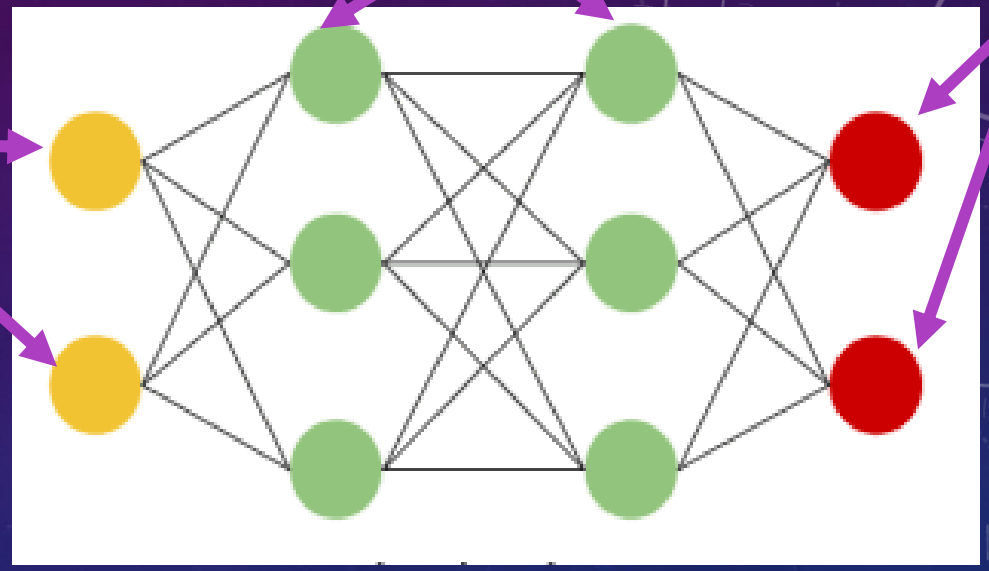
Presentazione sulle reti neurali

A cura di Alex Brunot 3°E

FUNZIONAMENTO RETE

Una rete neurale è come un «cervello» artificiale, fatto di tanti piccoli "neuroni» (chiamati nodi) collegati tra loro e organizzati in strati (layer).
Esistono tre tipi di nodi e layer

- Di input → Dati in ingresso alla rete
- Nascosti → Servono alla rete per elaborare la risposta
- Di Output → Ultimano la risposta fornita



Calcoli per ogni neurone:

- Per ogni neurone Input calcolare $w * Input + bias$
- Sommare i valori ottenuti per tutti i neuroni input = S
- Se S è positivo allora il neurone è attivato e il suo output = S
- Se S è negativo allora output = 0

