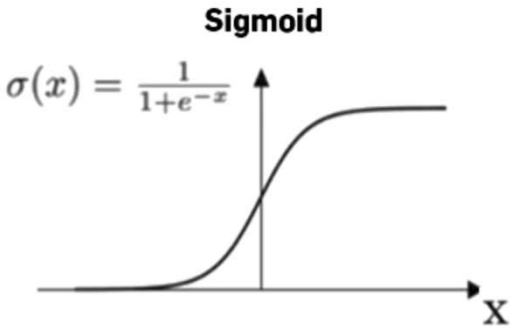
$\Sigma\, w_i * x_i \geq 0 \rightarrow f = 1$ → **CAT**

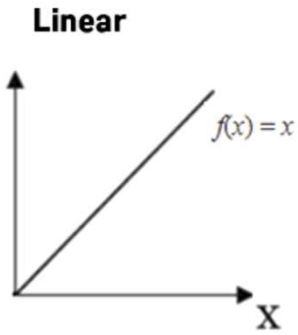
Selected Activation Functions





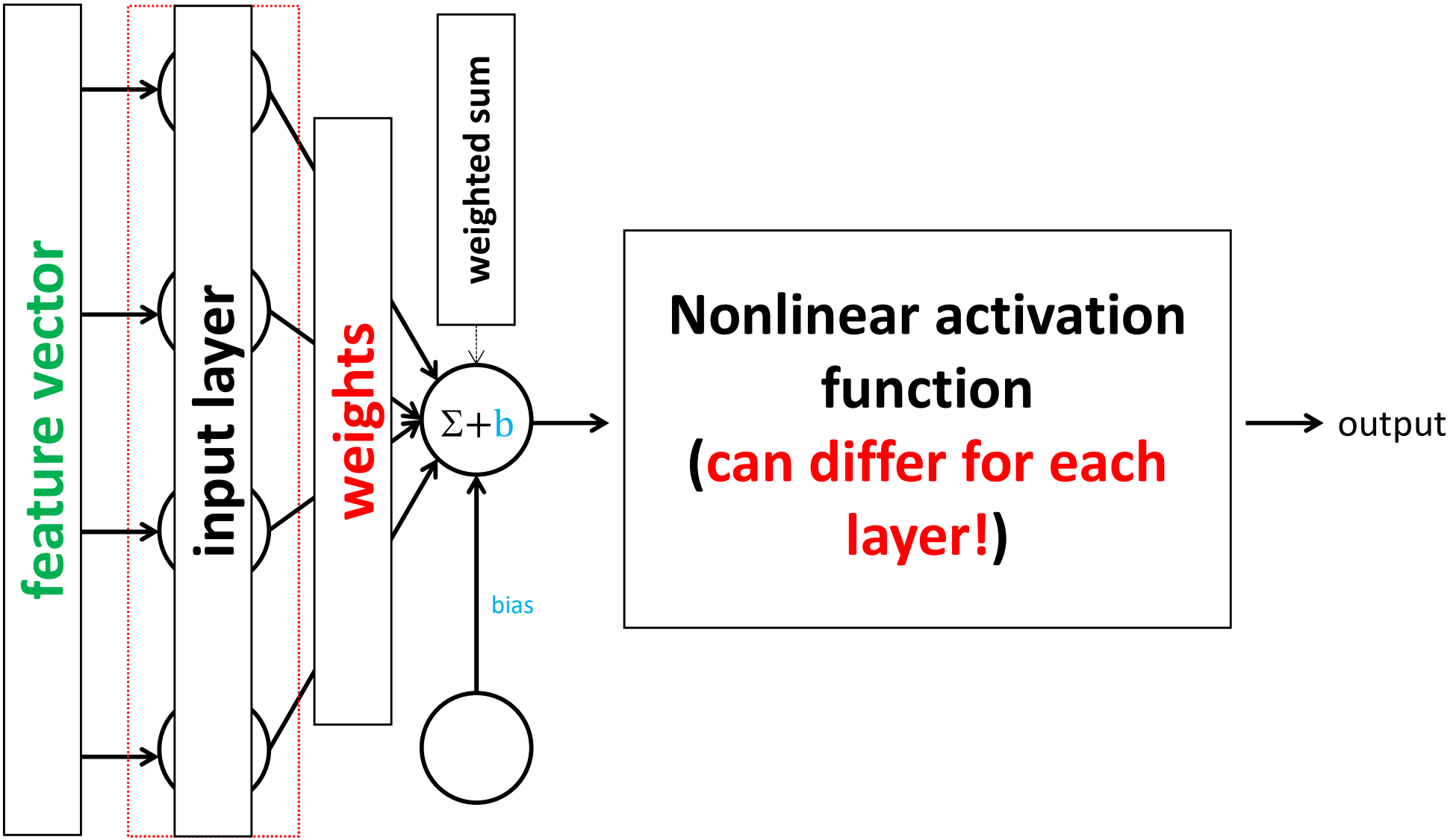




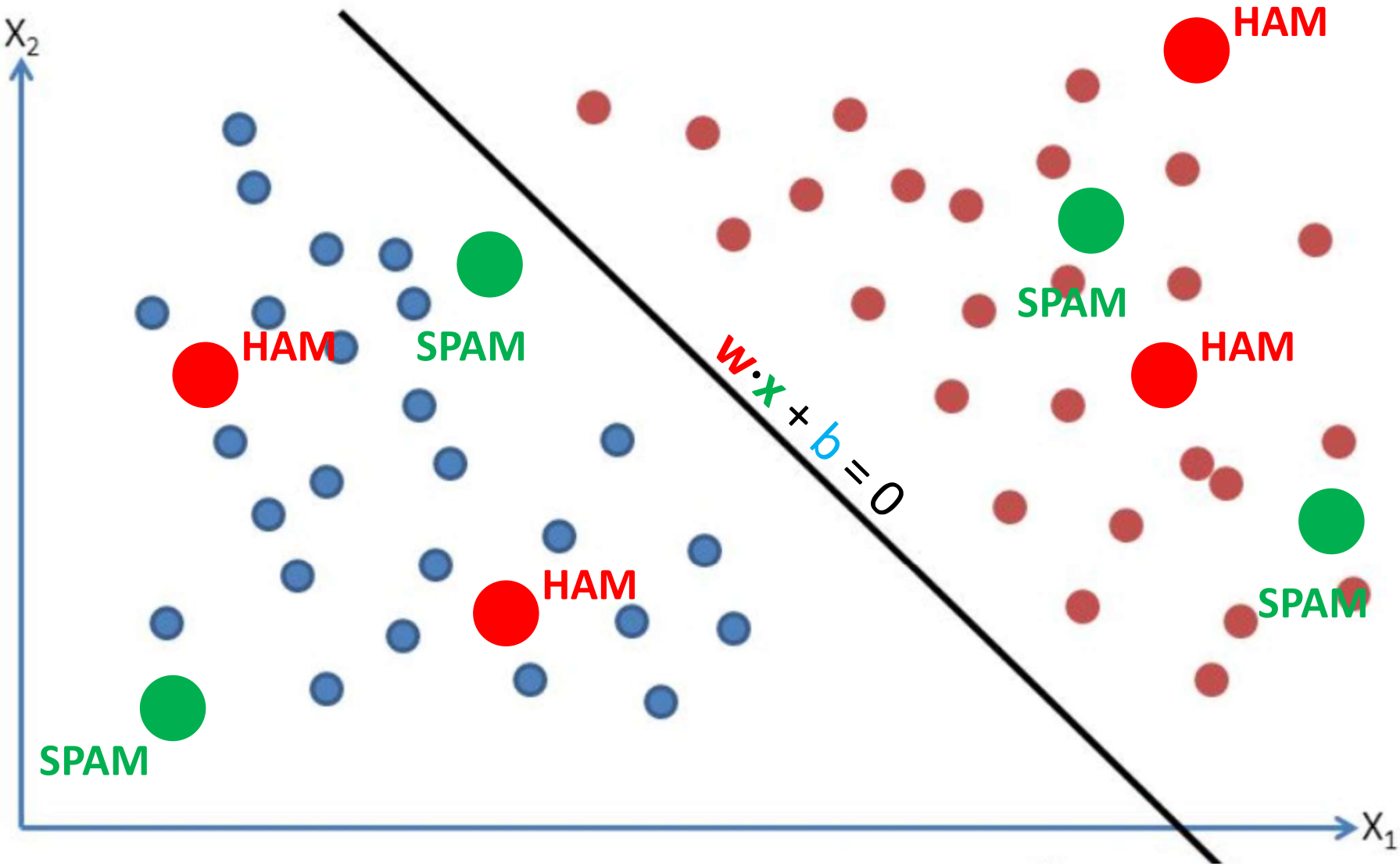


ReLU: Rectified Linear Unit

Basic Neural Unit

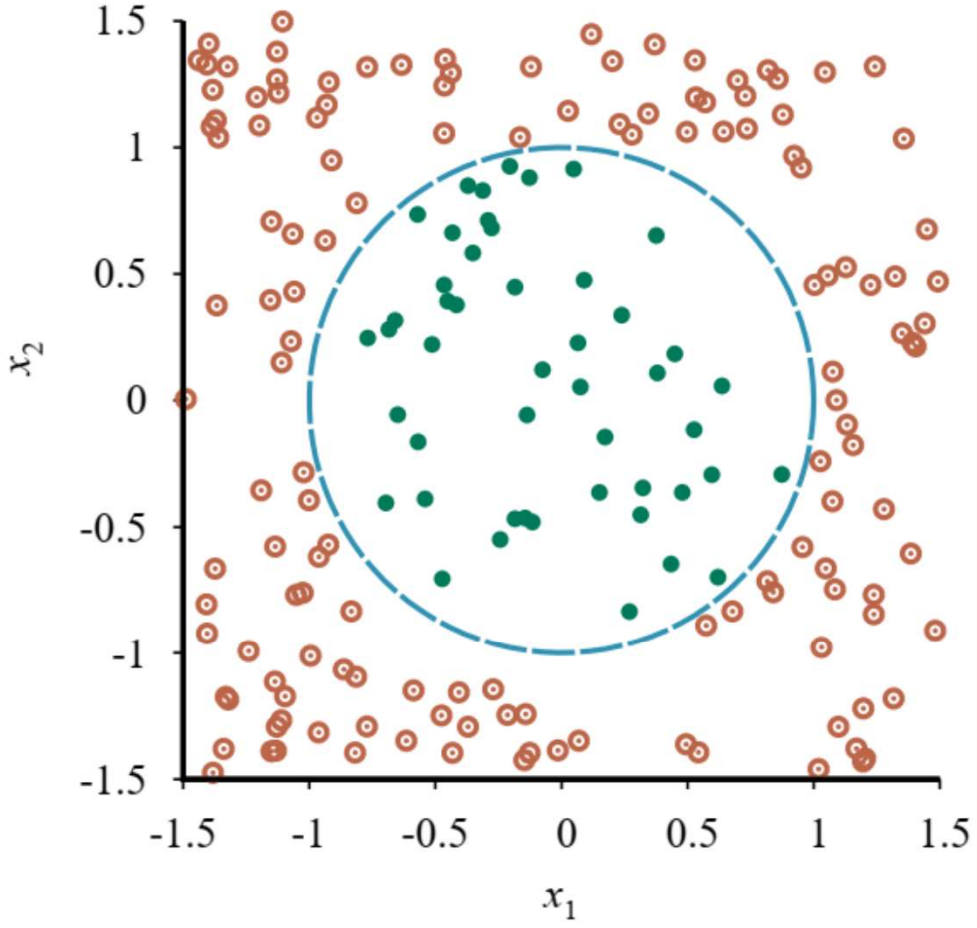


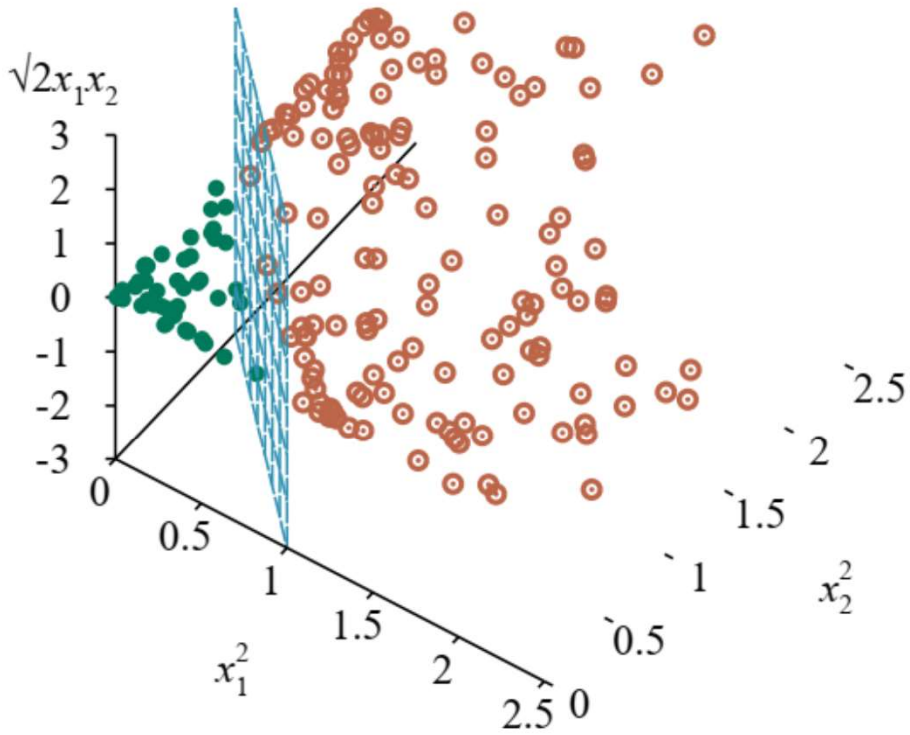
