# Introduzione alle tecniche di AI

Lezione 2.a

Prof. Ing. Loris Penserini, PhD

https://orcid.org/0009-0008-6157-0396

# Artificial Intelligence: cosa è…

L'intelligenza artificiale (AI) è la tecnologia informatica, formata da modelli logici e tecniche computazionali complesse, che consente di simulare i processi dell'intelligenza umana attraverso l'applicazione di algoritmi.

Uno degli obiettivi dell'AI è quello di creare computer in grado di pensare e agire come gli esseri umani.

Per realizzare questo obiettivo sono necessari tre componenti chiave:

- **Sistemi di calcolo**
- **Sistemi per la gestione dei dati**
- **Algoritmi e strutture dati avanzati (software)**

# Machine Learning con Supervisione

esempi di applicazioni

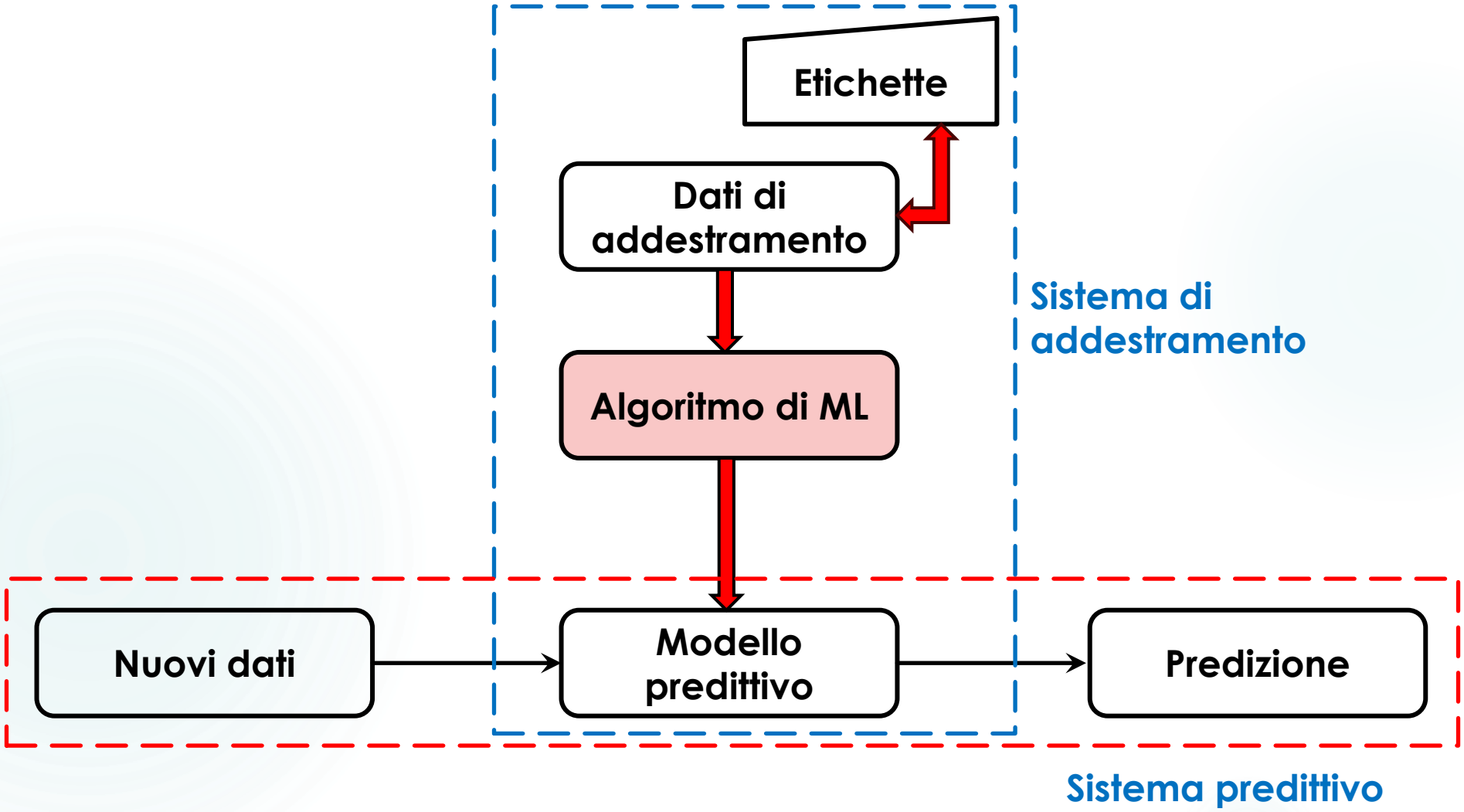
# ML con Supervisione

In questi sistemi di AI, lo scopo è quello di istruire un modello a partire da dati etichettati per rendere il sistema autonomo nel fare predizioni su dati mai visti prima o futuri.

