# 14. Regressione lineare

Corso di Python per il Calcolo Scientifico

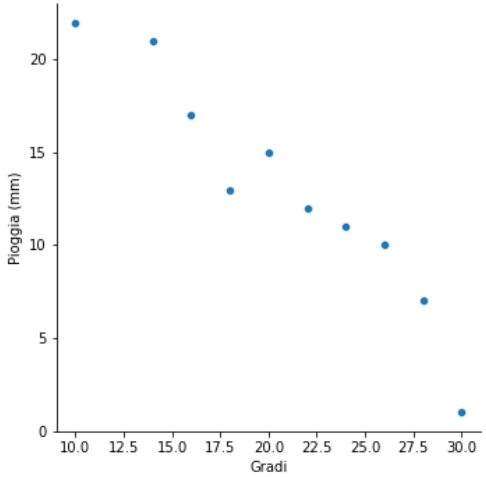
Angelo Cardellicchio
angelo.cardellicchio@stiima.cnr.it

# Outline

- La regressione lineare
- Addestramento e funzione di costo
- Un approccio iterativo

# La regressione lineare (1)

- Due variabili, ovvero pioggia e temperatura
- Queste si dispongono secondo il grafico a sinistra
- Appare evidente una relazione **lineare**: all'aumentare della temperatura, diminuiscono in maniera approssimativamente lineare i millimetri di pioggia
- Possiamo approssimare questa relazione mediante una **regressione lineare**

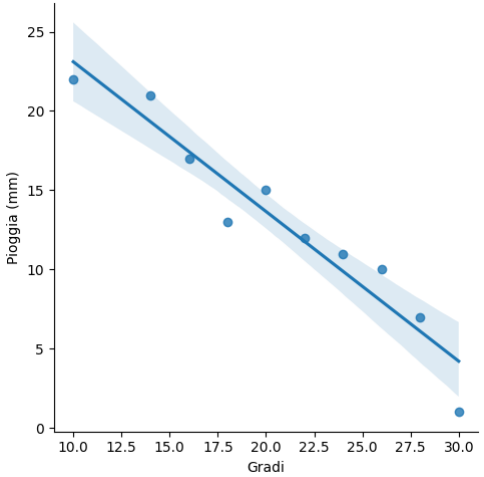


# La regressione lineare (2)

- La **regressione lineare** permette di definire una **retta di regressione**
- Ci consente di effettuare predizioni con un certo grado di confidenza su dati che non sono all'interno del dataset
- La sua espressione (nel caso univariato) è data da:

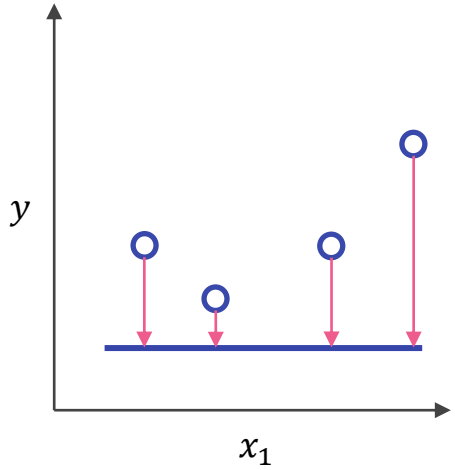
$$y' = b + w_1 x_1$$

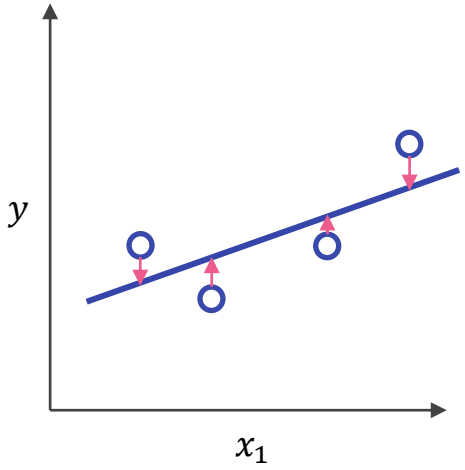
- Nella precedente $b$ è il **bias**, mentre $w_1$ è il **peso** della variabile indipendente
- Se avessimo più variabili indipendenti avremmo una regressione **multipla**, mentre con più variabili dipendenti abbiamo una **multivariata**