Classification: Linear Separation?



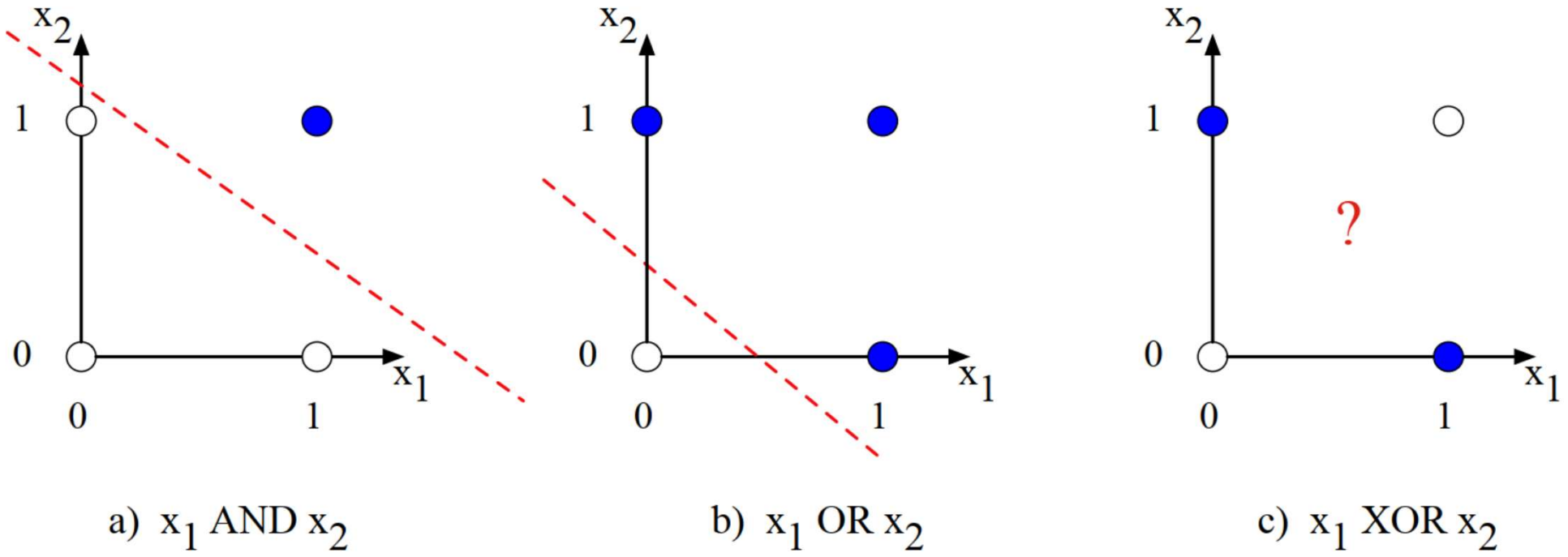
Sometimes decision boundary CANNOT be linear? Not linearly separable f()

Hypothesis: Classification "Boundary"



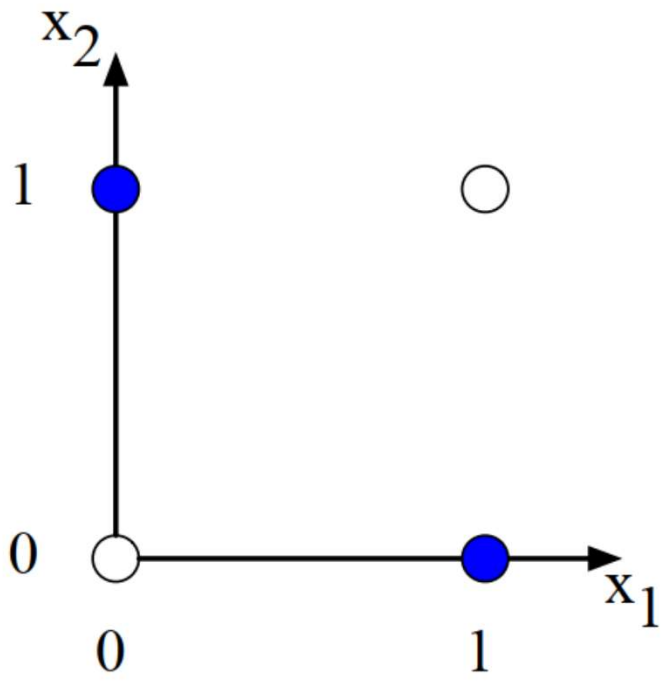


XOR: Not a Linearly Separable f()

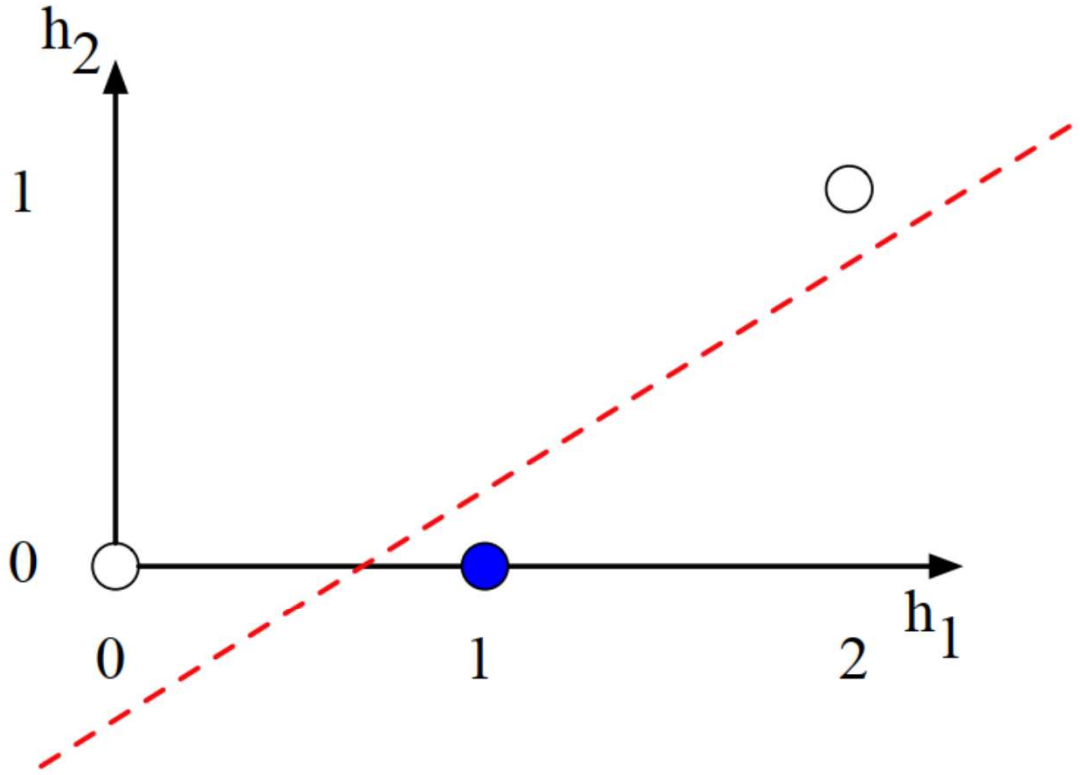


Logical XOR is an example of a function that is NOT linearly separable

XOR: Not a Linearly Separable f()?