Nel caso di filtraggio di messaggi di email, come spam oppure non-spam, si procede con un corpus di messaggi di addestramento già etichettati (spam/non-spam).

**Le tecniche di apprendimento con supervisione con etichette per classi discrete è chiamata anche compito di classificazione**.

# ML con Supervisione

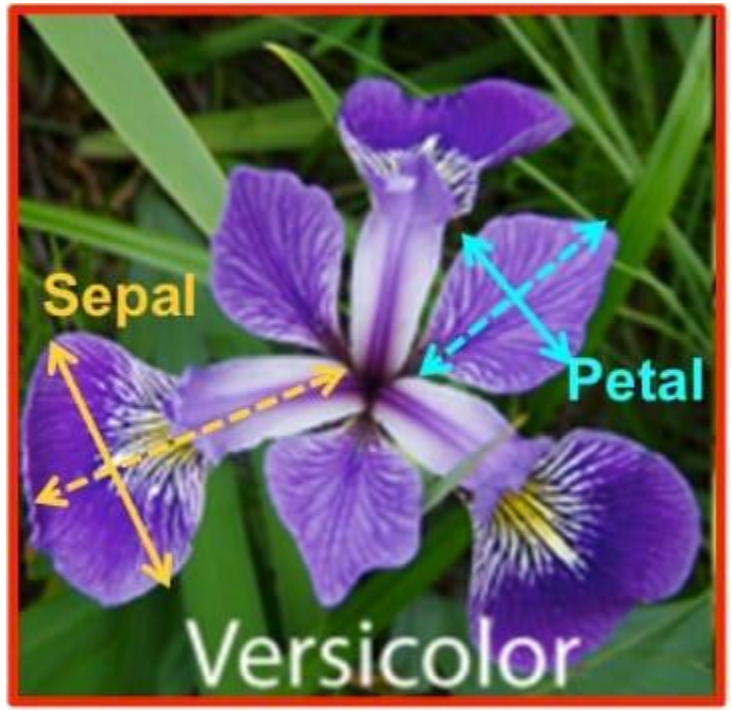


# Esempio: data set «Iris» [Raschka et al., 2020]

Costruiamo un classificatore di specie di Iris: Virginica, Setosa e Versicolor.

Il dataset contiene misurazioni di 150 iris delle tre specie elencate e raffigurate.



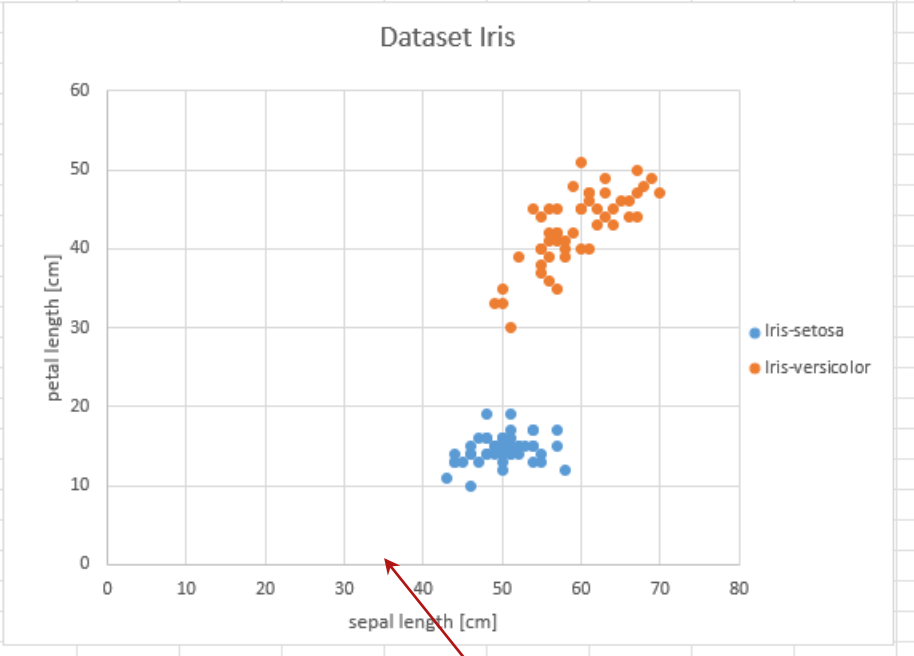


# Schema del data set «Iris» [Raschka et al., 2020]

Ciascun esempio di fiore rappresenta una riga del nostro dataset e le misurazioni del fiore in cm sono organizzate in colonne, che chiameremo le caratteristiche del dataset.

