Lezione 4: Dentro la "Scatola Nera"
Capire come la Macchina Corregge i Propri Errori

Laboratorio di Sistemi e Automazione

1 Introduzione

Nelle lezioni precedenti, `LinearRegression` ci è sembrata quasi una magia: abbiamo inserito dei dati disordinati e abbiamo ottenuto una retta perfetta. Ma l'Intelligenza Artificiale non è magia, è **matematica**.

In questa lezione risponderemo alla domanda fondamentale: *"Come fa il computer a capire che una retta è migliore di un'altra?"*. Scopriremo che la macchina impara per tentativi, cercando di minimizzare un valore chiamato "Funzione di Costo".

2 Il Concetto di "Residuo" (Lo Scarto)

Immaginiamo di essere in laboratorio. Il nostro multimetro legge una tensione di 1.9 V, ma la nostra formula teorica prevedeva 2.0 V.

$$\text{Realtà} = 1.9\,V \quad vs \quad \text{Teoria} = 2.0\,V$$

Quella differenza di $-0.1\,V$ è il **Residuo**. Per la macchina, questo residuo è un segnale di allarme: significa che la retta che ha tracciato non passa esattamente sopra il punto.

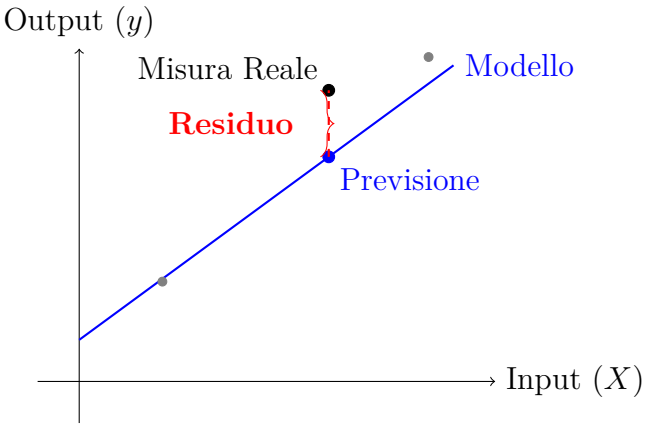

Fig. 1: Il "Residuo" (linea rossa) è l'errore commesso dalla macchina.

3 La Pagella della Macchina: MSE

L'algoritmo non può guardare un solo punto, deve guardare l'insieme di tutti i punti sperimentali. Deve darsi un "voto" complessivo. Questo voto si chiama **MSE (Mean Squared Error)**.

$$MSE = \frac{1}{N} \sum (\text{Errore})^2$$

Perché eleviamo al quadrato?

Se facessimo una media semplice, un errore di $+5$ e un errore di -5 si annullerebbero (media $= 0$), facendo credere alla macchina di essere perfetta anche se ha sbagliato tutto! Elevando al quadrato ($5^2 = 25$, $-5^2 = 25$), tutti gli errori diventano positivi e si sommano. Più è alto questo numero, peggiore è la retta.

4 Laboratorio: Uomo vs Macchina

Per capire davvero questo concetto, su Google Colab faremo una gara. Proveremo a indovinare noi la Resistenza (R) "a occhio" e calcoleremo il nostro MSE. Poi lasceremo fare a Python.

4.1 1. Prepariamo il campo di gara

Carichiamo i dati "sporchi" del laboratorio. Notate i commenti nel codice per capire cosa stiamo facendo.

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error # <--- Importiamo lo "strumento di
    misura" dell'errore

# Misure raccolte in laboratorio (Input: Corrente, Output: Tensione)
X = np.array([[1.0], [2.0], [3.0], [4.0], [5.0]])
y_true = np.array([1.1, 1.9, 3.2, 3.8, 5.1]) # Notate che non sono perfetti (rumore)
```

Listing 1: Setup dei Dati Sperimentali

4.2 2. La Sfida

Ora mettiamo a confronto l'intuizione umana contro l'ottimizzazione matematica.

