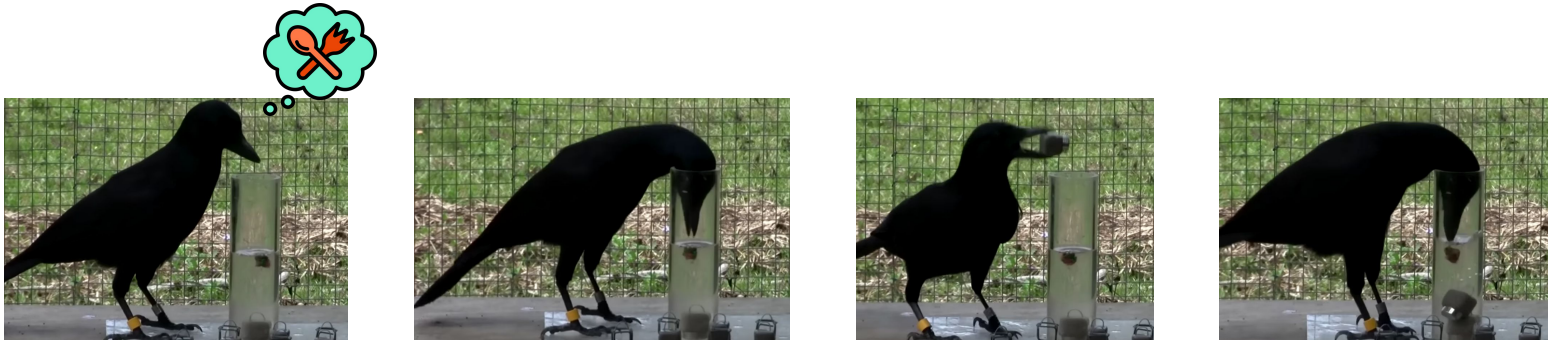
Introductory Seminar on Artificial Intelligence and Machine Learning

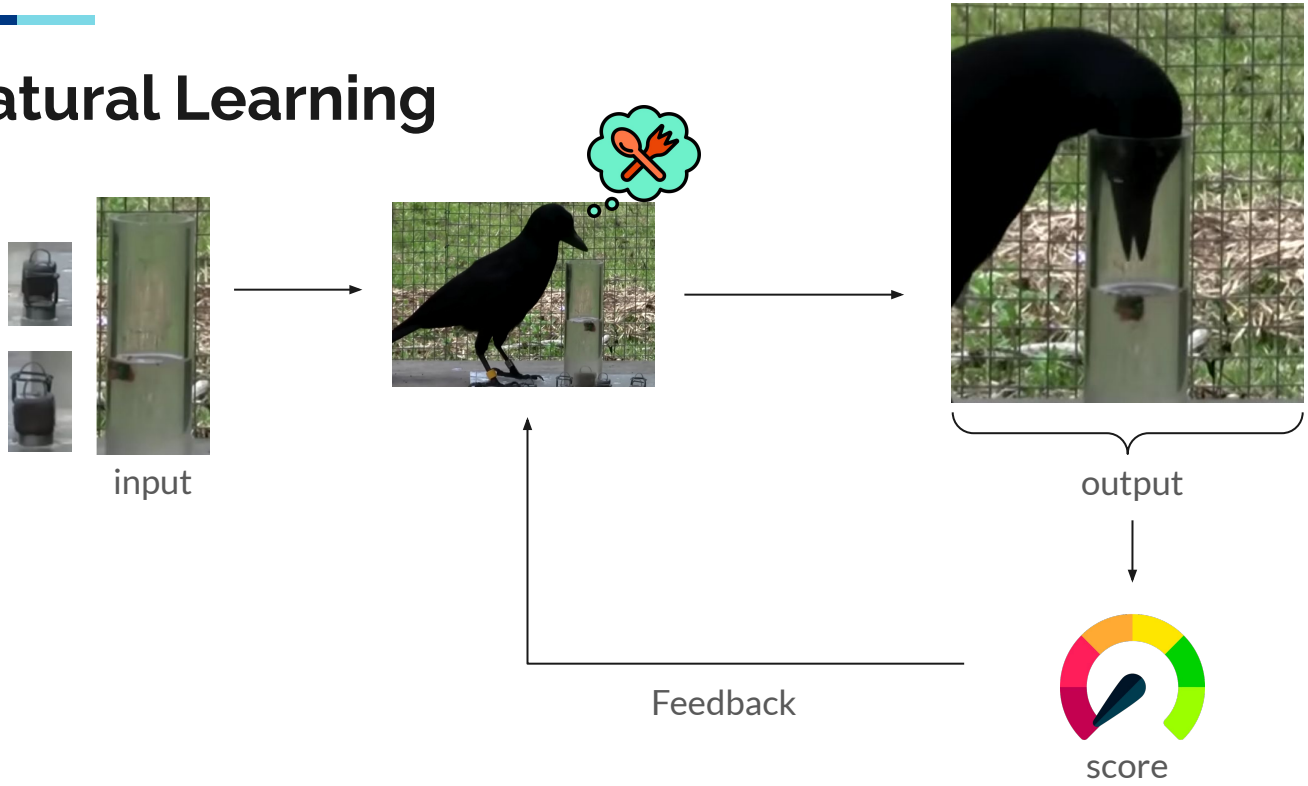
Emanuele Ledda, Cagliari Digital Lab 2024 - Day 2

Machine Learning

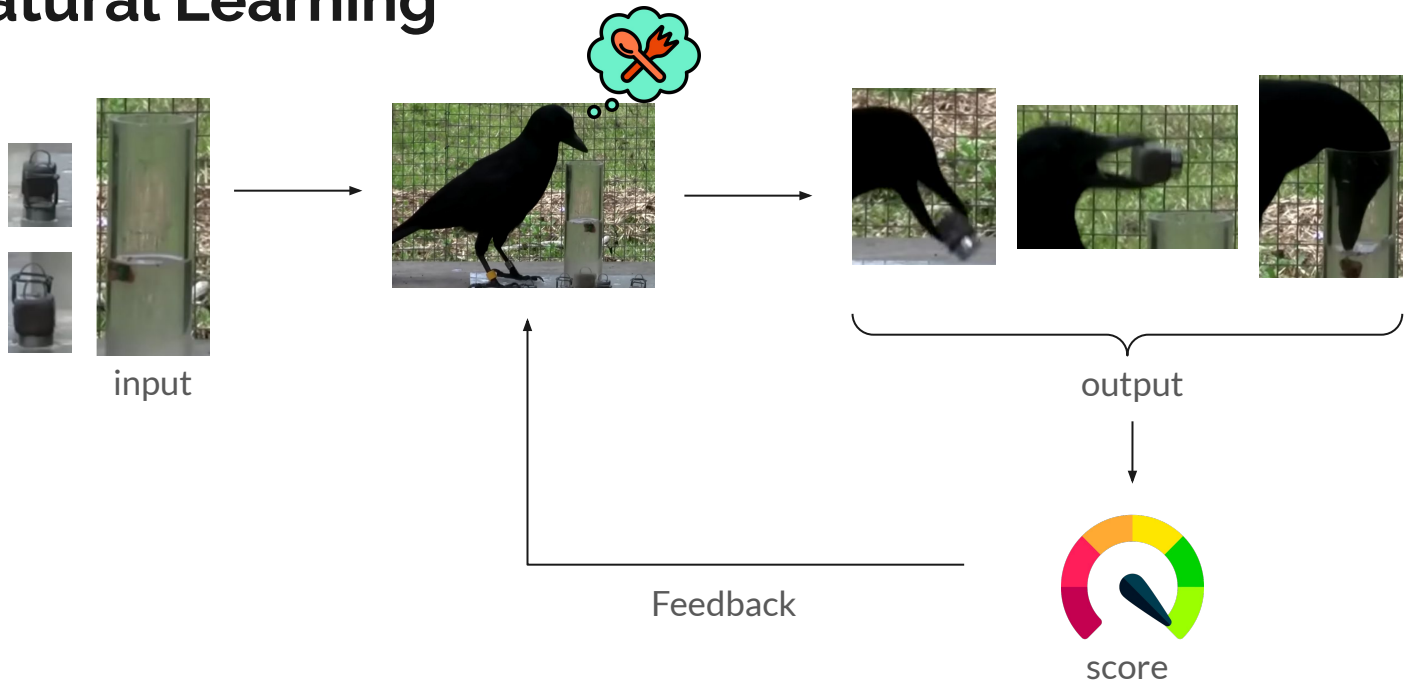
Natural Learning



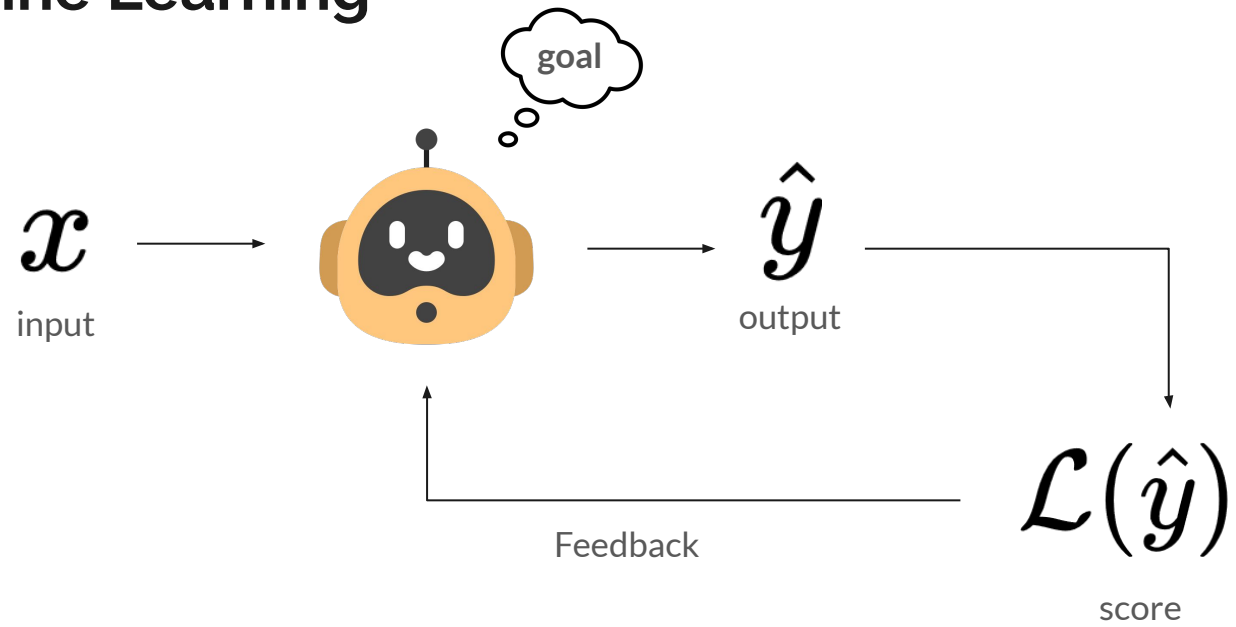
Natural Learning



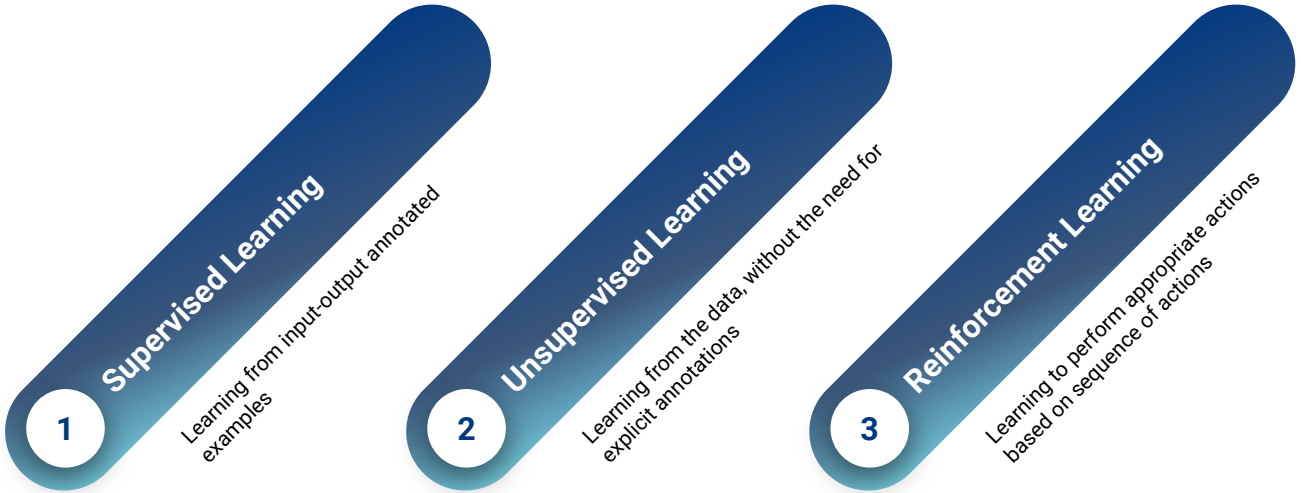
Natural Learning



Machine Learning



Paradigmi di Apprendimento



Supervised Learning

The algorithm learns by using **labels**, i.e., annotations of correct outputs associated with specific inputs that the machine can use for learning correct behaviors.

Examples of supervised task:

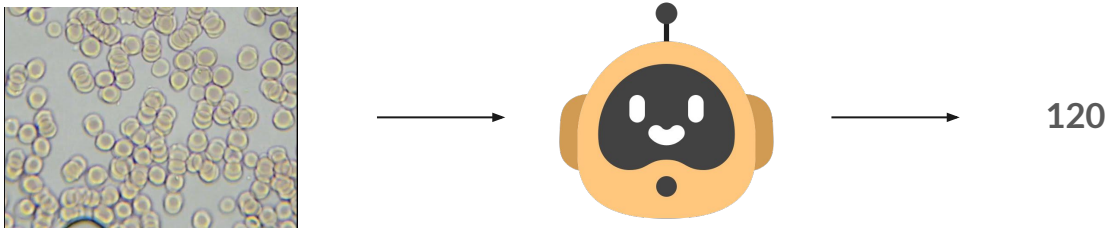
- **Classification**: assigning a **class** to an input object

Supervised Learning

The algorithm learns by using **labels**, i.e., annotations of correct outputs associated with specific inputs that the machine can use for learning correct behaviors.

Examples of supervised task:

- **Regression**: assigning a **numeric value** to an input object (e.g., counting)



Unsupervised Learning

The algorithm **cannot access labels** and learns by extracting information only from the input.

Examples of unsupervised task:

- **Clustering**: divide similar objects in clusters

input

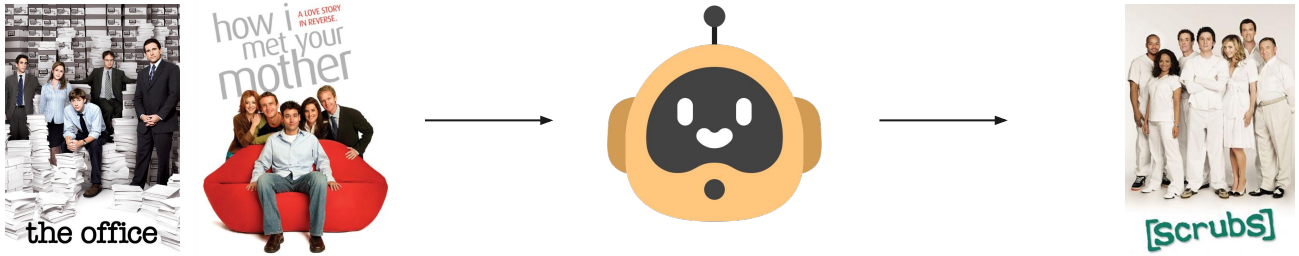
output

Unsupervised Learning

The algorithm **cannot access labels** and learns by extracting information only from the input.

Examples of unsupervised task:

- **Association Analysis**: Extract association between objects (e.g., recommending tv series)



input

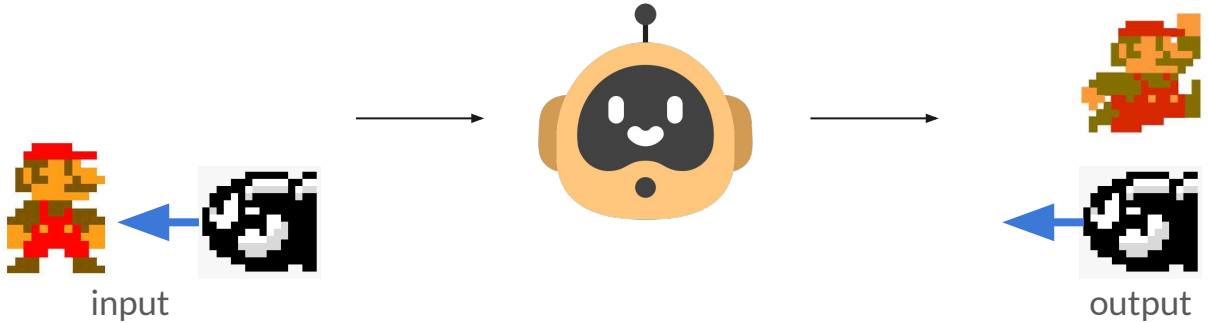
output

Reinforcement Learning

The algorithm should find an optimal action policy based on examples of the outcome of entire courses of actions.

