Regressione lineare come laboratorio interdisciplinare

Un percorso integrato tra informatica e matematica

Francesca Arrigoni · Francesca Voltolini

{francesca.arrigoni, francesca.voltolini}@liceodavincitn.it

Liceo Scientifico "Leonardo da Vinci", Trento

1 Introduzione e contesto

La regressione lineare è un terreno privilegiato per l'incontro tra matematica e informatica: a partire dal modello statistico intuitivo si arriva a introdurre alcuni concetti fondamentali del machine learning.

Il **percorso interdisciplinare** è stato progettato da una docente di matematica e una di informatica e è stato sperimentato in una classe quarta del Liceo Scientifico opzione Scienze Applicate nell'a.s. 2025-26.

Obiettivo: rendere visibile la continuità tra intuizione geometrica, formalizzazione algebrica e implementazione computazionale.

2 Fase 1 — Informatica

INFORMATICA

Due tutorial Google Colab, metodologie differenti.

Predict | Run | Investigate | Modify | Make

Matplotlib — metodo PRIMM: dal semplice `plot()` alla personalizzazione dei grafici, fino a un esercizio autonomo.

Pandas — schema Domanda-Risposta-Esercizio sul dataset *iris*: `head()`, `info()`, `describe()`, filtri e statistiche descrittive.

Insieme costituiscono la base tecnica: Pandas per caricare e preparare i dati, Matplotlib per visualizzarli.

3 Fase 1 — Matematica

MATEMATICA

Esplorazione

Simulazione PhET (Univ. of Colorado Boulder): apprendimento per esplorazione e scoperta.

Validazione

Gli studenti ricostruiscono l'applet in GeoGebra da una nuvola di punti assegnata.

Dimostrazione

Vincolo del passaggio per il baricentro: la somma dei residui quadratici dipende solo da m ed è una parabola.

Istituzionalizzazione

Discussione guidata sul criterio dei minimi quadrati.

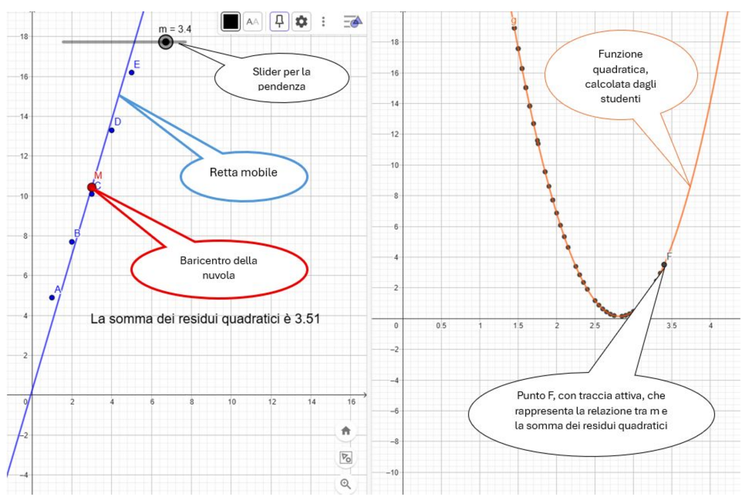


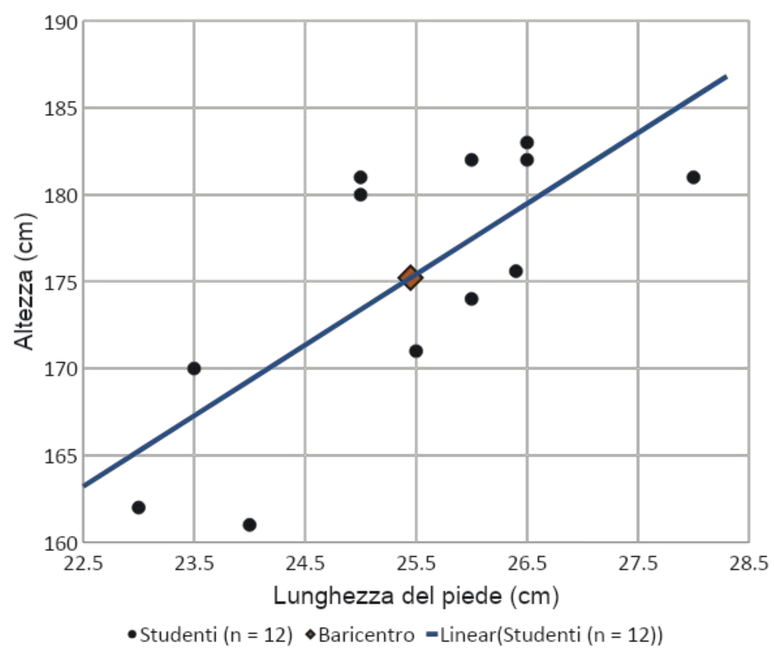
Fig. 1 — Ricostruzione in GeoGebra: retta mobile, baricentro della nuvola e somma dei residui quadratici in funzione di m.