```
# --- TENTATIVO UMANO ---
# Guardando i dati, ipotizziamo che la Resistenza sia 1.5 kOhm.
# Moltiplichiamo quindi tutti gli input per 1.5.
y_bad = X * 1.5

# --- TENTATIVO AI (Machine Learning) ---
# Chiediamo alla classe LinearRegression di trovare la retta ottima.
# La funzione .fit() esegue migliaia di calcoli per minimizzare l'errore.
modello = LinearRegression().fit(X, y_true)
y_ai = modello.predict(X)

```

```
# --- IL VERDETTO (Calcolo MSE) ---
# mean_squared_error confronta i dati VERI (y_true) con le nostre stime
errore_umano = mean_squared_error(y_true, y_bad)
errore_ai    = mean_squared_error(y_true, y_ai)

print(f"Il 'Voto' al modello Umano (Errore): {errore_umano:.4f}")
print(f"Il 'Voto' al modello AI (Errore):    {errore_ai:.4f}")
```

Listing 2: Confronto dei Modelli

Risultato atteso: L'errore umano sarà alto (es. > 2.0). L'errore dell'AI sarà piccolissimo (es. 0.02). Questo ci dimostra che la macchina ha trovato una linea che passa molto più "in mezzo" ai punti rispetto alla nostra ipotesi.

5 Visualizzare il Risultato

I numeri non dicono tutto. Guardando il grafico generato da Colab, vedrete chiaramente la differenza.

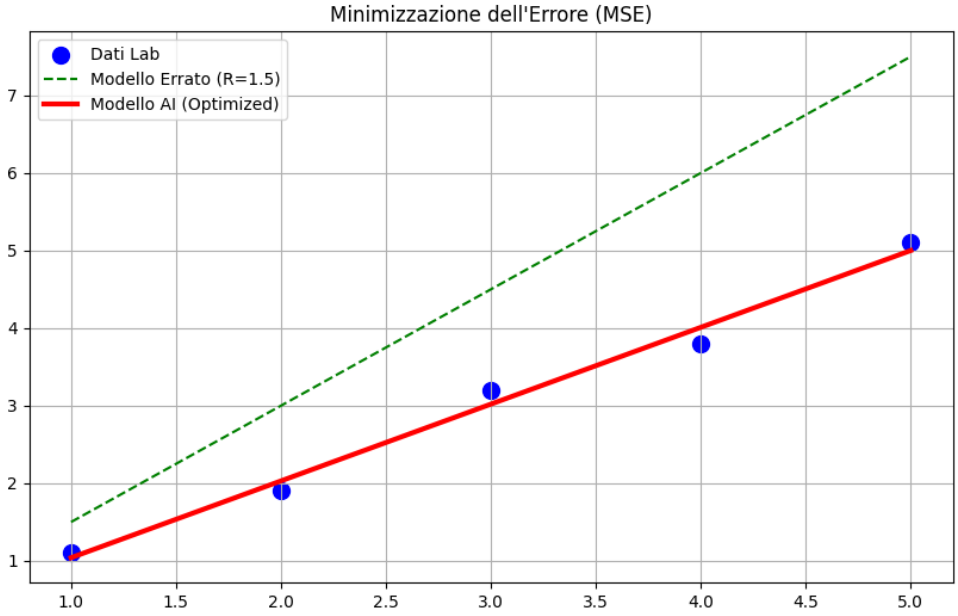

Figura 1: **Analisi Grafica:** La retta tratteggiata verde (ipotesi umana) passa molto lontana dai punti blu. La retta rossa (AI) taglia perfettamente la nuvola di punti, minimizzando le distanze (residui).

6 Attenzione all'Overfitting (Imparare a Memoria)

Potremmo chiederci: *"Perché non unire tutti i puntini con una linea a zig-zag? L'errore sarebbe zero!"*

In elettronica (e nel Machine Learning), questo è un errore grave chiamato **Overfitting**. Se unissimo i puntini, staremmo modellando gli errori del multimetro (il rumore), non la legge fisica del resistore. La retta rossa è migliore proprio perché **ignora le piccole oscillazioni** e cattura la tendenza generale (La Legge di Ohm).

→ TOCCA A TE SU COLAB

Nel notebook troverai il codice pronto.
Sfida: Prova a cambiare il valore `1.5` nel codice con altri numeri (es. `0.9`, `1.1`). Riesci ad abbassare l'errore e avvicinarti al risultato dell'AI?

[AVVIA ESERCITAZIONE]

→ CONTINUA IL PERCORSO

Ora che abbiamo capito come la macchina "pensa" e calcola l'errore, siamo pronti per il passo successivo.

Vai alla Lezione 5