Examples of reinforcement learning tasks:

- **Game Playing**: Learn which move should the agent take for winning a game



Linear Classifiers

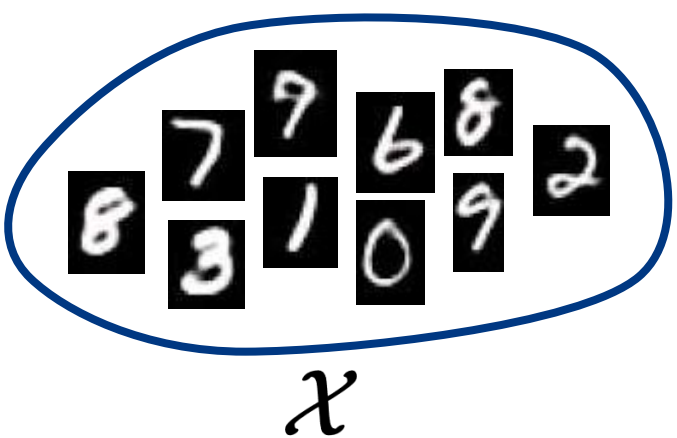
Supervised Classification

- Assigning a **class** to an input **object**

- One of the most common task:
  - Optical Character Recognition
  - Biometrics (e.g., face recognition)
  - Sentiment Analysis
  - Malware Detection
  - …

Supervised Classification

- Assigning a **class** to an input **object**
- e.g, the “objects” are images of hand written digits and the “classes” are the digits between 0 and 9

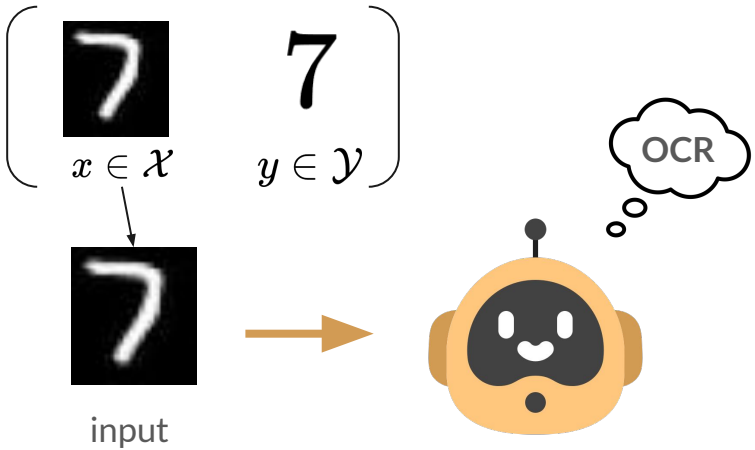


x

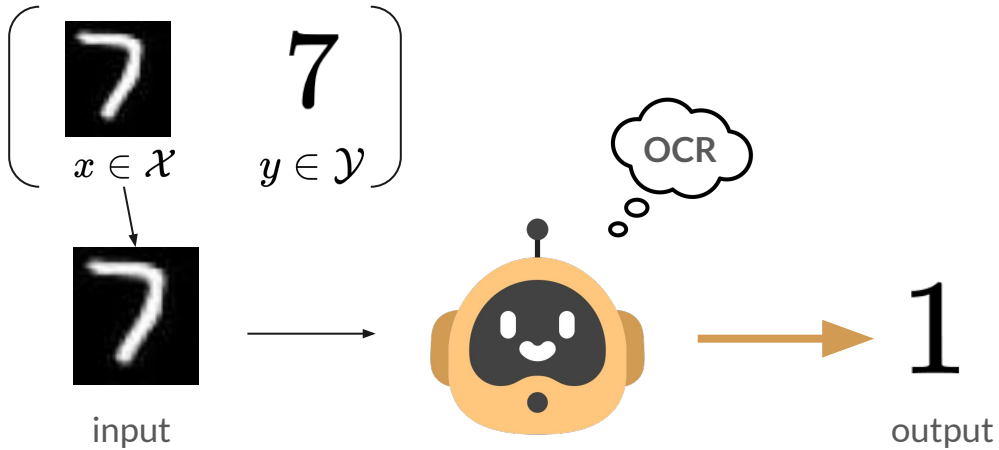
$$0, 1, 2, 3, 4,$$
$$5, 6, 7, 8, 9$$

y

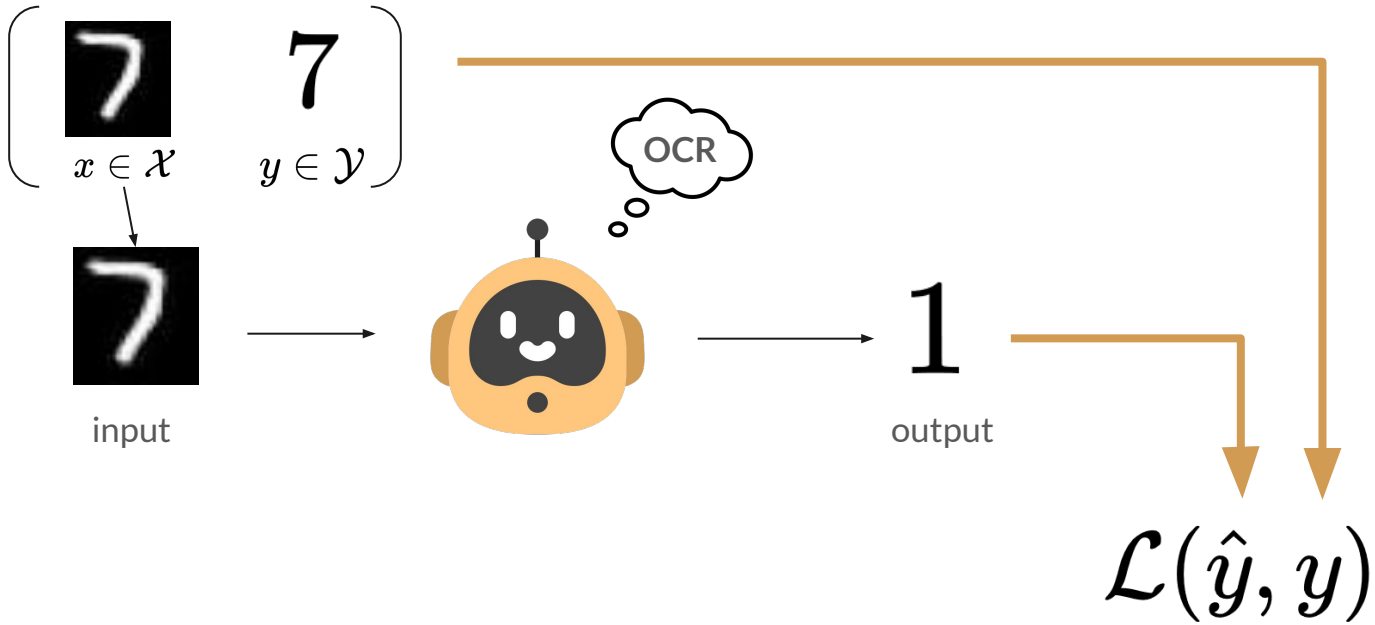
Supervised Classification



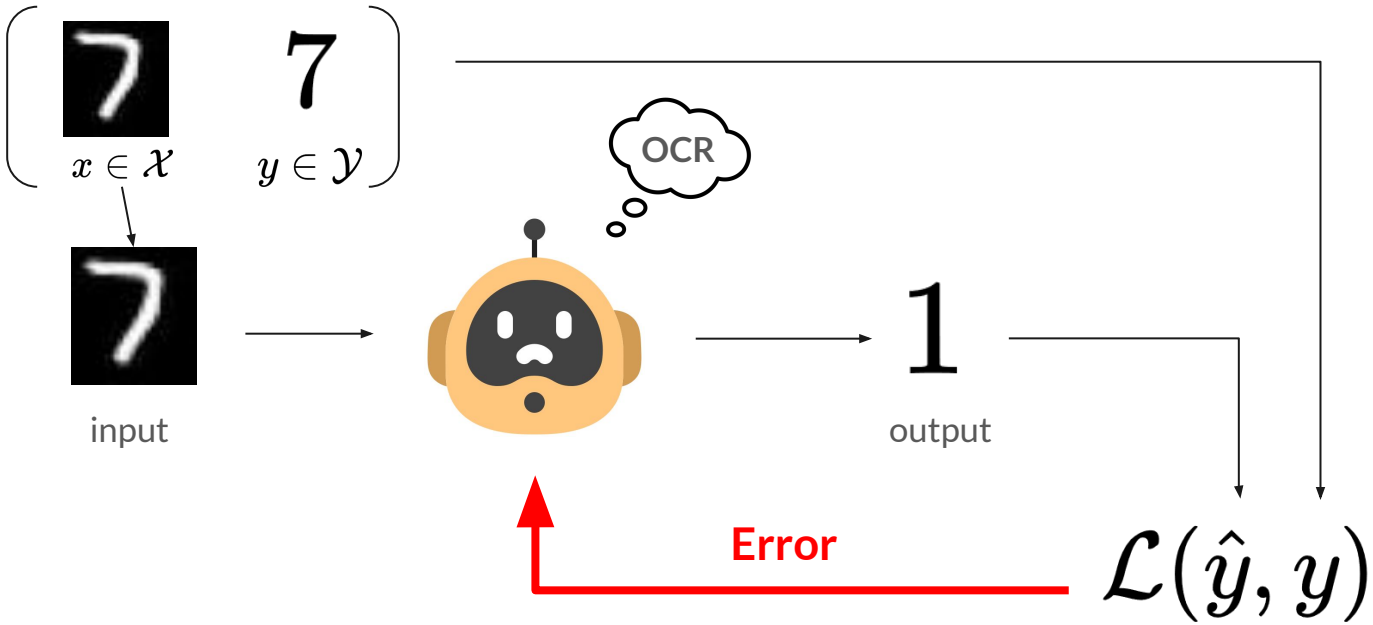
Supervised Classification



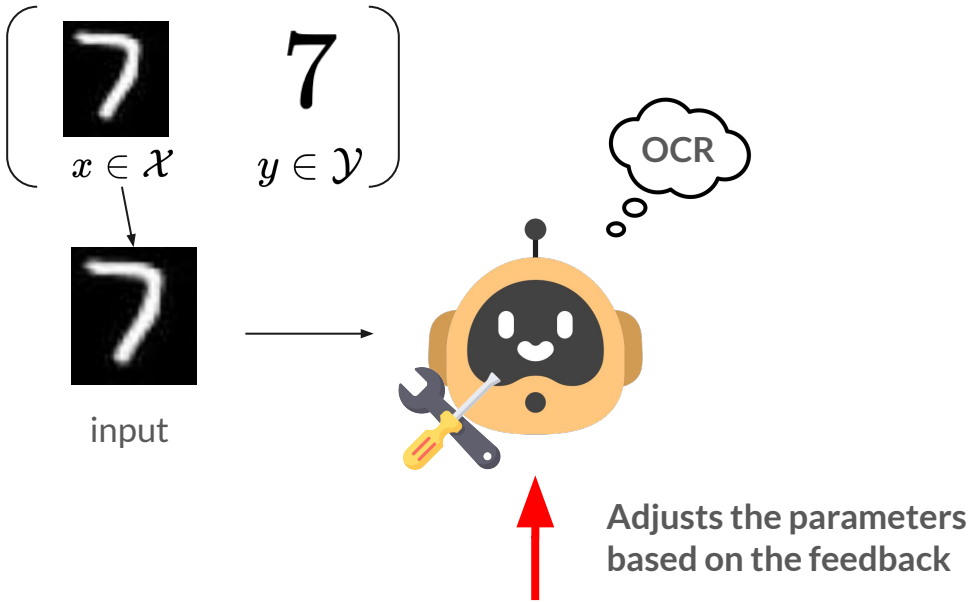
Supervised Classification



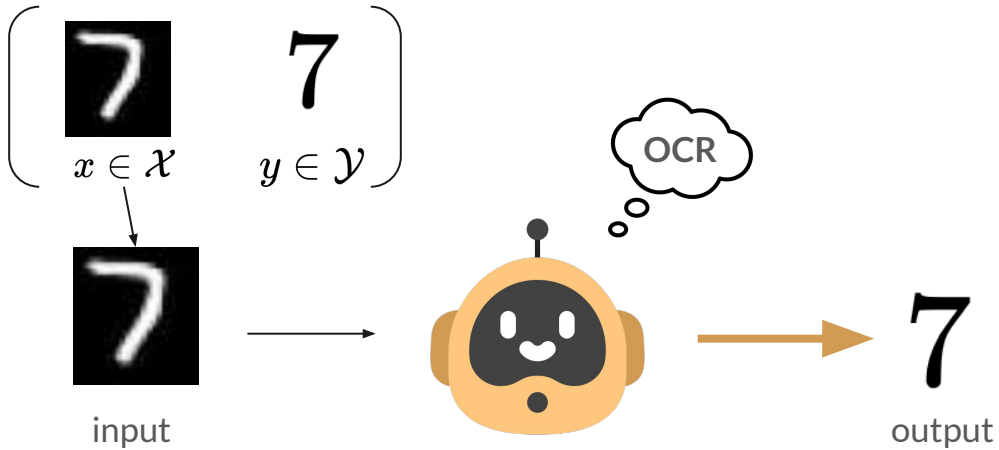
Supervised Classification



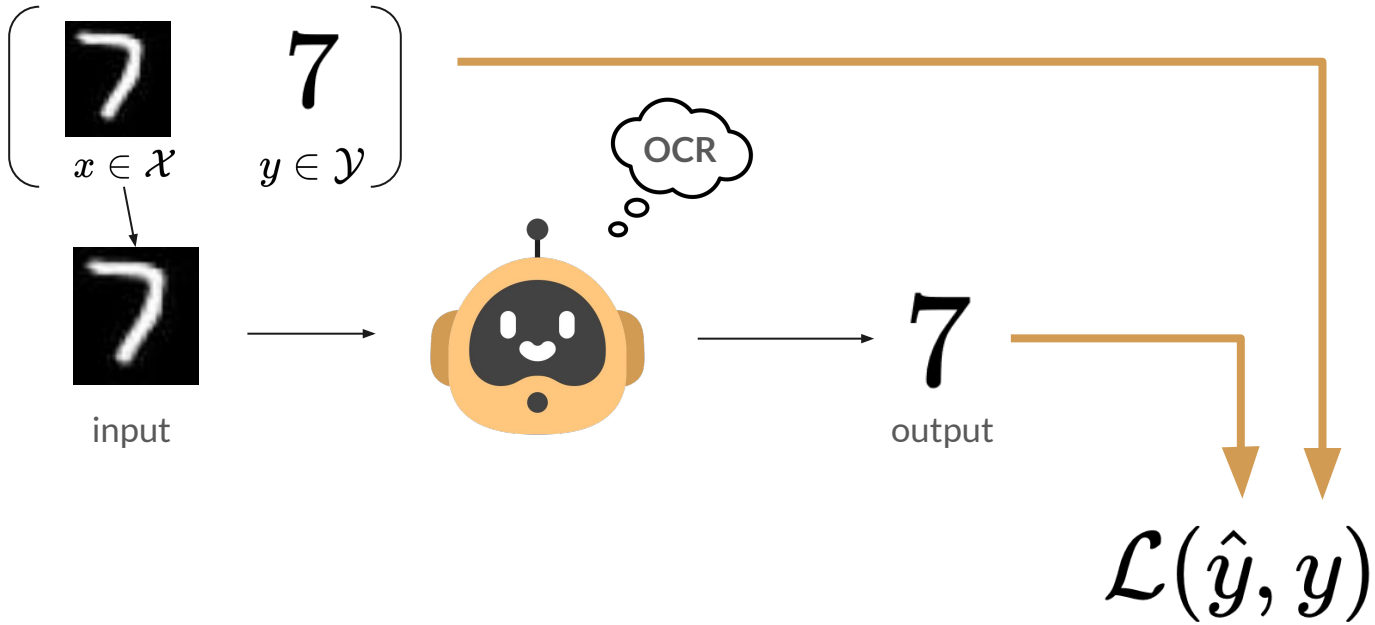
Supervised Classification



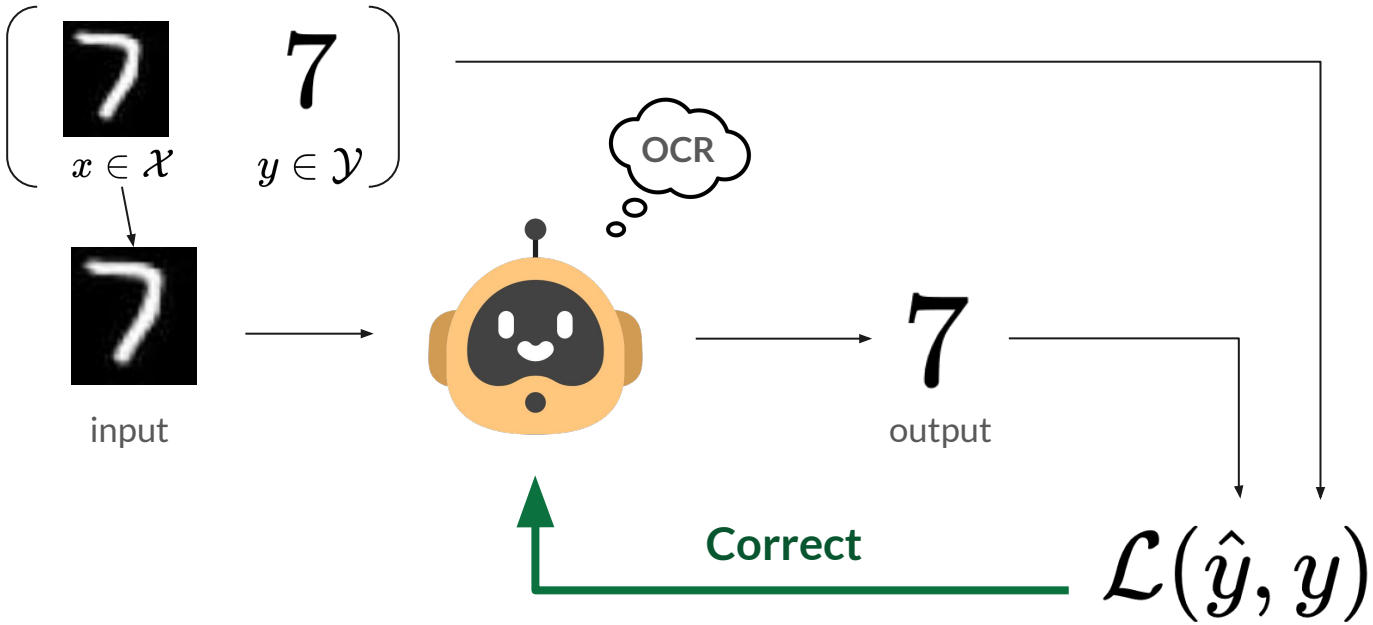
Supervised Classification



Supervised Classification



Supervised Classification



Supervised Classification

- Formally, we can see the input **x** as a **feature vector**, and the **y** as the **respective class**
- One can see a classification algorithm as a function **h** (called a **hypothesis**) mapping an input **x** to a class **y**.

