IA PER LA COMUNICAZIONE

Neural Netwoks

Prof. ssa Romina Eramo
Dipartimento di Scienze della Comunicazione
Università degli Studi di Teramo
reramo@unite.it

Introduzione

- Il connessionismo fornisce un substrato per l'intelligenza emergente.
- Oggi è rappresentato principalmente dalle reti neurali.
- Il termine "neurale" è ispirato ai neuroni biologici, ma la relazione tra i due è superficiale.

Neuroni biologici vs neuroni artificiali

- I neuroni biologici ricevono input dai dendriti.
- Si attivano solo quando un numero sufficiente di input è attivo.
- L'attivazione produce un picco di tensione sui loro assoni.
- 800 milioni di anni di evoluzione hanno reso il processo complesso, ma il principio base è semplice.

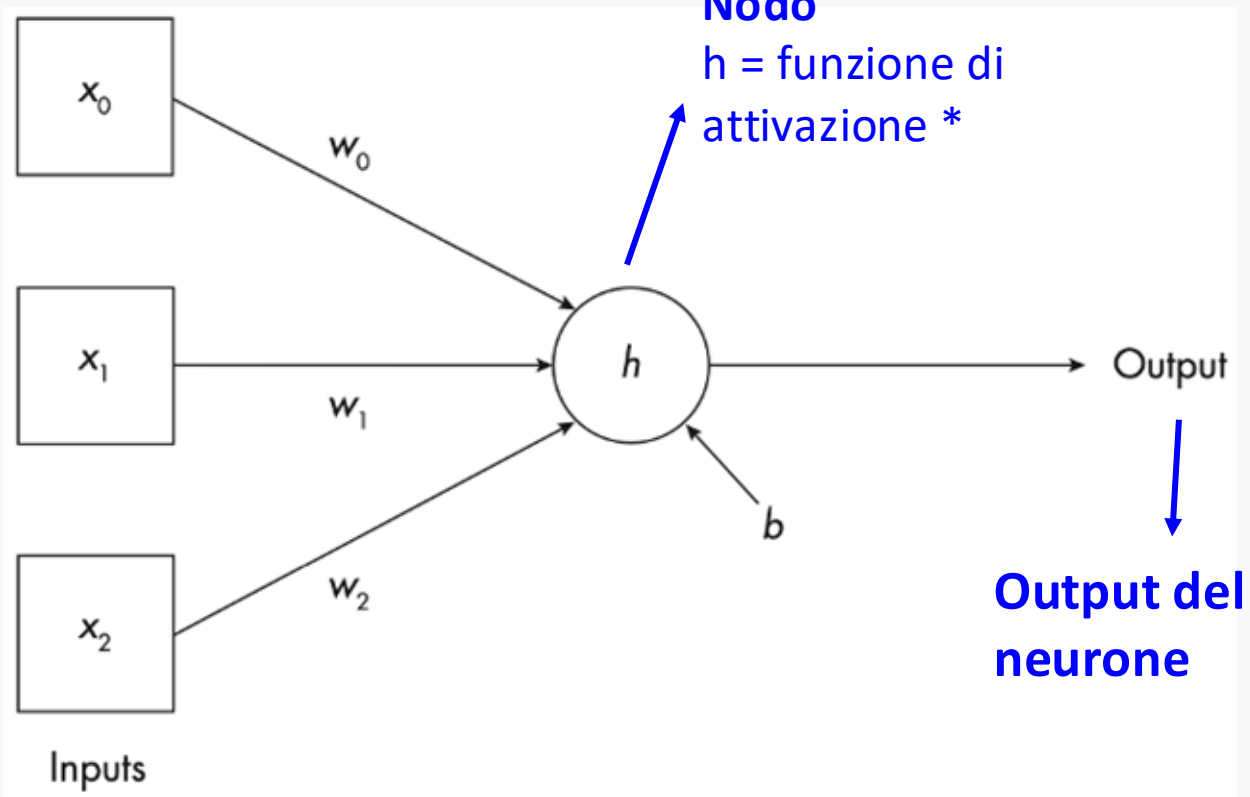
- I neuroni artificiali hanno input e output, ma operano in modo continuo.
- Funzionano come funzioni matematiche.
- Alcuni modelli imitano i picchi biologici, ma in questo contesto li ignoriamo.