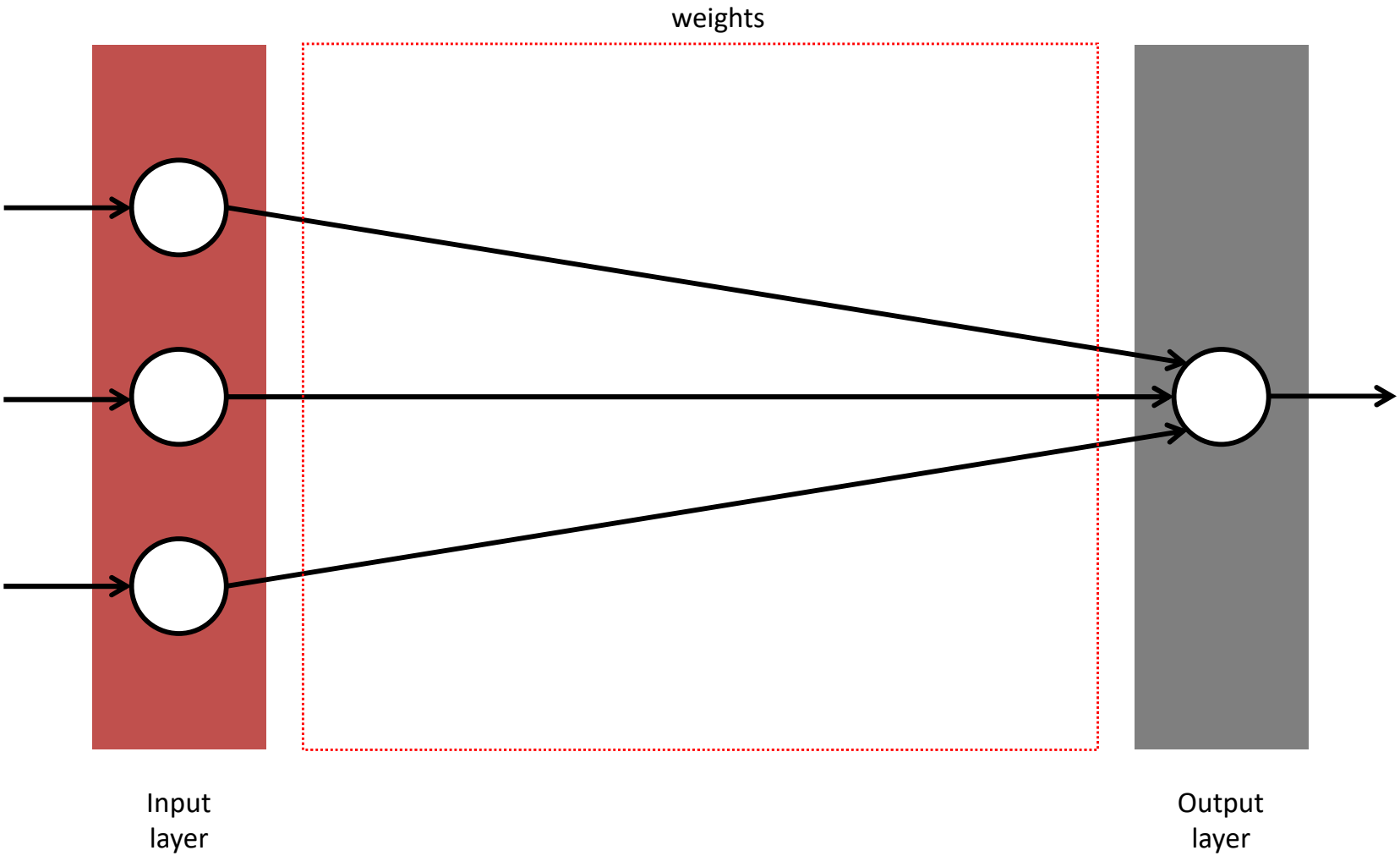


a) The original *x* space

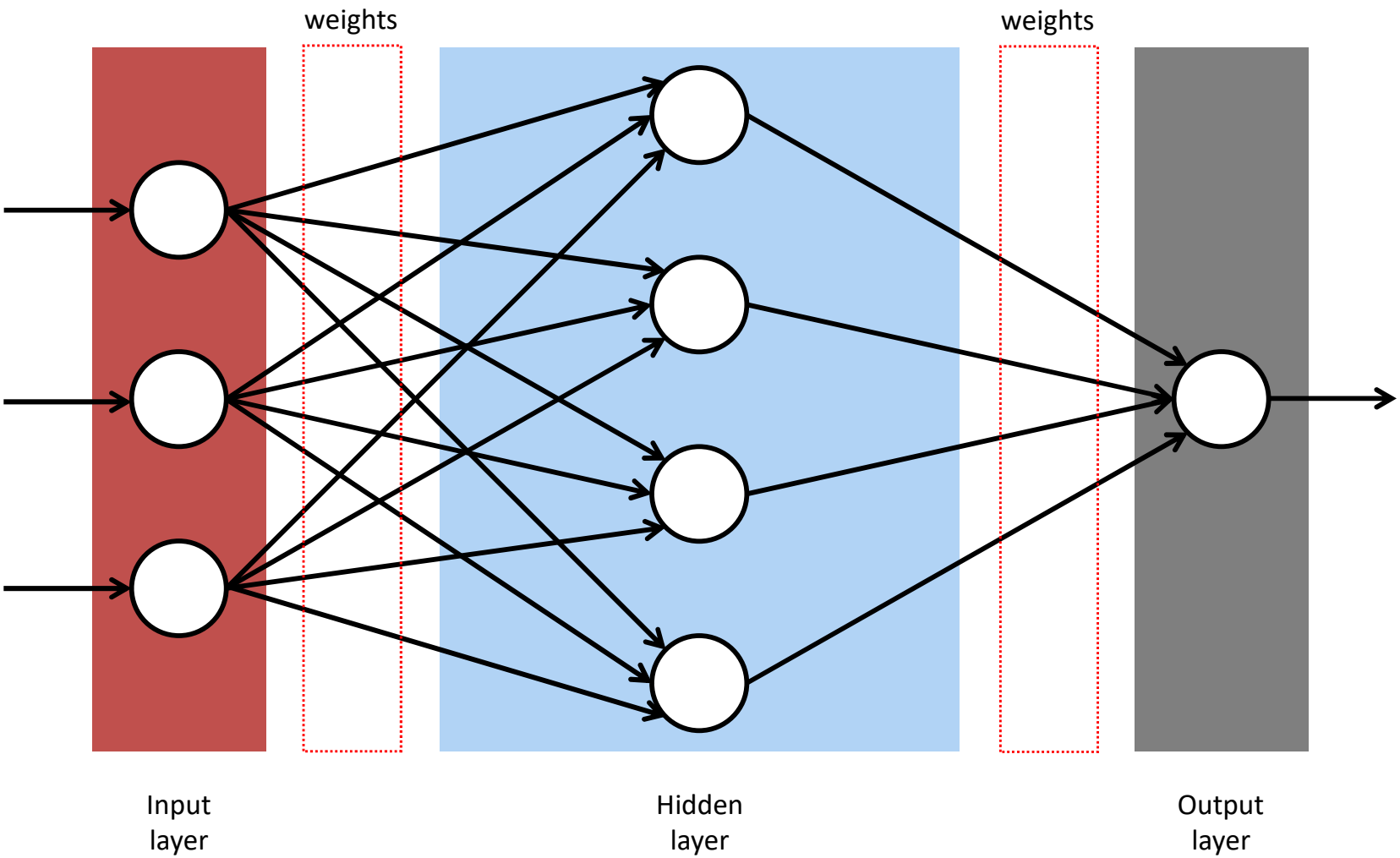


b) The new (linearly separable) *h* space

Basic Neural Unit

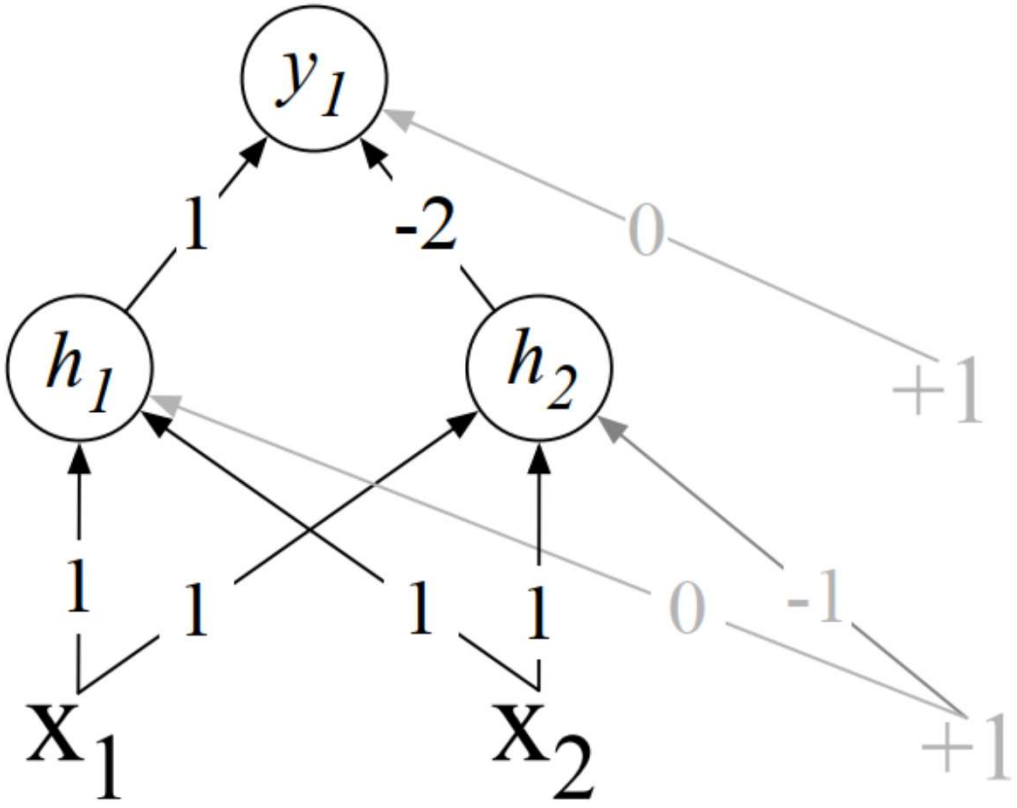