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

- A specific hypothesis function represents a "**classifier**"
- The set of all possible hypotheses represents a **model** (also known as **hypothesis space**).

$$h \in \mathcal{H}$$

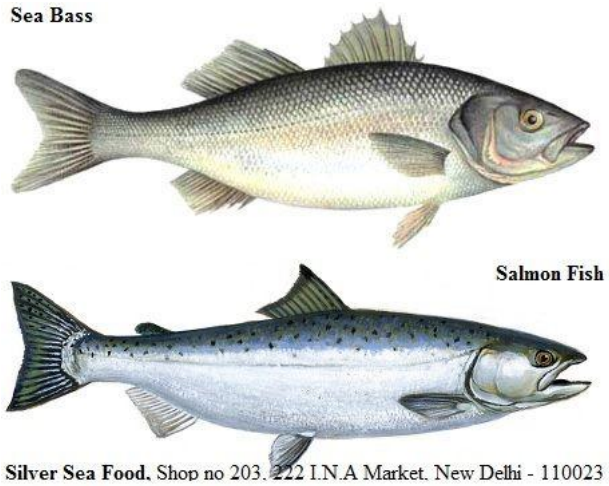
Supervised Classification

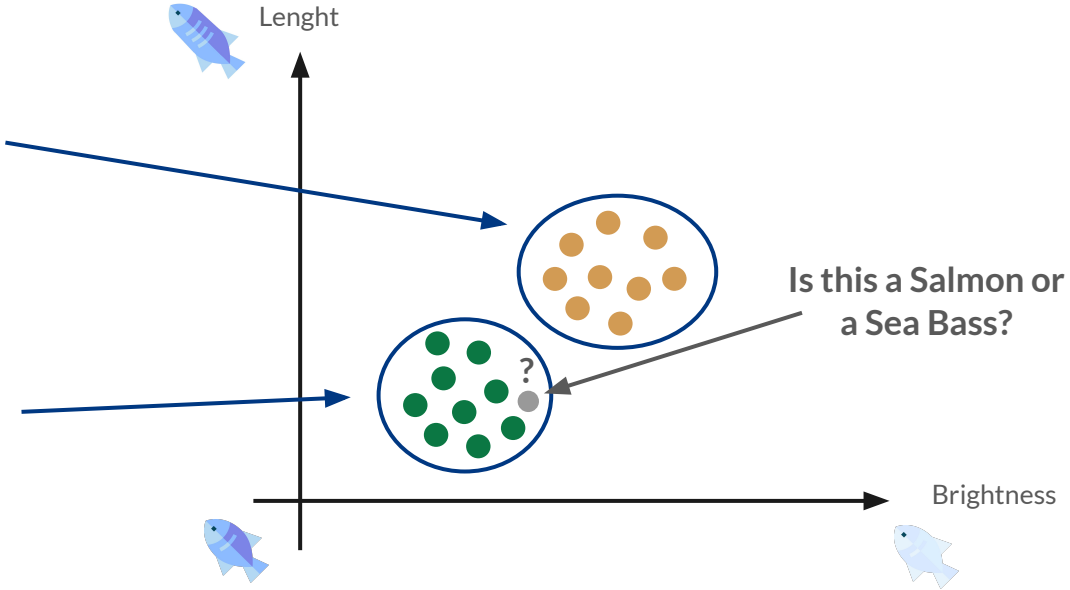
- Example of a simple model: **linear functions**

$$f(x) = \sum_{i=1}^{D} x_i \cdot w_i + b$$

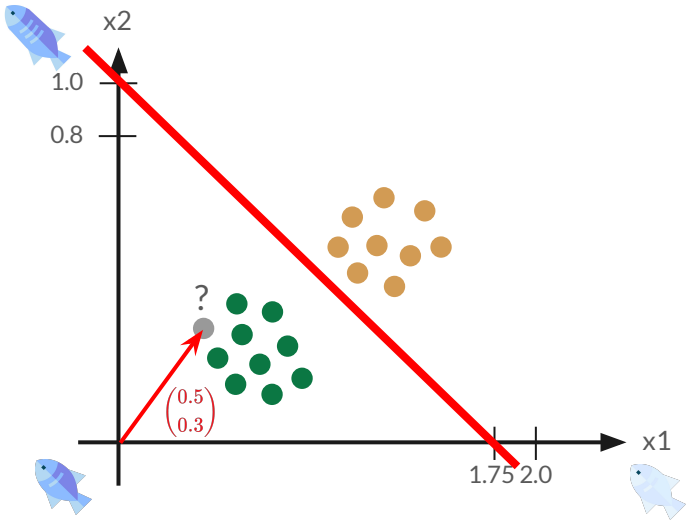
- **W** (weights) e **b** (bias) are the model parameters: a **classifier** is a specific instance of the weights and bias (**hyperplane**)

Binary Classification: Fishes Example





Classificazione Binaria Lineare



- Each point is a 2D vector [x1,x2]
- Intuitively, a hyperplane can separate the salmons from the sea basses.

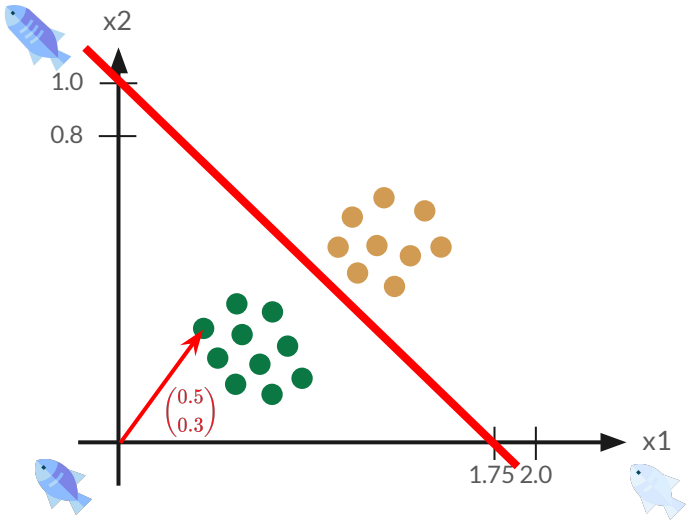
$$f(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

$$W = [w_1, w_2]^T = [1.0, 1.75]$$

$$b = -1.75$$

$$x = [x_1, x_2]^T = [0.5, 0.3]$$

Classificazione Binaria Lineare



- Each point is a 2D vector [x1,x2]
- Intuitively, a hyperplane can separate the salmons from the sea basses.

$$f(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

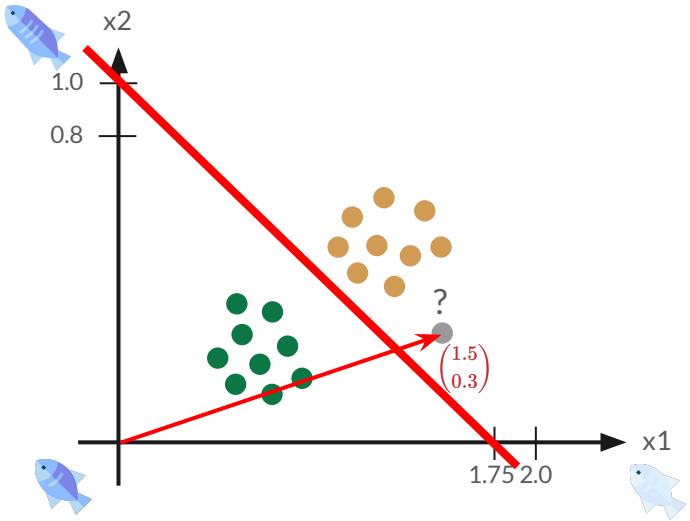
$$W = [w_1, w_2]^T = [1.0, 1.75]$$

$$b = -1.75$$

$$x = [x_1, x_2]^T = [0.5, 0.3]$$

$$0.5 \cdot 1.0 + 0.3 \cdot 1.75 - 1.75 = -0.725$$

Classificazione Binaria Lineare



- Each point is a 2D vector [x1,x2]
- Intuitively, a hyperplane can separate the salmons from the sea basses.

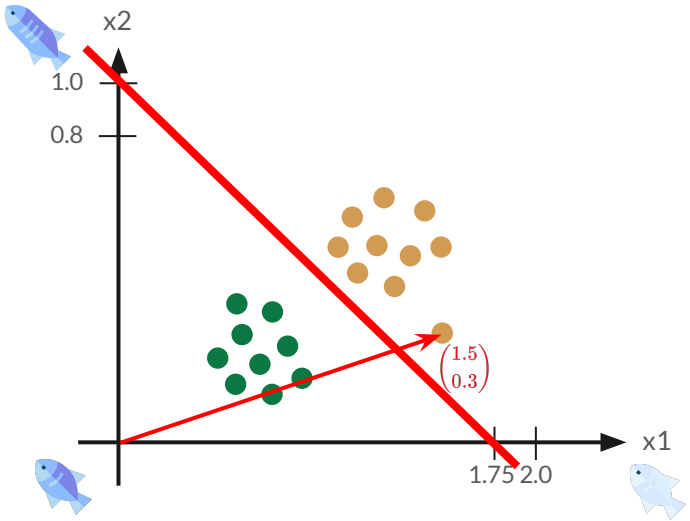
$$f(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

$$W = [w_1, w_2]^T = [1.0, 1.75]$$

$$b = -1.75$$

$$x = [x_1, x_2]^T = [1.5, 0.3]$$

Classificazione Binaria Lineare



- Each point is a 2D vector [x1,x2]
- Intuitively, a hyperplane can separate the salmons from the sea basses.

$$f(x) = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

$$W = [w_1, w_2]^T = [1.0, 1.75]$$

$$b = -1.75$$

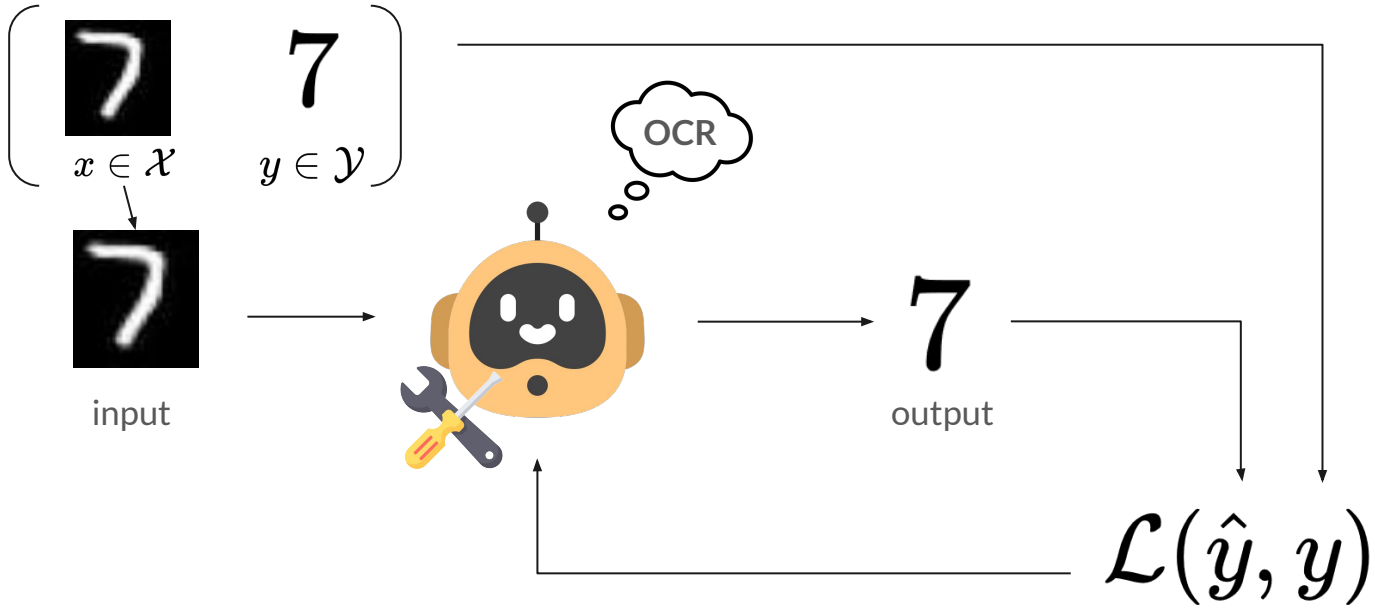
$$x = [x_1, x_2]^T = [1.5, 0.3]$$

$$1.5 \cdot 1.0 + 0.3 \cdot 1.75 - 1.75 = 0.275$$

Learning Algorithm

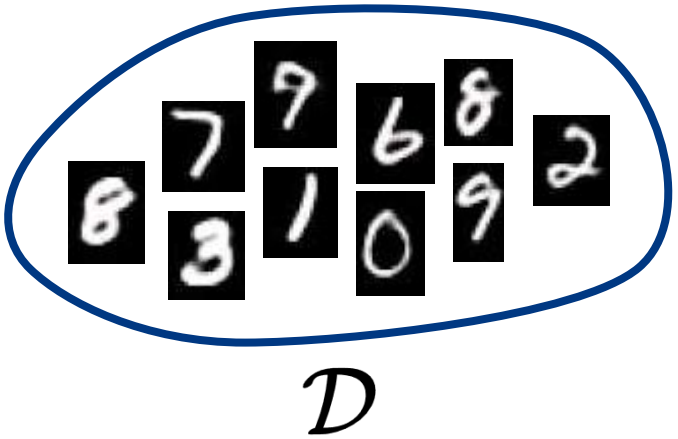
Learning Algorithm

- After selecting a proper model (e.g., linear model), we choose a proper **learning algorithm** 🛠️



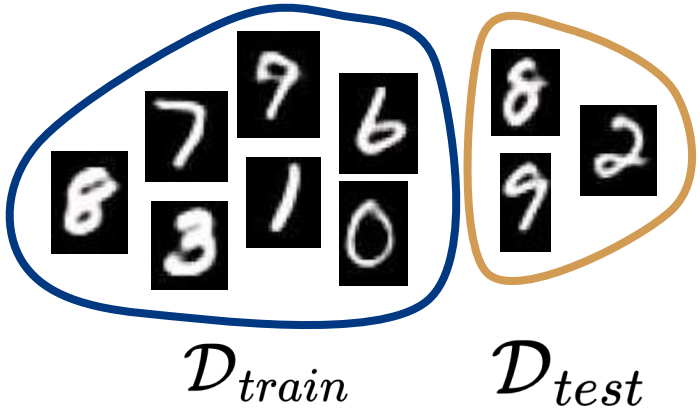
Learning Algorithm

- The objective is to obtain **generalization capabilities**, i.e., being able to give correct predictions on "new data"



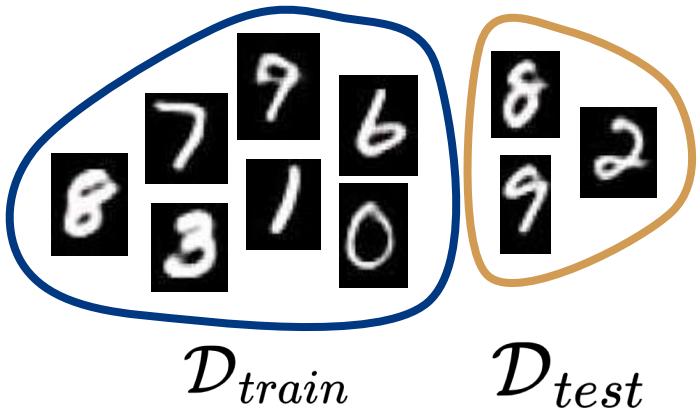
Learning Algorithm

- The objective is to obtain **generalization capabilities**, i.e., being able to give correct predictions on "new data"
- Accordingly, we divide the **data set into two partitions**, one dedicated to the training phase (**training set**) and one for evaluating the performances after training (**test set**).



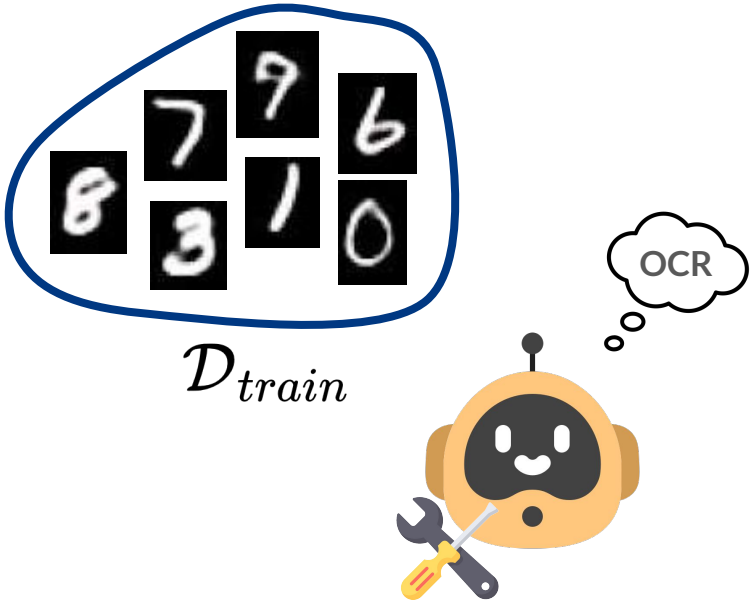
Learning Algorithm

- The objective is to obtain **generalization capabilities**, i.e., being able to give correct predictions on "new data"
- Accordingly, we divide the **data set into two partitions**, one dedicated to the training phase (**training set**) and one for evaluating the performances after training (**test set**).
- By doing so, we can obtain a classifier from a specific data set and evaluate it using separate data.