Simile a un interruttore della luce.
Spento finché non c'è sufficiente input per accendersi.
Lampeggia invece di rimanere acceso.

Simile a una luce con dimmer.
L'output cambia in proporzione all'input.

Funzionamento base

Input del neurone
x = caratteristiche di un vettore di caratteristiche
w = peso associato a ogni input



Nodo
h = funzione di attivazione *

Output del neurone

* **Unità lineare rettificata (ReLU):** l'input è minore di zero? Se è così, l'output è zero; altrimenti, è qualunque sia l'input.

Funzionamento base

1. Moltiplica ogni valore di input, x0, x1 e x2, per il suo peso associato, w0, w1 e w2.

2. Somma tutti i prodotti del passaggio 1 insieme al valore di bias, b. Questo produce un singolo numero.

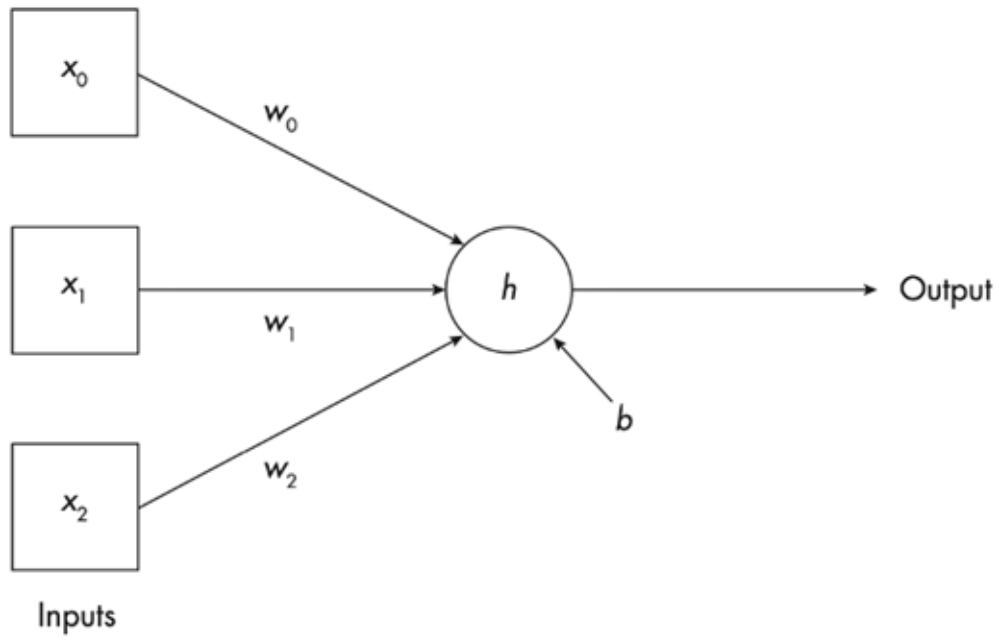


Figure 4-1: The humble (artificial) neuron

3. Assegna il singolo numero a h, la funzione di attivazione, per produrre l'output, anch'esso un singolo numero.

Esperimento (Iris)

- Addestrando il neurone con le tre caratteristiche dal set di dati del fiore di Iris
- Testando il neurone con un set di test inutilizzato che aveva 30 vettori di caratteristiche
- Il neurone ha classificato correttamente 28, per una precisione del 93%