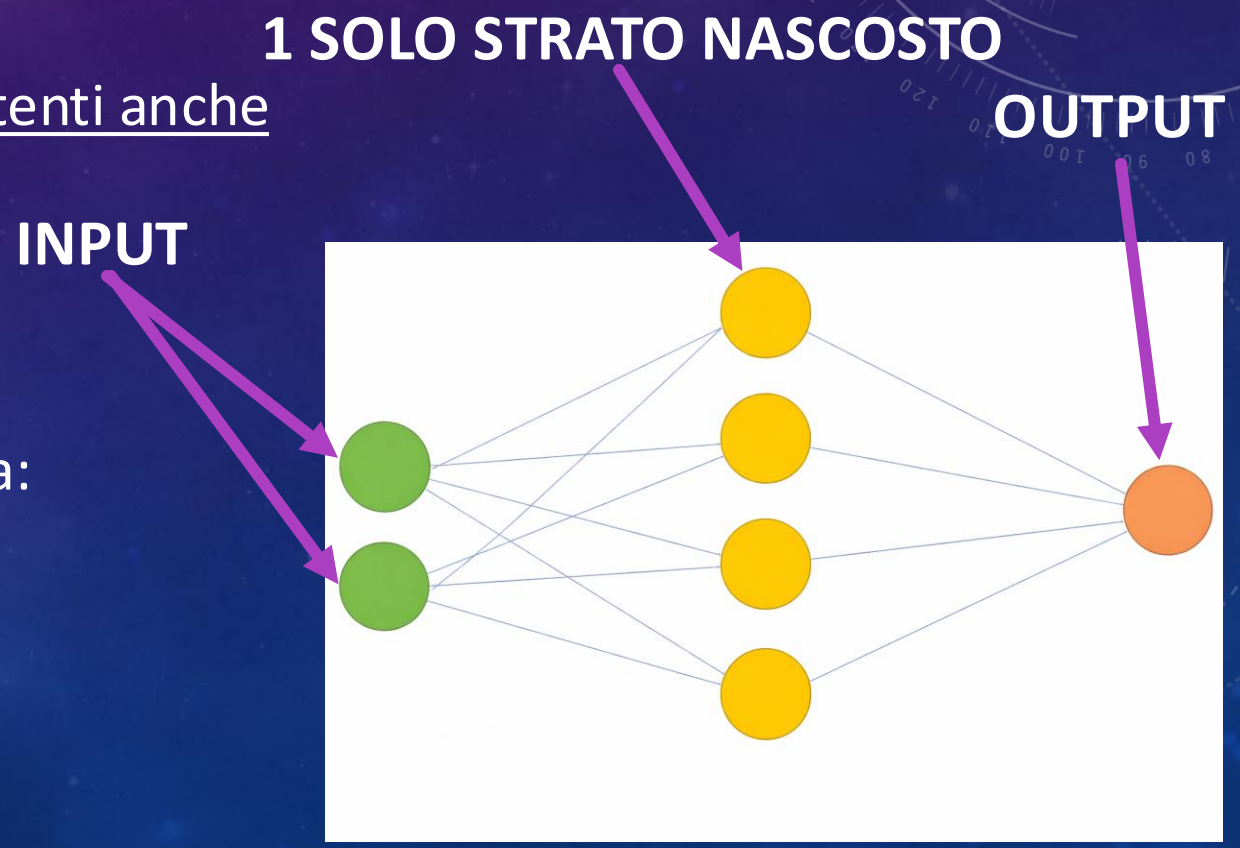
La rete neurale per imparare regola due gruppi di parametri:

- I pesi (W)
  - Sono uno dei valori dell'equazione che fa ogni neurone per capire se attivarsi
- I bias (B)
  - È un valore che si somma al risultato dell'equazione per aumentare la possibilità che questo si attivi

TEOREMA DI APPROSSIMAZIONE UNIVERSALE

Teorema di approssimazione universale: è stato dimostrato matematicamente che una rete neurale con uno singolo strato interno (ma abbastanza larga) è in grado di approssimare qualsiasi funzione continua, ovvero calcolare il valore della funzione con una precisione arbitraria.

Il teorema ci dice che le reti neurali sono molto potenti anche con 1 solo strato



Si potrebbe applicare al teorema di Pitagora ($\sqrt{c_1^2 + c_2^2} = i$) con una rete neurale così composta:

- 2 nodi di input (cateto 1, cateto 2)
- Un singolo strato interno di vari neuroni
- 1 nodo di output (ipotenusa)

CREARE UNA RETE NEURALE

```
from keras.models import Sequential
from keras.layers import Dense


rete = Sequential(
    [
        Dense(15, activation="relu", input_dim=2, name="Strato_nascosto"),
        Dense(1, activation="linear", name="Output_ipotenusa"),
    ]
)
```

STRATO NASCOSTO

INPUT

OUTPUT

È possibile creare una rete neurale utilizzando dei linguaggi di programmazione come Python, grazie a delle librerie apposite (funzioni disponibili aggiuntive) scaricabili online, io ho utilizzato «Keras».

Nell'esempio che ho scritto sopra, ho creato una rete che abbia:

- 2 input (Cateto1 e Cateto2)
- 15 neuroni nascosti (Elaborazione)
- 1 neurone di output (Ipotenusa)

Il numero di neuroni nascosti è regolabile in modo da svolgere risultare più preciso ed efficiente.

- Più neuroni ci sono più la rete è precisa
- Più neuroni ci sono più aumenta anche il numero di parametri
  - Nell'esempio sono presenti 15 neuroni in modo da poter mostrare una tabella alla fine della presentazione.