**Campioni**
(istanze, osservazioni)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Sepal length | Sepal width | Petal length | Petal width | Class label |
| 2 | 51 | 35 | 14 | 2 | Iris-setosa |
| 3 | 49 | 30 | 14 | 2 | Iris-setosa |
| 4 | 47 | 32 | 13 | 2 | Iris-setosa |
| 5 | 46 | 31 | 15 | 2 | Iris-setosa |
| 6 | 50 | 36 | 14 | 2 | Iris-setosa |
| 7 | 54 | 39 | 17 | 4 | Iris-setosa |
| 8 | 46 | 34 | 14 | 3 | Iris-setosa |
| 9 | 50 | 34 | 15 | 2 | Iris-setosa |
| 10 | 44 | 29 | 14 | 2 | Iris-setosa |
| 11 | 49 | 31 | 15 | 1 | Iris-setosa |
| 12 | 54 | 37 | 15 | 2 | Iris-setosa |
| 13 | 48 | 34 | 16 | 2 | Iris-setosa |
| 14 | 48 | 30 | 14 | 1 | Iris-setosa |
| 15 | 43 | 30 | 11 | 1 | Iris-setosa |
| 16 | 58 | 40 | 12 | 2 | Iris-setosa |
| 17 | 57 | 44 | 15 | 4 | Iris-setosa |
| 18 | 54 | 39 | 13 | 4 | Iris-setosa |
| 19 | 51 | 35 | 14 | 3 | Iris-setosa |
| 20 | 57 | 38 | 17 | 3 | Iris-setosa |
| 21 | 51 | 38 | 15 | 3 | Iris-setosa |
| 22 | 54 | 34 | 17 | 2 | Iris-setosa |




**Caratteristiche o features**
(attributi, misurazioni, dimensioni)

**Etichette della classe → target**

**Distribuzione degli esempi di fiori presenti nel dataset**

# Definire una Terminologia per il ML

E’ facile intuire che il ML è un campo molto ampio e altrettanto interdisciplinare, poiché utilizza nelle sue applicazioni tecniche e principi provenienti da altri settori di ricerca.

Per questa ragione molti termini e concetti usati nel ML potrebbero essere stati visti e studiati in altri settori.

**Per cui è bene definire e condividere una terminologia anche nel settore del ML.**

# Terminologia

***Esempio di addestramento***: rappresenta una istanza (o riga/record/campione) della tabella che definisce il dataset. Sinonimo di osservazione.

***Addestramento/training***: Durante il training, al modello viene fornito un insieme di dati di esempio, noto come training set (o *dataset*), e l'algoritmo di apprendimento utilizza questi dati per adattare i parametri interni del modello, con l'obiettivo di fare previsioni o prendere decisioni accurate quando verrà esposto a nuovi dati.

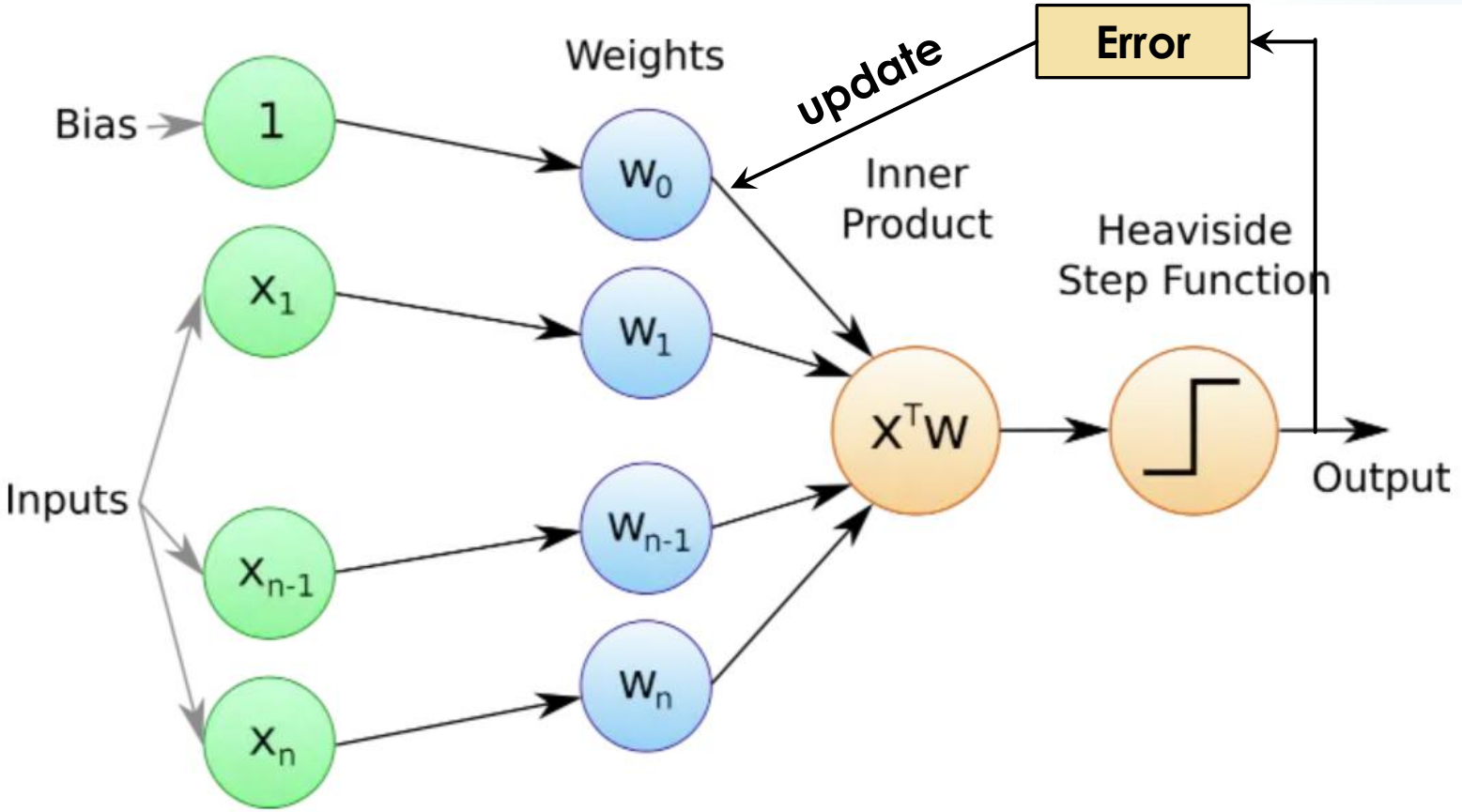
***Caratteristica/features***: si riferisce agli attributi o alle proprietà osservabili di un dato che vengono utilizzati per costruire un modello di apprendimento. Colonne o dimensioni nella tabella del dataset. Le features sono fornite come input a un algoritmo di machine learning e possono influenzare l'output o la previsione del modello. La scelta delle giuste features è cruciale per il successo di un modello di machine learning.

