

The background features several abstract, organic shapes in shades of purple and blue. A large, irregular shape dominates the right side, with a smaller, more circular shape above it and another smaller shape at the bottom right. The colors transition from a deep purple in the center to a lighter blue at the edges.

Come funziona una Rete Neurale Multilayer Perceptron

AI e Machine Learning per il Marketing

Paola Vigato

01

Come funziona una rete neurale MLP



In questo progetto analizzo il **funzionamento interno** di una **rete neurale artificiale di tipo Multilayer Perceptron (MLP)**.

L'obiettivo non è mostrare solo a cosa serve l'algoritmo, ma spiegare come, partendo dai dati, esso arriva progressivamente a un risultato.

Per farlo, presento il neurone artificiale, la struttura della rete, il forward pass, il calcolo dell'errore e il backward pass.

Il neurone artificiale

Il **neurone artificiale** è l'**unità base** della rete neurale.

Ogni neurone riceve **uno o più input**, assegna a ciascuno un **peso**, **somma i valori ottenuti** e **aggiunge un termine detto bias**.

Il risultato viene poi trasformato da una **funzione di attivazione**, che produce l'**output del neurone**.

$$z = w_1x_1 + w_2x_2 + b$$

$$\hat{y} = \sigma(z)$$

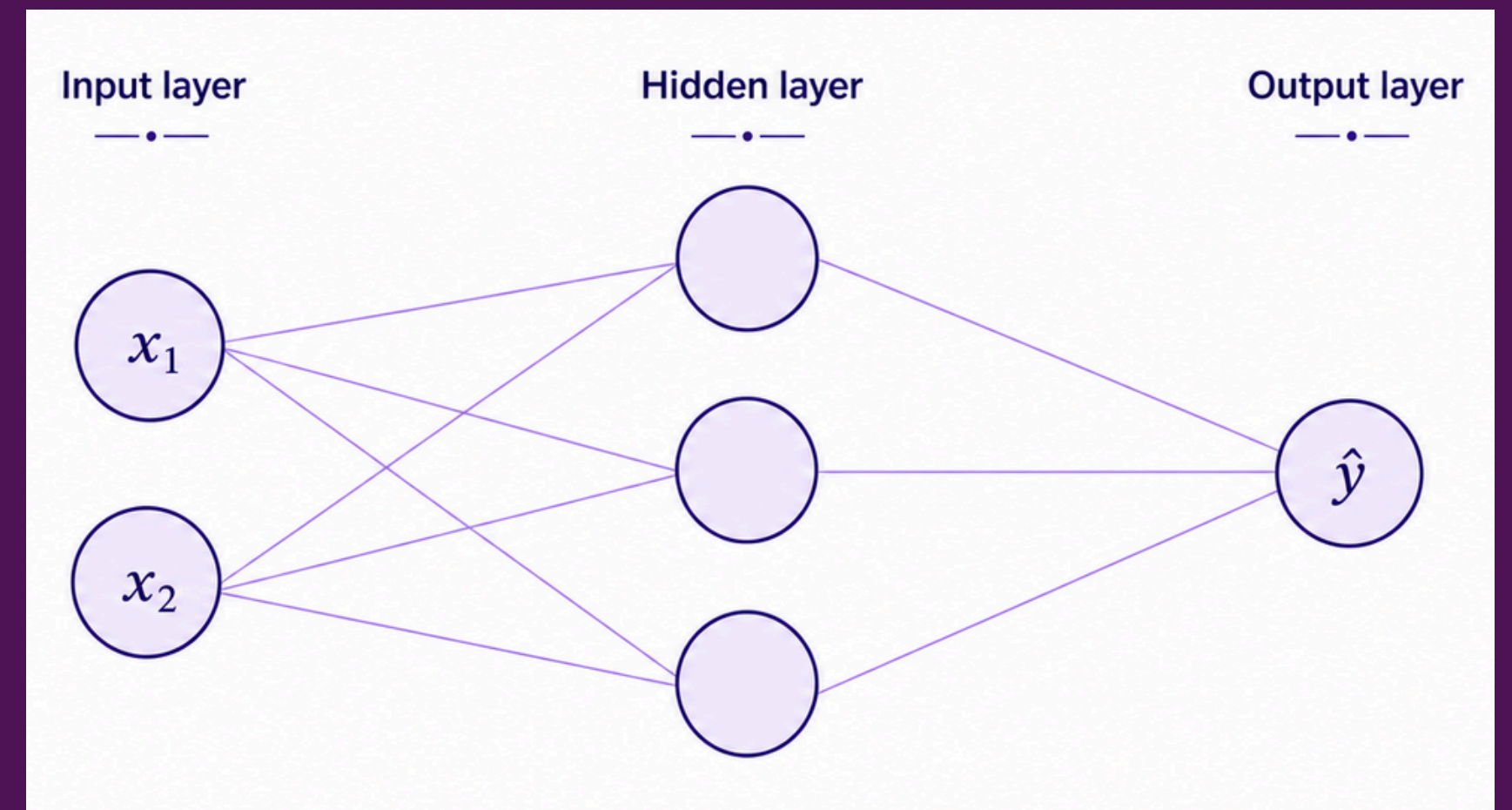
In questo modo il neurone non si limita a sommare i dati, ma li trasforma in una **risposta interpretabile dalla rete**.