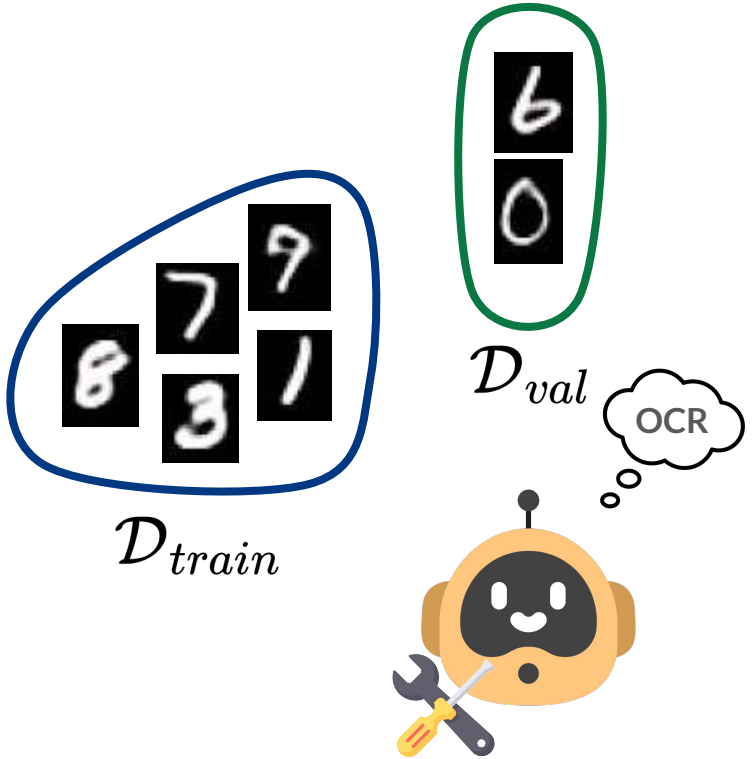
Learning Algorithm

- The learning algorithm can use only the data dedicated to the training phase

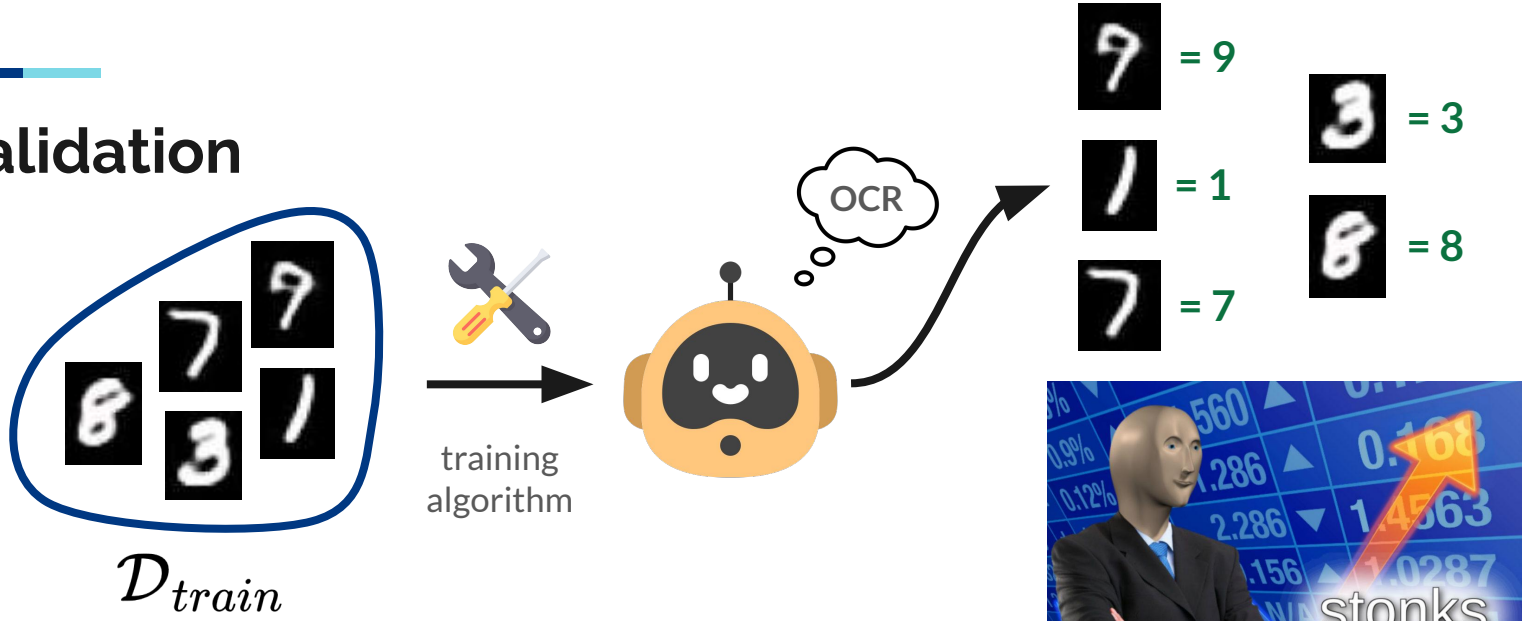


Validation

- The learning algorithm can use only the data dedicated to the training phase
- We can use a dedicated portion of the training set, called the "**validation set**," to **validate** the performances during the training.

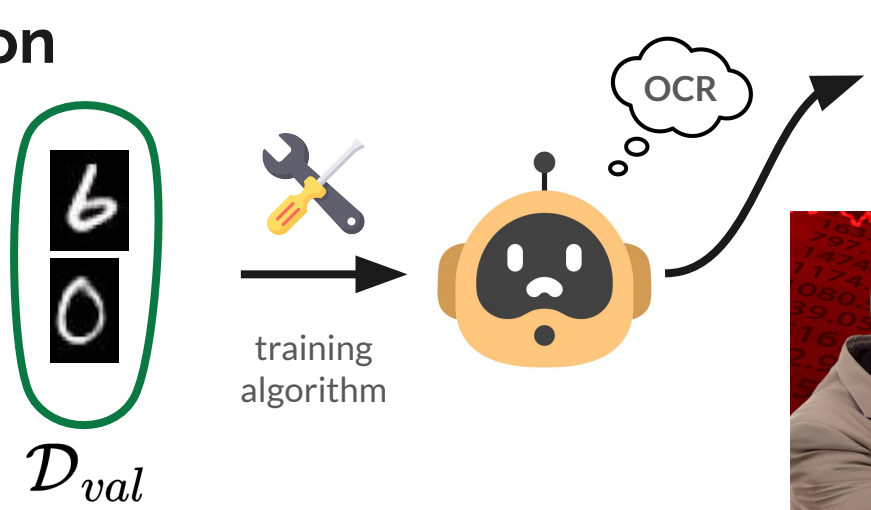


Validation



- Many classifiers can be pretty accurate on the training data
- But not so accurate con “new data”

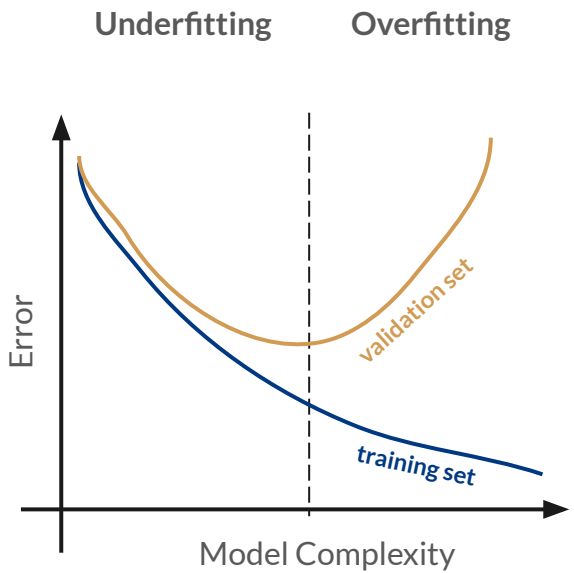
Validation



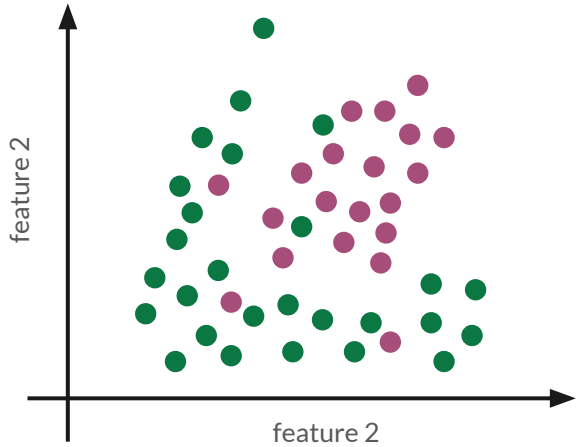
- Many classifiers can be pretty accurate on the training data
- But not so accurate con “new data”

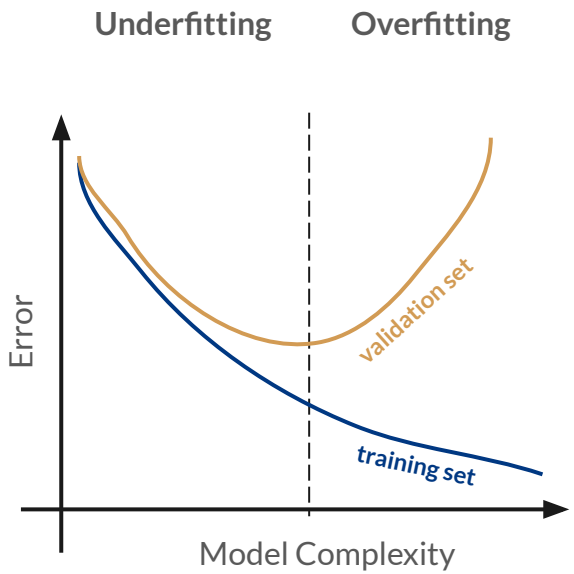
Overfitting

- Un modello opportuno dovrebbe essere adeguatamente complesso per descrivere i dati (evitando **underfitting**)
- Ma allo stesso tempo non deve essere eccessivamente complesso/specifico, per non rischiare di andare in **overfitting**
- L'overfitting è uno dei principali problemi del machine learning odierno, come vedremo più avanti con le reti neurali

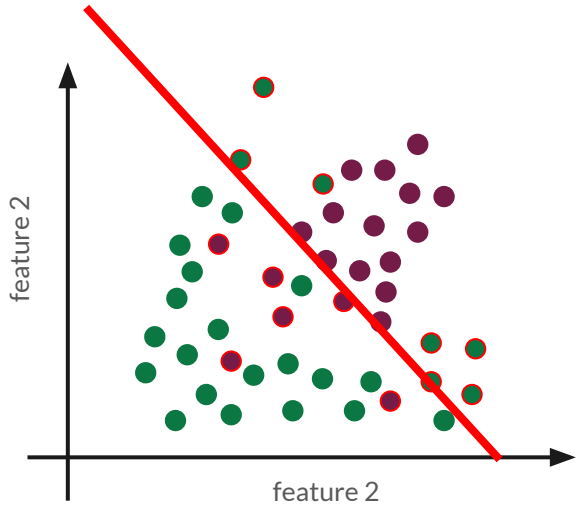


Overfitting

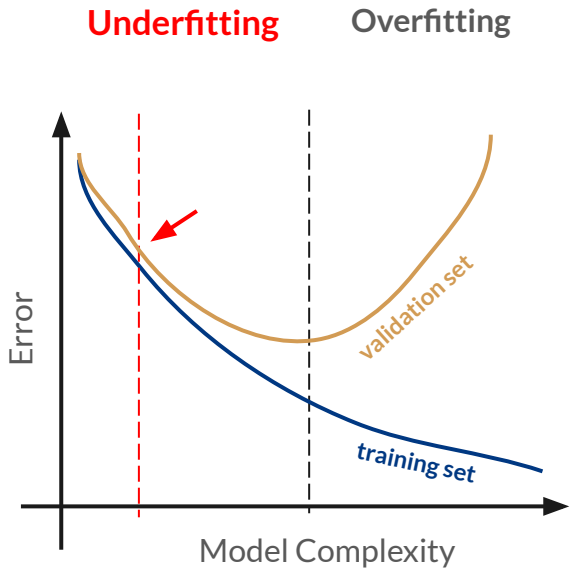




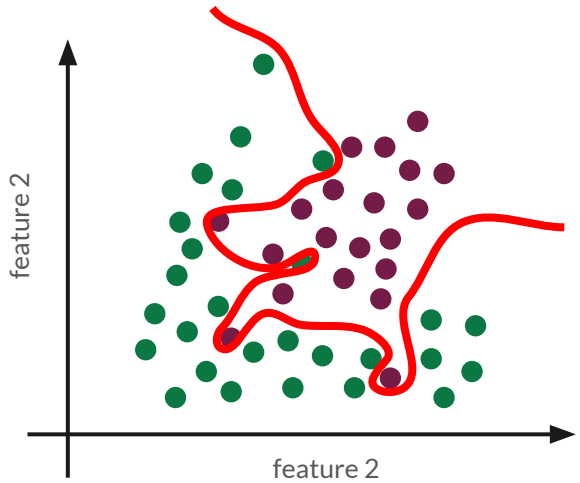
Overfitting



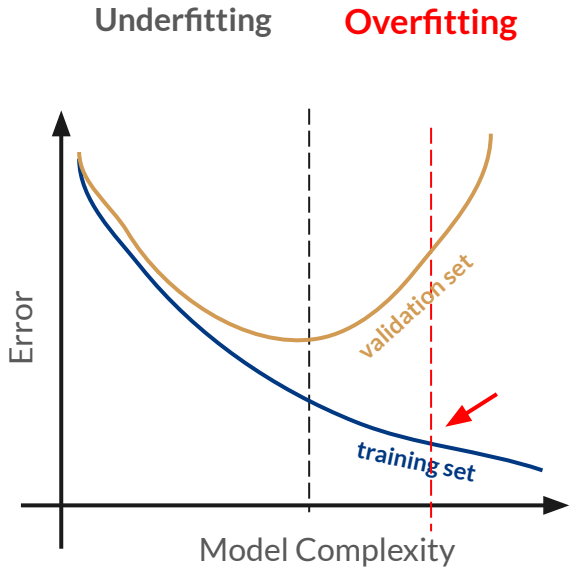
Errore sul training set: 13



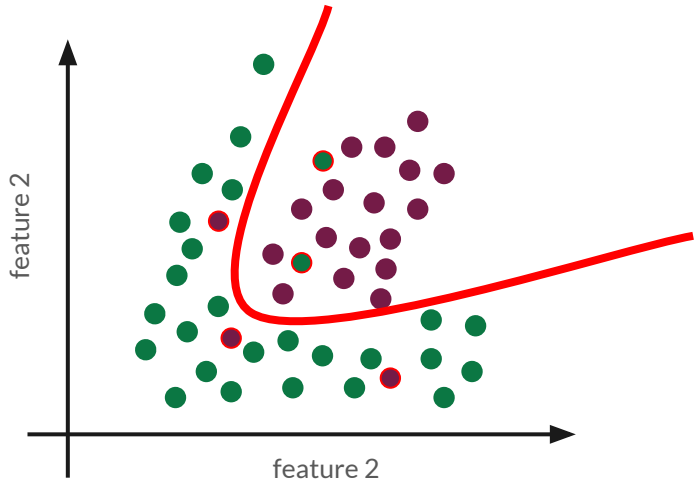
Overfitting

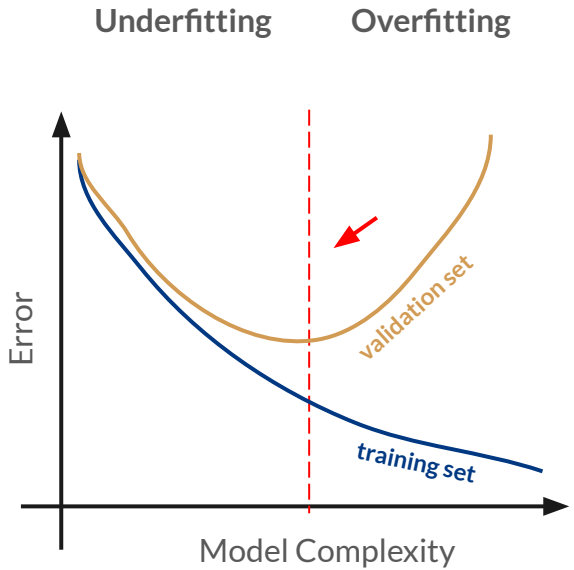


Errore sul training set: 0



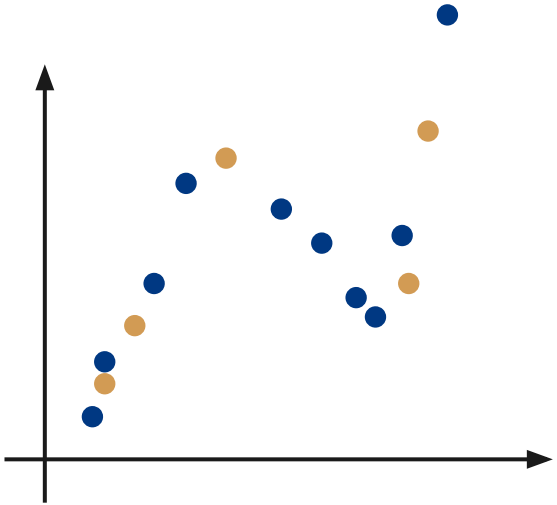
Overfitting

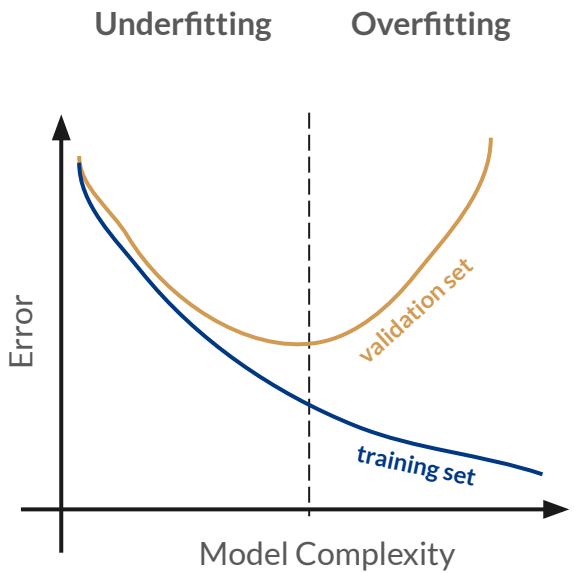




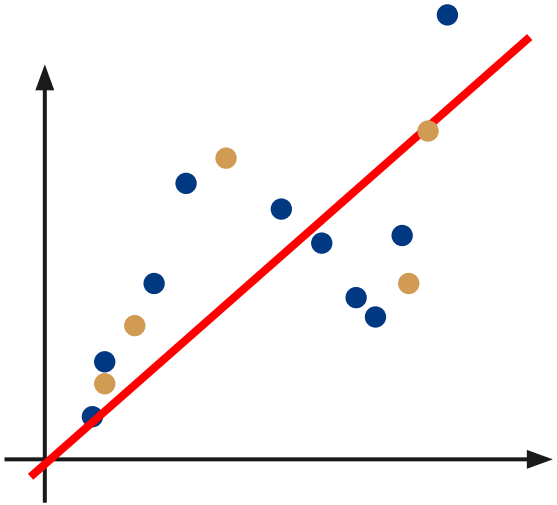
Errore sul training set: 5

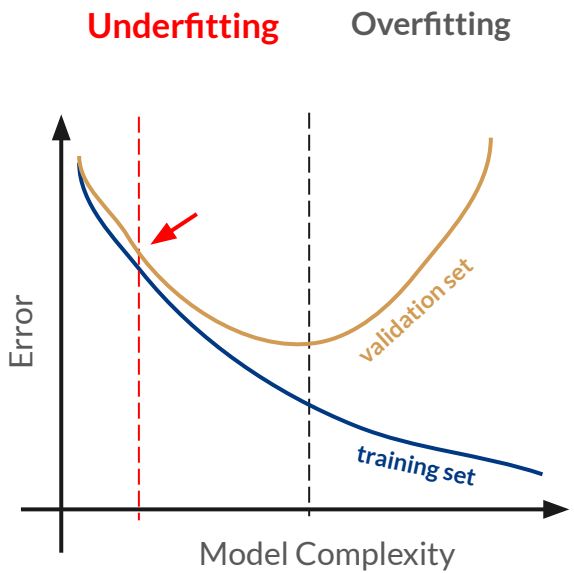
Overfitting (regression)