***Etichetta/Target***: si riferisce al valore o alla variabile che il modello deve predire o stimare. In altre parole, il target rappresenta l'uscita desiderata, la risposta corretta o la variabile dipendente in un problema di apprendimento supervisionato.

***Funzione di loss/di costo***: è una funzione matematica che misura quanto le previsioni di un modello siano lontane dai valori reali (o target). È uno degli elementi fondamentali per l'addestramento di un modello, poiché fornisce una metrica per valutare le prestazioni del modello.

# Classificazione con «Perceptron»

Il **perceptron** è un algoritmo di **classificazione binaria** di apprendimento supervisionato, originariamente sviluppato da Frank Rosenblatt nel 1957. Classifica i dati di input in uno di due stati separati sulla base di una procedura di addestramento eseguita su dati di input precedenti.



# Convenzioni notazionali

Useremo l'apice $\boldsymbol{i}$ per far riferimento all'i-esimo esempio di addestramento e il pedice $\boldsymbol{j}$ per far riferimento alla j-esima dimensione del dataset di addestramento. Lettere minuscole in grassetto per i vettori (es. $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{m}\times\boldsymbol{1}}$ ) e lettere maiuscole in grassetto per le matrici (es. $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{m}\times\boldsymbol{n}}$ ). In particolare, segue che:

$x_1^{(150)}$: fa riferimento alla prima dimensione dell'esempio/istanza e alla 150-esima riga del dataset di esempi di addestramento. Nell'esempio di Iris, è il fiore *150:lunghezza del sepalo*. Quindi, **ogni riga rappresenta un'istanza di** fiore e può essere scritta come vettore quadridimensionale:

$$\boldsymbol{x^{(i)}} \in \mathbb{R}^{\mathbf{1}\times\mathbf{4}} \quad \boldsymbol{x^{(i)}} = \left[\boldsymbol{x_1^{(i)}}\ \boldsymbol{x_2^{(i)}}\ \boldsymbol{x_3^{(i)}}\ \boldsymbol{x_4^{(i)}}\right]$$

Mentre ciascuna dimensione di caratteristiche è un vettore colonna 150-dimensionale,

$$\boldsymbol{x^{(i)}} \in \mathbb{R}^{\mathbf{150}\times\mathbf{1}} \quad \boldsymbol{x_j} = \begin{bmatrix} \boldsymbol{x_j^{(1)}} \\ \boldsymbol{x_j^{(2)}} \\ \dots \\ \boldsymbol{x_j^{(150)}} \end{bmatrix}$$

anche le variabili target: $\boldsymbol{y^{(i)}} \in \mathbb{R}^{\mathbf{150}\times\mathbf{1}}$ $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y_j^{(1)}} \\ \boldsymbol{y_j^{(2)}} \\ \dots \\ \boldsymbol{y_j^{(150)}} \end{bmatrix}$ con $\boldsymbol{y} \in$ {Setosa, Versicolor, Virginica}