Hidden Layer

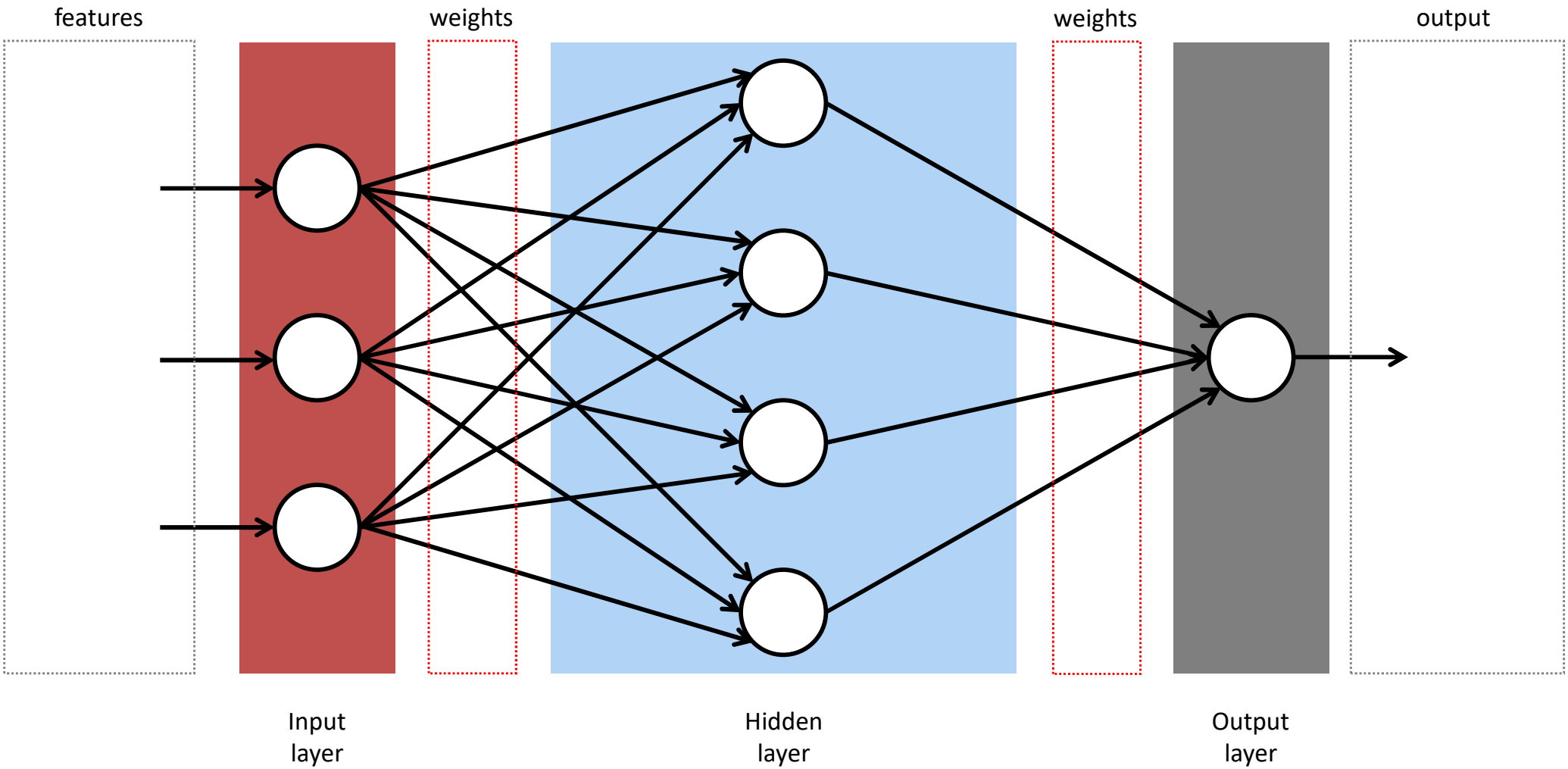


XOR: Hidden Layer Approach

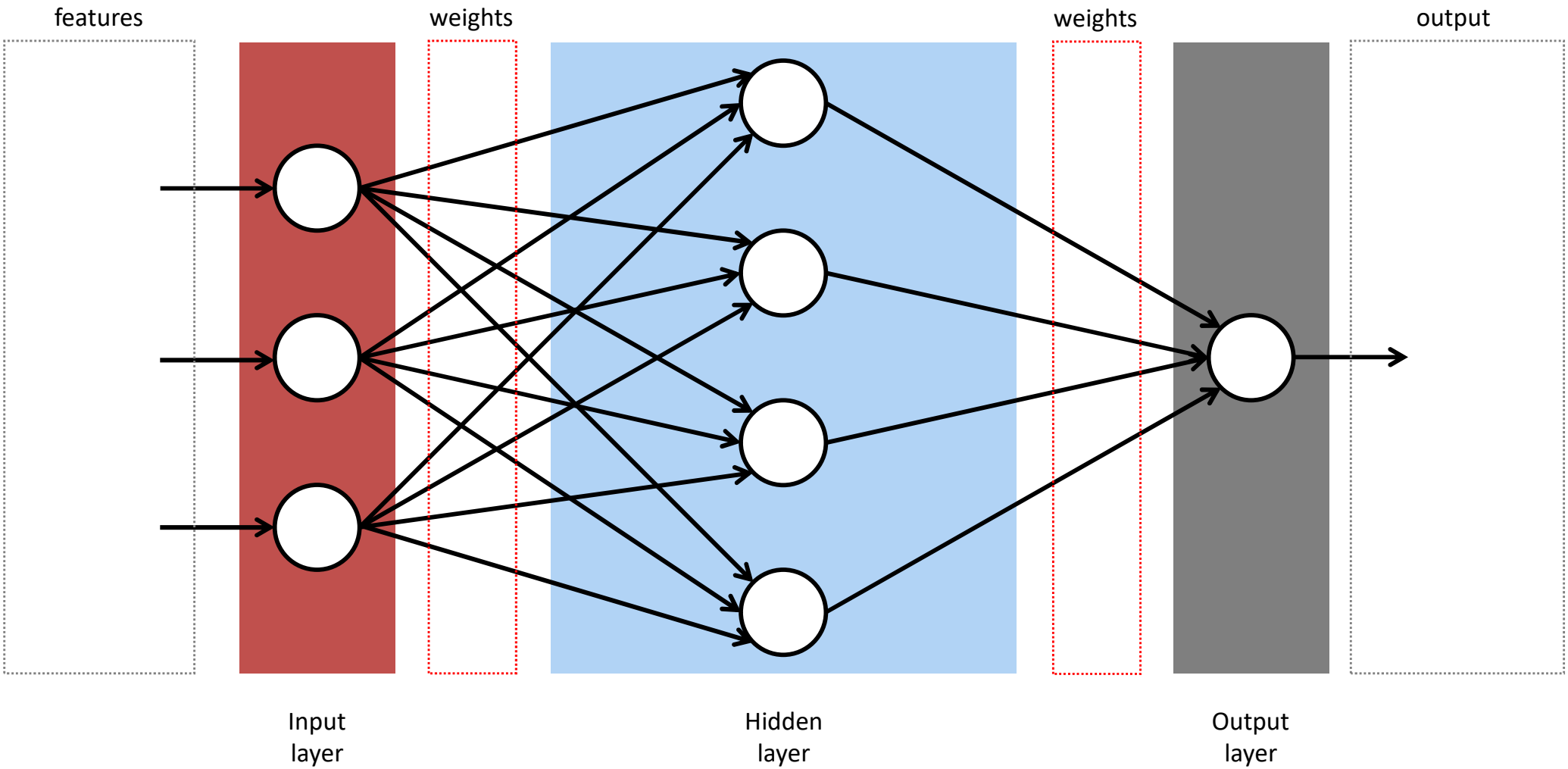
XOR		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