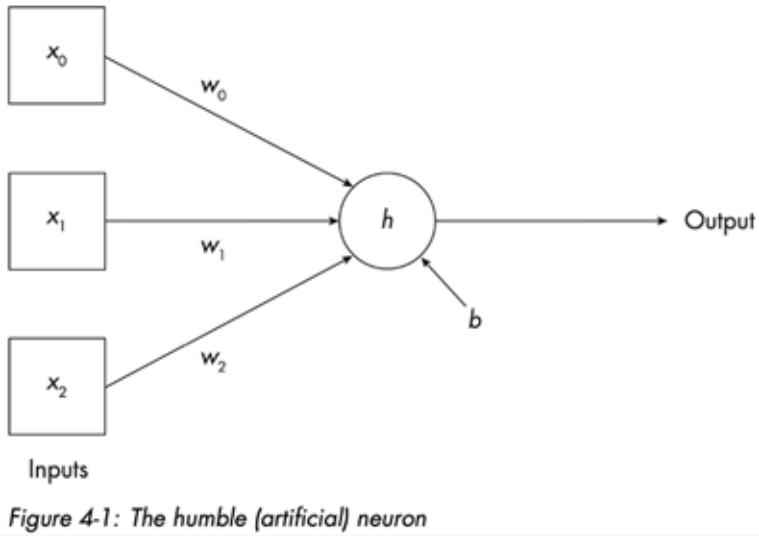


Figure 4-1: The humble (artificial) neuron

- Addestrando il neurone con un set di tre pesi e un valore di bias, produceva un output che, quando arrotondato al numero intero più vicino, corrispondeva all'etichetta di classe per un fiore di iris, ovvero 0, 1 o 2.
- Questo non è l'addestramento standard (lo vedremo più avanti)