ALLENARE UNA RETE NEURALE

- **Generare dati per l'allenamento**
  - Scegliere a caso due lunghezze di cateti C1, C2
  - Calcolare il risultato $\sqrt{c_1^2 + c_2^2}$
  - Preparare così 2000 dati di allenamento

```
numero_campioni = 2000  # Numero esempi
cateti1 = np.random.uniform(0, 15, numero_campioni)  # Cateto A: 0-15
cateti2 = np.random.uniform(0, 15, numero_campioni)  # Cateto B: 0-15
ipotenuse = np.sqrt(cateti1**2 + cateti2**2)  # Target esatto
```

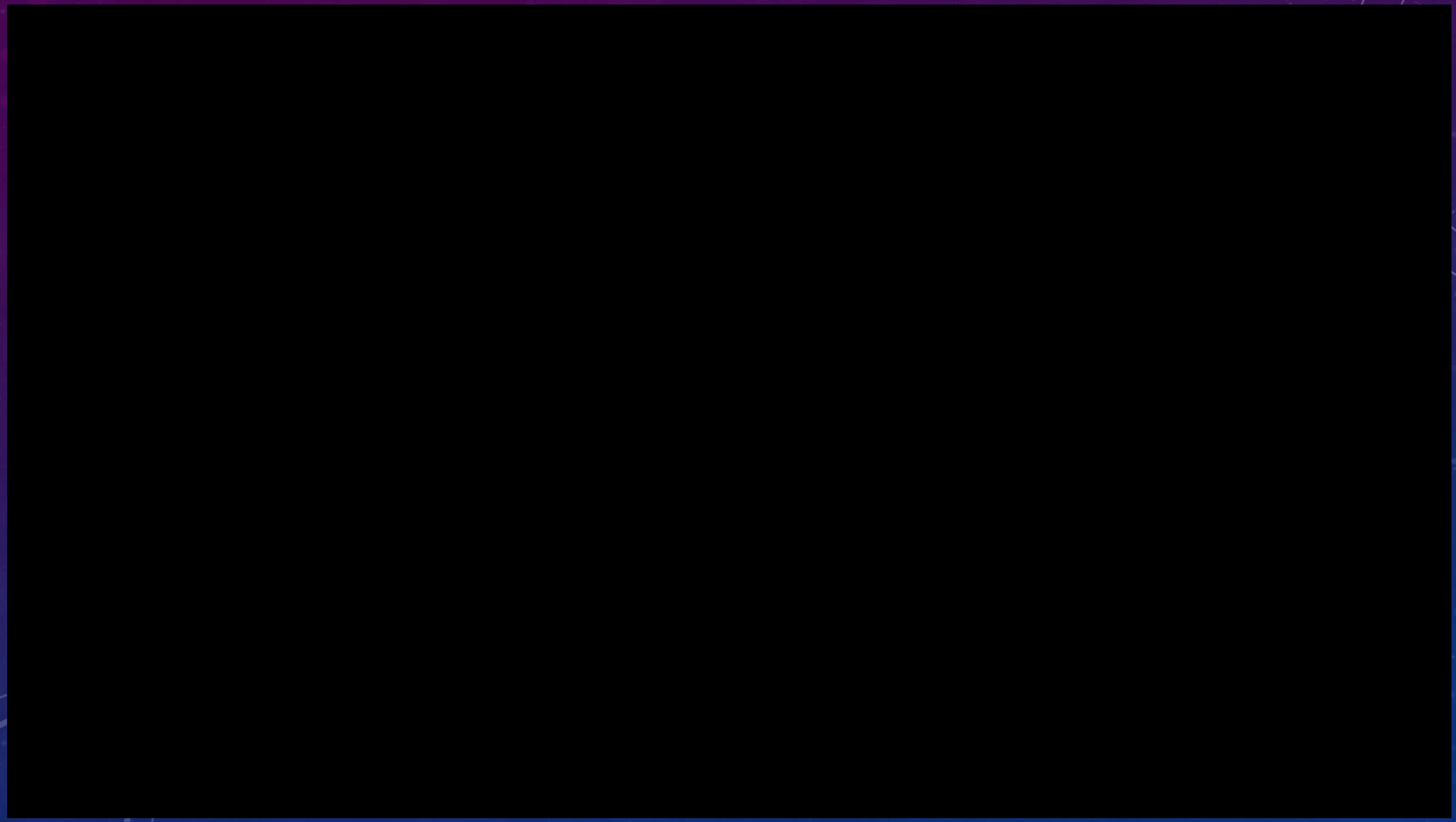
- **Allenare la rete (calcolare i parametri)**
  - La libreria Python lo fa da sola in pochi minuti, bisogna specificare quante volte ricalcolare i parametri («epochs»)

```
rete.compile(loss="mse", optimizer="adam", metrics=["mae"])

history = rete.fit(
    x=cateti_training,     # Dati generati
    y=ipotenuse_training,  # Dati generati
    epochs=250,
    validation_split=0.2,  # 20% per validazione automatica
    shuffle=True,  # Mescola dati
)
```