Struttura di una rete MLP

Una rete MLP è composta da **più neuroni distribuiti in diversi livelli**.

L'input layer riceve i **dati iniziali**, uno o più hidden layer elaborano le **informazioni interne**, mentre l'output layer restituisce il **risultato finale**. La presenza dei layer nascosti consente alla rete di **apprendere relazioni più complesse** rispetto a un semplice modello lineare.

La rete quindi **non applica una regola fissa** già nota, ma costruisce la propria logica interna attraverso i **pesi delle connessioni**.



Mini-dataset simulato

Per spiegare il funzionamento dell'algoritmo ho costruito un mini dataset molto semplice.

Gli **input** rappresentano due variabili osservabili nel marketing digitale: **il numero di volte in cui è stata aperta una mail e il numero di click**.

Mentre **l'output** indica **se il cliente effettua o meno un acquisto**.

La rete deve imparare come combinare gli input per stimare correttamente l'output.

Cliente	Aperture email	Click	Acquisto
A	2	0	0
B	4	1	0
C	6	2	1
D	8	3	1

Forward pass: dalla combinazione degli input all'output

Nel forward pass la rete parte dagli input e calcola una **prima previsione**.

All'inizio i **pesi** sono assegnati in modo **casuale** e permettono di costruire una **combinazione pesata delle variabili in ingresso**.

Il valore ottenuto viene trasformato dalla funzione di attivazione e produce l'output stimato.

Supponiamo di osservare un cliente con:

- **aperture email = 2**
- **click = 0**

E di usare i pesi iniziali:

- **w1 = 0.2**
- **w2 = 0.4**
- **b = -1**

Allora:

$$z = (0,2 \cdot 2) + (0,4 \cdot 0) - 1 = -0,6$$

$$\hat{y} = \sigma(-0,6) \approx \mathbf{0.35}$$

La rete stima quindi una **probabilità di acquisto** pari a circa **0,35**.

Errore e backward pass

Una volta **ottenuto l'output previsto**, la **rete lo confronta con il valore reale**.