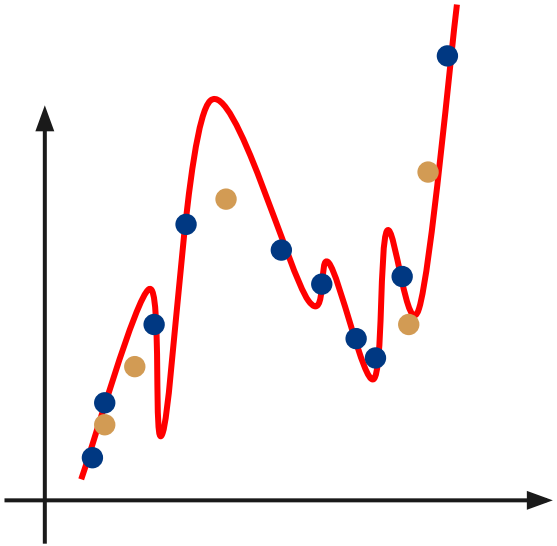


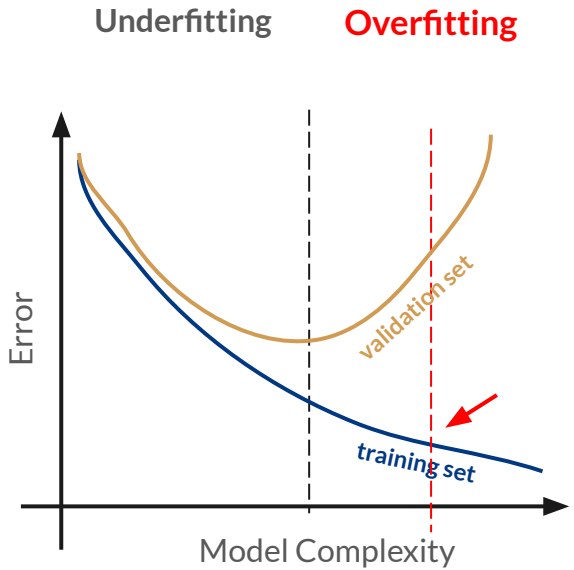
Overfitting (regression)



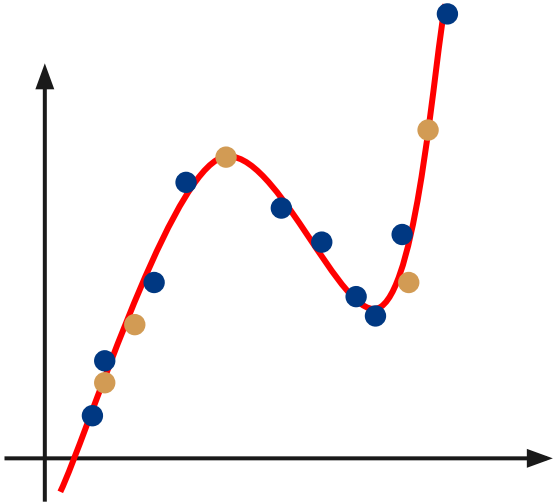


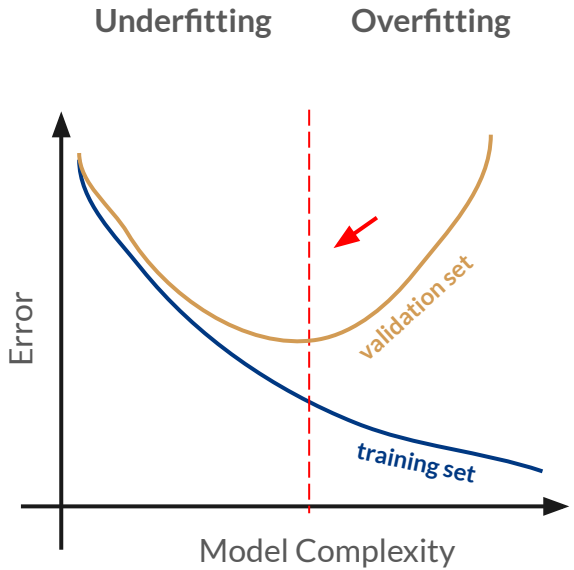
Overfitting (regression)



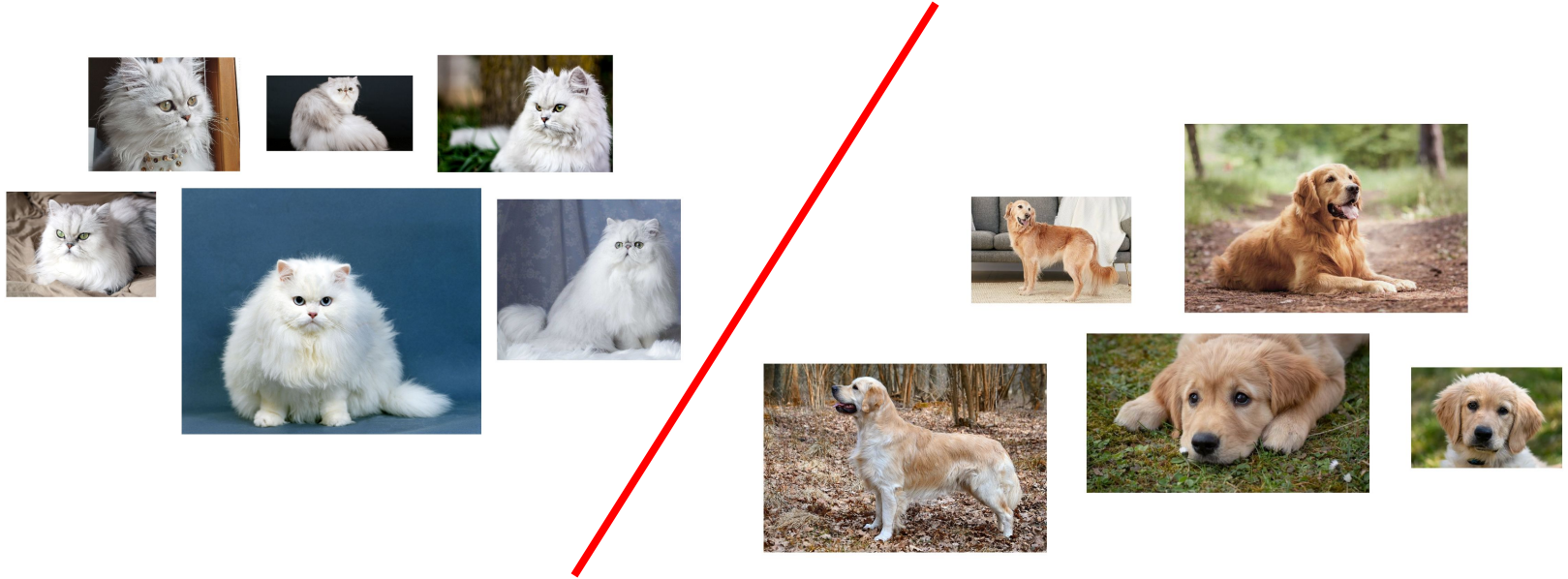


Overfitting (regression)

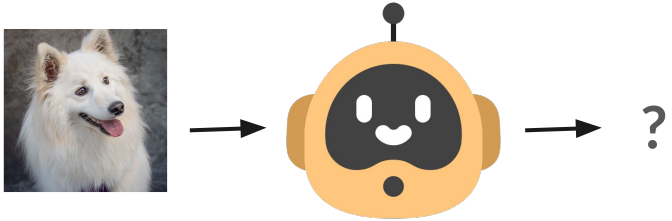




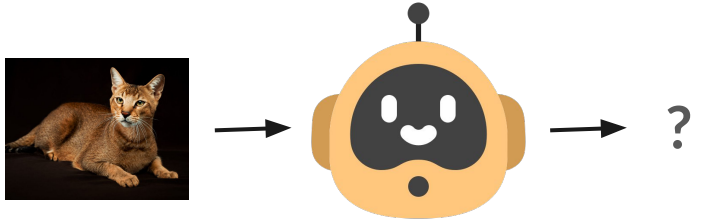
Generalization Capability



Generalization Capability



Generalization Capability



Modelli di Classificazione

- Esistono diversi modelli per risolvere problemi di classificazione, e ognuno di essi presenta dei learning algorithm peculiari

- Esempi di modelli di classificazione:
  - Alberi Decisionali
  - Naive Bayes
  - K-Nearest-Neighbor
  - Support Vector Machine

Shallow Learning

Alberi Decisionali

Alberi Decisionali

- Rappresenta una delle tecniche di classificazioni più classiche
- Permette di rappresentare con un albero un insieme di regole
- Spesso utilizzati per dati tabellari

Alberi Decisionali

features | classi

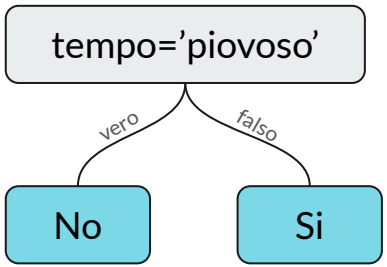
Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	No
Piovoso	20°	Alta	5 Km/h	Si
Nuvoloso	22°	Alta	12 Km/h	Si
Piovoso	15°	Normale	20 Km/h	No
Soleggiato	23°	Normale	15 Km/h	Si
Soleggiato	25°	Bassa	50 Km/h	No
Soleggiato	24°	Bassa	10 Km/h	Si
Piovoso	20°	Normale	45 Km/h	No

Alberi Decisionali

- Immaginiamo di effettuare un **test** su un attributo per discriminare gli esiti
- In particolare, notiamo che **quando il tempo è piovoso, quasi sempre l'esito è negativo**

Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	No
Piovoso	20°	Alta	5 Km/h	Si
Nuvoloso	22°	Alta	12 Km/h	Si
Piovoso	15°	Normale	20 Km/h	No
Soleggiato	23°	Normale	15 Km/h	Si
Soleggiato	25°	Bassa	50 Km/h	No
Soleggiato	24°	Bassa	10 Km/h	Si
Piovoso	20°	Normale	45 Km/h	No

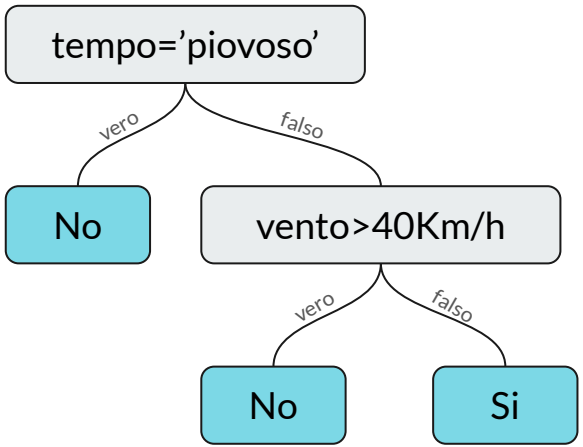
Alberi Decisionali



- Così facendo però commetteremmo due errori.
- Possiamo correggerli aggiungendo altri nodi all’albero

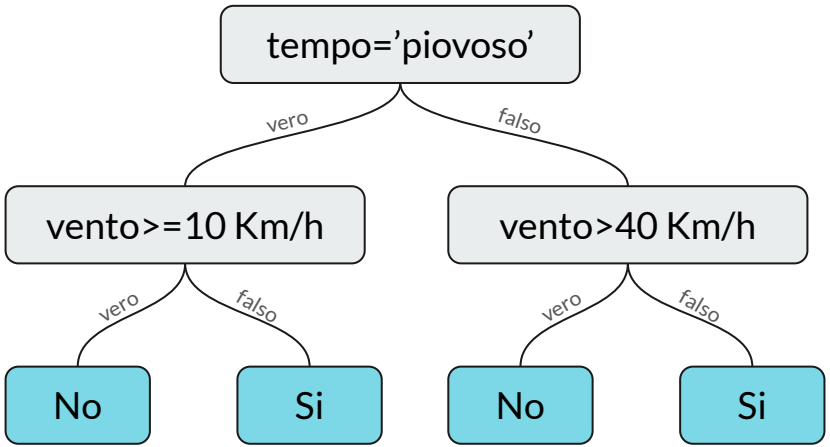
Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	No
Piovoso	20°	Alta	5 Km/h	Si
Nuvoloso	22°	Alta	12 Km/h	Si
Piovoso	15°	Normale	20 Km/h	No
Soleggiato	23°	Normale	15 Km/h	Si
Soleggiato	25°	Bassa	50 Km/h	No
Soleggiato	24°	Bassa	10 Km/h	Si
Piovoso	20°	Normale	45 Km/h	No

Alberi Decisionali



Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	No
Piovoso	20°	Alta	5 Km/h	Si
Nuvoloso	22°	Alta	12 Km/h	Si
Piovoso	15°	Normale	20 Km/h	No
Soleggiato	23°	Normale	15 Km/h	Si
Soleggiato	25°	Bassa	50 Km/h	No
Soleggiato	24°	Bassa	10 Km/h	Si
Piovoso	20°	Normale	45 Km/h	No

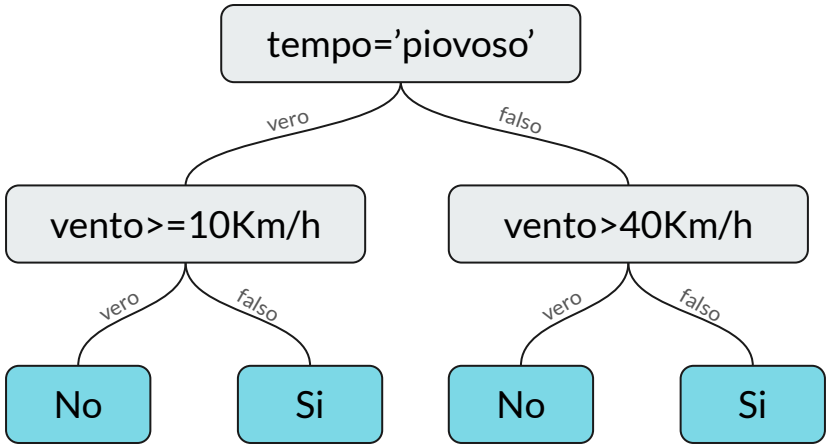
Alberi Decisionali



Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	No
Piovoso	20°	Alta	5 Km/h	Si
Nuvoloso	22°	Alta	12 Km/h	Si
Piovoso	15°	Normale	20 Km/h	No
Soleggiato	23°	Normale	15 Km/h	Si
Soleggiato	25°	Bassa	50 Km/h	No
Soleggiato	24°	Bassa	10 Km/h	Si
Piovoso	20°	Normale	45 Km/h	No

Alberi Decisionali

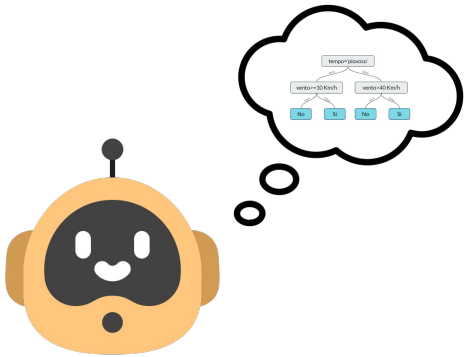
- I nodi interni rappresentano **test sugli attributi**
- Ogni ramo è una possibile **risposta** al test (vero o falso)
- Ogni nodo foglia è una **classe**
- Ogni percorso rappresenta una **regola di classificazione**



Induzione dell'albero

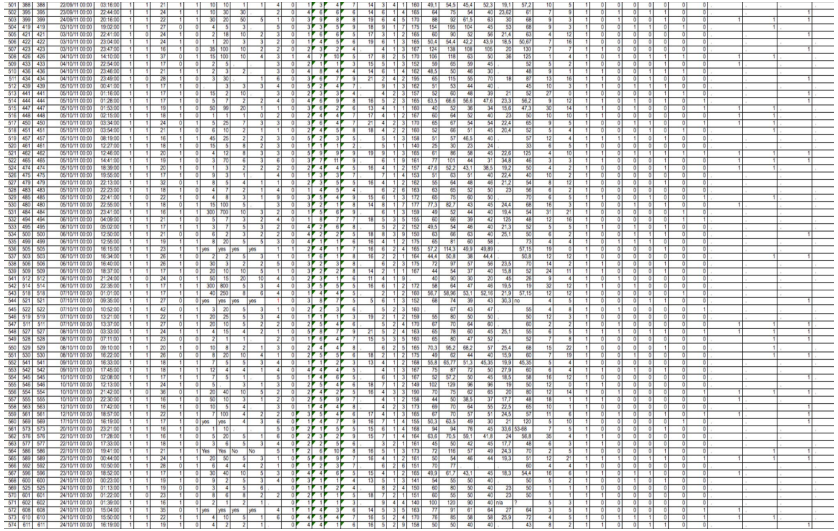
- Trovare una serie di regole per costruire un albero può sembrare facile a prima vista…

Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	No
Piovoso	20°	Alta	5 Km/h	Si
Nuvoloso	22°	Alta	12 Km/h	Si
Piovoso	15°	Normale	20 Km/h	No
Soleggiato	23°	Normale	15 Km/h	Si
Soleggiato	25°	Bassa	50 Km/h	No
Soleggiato	24°	Bassa	10 Km/h	Si
Piovoso	20°	Normale	45 Km/h	No



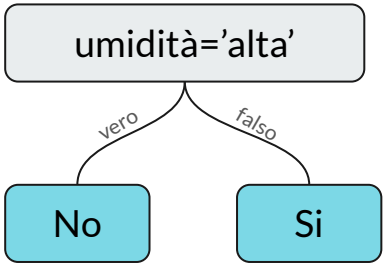
Induzione dell'albero

- Trovare una serie di regole per costruire un albero può sembrare facile a prima vista
- Ma purtroppo lo spazio di ricerca esponenziale rispetto al numero di features
- Pertanto, è richiesto l'utilizzo di **metodi euristici**



Induzione dell'albero

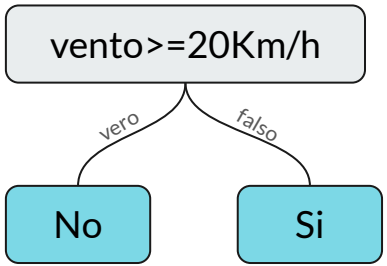
- Se scegliessimo di discriminare tramite l'attributo 'umidità' non avremmo una partizione ottimale…



Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	No
Piovoso	20°	Alta	5 Km/h	Si
Nuvoloso	22°	Alta	12 Km/h	Si
Piovoso	15°	Normale	20 Km/h	No
Soleggiato	23°	Normale	15 Km/h	Si
Soleggiato	25°	Bassa	50 Km/h	No
Soleggiato	24°	Bassa	10 Km/h	Si
Piovoso	20°	Normale	45 Km/h	No

Induzione dell'albero

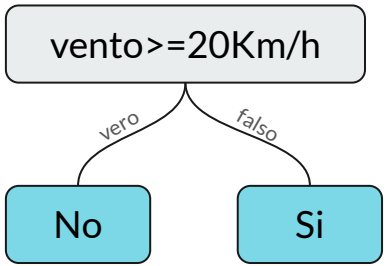
- Se scegliessimo di discriminare tramite l'attributo 'umidità' non avremmo una partizione ottimale…
- Ma se scegliessimo l'attributo 'vento' avremmo una miglior discriminazione



Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	No
Piovoso	20°	Alta	5 Km/h	Si
Nuvoloso	22°	Alta	12 Km/h	Si
Piovoso	15°	Normale	20 Km/h	No
Soleggiato	23°	Normale	15 Km/h	Si
Soleggiato	25°	Bassa	50 Km/h	No
Soleggiato	24°	Bassa	10 Km/h	Si
Piovoso	20°	Normale	45 Km/h	No

Induzione dell'albero

- Se scegliessimo di discriminare tramite l'attributo 'umidità' non avremmo una partizione ottimale…
- Ma se scegliessimo l'attributo 'vento' avremmo una miglior discriminazione



Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	No
Piovoso	20°	Alta	5 Km/h	Si
Nuvoloso	22°	Alta	12 Km/h	Si
Piovoso	15°	Normale	20 Km/h	No
Soleggiato	23°	Normale	15 Km/h	Si
Soleggiato	25°	Bassa	50 Km/h	No
Soleggiato	24°	Bassa	10 Km/h	Si
Piovoso	20°	Normale	45 Km/h	No

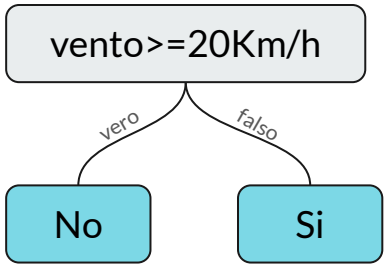
Scegliere come radice l'attributo capace di discriminare meglio le classi

Algoritmi di Induzione dell'Albero

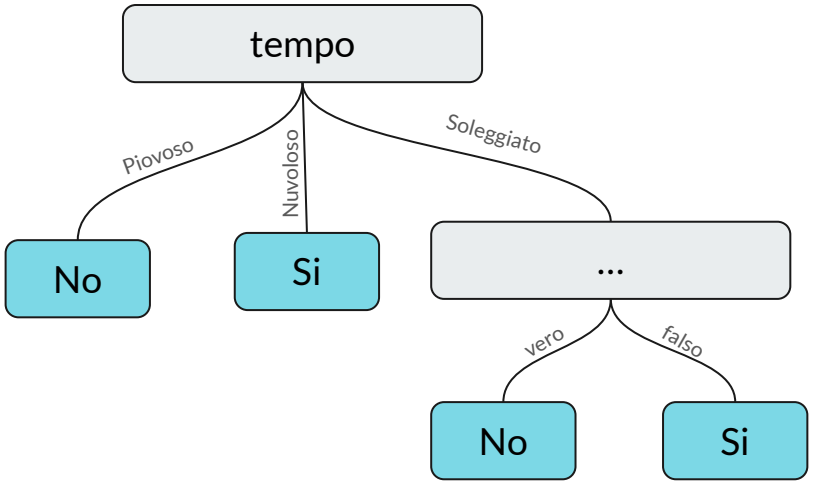
- Hunt
- Cart
- ID3 / C4.5
- SLIQ / SPRINT
- …

Come partizionare

- Binary Split



- Multi-Way Split



Bontà di uno split

Sì: 4 – No: 4

Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	**No**
Piovoso	20°	Alta	5 Km/h	**Si**
Nuvoloso	22°	Alta	12 Km/h	**Si**
Piovoso	15°	Normale	20 Km/h	**No**
Soleggiato	23°	Normale	15 Km/h	**Si**
Soleggiato	25°	Bassa	50 Km/h	**No**
Soleggiato	24°	Bassa	10 Km/h	**Si**
Piovoso	20°	Normale	45 Km/h	**No**

Sì: 2 – No: 1

Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	**No**
Piovoso	20°	Alta	5 Km/h	**Si**
Nuvoloso	22°	Alta	12 Km/h	**Si**

Sì: 2 – No: 3

Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	15°	Normale	20 Km/h	**No**
Soleggiato	23°	Normale	15 Km/h	**Si**
Soleggiato	25°	Bassa	50 Km/h	**No**
Soleggiato	24°	Bassa	10 Km/h	**Si**
Piovoso	20°	Normale	45 Km/h	**No**

Bontà di uno split

Sì: 4 – No: 4

Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	**No**
Piovoso	20°	Alta	5 Km/h	**Si**
Nuvoloso	22°	Alta	12 Km/h	**Si**
Piovoso	15°	Normale	20 Km/h	**No**
Soleggiato	23°	Normale	15 Km/h	**Si**
Soleggiato	25°	Bassa	50 Km/h	**No**
Soleggiato	24°	Bassa	10 Km/h	**Si**
Piovoso	20°	Normale	45 Km/h	**No**

Sì: 4 – No: 1

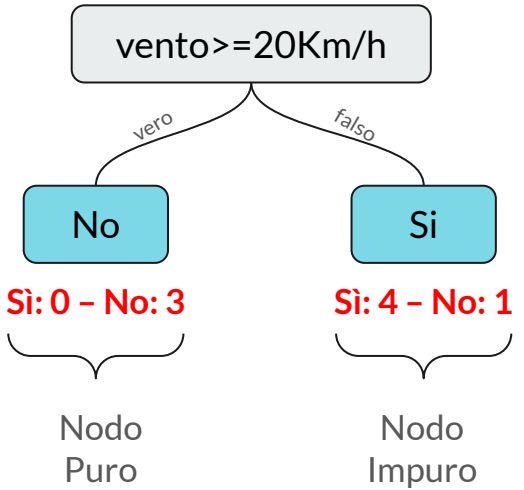
Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	**No**
Piovoso	20°	Alta	5 Km/h	**Si**
Nuvoloso	22°	Alta	12 Km/h	**Si**
Soleggiato	23°	Normale	15 Km/h	**Si**
Soleggiato	24°	Bassa	10 Km/h	**Si**

Sì: 0 – No: 3

Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	15°	Normale	20 Km/h	**No**
Soleggiato	25°	Bassa	50 Km/h	**No**
Piovoso	20°	Normale	45 Km/h	**No**

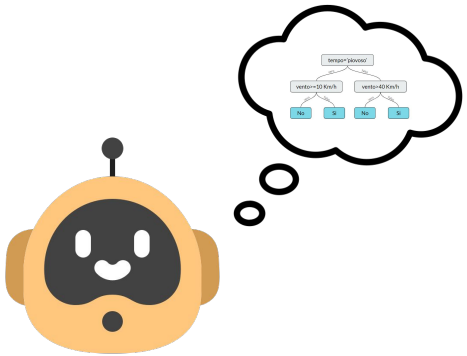
Purezza di un nodo

- **Massima purezza**: tutti i records alla stessa classe
- **Minima purezza**: records equamente distribuiti
- Misure di impurezza:
  - **Indice di Gini**
  - **Entropia**
  - **Classification error**
- Per valutare la bontà dello split occorre confrontare l'impurezza del padre contro i figli (misura chiamata **guadagno**)



Criterio d'arresto

- Le misure di impurezza tendono a creare alberi con molti split…
- Infatti, con molti split prima o poi si ottengono nodi foglie tutte appartenenti alla stessa classe
  - Possibile soluzione: esprimere tutto in forma binaria
  - Oppure, pesare sul numero di split con il **gain ratio**
- Criteri d'arresto:
  - Tutti i sample di ogni split appartengono alla stessa classe
  - **early termination**



Alberi Decisionali: altre note da incorporare

- Decision Region
- Oblique tree?
- Forests?
- Pruning?

Classificatori Bayesiani

Probabilità

- Qual è la probabilità di **ottenere 6** lanciando un dado a sei facce non truccato?

P(X=6)=1/6

- E la probabilità di **pescare un asso di quadri** in un mazzo di 52 carte non truccato?

P(X=1 ♦)=1/52

Probabilità... Condizionata

- Qual è la probabilità di **ottenere 6** lanciando un dado a sei facce non truccato, **sapendo che è uscito un numero pari**?

P(X=6 | X è pari)=1/3

- E la probabilità di **pescare un asso di quadri** in un mazzo di 52 carte non truccato, **sapendo che il seme è rosso**?

P(X=1 ♦ | seme è rosso)=1/26

Probabilità - Idea per il Machine Learning

- Qual è la probabilità di **assegnare una determinata classe** dato un insieme di features?

P(Escursione=Sì|Tempo='piovoso', Temperatura=18, Umidita='alta', Vento=10Km/h)=???

Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	???

- Possiamo esprimere (in questo caso) delle classi seguendo delle regole probabilistiche come per un lancio di un dado o l'estrazione di una carta da un mazzo?

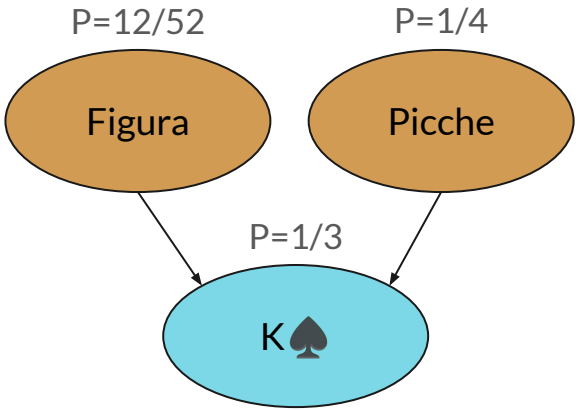
Approccio Bayesiano

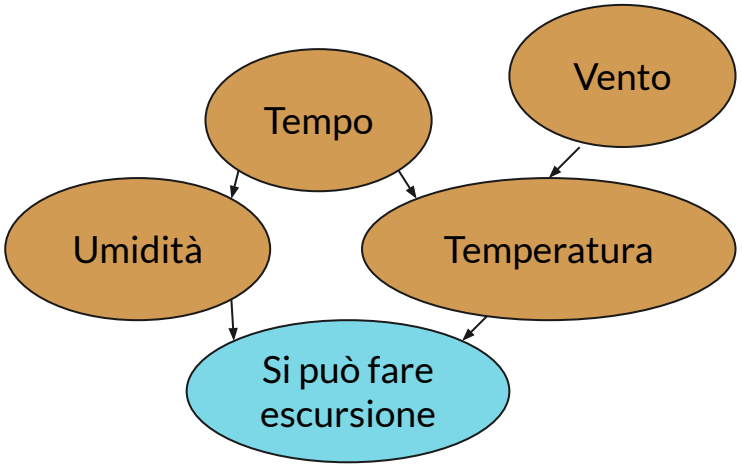
- Attraverso regole definite sulla base della probabilità classica (a partire dagli assiomi di Kolmogorov) possiamo arrivare a definire la probabilità di eventi senza la necessità di doverli misurare
- Esempio: Qual è la probabilità di estrarre un K sapendo di aver estratto picche?
- Non serve ripetere l'esperimento N# volte (approccio **frequentista**), possiamo usare le regole della probabilità (approccio **bayesiano**)

$$P(N=K \mid S=\spadesuit) = P(N=K \text{ AND } S=\spadesuit) / P(S=\spadesuit)$$

Rete Bayesiana

- Qual è la probabilità di **estrarre una figura** lanciando un dado a sei facce non truccato, **sapendo che il seme è rosso**?



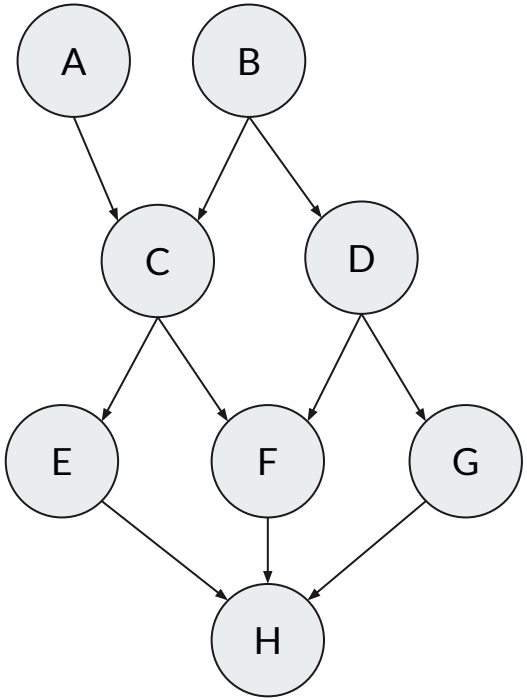


- Qual è la probabilità di avere una determinata classe sulla base di un insieme di features?

Rete Bayesiana

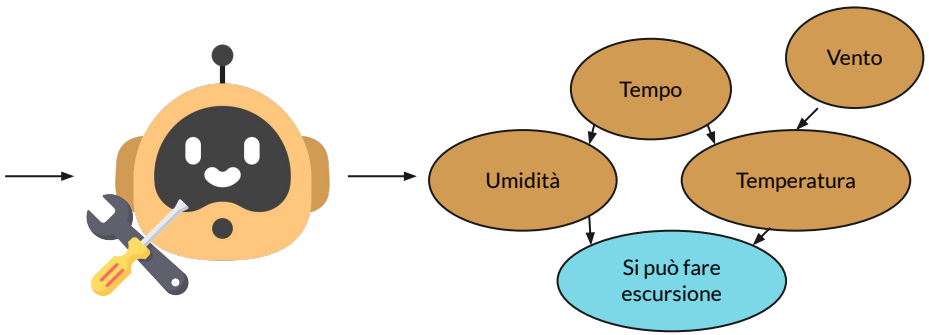
- Se però le probabilità sono articolare, e le relazioni molto complesse, può essere dispendioso il calcolo delle probabilità congiunte…

- Ma grazie alle regole delle probabilità alcune dipendenze possono essere semplificate: ogni nodo dipende solo dai propri discendenti (**Local Markov Property**)



Costruire un Classificatore con una Rete Bayesiana

Tempo	Temperatura	Umidità	Vento	Si può fare escursione?
Piovoso	18°	Alta	10 Km/h	**No**
Piovoso	20°	Alta	5 Km/h	**Si**
Nuvoloso	22°	Alta	12 Km/h	**Si**
Piovoso	15°	Normale	20 Km/h	**No**
Soleggiato	23°	Normale	15 Km/h	**Si**
Soleggiato	25°	Bassa	50 Km/h	**No**
Soleggiato	24°	Bassa	10 Km/h	**Si**
Piovoso	20°	Normale	45 Km/h	**No**