# Formalizzazione del «Perceptron»

Per un determinato esempio/istanza $\boldsymbol{x^{(i)}}$ , si definisce una funzione decisionale $\boldsymbol{\Phi(z)}$ che prende una combinazione lineare di determinati valori di input **x** e un corrispondente valore di pesi **w**, dove **z** è il cosiddetto input della rete, per cui la somma ponderata:

$$z = w_0 \cdot x_0 + w_1 \cdot x_1 + \cdots + w_m \cdot x_m = \sum_{j=0}^{m} w_j \cdot x_j = w^T \cdot x$$

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w_0} \\ \cdots \\ \boldsymbol{w_m} \end{bmatrix}; \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_0} \\ \cdots \\ \boldsymbol{x_m} \end{bmatrix} \quad \boldsymbol{con} \quad \boldsymbol{w, x} \in \mathbb{R}^{(\boldsymbol{m+1})\times \boldsymbol{1}} \ \boldsymbol{vettori}$$

Se l'input di rete di un determinato esempio, $\boldsymbol{x^{(i)}}$ , è maggiore di una determinata soglia, **θ**, possiamo predire la classe 1 e viceversa la classe -1:

$$\boldsymbol{\Phi(z)} = \begin{cases} \boldsymbol{1} & \boldsymbol{if} \quad \boldsymbol{z \geq 0} \\ \boldsymbol{-1} & \boldsymbol{altrimenti} \end{cases}$$

# Apprendimento del «Perceptron»

Sia il neurone di McCulloch-Pitts sia il modello con soglia del perceptron di Rosenblatt si basano su una semplificazione che replica il funzionamento del neurone cerebrale, che può reagire o non reagire, cioè un **classificatore binario**. Per cui la **regola del neurone di Rosemblatt** è piuttosto semplice e si riflette sull'analogo algoritmo:

- **Inizializzare i pesi a 0 o a piccoli numeri casuali**
- **Per ogni esempio di addestramento, $x^{(i)}$:**
  - **Calcolare (predire) il valore di output, $\hat{y}$;**
  - **Aggiornare i pesi**

**In questo esempio l'output è l'etichetta della classe che è stata predetta dalla funzione a passo unitario (es. setosa o versicolor).**

# Aggiornamento con «Perceptron»

Simultaneo aggiornamento di ciascun peso $w_j$:

$$\boldsymbol{w_j} := \boldsymbol{w_j} + \triangle \boldsymbol{w_j}$$

$con\ \triangle w_j$ aggiornamento di $w_j$

L'aggiornamento è calcolato con la regola d'apprendimento del perceptron:

$$\triangle \boldsymbol{w_j} = \eta \cdot (y^{(i)} - \hat{y}^{(i)}) \cdot x_j^{(i)}$$

dove:

$\boldsymbol{\eta}$ è il tasso di apprendimento (in genere varia da 0 a 1);

$\boldsymbol{y^{(i)}}$ è l'etichetta effettiva della classe per l'i-esimo esempio di addestramento;

$\boldsymbol{\hat{y}^{(i)}}$ è l'etichetta della classe che è stata predetta.

# Esempio

Per un dataset bidimensionale l'aggiornamento sarà calcolato come segue:

$$\triangle \boldsymbol{w_0} = \eta \cdot (y^{(i)} - output^{(i)})$$

$$\triangle \boldsymbol{w_1} = \eta \cdot (y^{(i)} - output^{(i)}) \cdot x_1^{(i)}$$

$$\triangle \boldsymbol{w_2} = \eta \cdot (y^{(i)} - output^{(i)}) \cdot x_2^{(i)}$$

Per cui si vuol far notare che non si ricalcola l'etichetta predetta, prima che tutti i pesi siano stati aggiornati.

# Project Work

Utilizzando l'architettura del Perceptron, prima illustrata, formulare i due possibili scenari di: predizione corretta e di predizione errata evidenziando come variano i pesi.

Considerare:

- il tasso di apprendimento $\boldsymbol{\eta = 1}$
- valore di input $\boldsymbol{x_j^{(i)} = 0,5}$
- $\boldsymbol{\triangle w_j}$**??**

# PW: Regola di Apprendimento

Nei due scenari in cui il perceptron predice correttamente l'etichetta della classe, i pesi rimangono immutati, in quanto i valori di aggiornamento sono uguali a 0:

$$y^{(i)} = -1,\ \hat{y}^{(i)} = -1,\ \triangle \boldsymbol{w_j} = \eta \cdot (-1 - (-1)\ ) \cdot x_j^{(i)} = 0$$

$$y^{(i)} = 1,\ \hat{y}^{(i)} = 1,\ \triangle \boldsymbol{w_j} = \eta \cdot (1 - 1) \cdot x_j^{(i)} = 0$$