Reti con 2, 3 e 8 nodi

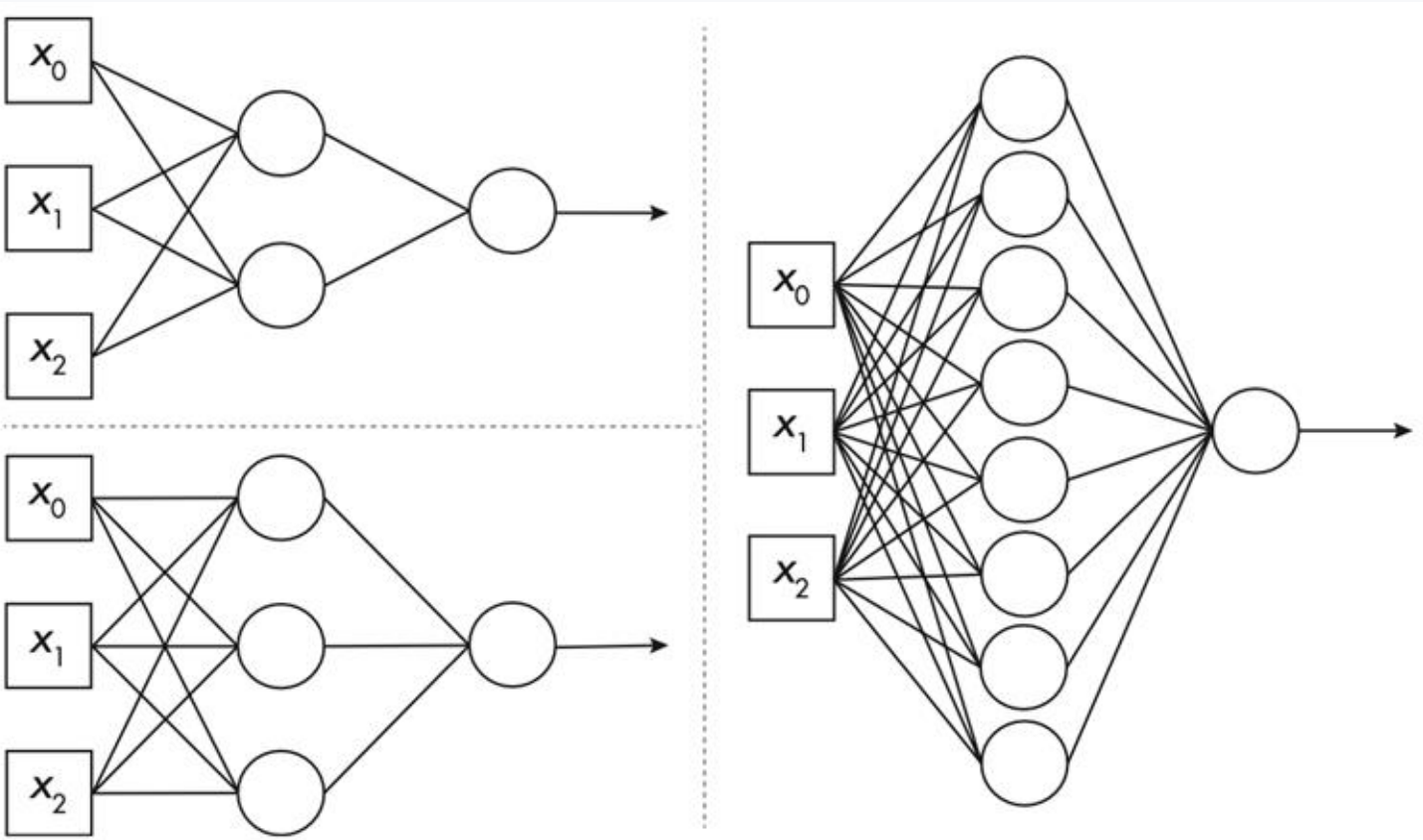


Figure 4-2: Two-, three-, and eight-node networks

Reti con 2, 3 e 8 nodi

Abbiamo bisogno di **un peso per ogni linea** (tranne la freccia di output) e **un valore di bias per ogni nodo**