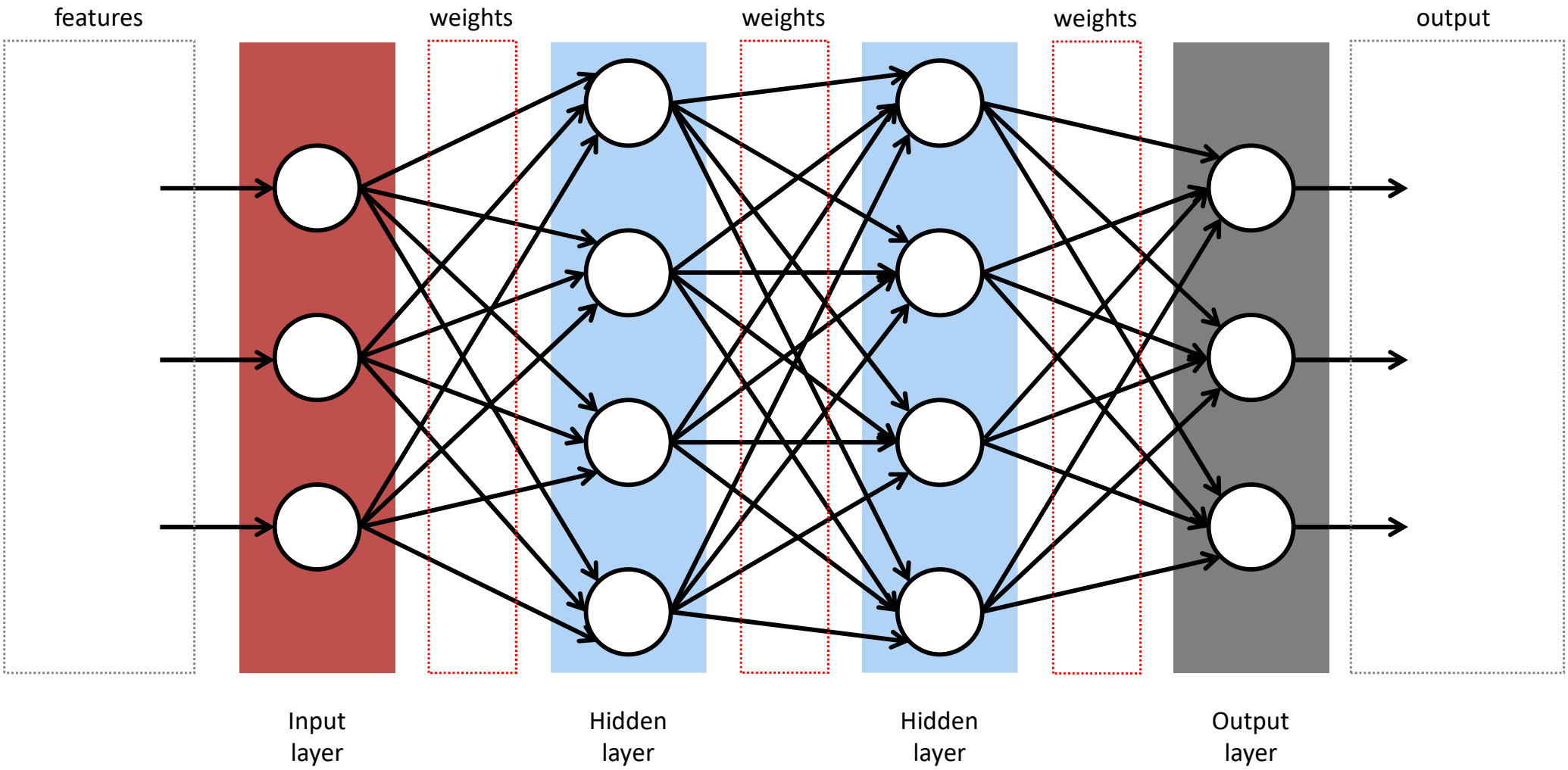
Hidden Layer



Hidden Layer



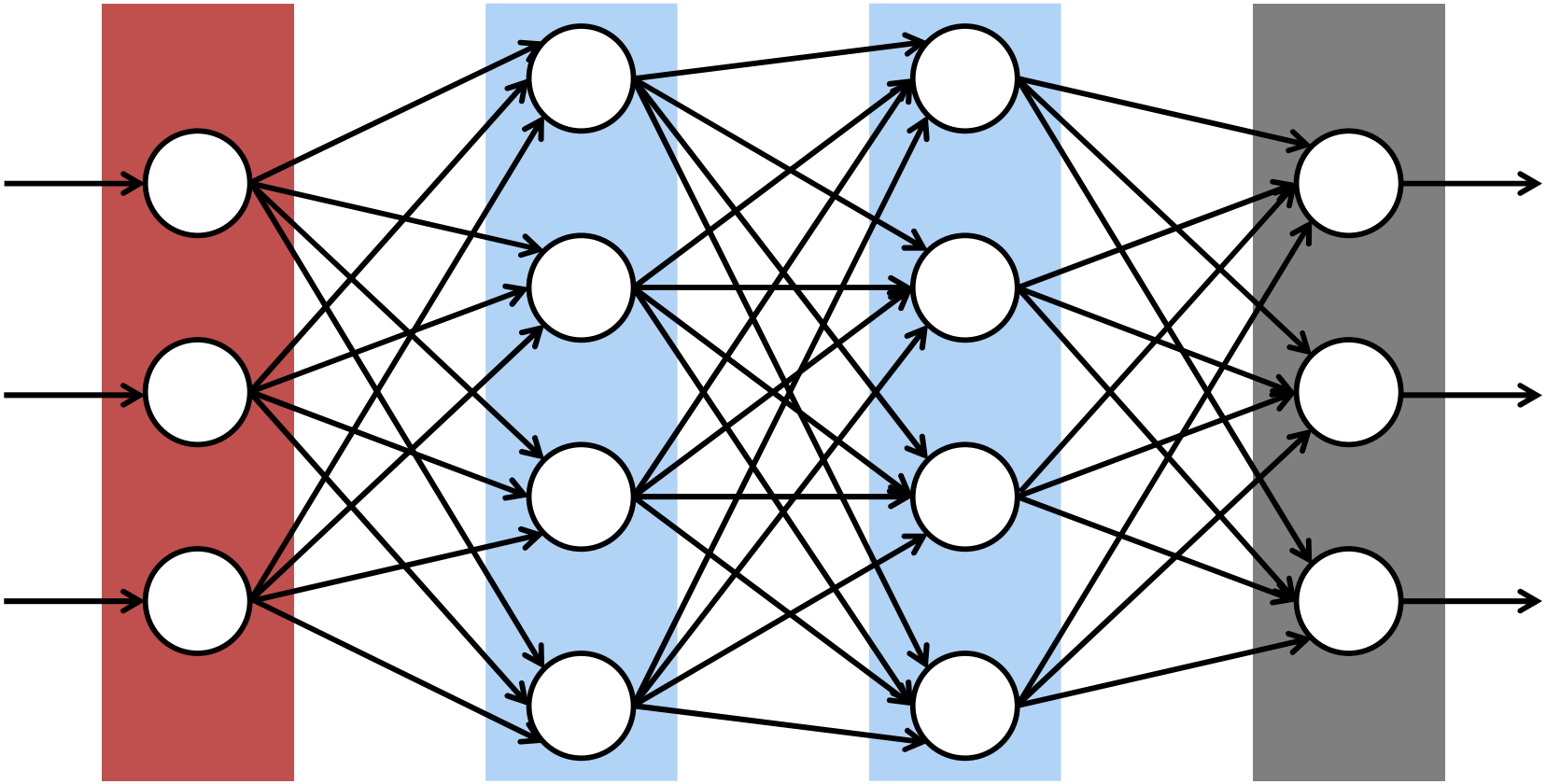
Feedforward Neural Network



Also called (historically): multi-layer perceptron

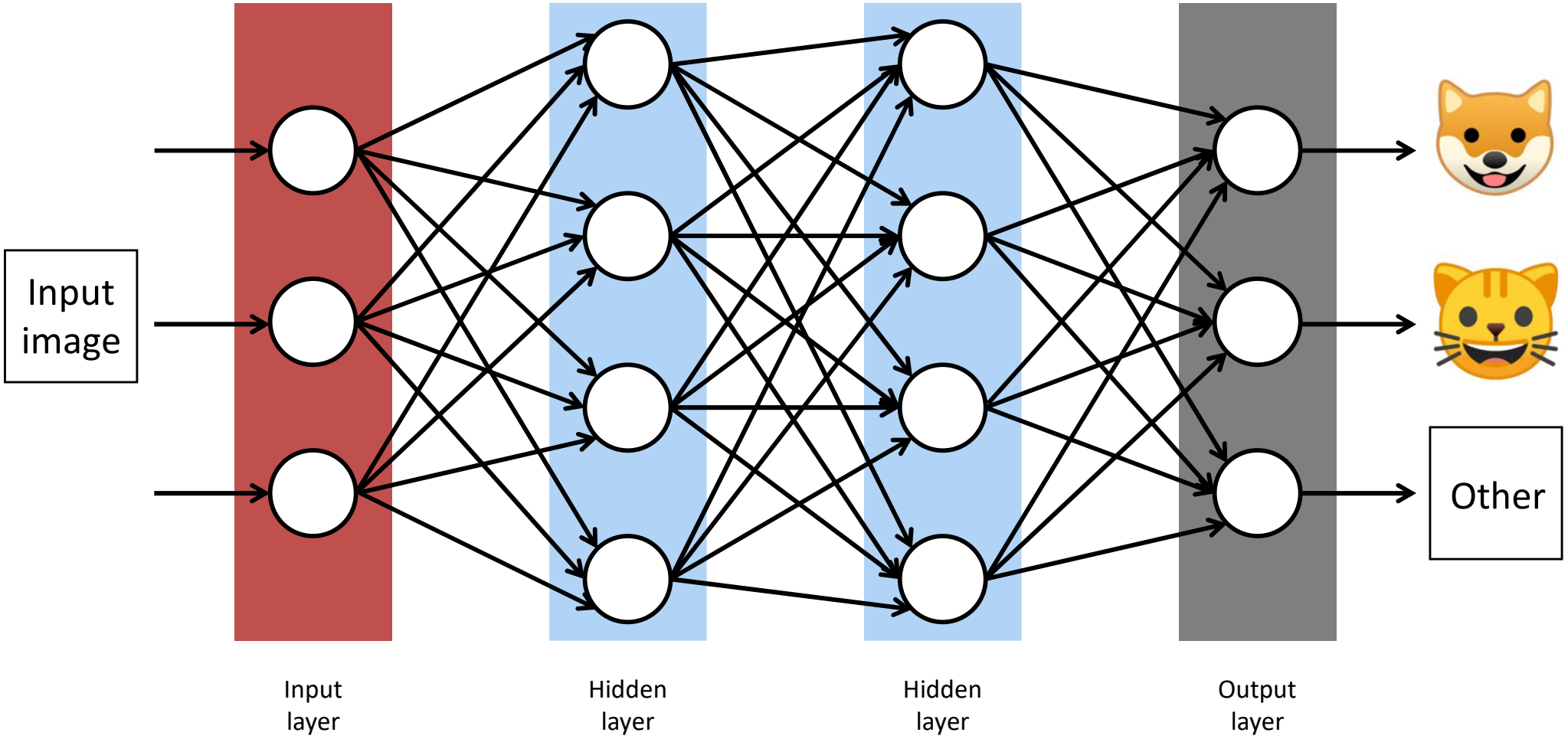
Artificial Neural Network (ANN)

An artificial neural network is made of **multiple artificial neuron layers**.

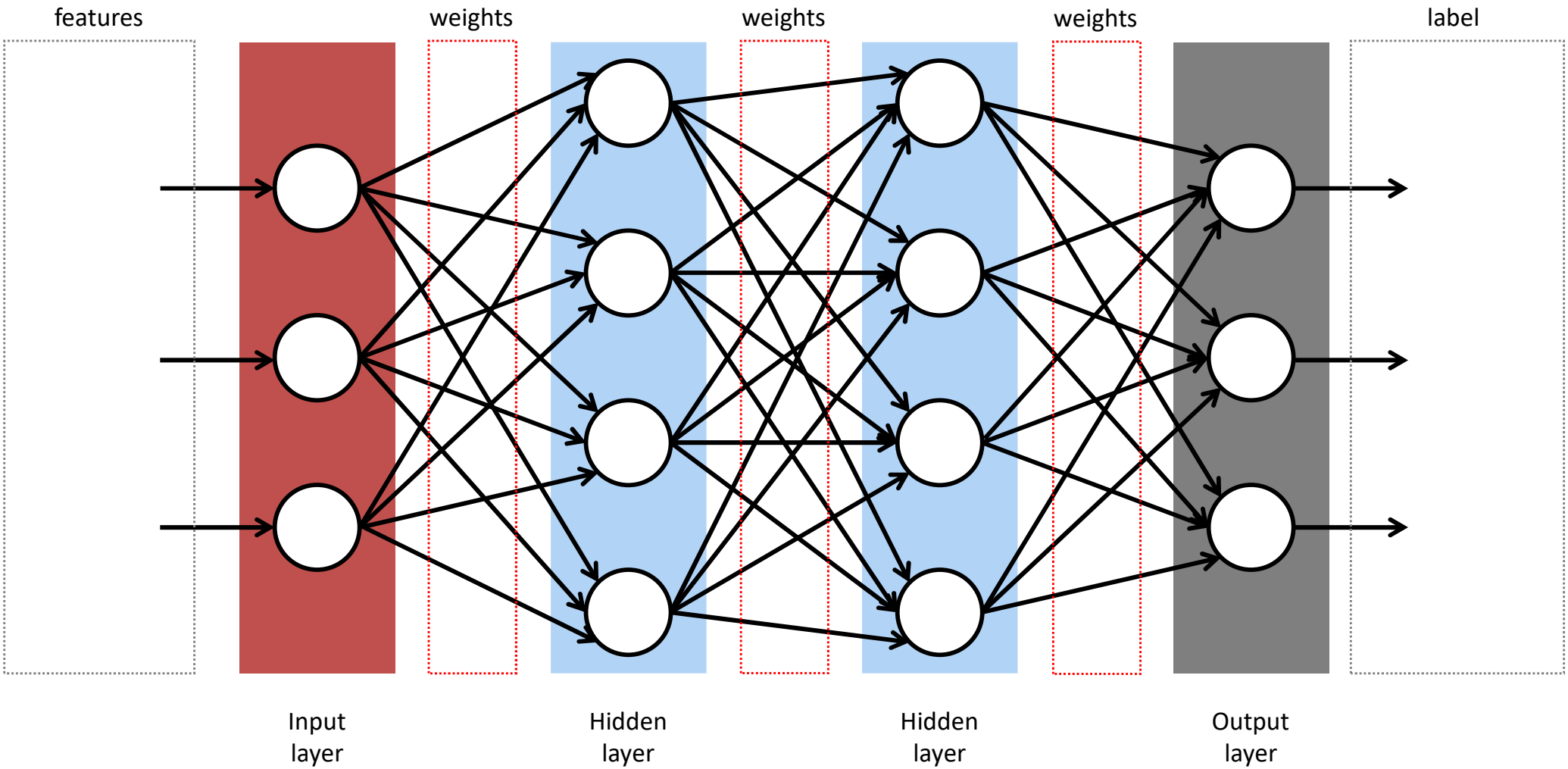


ANN as an Image Classifier

An artificial neural network can be used as a **classifier** as well.