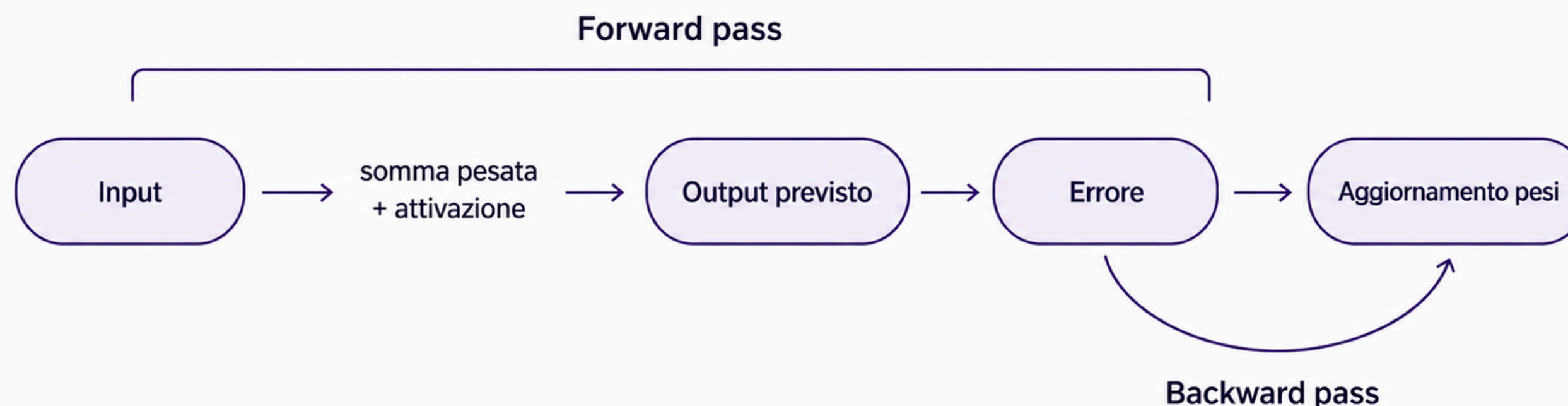
Se la **previsione non coincide** con il risultato corretto, viene calcolato un **errore**.

Il backward pass utilizza questo errore per **aggiornare i pesi** e migliorare la previsione successiva.

Nel nostro esempio, il valore **reale** è **0** e il valore **previsto** è circa **0,35**.

La rete ha quindi commesso un errore.

Questo errore non viene ignorato, ma viene usato come informazione per **correggere** i collegamenti interni tra i neuroni. In questo modo **l'algoritmo apprende progressivamente dai propri sbagli**.



Il backward pass permette alla rete di **imparare**: se sbaglia, corregge i pesi e riprova in modo più preciso.

Apprendimento iterativo

07

L'apprendimento della rete non avviene in un solo passaggio, ma attraverso **più iterazioni successive**.

A ogni iterazione la rete:

1. calcola **l'output previsto** per tutti i dati;
2. misura **l'errore medio**;
3. **aggiorna pesi e bias** tramite backward pass.

iterazione	peso aperture email	peso click	bias	errore medio
1	0.200000	0.400000	-1.000000	0.130082
2	0.193722	0.400573	-1.003675	0.129077
3	0.188325	0.401347	-1.007184	0.128281
4	0.183708	0.402400	-1.010546	0.127639
5	0.179774	0.403666	-1.013779	0.127113

Nelle **prime 5 iterazioni** si osserva che i parametri **w1, w2 e b cambiano progressivamente**, mentre **l'errore medio diminuisce**.