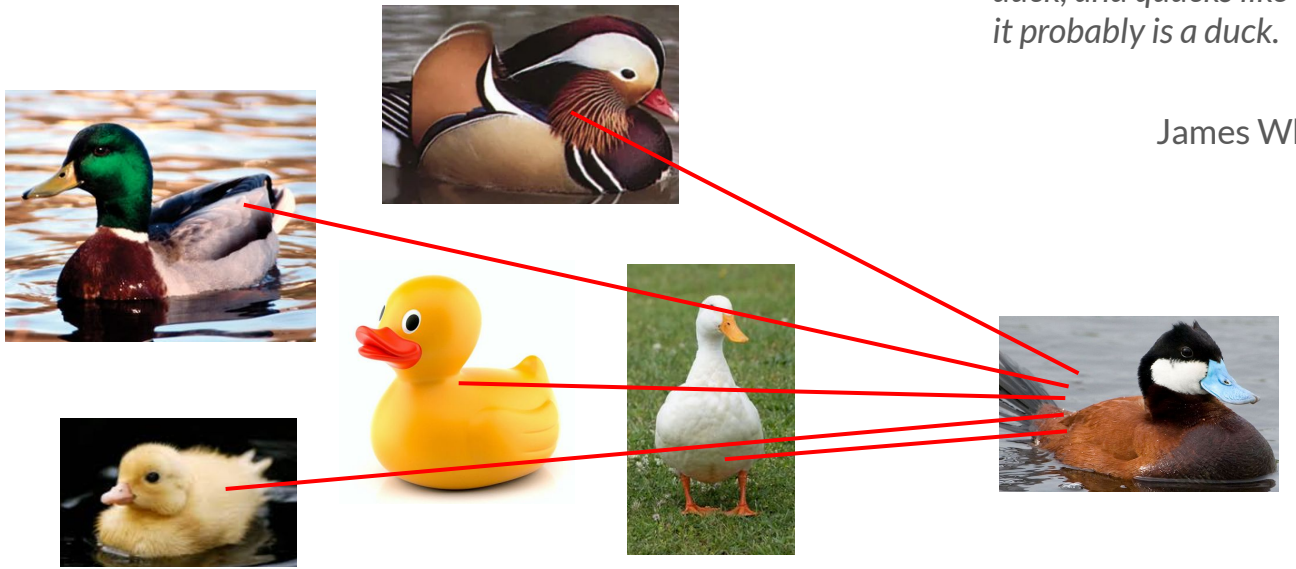
Naive Bayes

- In fase di training occorre:
  - Definire struttura della rete
  - Stimare valori della tabella di probabilità per ogni nodo
- Vista la complessità, risolviamo con un algoritmo naive: **Naive Bayes**
- **Assunzione: tutti gli attributi sono condizionalmente indipendenti dato il valore della classe**
- Occorre semplicemente definire delle regole per stimare la probabilità sugli attributi (a seconda del tipo) e sulla classe, stimandole dal training set
- In fase di inferenza basterà usare la regola di bayes sfruttando dell'indipendenza degli attributi:

$$P(\text{classe} \mid f1, f2, \ldots, fn) = P(f1, f2, \ldots, fn \mid \text{classe})\, P(\text{classe})$$

K-Nearest-Neighbor (KNN)

K-Nearest-Neighbor

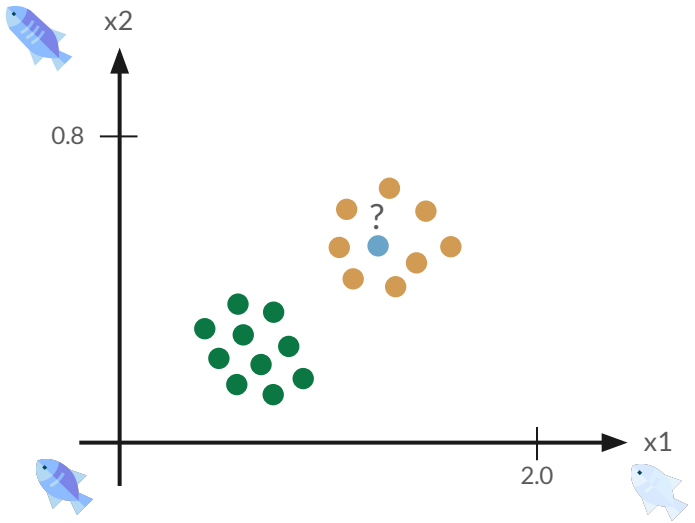


If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.

James Whitcomb Riley (1849–1916)

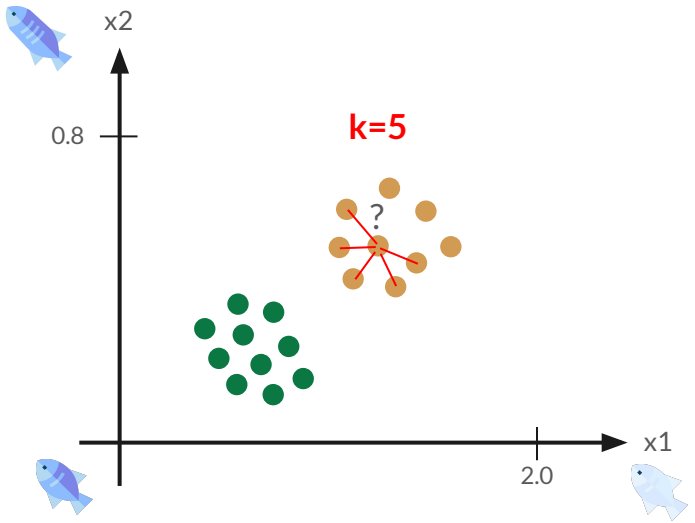
K-Nearest-Neighbor

- Il K-Nearest-Neighbor (KNN) è un algoritmo di classificazione basato su istanze
- Si basa sul concetto di “prossimità” (nearest neighbor = “il vicino più vicino”)



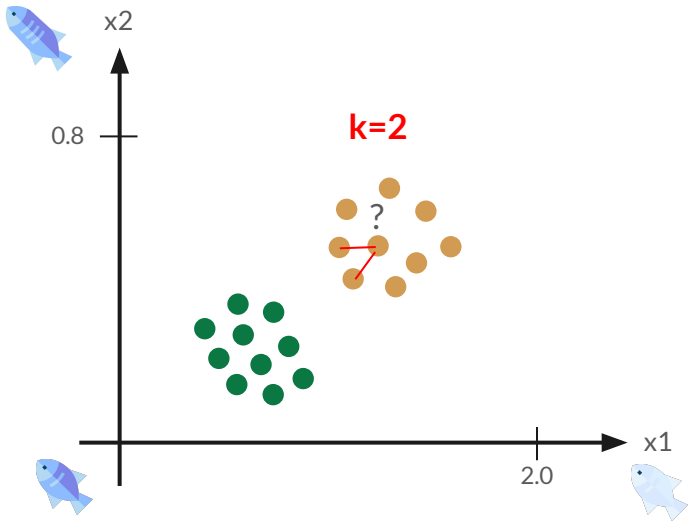
K-Nearest-Neighbor

- Il K-Nearest-Neighbor (KNN) è un algoritmo di classificazione basato su istanze
- Si basa sul concetto di “prossimità” (nearest neighbor = “il vicino più vicino”)
- La classificazione avviene sulla base delle **k** istanze simili



K-Nearest-Neighbor

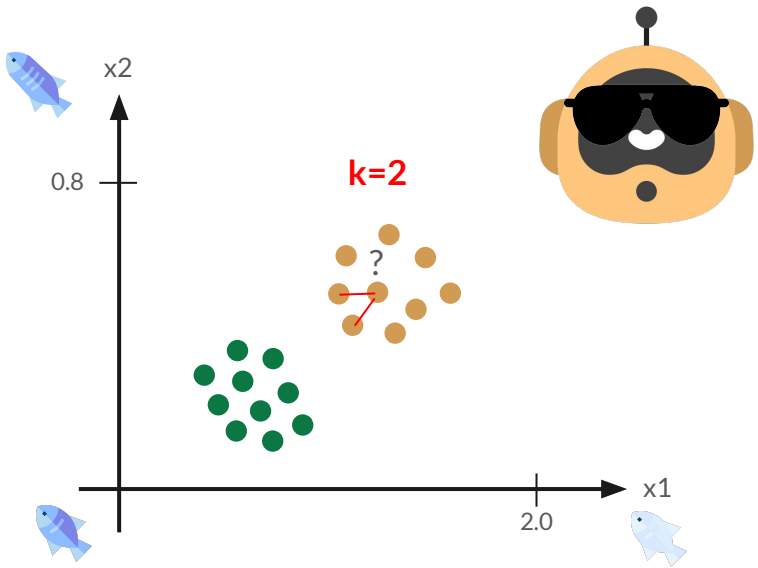
- Il K-Nearest-Neighbor (KNN) è un algoritmo di classificazione basato su istanze
- Si basa sul concetto di “prossimità” (nearest neighbor = “il vicino più vicino”)
- La classificazione avviene sulla base delle **k** istanze simili



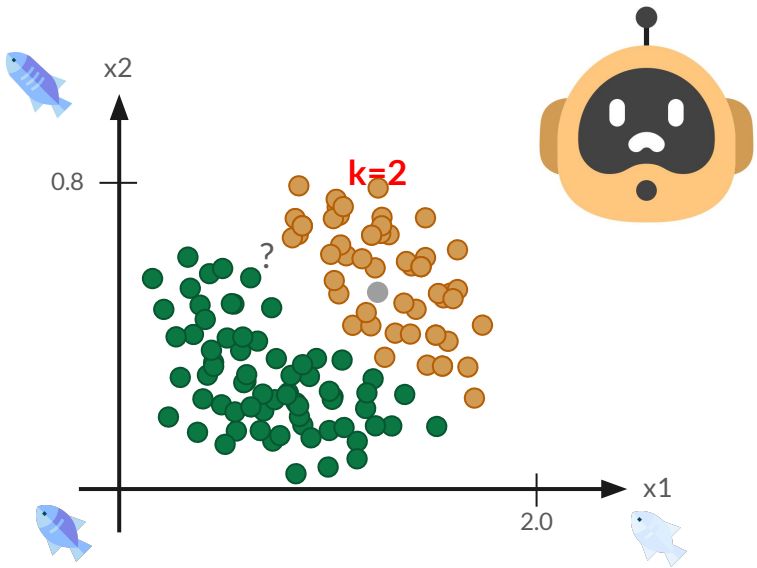
Lazy Learner 

- **Non c’è un vero e proprio “apprendimento”**, bensì una memorizzazione: il modello memorizza le istanze, e le utilizza poi in fase di inferenza. Pertanto viene chiamato **“pigro” (lazy)**
- E’ un modello non-parametrico
- Sebbene sia “veloce” (nullo) il tempo di apprendimento, **l’inferenza scala sulla base della grandezza del dataset**

Lazy Learner



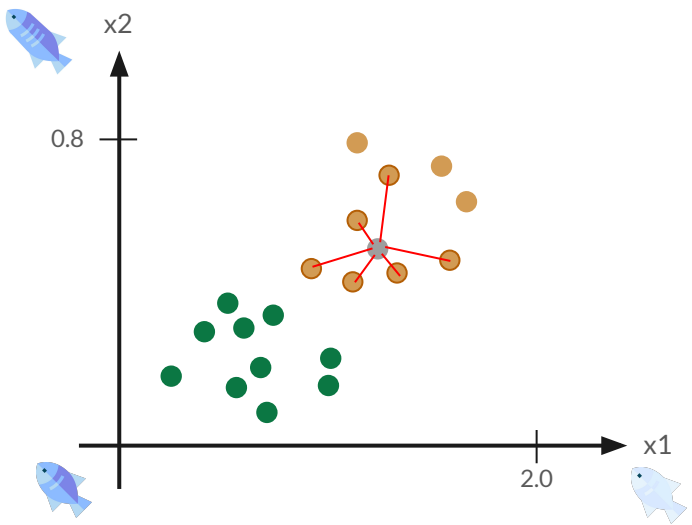
Numero di controlli per trovare i primi due vicini: **~19**



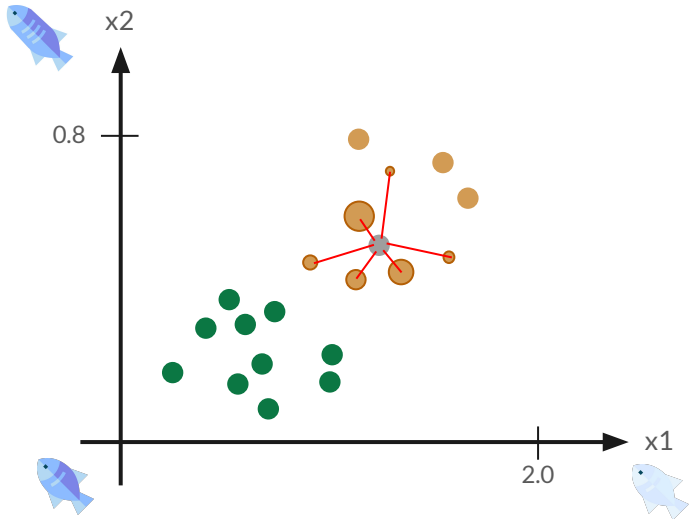
Numero di controlli per trovare i primi due vicini: **~200**

Pesare i contributi dei samples

Majority Voting



Distance Weight Voting



Support Vector Machines

Support Vector Machines

- Tecnica impiegata con successo ancora oggi (2024)
- Concepita per binario ma estendibile a piu classi
- Linear SVM / Non Linear SVM (???????) Kernel
- Idea: miglior iperpiano separatore
- Support Vectors + Margine
- Maximum Margin Hyperplane
- Moltiplicatori di Lagrange
- Soft Margin / Hard Margin
- slack variables
- Kernel Non lineare + Kernel Trick
- Kernel Polinomiale
- Kernel RBF
- Multiclass SVM

Support Vector Machines

- Sono uno strumento inventato dal matematico Vladmir Vapnik negli anni ‘60 (con successive estensioni del lavoro negli anni ‘90)
- Ottimi anche per problemi **non lineari**
- **Ottimizzazione Convessa / Quadratic Programming**
- Rappresentano un importante strumento allo stato dell’arte, e sfruttano **forti interpretazioni matematiche**…

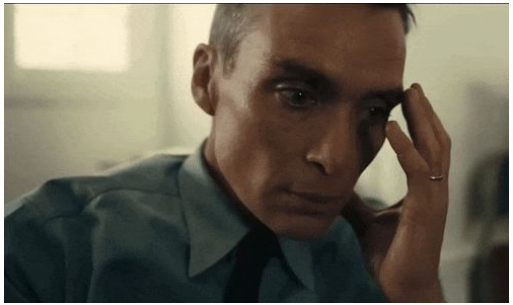
Molte “forti” interpretazioni matematiche…

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\{y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i$$

$$\sum_{i=1}^{\ell} y_i \alpha_i \left(\mathbf{x}_i \cdot \mathbf{x}\right) + b = 0$$

$$\frac{\partial L_P}{\partial w_\nu} = w_\nu - \sum_i \alpha_i y_i x_{i\nu} = 0 \qquad \frac{\partial L_P}{\partial b} = -\sum_i \alpha_i y_i = 0 \qquad \frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0$$

$$\min_{\mathbf{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t. } y_i f(\mathbf{x}_i) \geq +1, \ \forall i$$



$$L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i \left[y_i\left(\mathbf{w}\cdot\mathbf{x}_i + b\right) - 1\right]$$

$$\max_{\alpha_1,\ldots,\alpha_\ell} \min_{\mathbf{w},b} L\left(\alpha_1,\ldots,\alpha_\ell,\mathbf{w},b\right)$$
$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1,\ldots,N$$

$$\text{minimise} \quad \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to} \quad y_i f(\mathbf{x}_i) \geq 1, \quad i = 1,\ldots,N$$

$$\max_{\boldsymbol{\alpha}} \ \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i \alpha_i \, \mathbf{x}_i^T \mathbf{x}_j \alpha_j y_j$$
$$\text{s.t.} \ \ 0 \leq \alpha_i \leq C, \ \forall i$$
$$\sum_i \alpha_i y_i = 0, \forall i$$

$$L_D(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i \alpha_i \, \mathbf{x}_i^T \mathbf{x}_j \alpha_j y_j$$

$$\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$
$$\text{s.t. } y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \ \forall i$$
$$\xi_i \geq 0, \ \forall i$$

$$\max_{\boldsymbol{\alpha}} \ \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i \alpha_i \, \mathbf{x}_i^T \mathbf{x}_j \alpha_j y_j$$
$$\text{s.t.} \ \ \alpha_i \geq 0, \ \forall i$$
$$\sum_i \alpha_i y_i = 0, \forall i$$

Ma altrettanto “intuitive”!

- In questo seminario ci limiteremo a spiegare i concetti base dietro al significato di questa elegante tecnica
- Esistono svariate librerie che implementano già magistralmente queste tecniche (es. SciKit-Learn)