Al contrario, nel caso di una **predizione errata**:

$$y^{(i)} = 1,\ \hat{y}^{(i)} = -1,\ \triangle \boldsymbol{w_j} = \eta \cdot (1 - (-1)\ ) \cdot x_j^{(i)} = \eta \cdot (2) \cdot x_j^{(i)}$$

$$y^{(i)} = -1,\ \hat{y}^{(i)} = 1,\ \triangle \boldsymbol{w_j} = \eta \cdot (-1 - 1) \cdot x_j^{(i)} = \eta \cdot (-2) \cdot x_j^{(i)}$$

**Cioè quando avviene una predizione errata si introduce un fattore moltiplicativo che spinge l'esempio ad essere predetto meglio nella successiva iterazione predittiva.**

# PW: Applicazione della Regola

Nel caso:

$$y^{(i)} = -1, \quad \hat{y}^{(i)} = +1, \quad \eta = 1$$

Cioè stiamo classificando erroneamente, e supponiamo di avere: $x_j^{(i)} = 0{,}5$

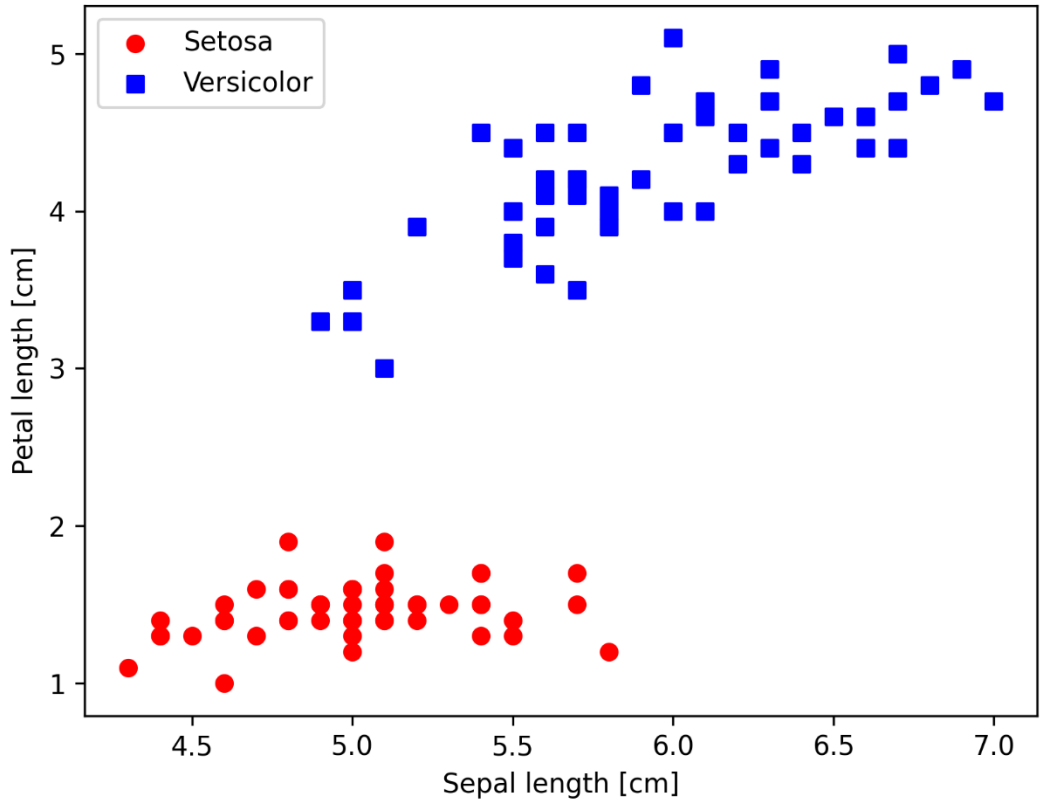
$$\triangle \boldsymbol{w_j} = \eta \cdot \big(1 - (-1)\big) \cdot x_j^{(i)} = 1 \cdot (2) \cdot 0{,}5 = 1$$

$$\boldsymbol{w_j} := \boldsymbol{w_j} + \triangle \boldsymbol{w_j}$$

Per cui il peso, $w_j$, è ora aggiornato con peso maggiore al precedente, per cui la prossima volta lo stesso esempio, $x_j$, sarà predetto con più probabilità.