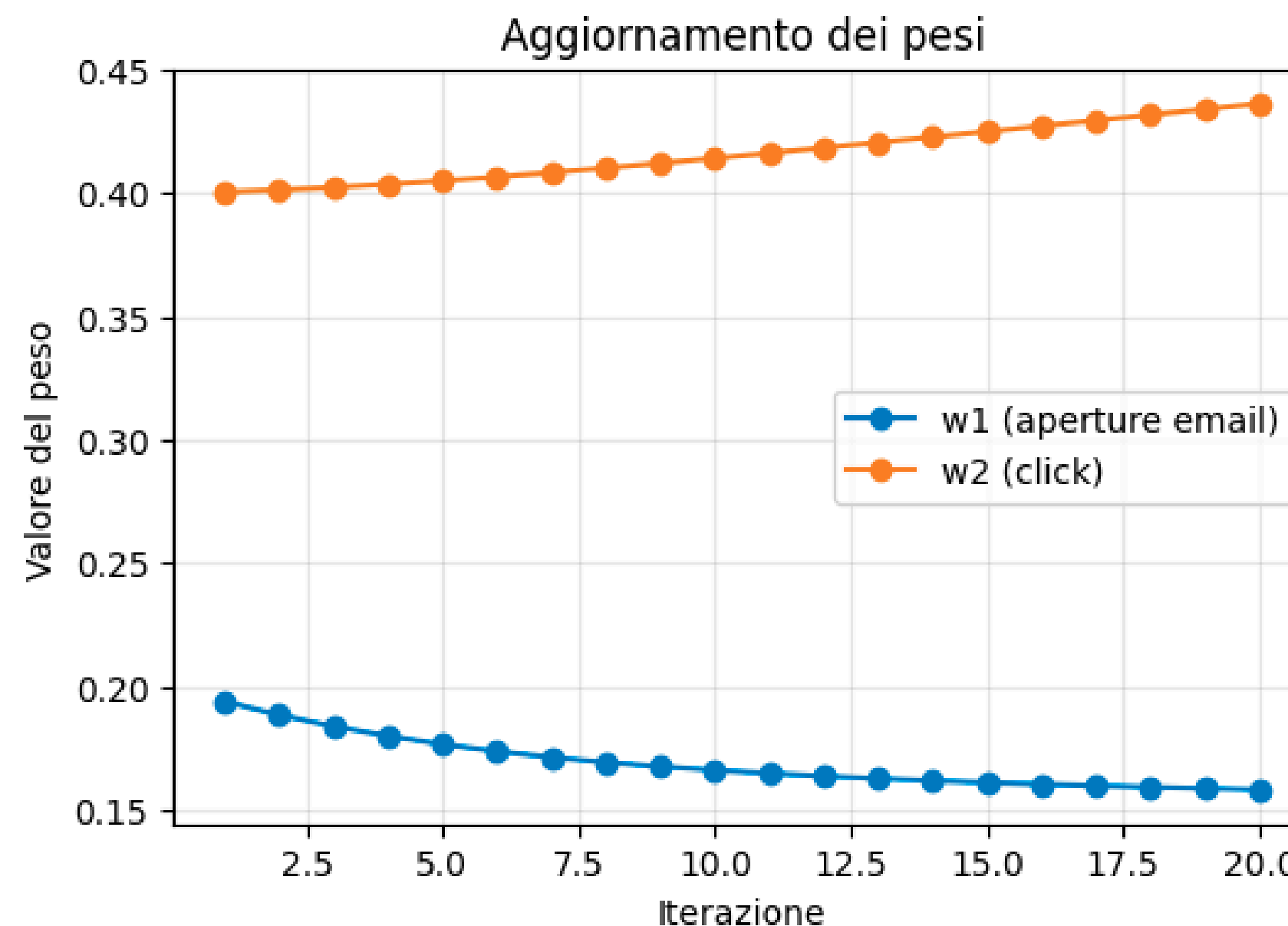
Nei grafici nella slide successiva vedremo l'evoluzione completa su 20 iterazioni.

Come cambiano i pesi

08

La rete parte da pesi casuali ($w1=0.2$, $w2=0.4$).
A ogni iterazione il backward pass li aggiusta
nella direzione che riduce l'errore.

La rete 'scopre' da sola che **i click** ($w2$) sono
un segnale **più forte** di acquisto rispetto alle
aperture email ($w1$): non lo sa a priori, lo
impara dai dati.



Evoluzione dell'output previsto

Per ogni iterazione calcoliamo la probabilità prevista per **due clienti reali del dataset**:

- Cliente A ($y=0$, non acquista);
- il Cliente C ($y=1$, acquista).

Se il modello funziona, la curva del Cliente A deve scendere **verso 0** e quella del Cliente C salire **verso 1**.