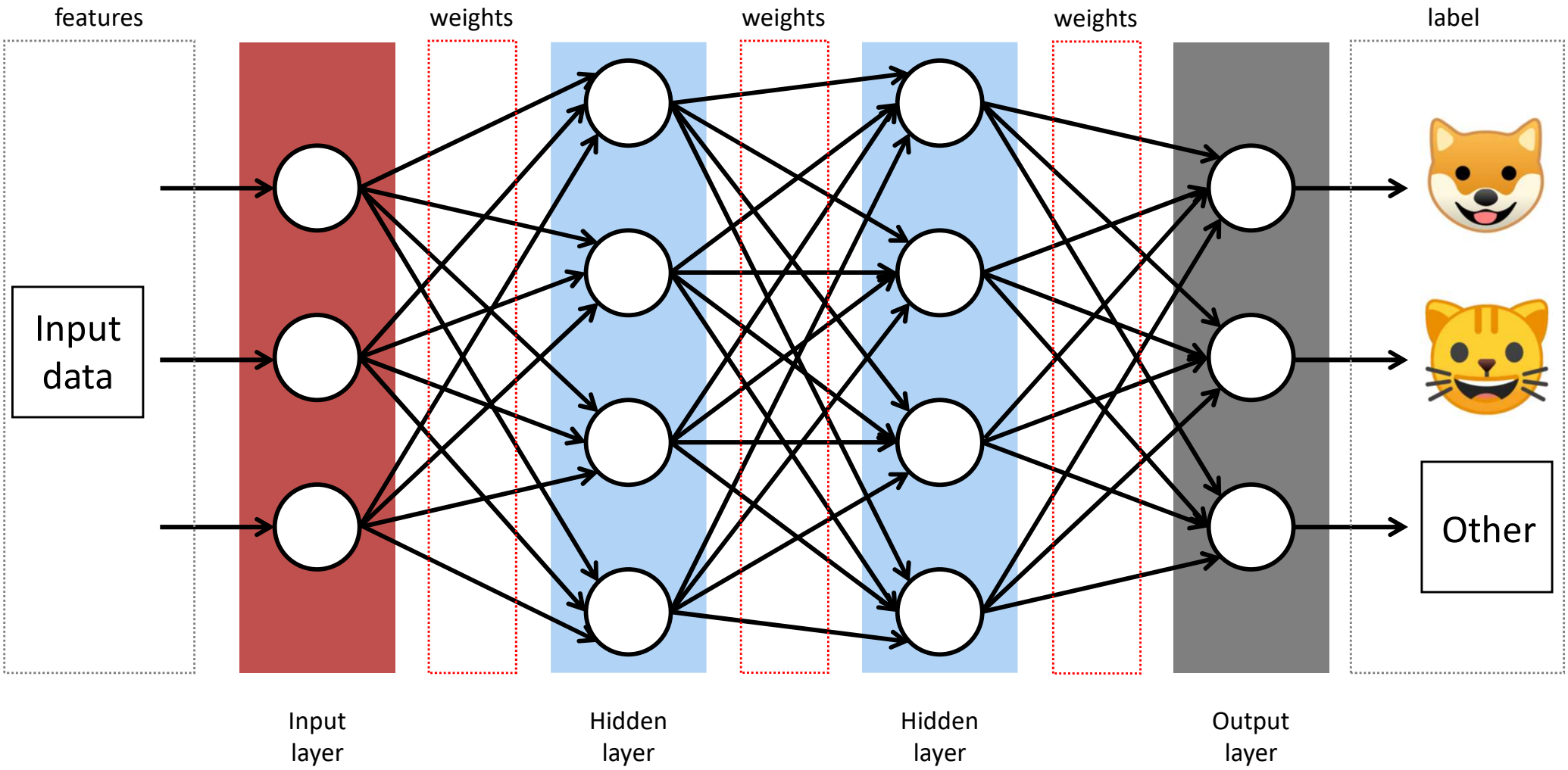


ANN as a Classifier



ANN: Supervised Learning

In order to work properly a classifier **needs to be trained** first with **labeled data**.



Training will **adjust all the weights** within this artificial neural network.

Training Data: Features + Labels

Typically input data will be represented by a **limited set of features**.

Features:
Wheels: 4
Weight: 8 tons
Passengers: 1

Label:
Truck

Features:
Wheels: 6
Weight: 8 tons
Passengers: 1

Label:
Truck

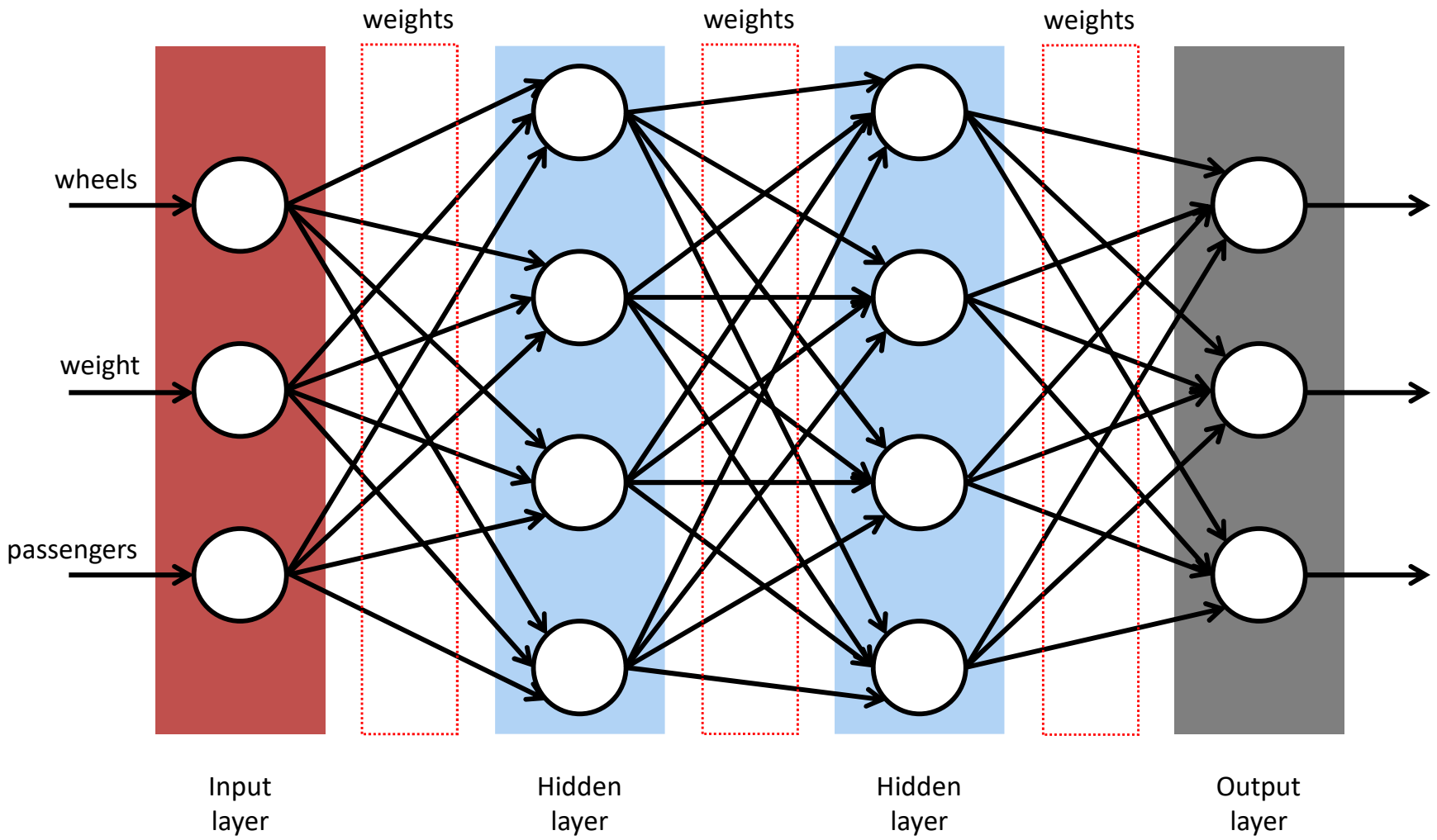
Features:
Wheels: 4
Weight: 1 ton
Passengers: 4

Label:
Car

Features:
Wheels: 4
Weight: 2 tons
Passengers: 4

Label:
Car

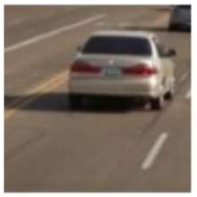
ANN: Supervised Learning



Training Data: Images + Labels

A classifier **needs to be "shown" thousands of labeled examples to learn**.