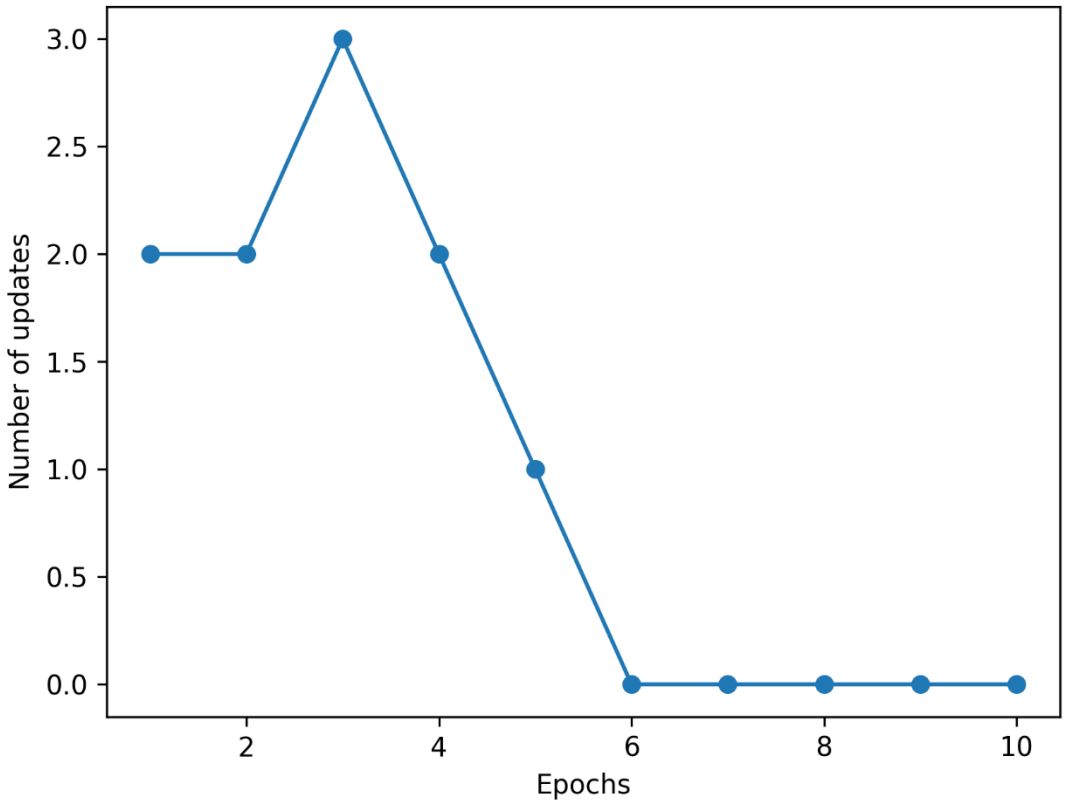
# Esempio con dataset Iris

Implementazione in Python del perceptron: caricamento e visualizzazione del dataset Iris