Il grafico mostra che questo sta accadendo, iterazione dopo iterazione. Il modello si sta muovendo nella direzione giusta: con più iterazioni la separazione tra le due curve si farebbe ancora più netta.

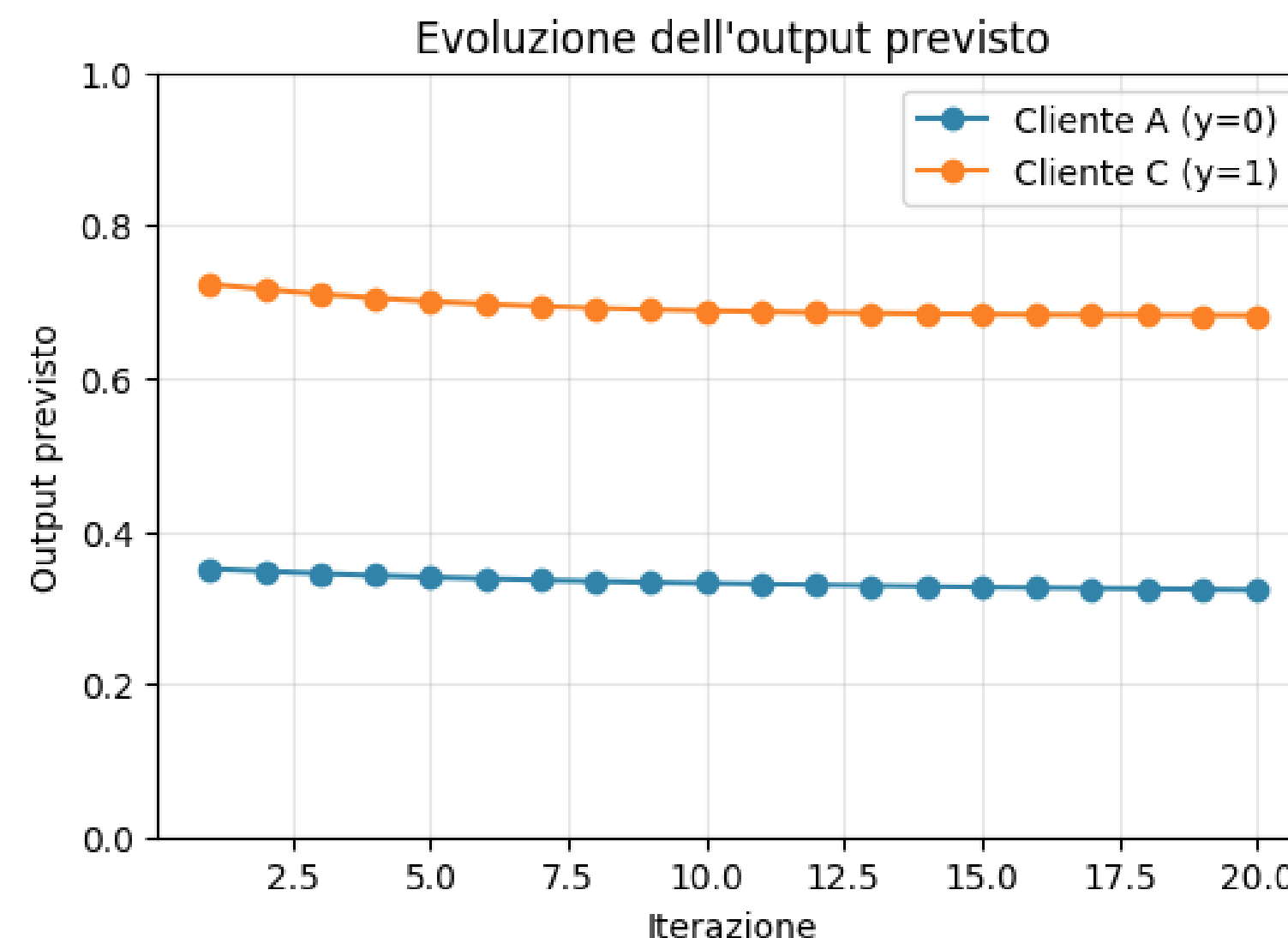


Grafico dell'errore

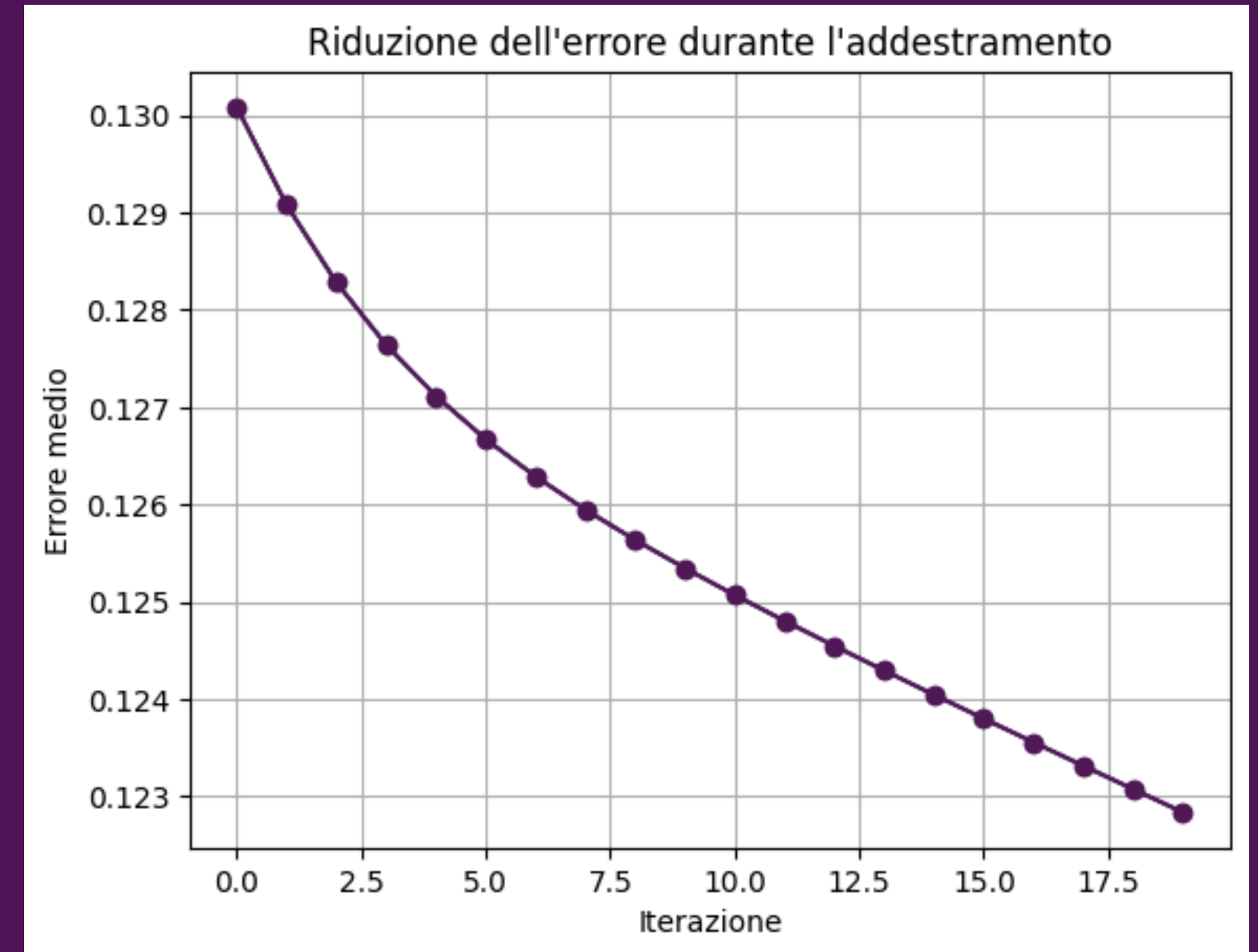
10

All'inizio i pesi della rete sono scelti in modo arbitrario, quindi le previsioni sono meno precise e l'errore medio è più alto.