  - Alla fine dell'allenamento sono stati calcolati i pesi e i bias per tutti i neuroni della rete (61 in totale)
  - E' possibile iniziare a usarla per calcolare ipotenuse

(M)ANIMAZIONE DELL'ALLENAMENTO



TESTARE UNA RETE NEURALE

Ora che abbiamo una rete allenata, come possiamo provarla?

Provare alcuni valori che diano ipotenuse intere

```
Ipotenusa di 3.0 e 4.0: 4.9939
Ipotenusa di 6.0 e 8.0: 10.0080
```

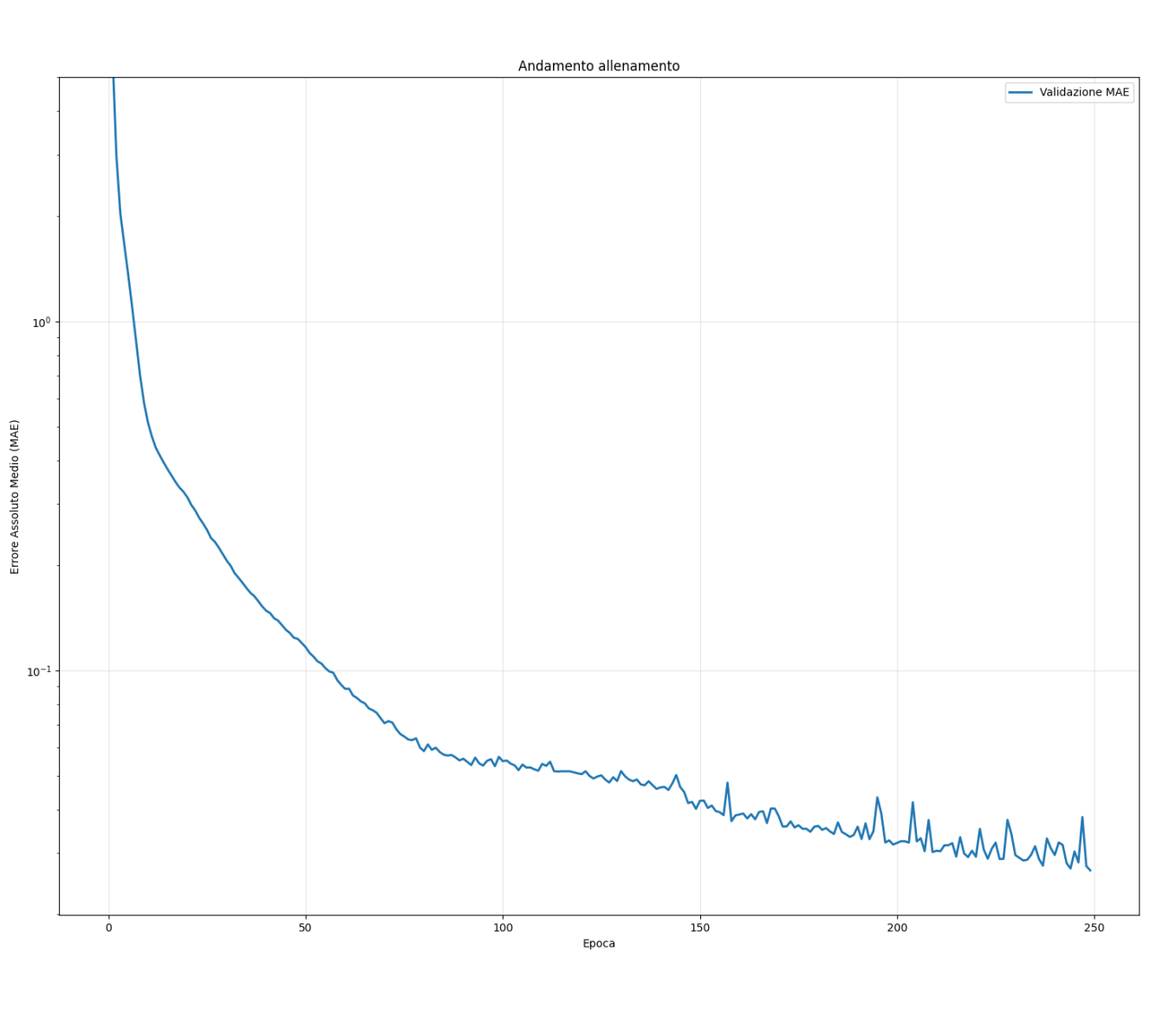
Esistono delle funzioni di valutazione standard come l'Errore medio assoluto (**MAE**)