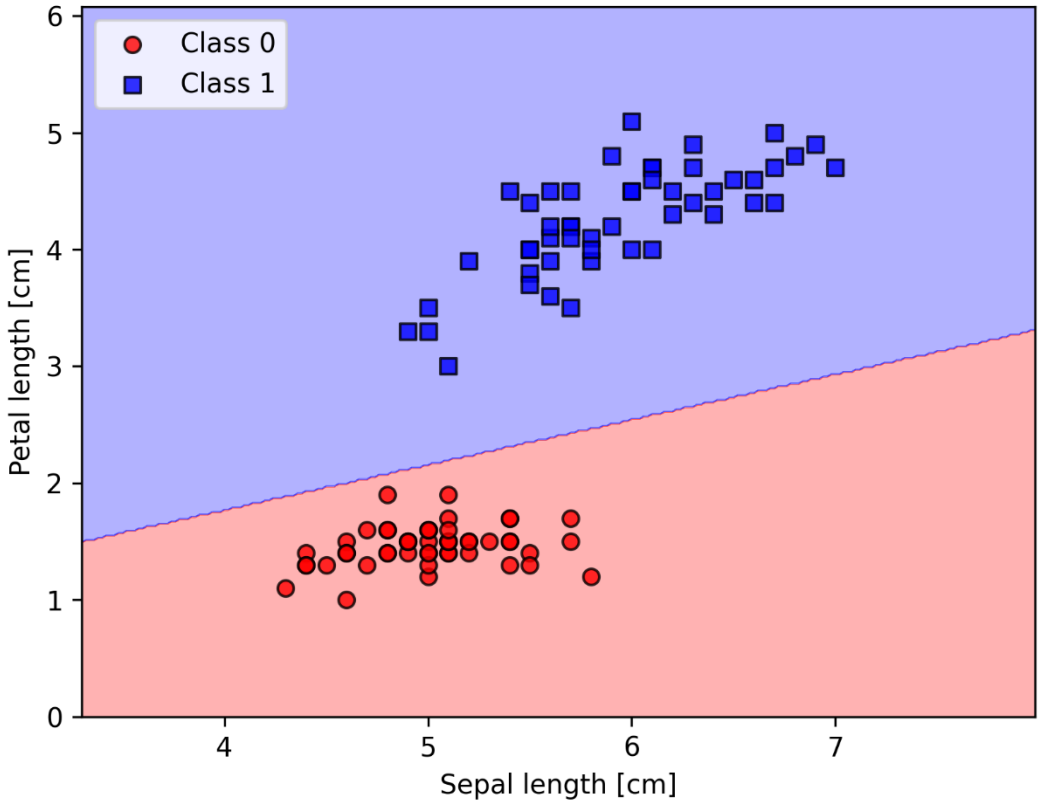
# Esempio con dataset Iris

Implementazione in Python del perceptron: l'errore si azzera dopo sei epoche

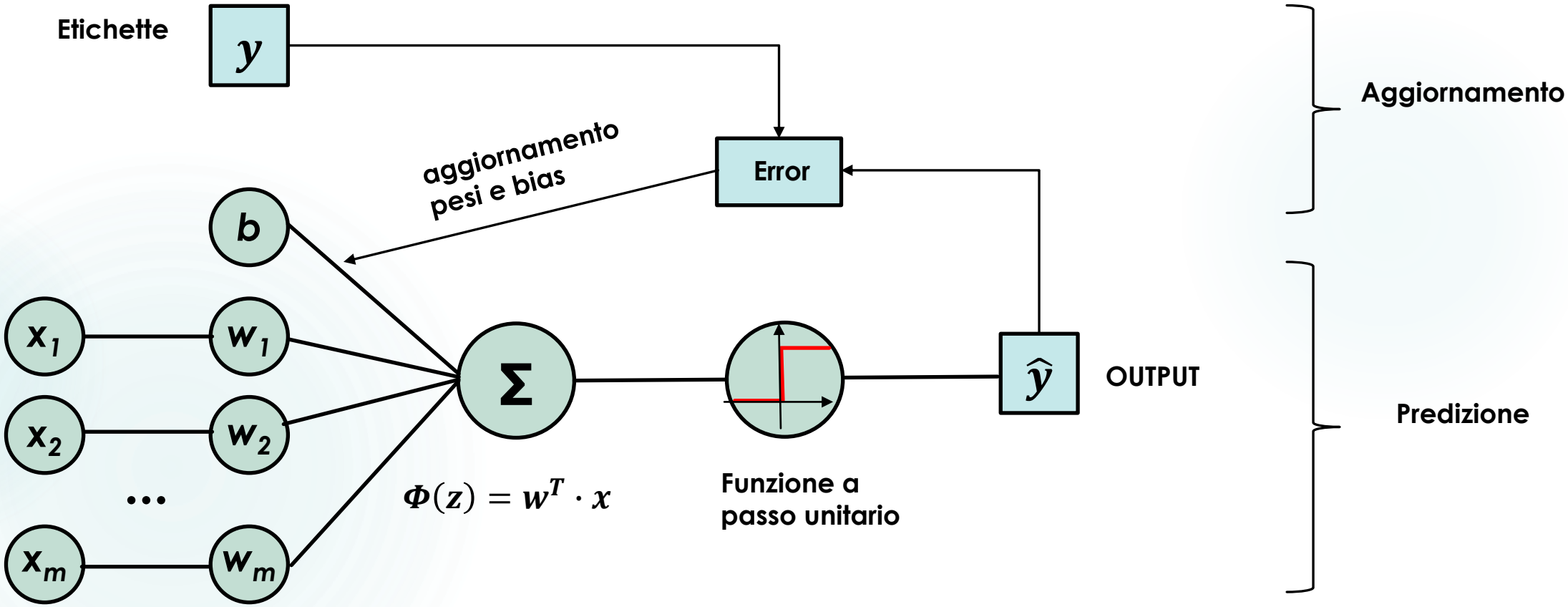


# Esempio con dataset Iris

Implementazione in Python del perceptron: risultato della classificazione (regioni decisionali)



# Esempio con dataset Iris



# Bibliografia

[Esteva et al., nature 2017] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun, ‘’Dermatologist-level classification of skin cancer with deep neural networks’’, Nature 542, 115–118 (2017). https://doi.org/10.1038/nature21056

[Panti et al., CoopIS-01] Maurizio Panti, Luca Spalazzi, Loris Penserini, "Cooperation Strategies for Information Integration", in Proc. of Sixth International Conference on Cooperative Information Systems (CoopIS 2001), Springer Verlag, LNCS 2172, Trento, Italy, September 5-7, 2001.

[Panti et al., IJCAI-01] Maurizio Panti, Luca Spalazzi, Loris Penserini, "A Distributed Case-Based Query Rewriting", in Proc. of 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Morgan Kaufmann Publishers, vol.2, p.1005-1010, Seattle, Washington, USA, August 4-10, 2001.

[Ridi, AIB studi 2020] Riccardo Ridi, "La piramide dell'informazione: una introduzione’’ , journal AIB studi, n. vol. 59/1-2, p.69-96, 2020. https://dx.doi.org/10.2426/aibstudi-12216

[Raschka et al., 2020] Sebastian Raschka, Vahid Mirjalili, Machine Learning con Python – Costruire algoritmi per generare conoscenza. Apogeo, 2020. ISBN 978-88-503-3524-4

[DeepMind] AlphaFold reveals the structure of the protein universe. https://deepmind.google/discover/blog/alphafold-reveals-the-structure-of-the-protein-universe/