Ad ogni iterazione avvengono due passaggi:

1. **Forward pass:** la rete combina gli input con i pesi, calcola la somma pesata, applica la funzione di attivazione e produce una previsione.
2. **Backward pass:** confronta la previsione con il valore reale, calcola l'errore e lo propaga all'indietro per aggiornare i pesi in modo da ridurre quell'errore.

Ripetendo più volte questa sequenza, la rete apprende la relazione tra input e output: lo si vede dal fatto che la **curva dell'errore medio scende progressivamente** man mano che aumentano le iterazioni.



1 Riceve i dati

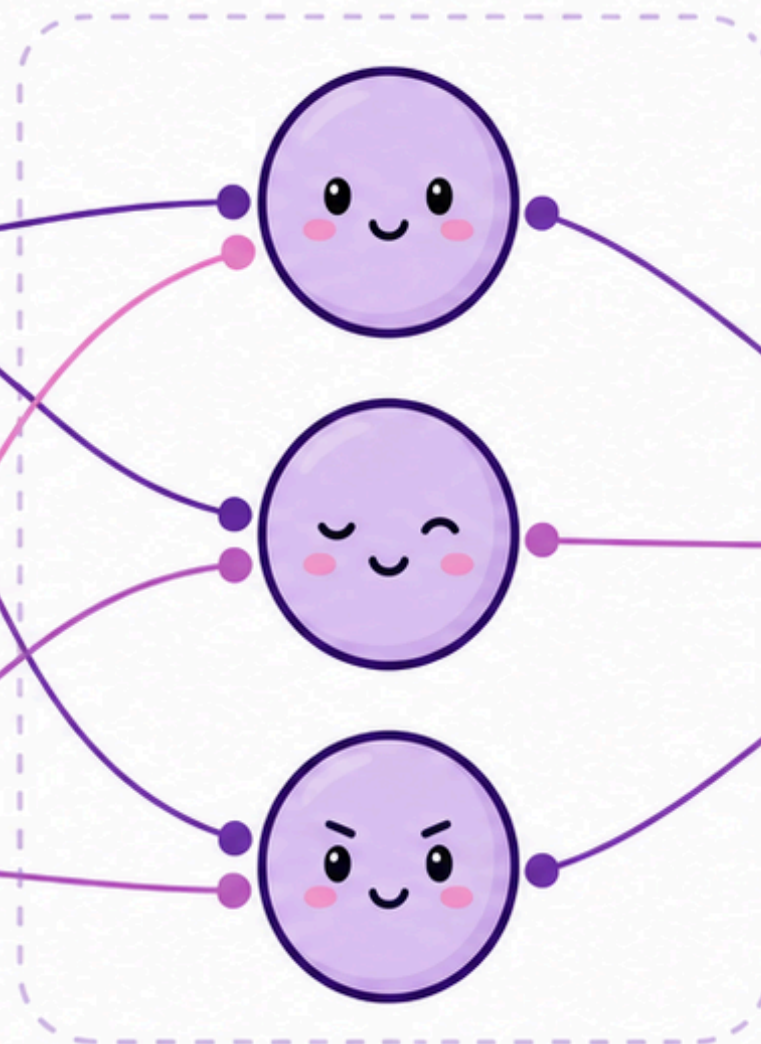
2 Li combina

3 Decide il risultato

Input 1



Input 2



Strato nascosto



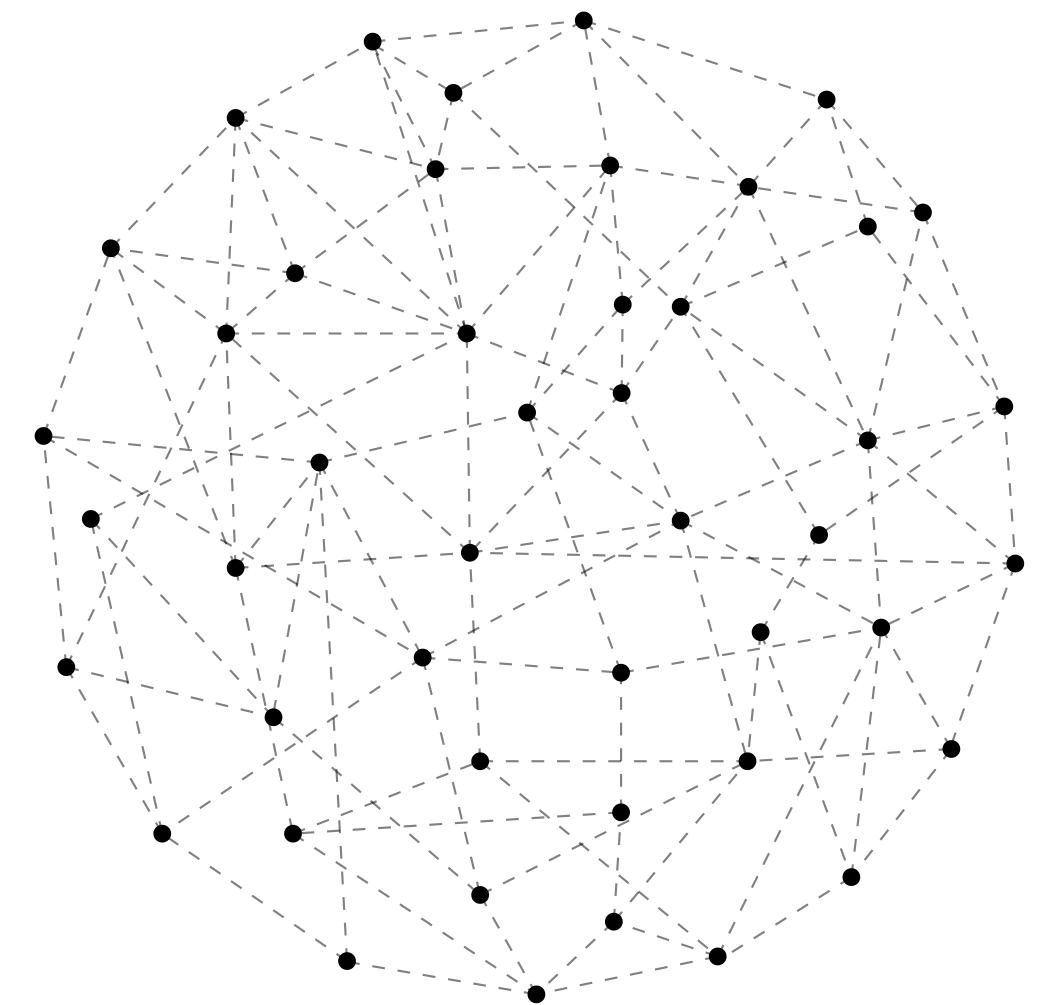
Output

Conclusione

La rete neurale MLP funziona assegnando pesi agli input, combinandoli nei neuroni e trasformandoli in un output tramite una funzione di attivazione. L'output previsto viene poi confrontato con il valore reale, e l'errore ottenuto viene propagato all'indietro per aggiornare i pesi e migliorare la previsione successiva.

Il modello apprende quindi attraverso un processo iterativo di adattamento: iterazione dopo iterazione, la relazione tra dati in ingresso e risultato stimato diventa progressivamente più precisa.

La rete neurale apprende perché usa l'errore come guida per migliorare progressivamente la previsione.



Notebook Colab:

https://colab.research.google.com/drive/1aIXjCGFQjHcFGFVyo_fl3fcKlRjmUiJ0?usp=sharing_