4 Fase 2 — Integrazione: dataset reale

Modulo ispirato al progetto ministeriale canadese TIPS4RM sulla best fit line. Gli studenti misurano avambraccio (X_1), apertura delle braccia (X_2), lunghezza del piede (X_3) e altezza (Y), condividendo i dati in un foglio Google.

Studente	Altezza	Avambr.	Apert. br.	Piede
S01	182,0	29,5	198,0	26,0
S04	175,6	26,3	181,0	26,4
S07	174,0	26,0	181,0	26,0
S09	181,0	28,0	180,0	28,0
S12	161,0	25,0	165,0	24,0

Estratto della tabella condivisa in classe (misure in cm).



Tutti i 12 studenti: y ≈ 4,07·x + 71,7 (r ≈ 0,75); la retta passa per il baricentro (25,45; 175,2).

- Ogni studente crea un dataset personalizzato eliminando la propria riga.
- Retta dei minimi quadrati con NumPy, Pandas e Matplotlib in Google Colaboratory.
- Discussione finale sull'errore: la previsione può discostarsi dalla reale altezza.

```
# 1. Selezione variabili
x = df_altezze['Piede']
y = df_altezze['Altezza']

# 2. Calcolo regressione (m, q)
m, q = np.polyfit(x, y, 1)

# 3. Grafico
plt.scatter(x, y)
plt.plot(x, m*x + q)
plt.scatter(x.mean(), y.mean())
plt.xlabel("Lunghezza Piede (cm)")
plt.ylabel("Altezza (cm)")
plt.show()
```

Listato 1 — Calcolo di m e q e visualizzazione della retta sul dataset piede–altezza.

5 Valutazione

Formativa con l'osservazione delle varie fasi e sommativa sugli elaborati, con una rubrica condivisa e un giudizio congiunto delle due docenti in base ai seguenti criteri:

1. correttezza tecnica del codice Python
2. qualità e pertinenza dei commenti
3. spiegazione chiara delle scelte effettuate
4. correttezza dell'interpretazione matematica dei risultati
5. uso critico dei modelli: limiti delle previsioni su individui non presenti nel training set

6 Fase 3 — Modelli lineari a confronto

Visione di un video divulgativo che mostra come ottenere la retta di regressione lineare con PyTorch.
Gli studenti replicano quanto appreso sul dataset piede–altezza, trattando i dettagli matematici della libreria come black-box.

```
SEED = ...
LEARNING_RATE = ...
EPOCHS = ...

x = torch.tensor(df_altezze[...],
      dtype=torch.float32).unsqueeze(1)
y = torch.tensor(df_altezze[...],
      dtype=torch.float32).unsqueeze(1)

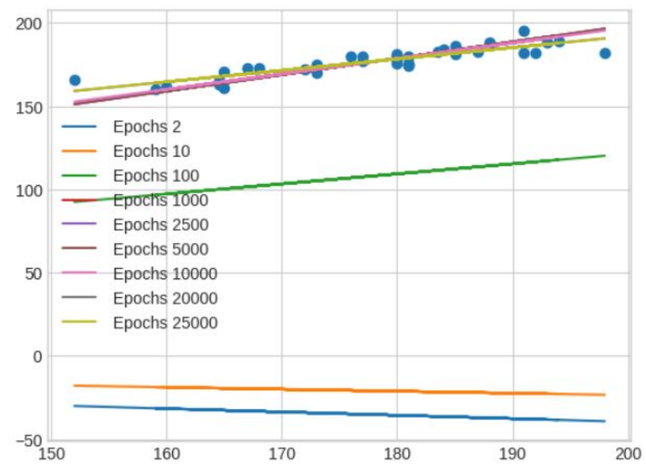
model = torch.nn.Linear(1, 1)
loss_fn = torch.nn.MSELoss()
optimizer = torch.optim.Adam(
      model.parameters(), lr=LEARNING_RATE)
```

Listato 2 — Notebook da completare: configurazione, modello a un solo neurone lineare, training loop, stampa di pesi e bias finali.

Minimi quadrati
pendenza m · intercetta q

Rete neurale
peso w · bias b

- Dataset finale fornito dal docente: stessa età, campione più ampio e in parte diverso.
- Confronto sulla relazione apertura delle braccia – altezza: retta dei minimi quadrati e rete neurale lineare (PyTorch).
- Relazione finale: due Notebook commentati e un documento che spiega come si ottengono i due modelli, con esempi d'uso.



Ciclo di iterazioni del modello con rete neurale

```
SEED = 42
LEARNING_RATE = 0.01
EPOCHS = 25000
```

(estratto da una relazione finale)

7 Punti di forza e criticità

+ Forte coinvolgimento: dati personali che rendono concreto un tema astratto.
+ Progressione efficace degli strumenti (PhET ⟶ GeoGebra ⟶ Python ⟶ PyTorch).
− Differenze significative nelle competenze di programmazione.
− L'approccio black-box facilita il focus sul training ma lascia dubbi sul funzionamento interno.

8 Sviluppi futuri

Nel prossimo anno scolastico si aprirà la black-box: forward pass, funzione di perdita, aggiornamento dei pesi. Poi, gradualmente, da un singolo neurone a reti più articolate, introducendo overfitting, generalizzazione e qualità dei dati.