# Addestramento e funzione di costo

- L'addestramento ha come obiettivo quello di **minimizzare una funzione di costo**
- Nel caso della regressione, un possibile costo è lo scarto tra la retta di regressione ed i punti 'veri'
- Una possibile formulazione è data dall'errore quadratico medio:

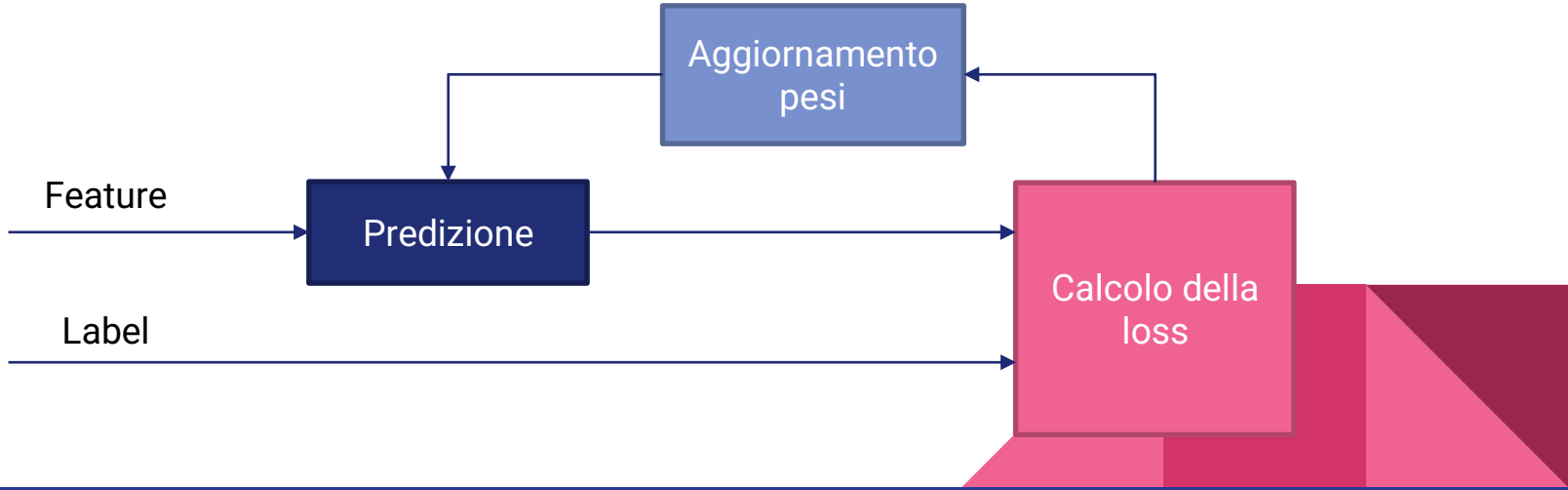



$$MSE = \frac{1}{N} \sum_{(x,y) \in D} (y - y')^2$$

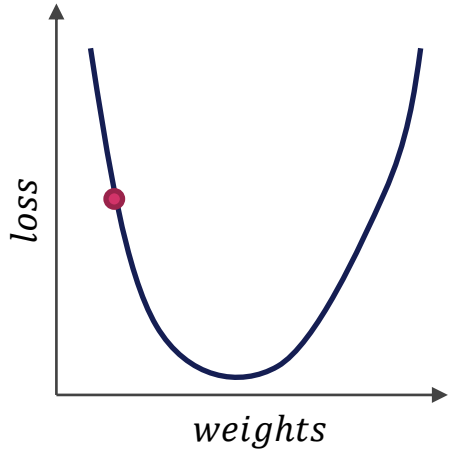
# Un approccio iterativo (1)