Label: BUS

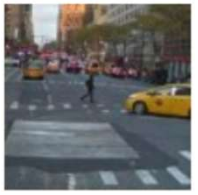


Label: CAR

Label: BRIDGE

Label: PALM

Label: TRAFFIC LIGHT



Label: TAXI

Label: CROSSWALK

Label: CHIMNEY

Label: MOTORCYCLE

Label: STREET SIGN

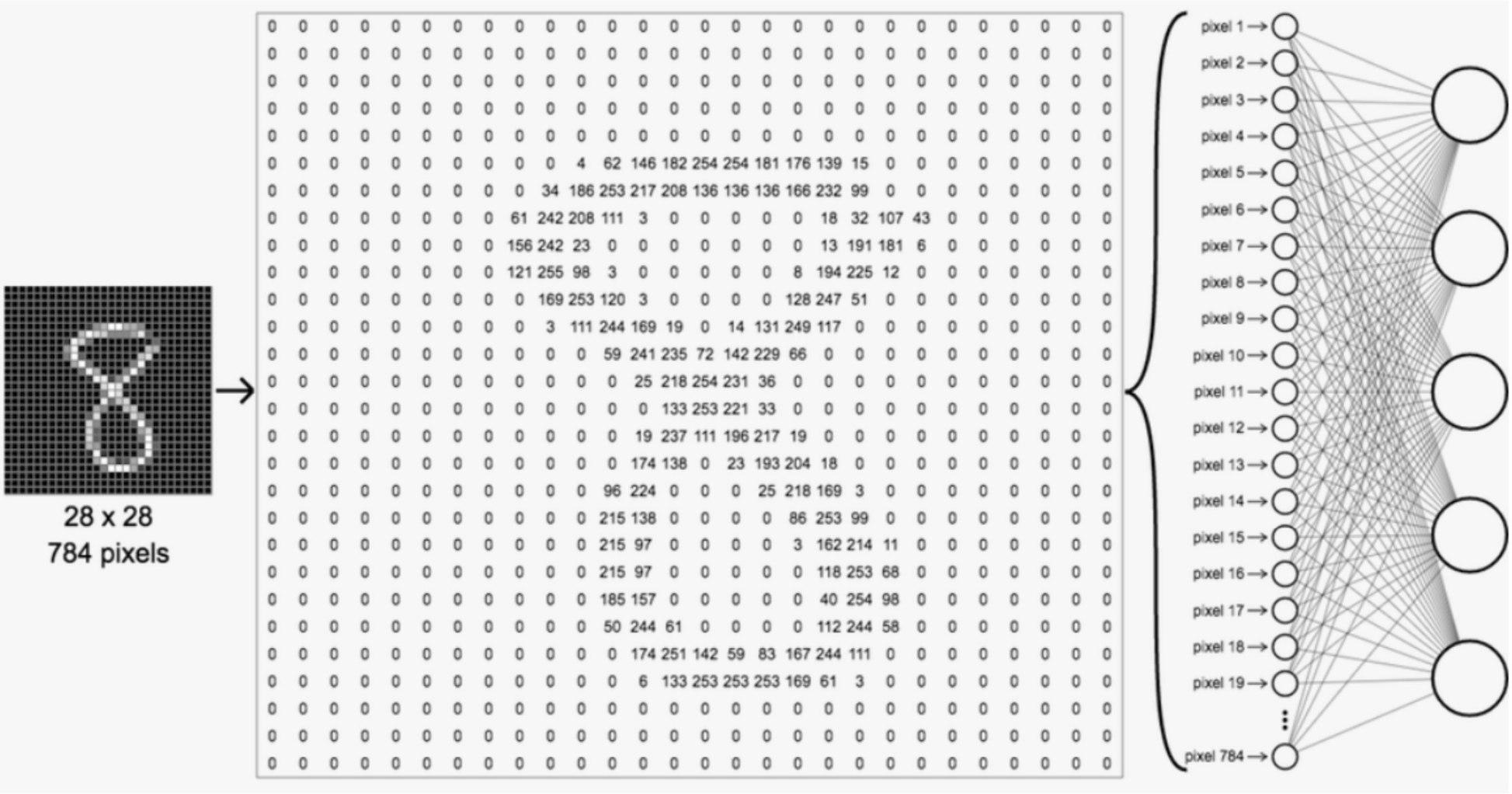
Label: HYDRANT

Label: BICYCLE

Note how some images are "incomplete" and "flawed".

Digit Image as ANN Feature Set

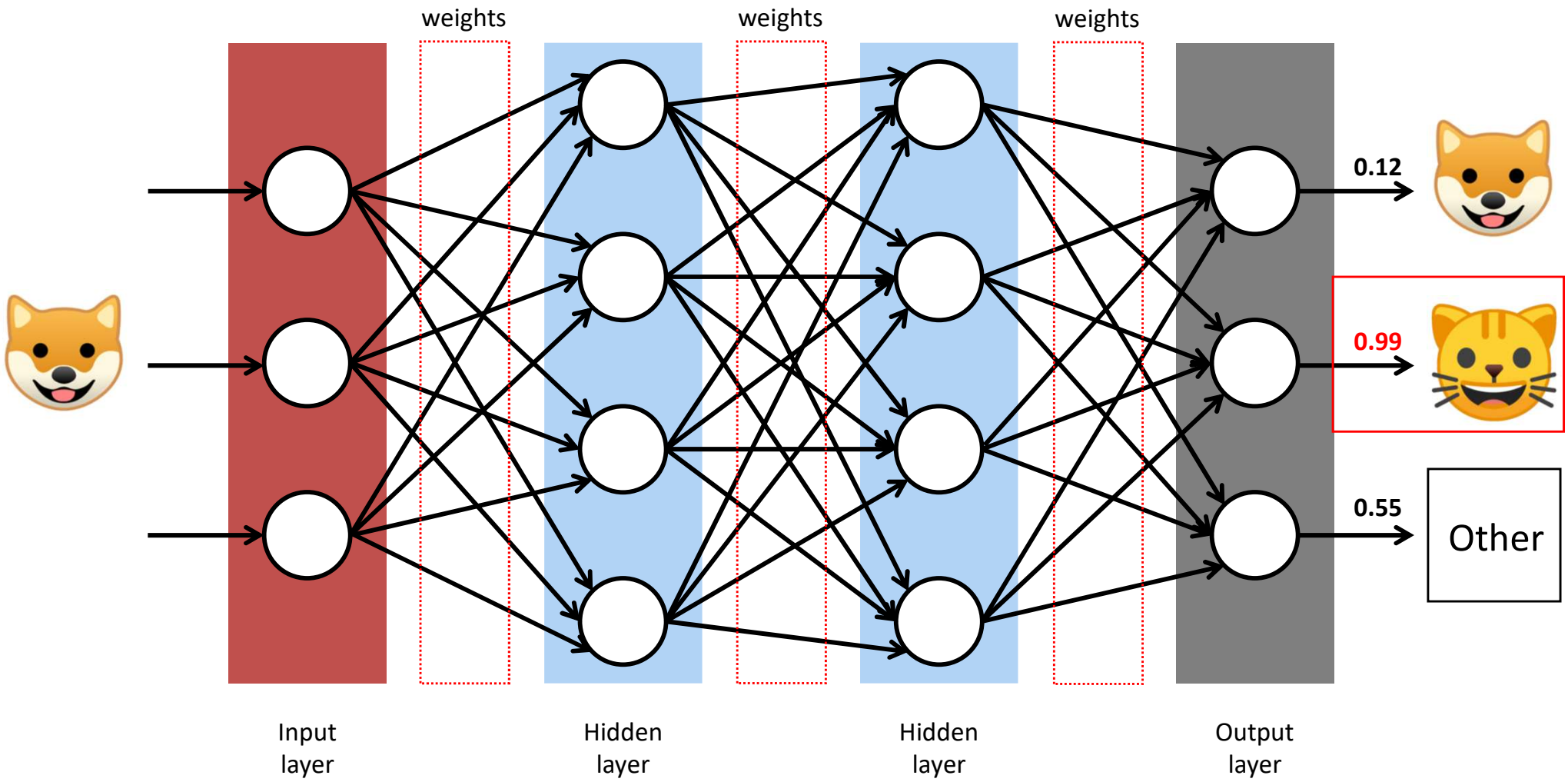
Individual features need to be “extracted” from an image. An image is numbers.



Source: https://nikolanews.com/not-just-introduction-to-convolutional-neural-networks-part-1/

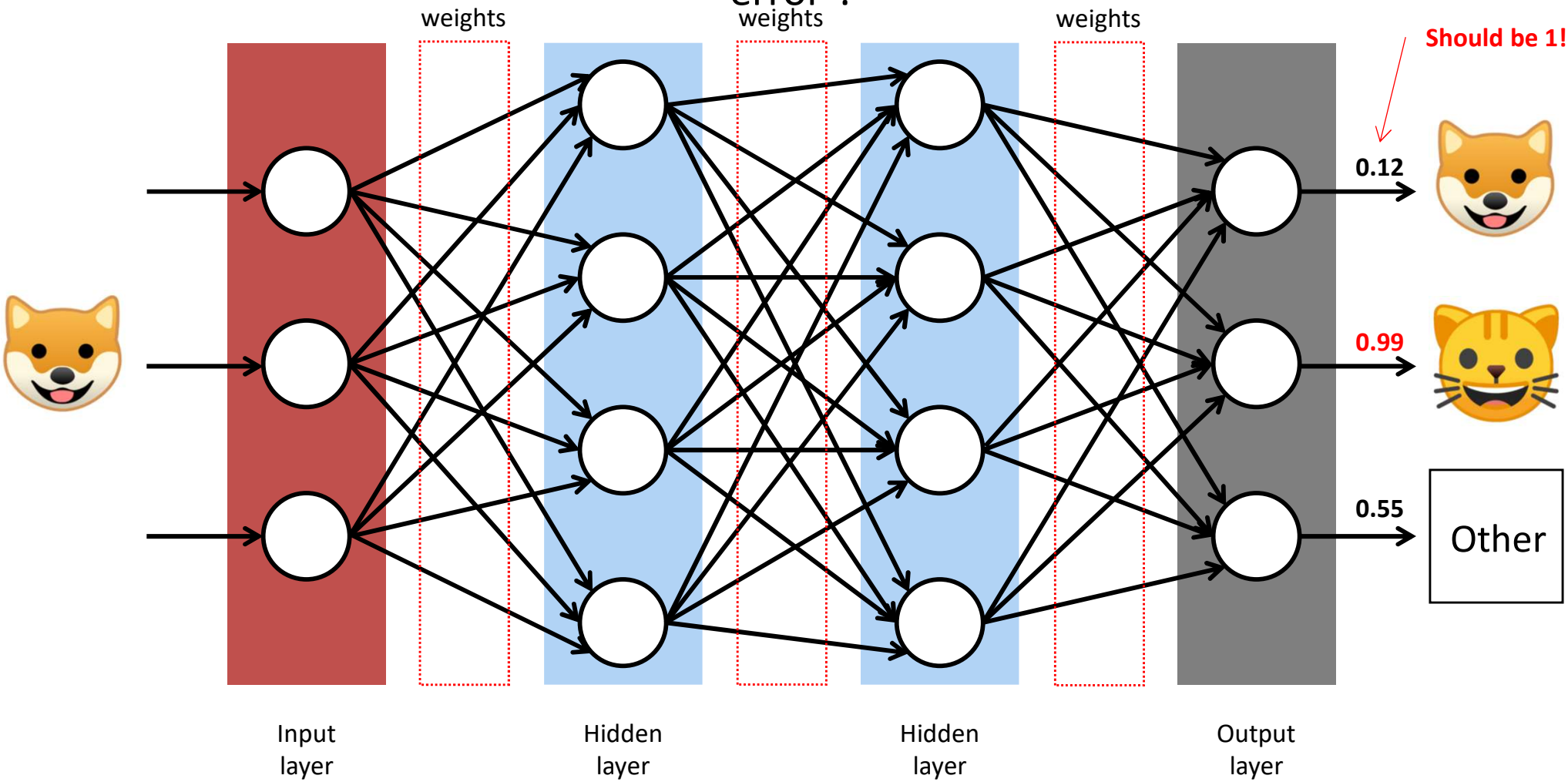
ANN: Supervised Learning

An **untrained classifier** will **NOT label input data correctly**.