- **MAE** = (1/n) * Σ|Valore Effettivo - Valore Previsto|

Si può rappresentare con un grafico come la rete sta migliorando come si può veder a lato.

In questo caso la scala è detta «logaritmica»

- Vuol dire che man mano che l'asse si avvicina allo 0, l'unità di misura aumenta o diminuisce.
- All'inizio è >4, alla fine circa 0.03
  - Serve a vedere meglio l'andamento man mano che le epoche avanzano



PROGRAMMA PYTHON COMPLETO

```python
from keras.models import Sequential
from keras.layers import Dense
import numpy as np

rete = Sequential([Dense(15, activation="relu", input_dim=2, name="Strato_interno"),
     Dense(1, activation="linear", name="Output_ipotenusa")])

numero_campioni = 2000  # Numero esempi
cateti1 = np.random.uniform(0, 15, numero_campioni)  # Cateto A: 0-15
cateti2 = np.random.uniform(0, 15, numero_campioni)  # Cateto B: 0-15
ipotenuse = np.sqrt(cateti1**2 + cateti2**2)  # Target esatto

cateti_training = np.column_stack((cateti1, cateti2))  # Preparara array (10000, 2)
ipotenuse_training = ipotenuse.reshape(-1, 1)  # Prepara output (10000, 1)

rete.compile(loss="mse", optimizer="adam", metrics=["mae"])

history = rete.fit(
    x=cateti_training, y=ipotenuse_training,  # Dati generati
    epochs=100, validation_split=0.2,  # 20% per validazione automatica
    shuffle=True,  # Mescola dati
    verbose=0
)

while True:
    c1 = float(input("Cateto 1="))
    c2 = float(input("Cateto 2="))
    pred = rete.predict(np.array([[c1, c2]]))
    print(f"Ipotenusa calcolata: {pred[0][0]:.4f}")

    # Mostra anche errore di calcolo
    ipotenusa_esatta = np.sqrt(c1**2 + c2**2)
    print(f"Ipotenusa esatta: {ipotenusa_esatta:.4f}")
    print(f"Errore di calcolo: {abs(pred[0][0]-ipotenusa_esatta):.4f} (% di errore = {abs(pred[0][0]-ipotenusa_esatta)/ipotenusa_esatta * 100:0.1f})")
```

CONCLUSIONI

Ho deciso di svolgere questo lavoro perché l'argomento delle reti neurali è un argomento che mi interessa molto e che posso integrare con altre mie competenze come la programmazione.

Grazie a questa presentazione ho imparato:

- Cos'è il teorema di approssimazione universale,
- Come funzionano nel dettaglio bias e pesi,
- Le formule e i parametri che sono dietro al funzionamento di una rete neurale
- Come allenare una rete neurale.

Mi ha sorpreso il fatto che dopo diversi allenamenti la rete neurale non desse mai il risultato esatto (ad es: 5,034 al posto di 5), come invece mi sarei aspettato.

Un'altra cosa che non mi sarei aspettato è che, durante uno dei miei test, quando ho aumentato molto il numero di neuroni, al posto di migliorare, il risultato ha iniziato a peggiorare.

Ho scoperto che l'argomento è molto vasto e che ci sono tanti argomenti trattabili con le reti neurali oltre che l'approssimazione di una funzione come il Teorema di Pitagora, mi piacerebbe trovare altre applicazioni in futuro.

FINE