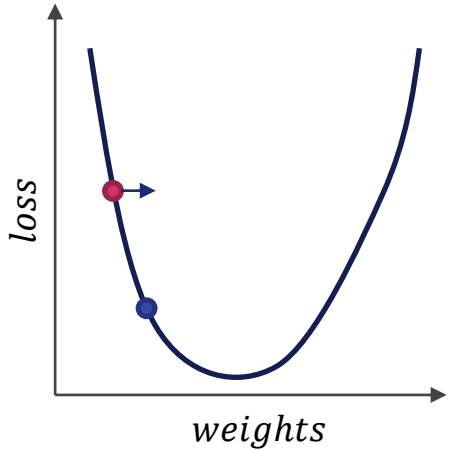
- L'addestramento prevede un approccio **iterativo**
- Ad ogni iterazione, il valore della funzione di costo viene **minimizzato**
- Per farlo, si utilizzano algoritmi di **ottimizzazione**

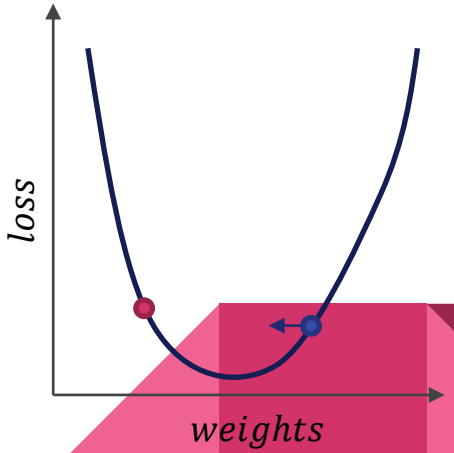


# Un approccio iterativo (2)

- L'obiettivo dell'ottimizzazione è cercare di trovare un insieme di parametri che minimizzi la funzione di costo
- Questo non è sempre vero: ad esempio, la regressione lineare utilizza il metodo dei minimi quadrati mediante la funzione **`scipy.linalg.lstsq`**!

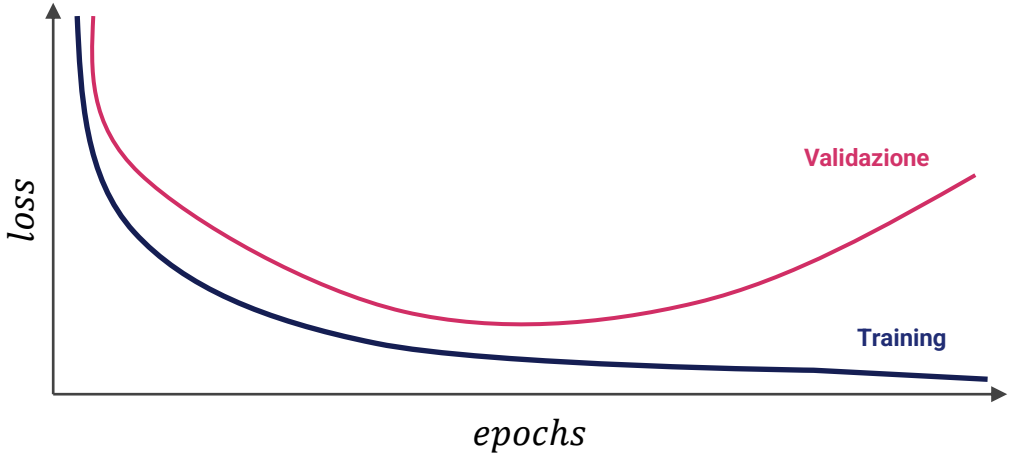


# Regolarizzazione

- La **regolarizzazione** agisce sulla complessità del modello
- In particolare, si fa in modo di preferire modelli con rapporti tra i termini **poco complessi**
- Per farlo, si minimizzano congiuntamente la loss ed il termine di regolarizzazione
- Molto usata è la **regolarizzazione $L_2$**




$$L_2 = w_1^2 + w_2^2 + \cdots + w_n^2$$

$$w_1 = 0.1, w_2 = 3, w_3 = 0.2 \Rightarrow$$
$$\Rightarrow L_2 = 0.01 + 9 + 0.04$$

# Regressione lineare in Scikit Learn

- In Scikit Learn la regressione lineare è implementata da oggetti di classe **LinearRegression()**
- Questi oggetti sono degli stimatori che usano l'algoritmo ai minimi quadrati per il fitting

```
import numpy as np
from sklearn.linear_model import LinearRegression

reg = LinearRegression()
data = np.array([[0, 0], [1, 1], [2, 2]])
reg.fit(data)
```

# Domande?

42