ANN: Training

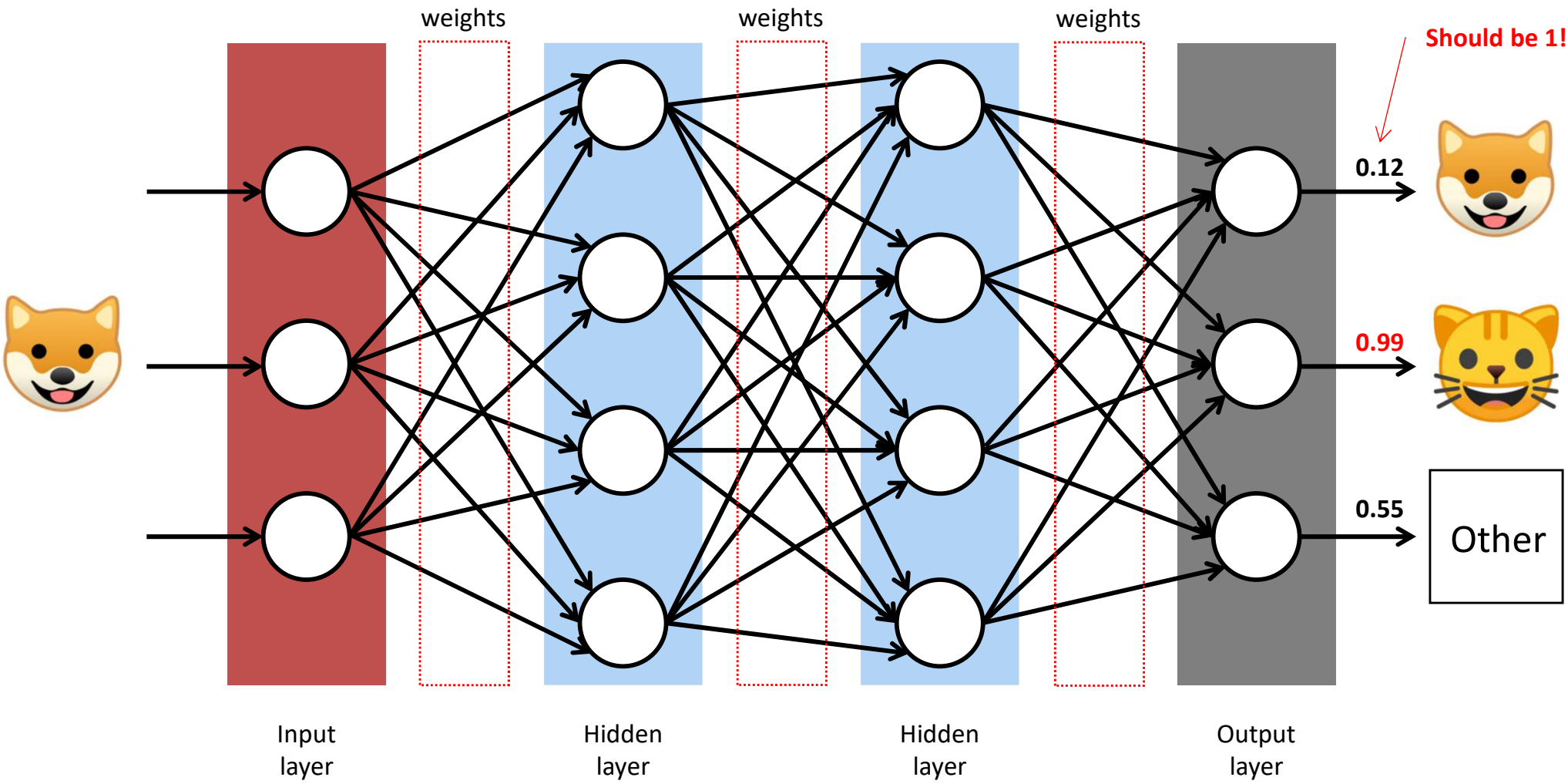
Given: input d  and it's corresponding **expected** label: DOG calculate "error".



"Error" = 0.88. Go back and **adjust all the weights** to ensure it is lower next time.

ANN: Training

Show data / label pair:  / DOG.



Correct all the weights. Repeat many times.

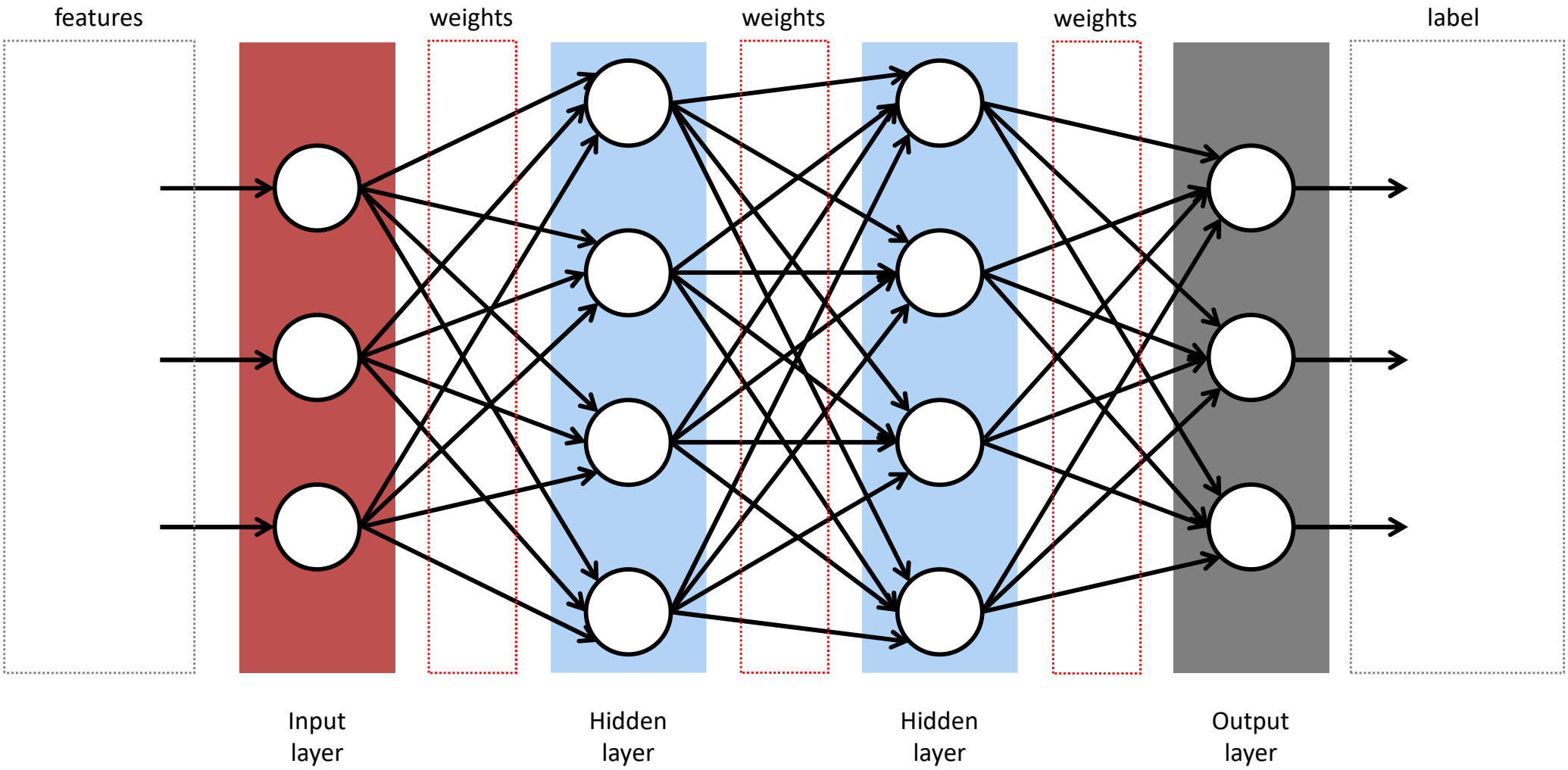
Exercise: ANN Demo

http://playground.tensorflow.org/

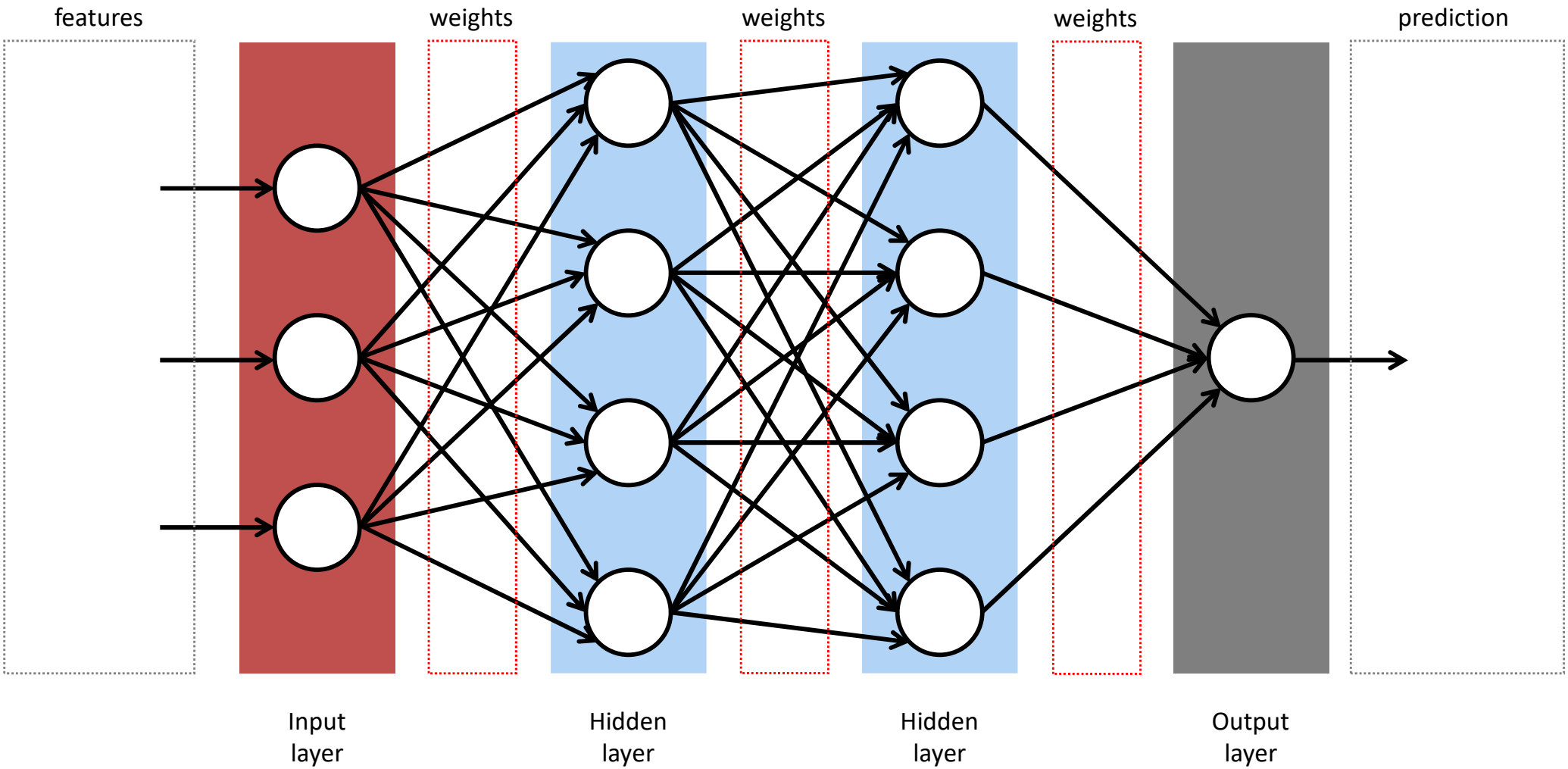
Exercise: Train a Classifier!

https://teachablemachine.withgoogle.com/

ANN for Classification



ANN for Regression



ANN for Regression: Used Car Price

Used car price predictor: train it first with used **car data - price** pairs.