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i \alpha_i \boldsymbol{x}_i^T \boldsymbol{x}_j \alpha_j y_j$$

$$\text{s.t. } 0 \le \alpha_i \le C, \ \forall i$$

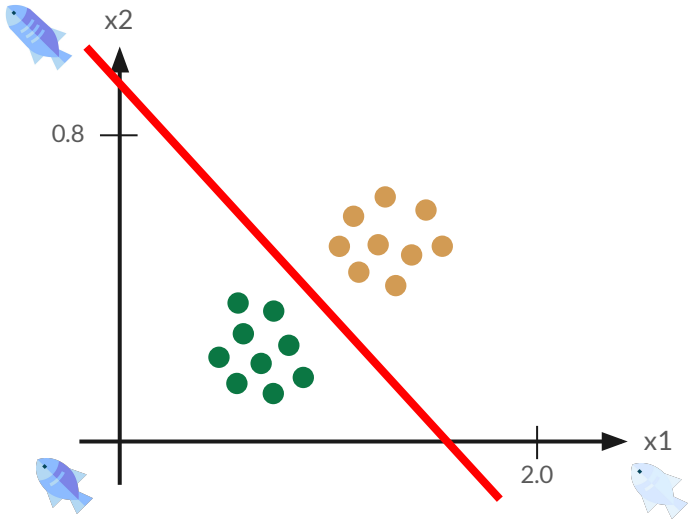
$$\sum_i \alpha_i y_i = 0$$

$$\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + C$$

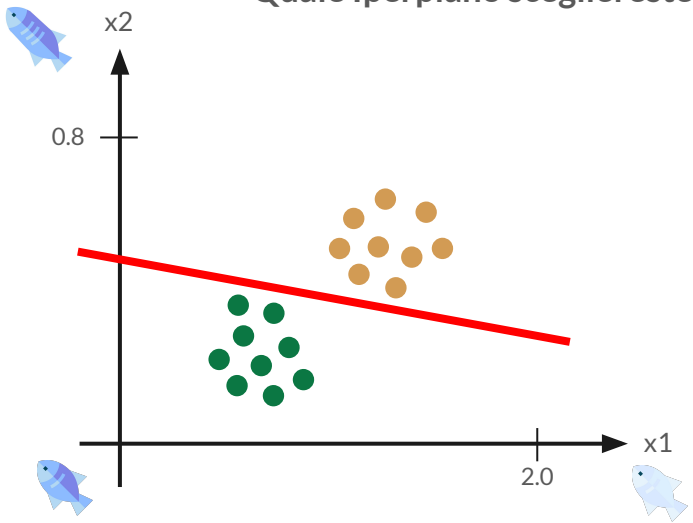
$$\text{s.t. } y_i f(\mathbf{x}_i) \ge 1 - \xi_i \ge 0, \ \forall i$$

$$L_D(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i \alpha_i$$

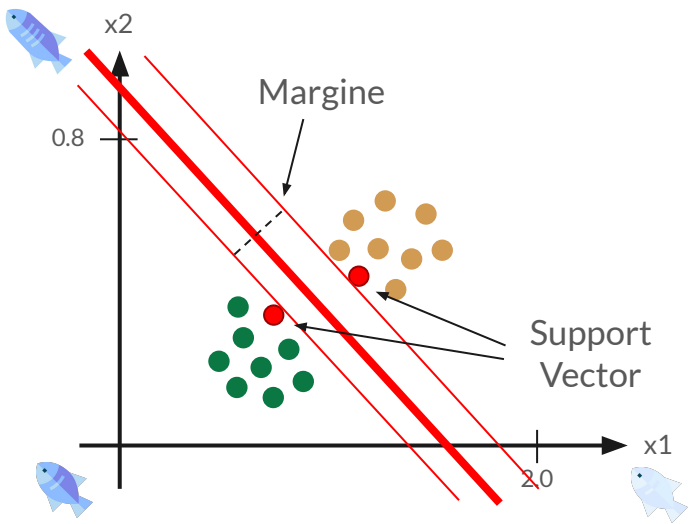
Support Vector Machines

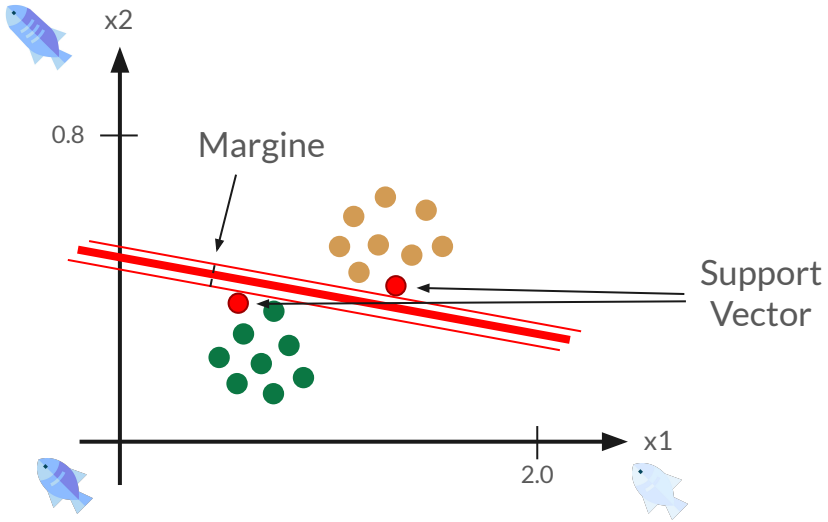


Quale Iperpiano sceglieresti?



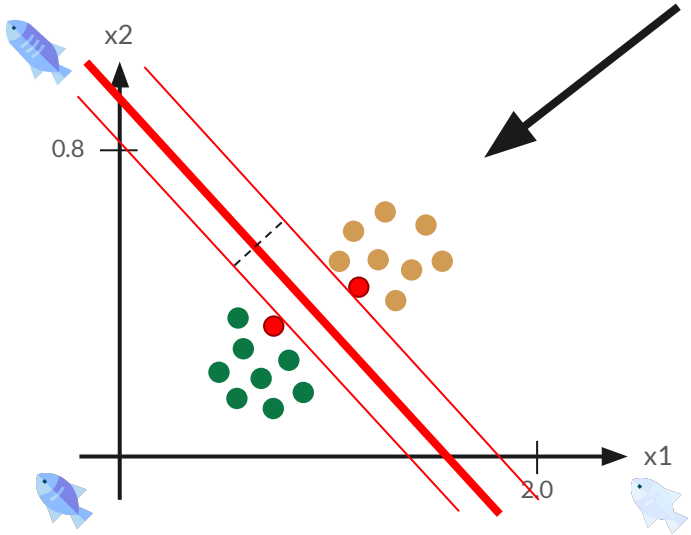
Support Vector Machines

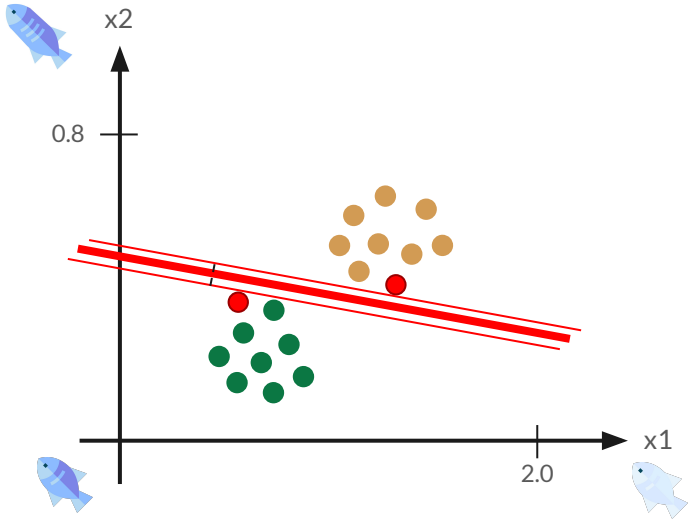




Support Vector Machines

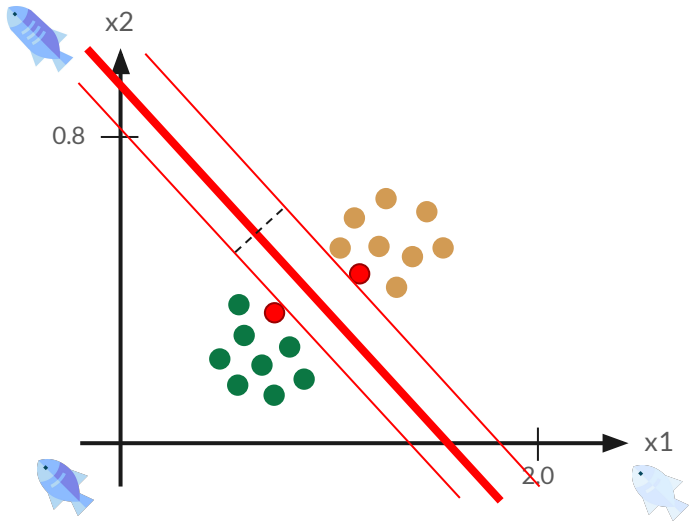
Mi conviene scegliere l'iperpiano con il margine maggiore possibile



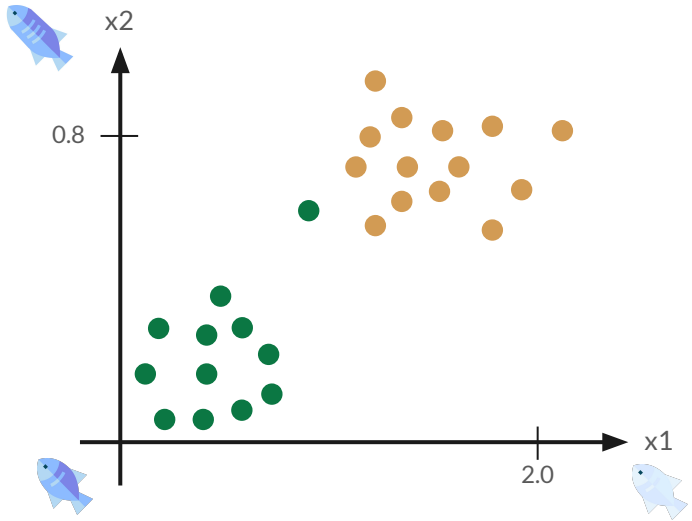


Support Vector Machines

- Trovare l'equazione del miglior iperpiano separatore coincide con il **risolvere un problema di ottimizzazione convessa**
- Esistono diversi solver automatici che riescono a farlo oggi

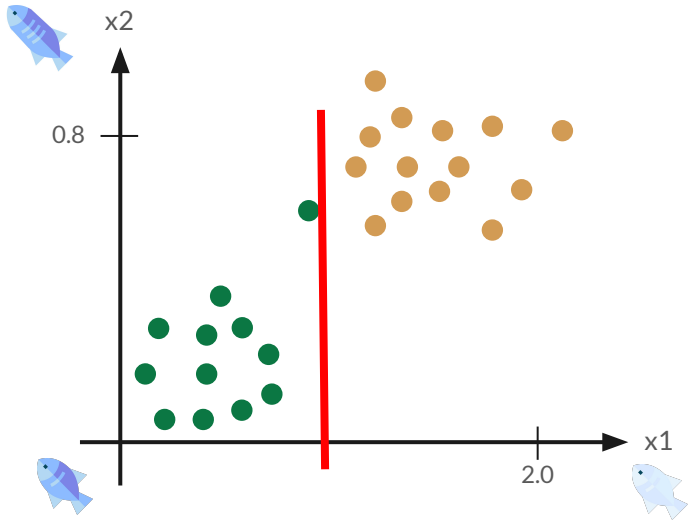


Support Vector Machines - Problema del rumore



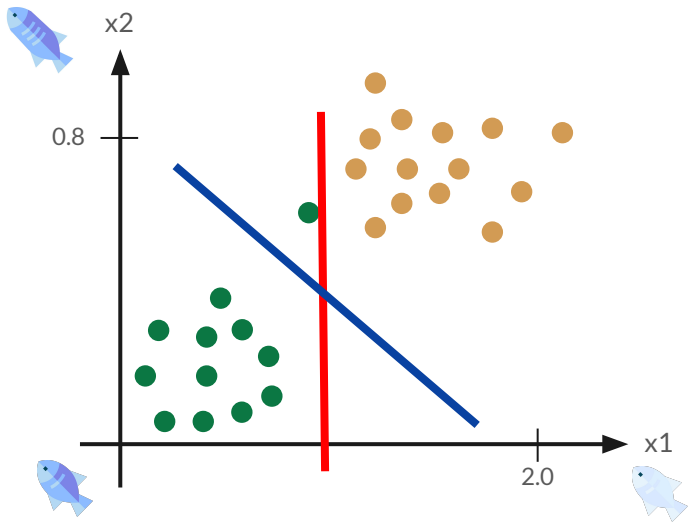
- Sfortunatamente, la tecnica descritta prima potrebbe creare delle situazioni spiacevoli…

Support Vector Machines - Problema del rumore



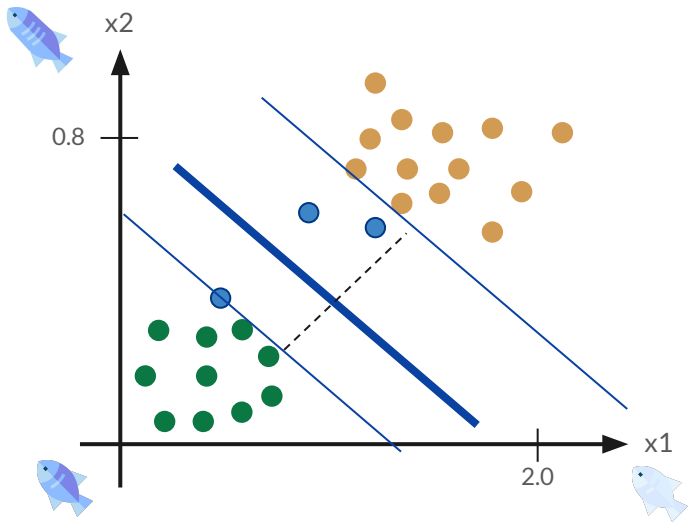
- Sfortunatamente, la tecnica descritta prima potrebbe creare delle situazioni spiacevoli…
- In una situazione del genere, otteniamo il miglior margine con l'**iperpiano rosso**…

Support Vector Machines - Problema del rumore



- Sfortunatamente, la tecnica descritta prima potrebbe creare delle situazioni spiacevoli…
- In una situazione del genere, otteniamo il miglior margine con l'**iperpiano rosso**…
- … ma è chiaro che si tratta di una soluzione causata da una **predizione rumorosa**, e sarebbe meglio usare l'**iperpiano blu**

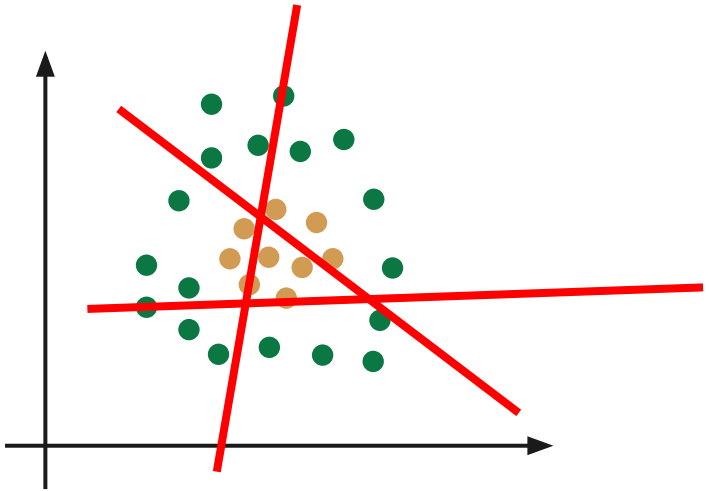
Support Vector Machines - Problema del rumore



- Sfortunatamente, la tecnica descritta prima potrebbe creare delle situazioni spiacevoli…
- In una situazione del genere, otteniamo il miglior margine con l'**iperpiano rosso**…
- … ma è chiaro che si tratta di una soluzione causata da una **predizione rumorosa**, e sarebbe meglio usare l'**iperpiano blu**
- In questo caso possiamo "addolcire" le condizioni dettate all'SVM, **tollerando una piccola porzione di errori (slack variables)**

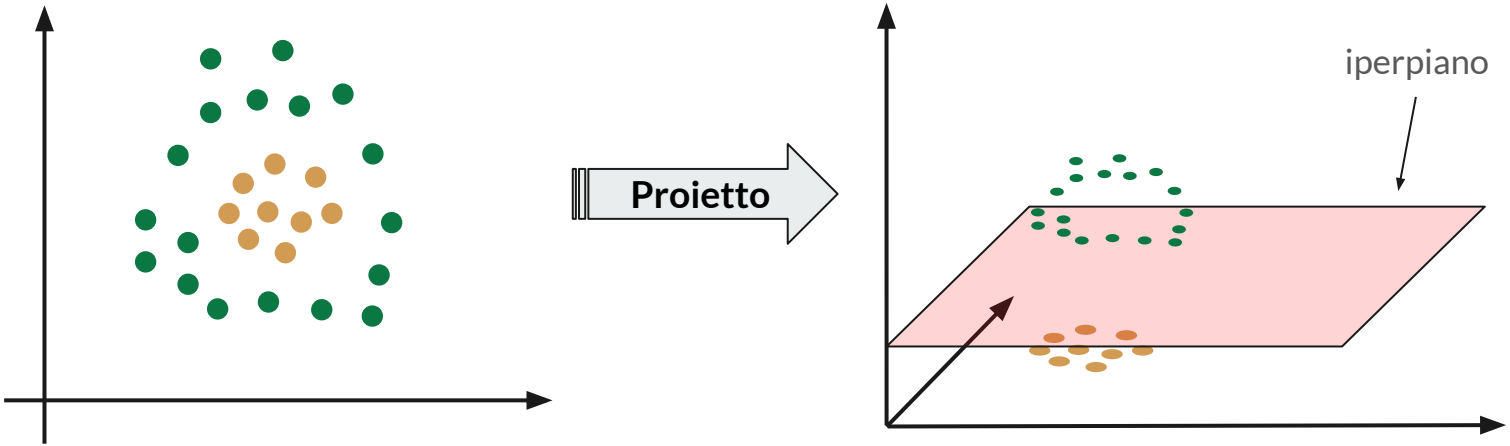
Support Vector Machines - Problemi non Lineari

- Avevamo detto che gli SVM sono utili anche per problemi non lineari...
- Ma come possono essere d'aiuto se sono capaci di creare solo iperpiani?

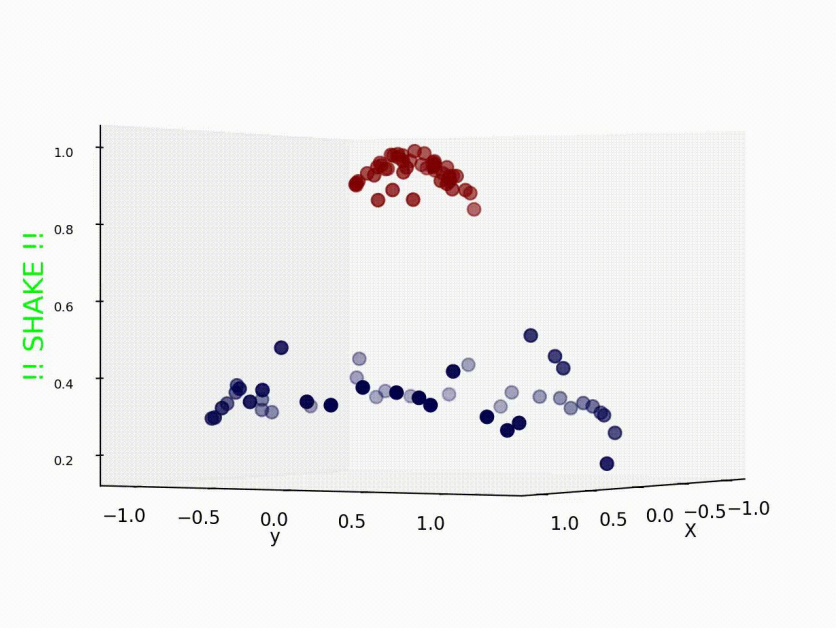


Kernel Trick

- Anziché separare i punti nel loro “spazio originale”, possiamo **proiettarli** in uno spazio a più alta dimensionalità, nel **quale esiste un iperpiano separatore**



Kernel Trick



- Nella dimensionalità originale i dati non sono linearmente separabili
- Ma in una più alta dimensionalità questi potrebbero esserlo
- Esistono vari “kernel”che ci permettono di proiettare i dati:
  - Polinomiali
  - Radial Basin Function (RBF)

Validazione e Overfitting