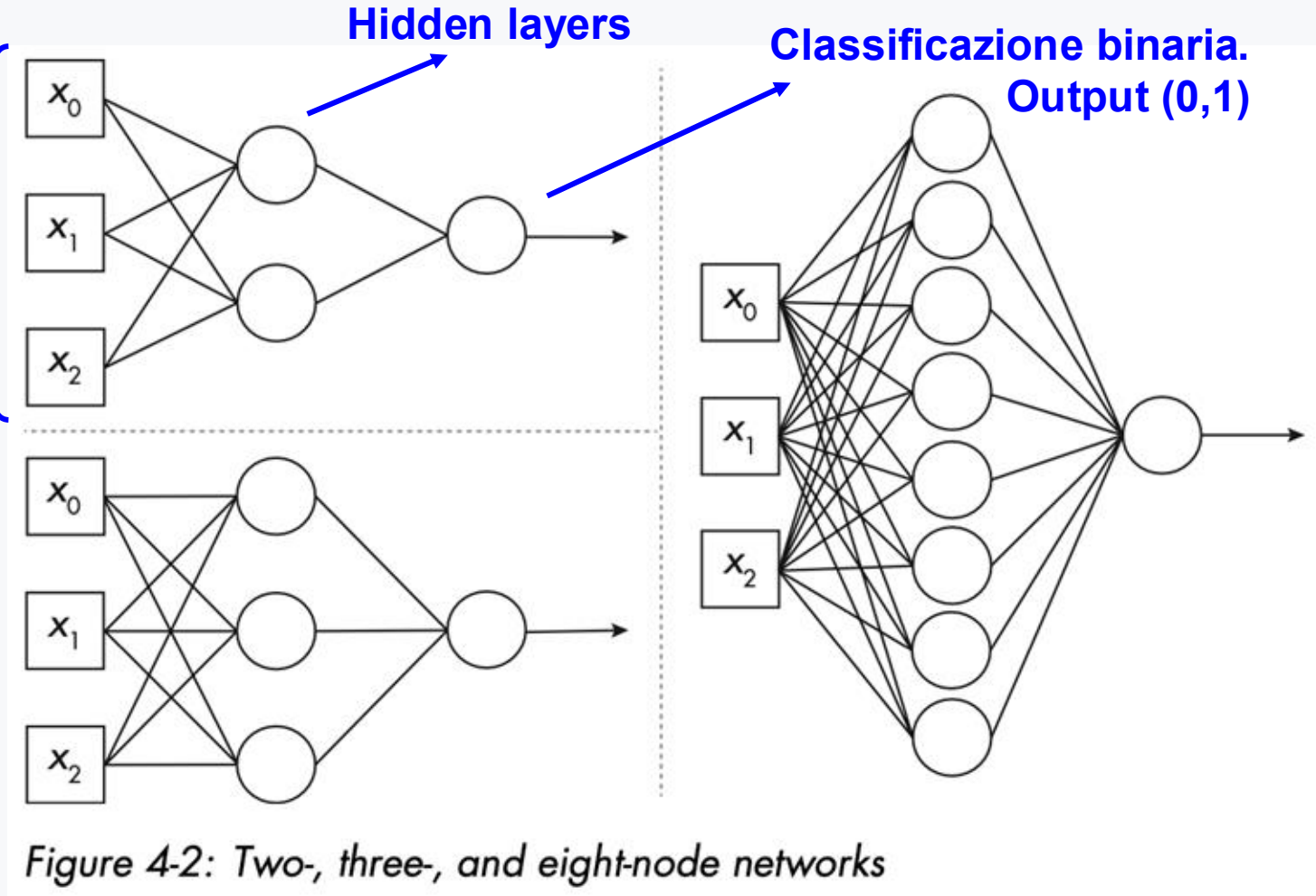


Figure 4-2: Two-, three-, and eight-node networks

ESEMPIO

Dataset: Varietà di uva per un vino italiano

- **Obiettivo**: addestrare i modelli in figura e vedere quanto bene ognuno si comporta nell'identificare un vino sconosciuto date le misurazioni delle tre caratteristiche.
- **Caratteristiche**: contenuto di alcol, acido malico e fenoli totali.

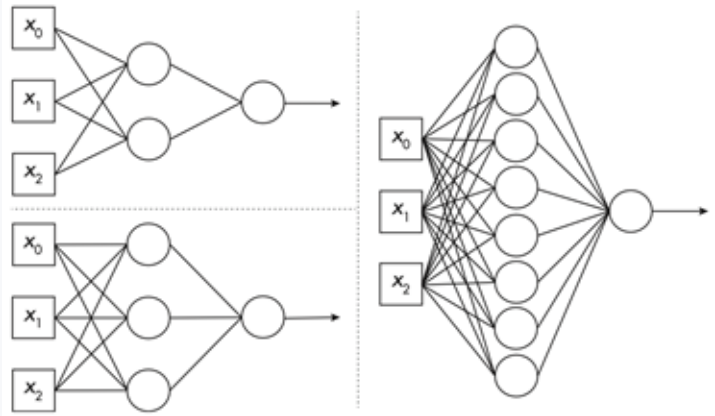


Figure 4-2: Two-, three-, and eight-node networks

- Addestrato il modello a due neuroni usando un training set di 104 campioni e un test set di 26 campioni.
  - 104 triplette di contenuto alcolico misurato, livello di acido malico e fenoli totali, conoscendo l'etichetta di output corretta, classe 0 o classe 1.
  - Il set di addestramento ha condizionato il modello a due neuroni per dare valori a tutti gli otto pesi e tre bias.
- Il modello addestrato ha raggiunto un'**accuratezza sul set di test dell'81%**, il che significa che è stato corretto più di 8 volte su 10

ESEMPIO

Dataset: Varietà di uva per un vino italiano

aggiunti i pesi ai collegamenti e i bias ai nodi

Campioni di prova:

Sample 1	(−0.7359, 0.9795, −0.1333)
Sample 2	(0.0967, −1.2138, −1.0500)

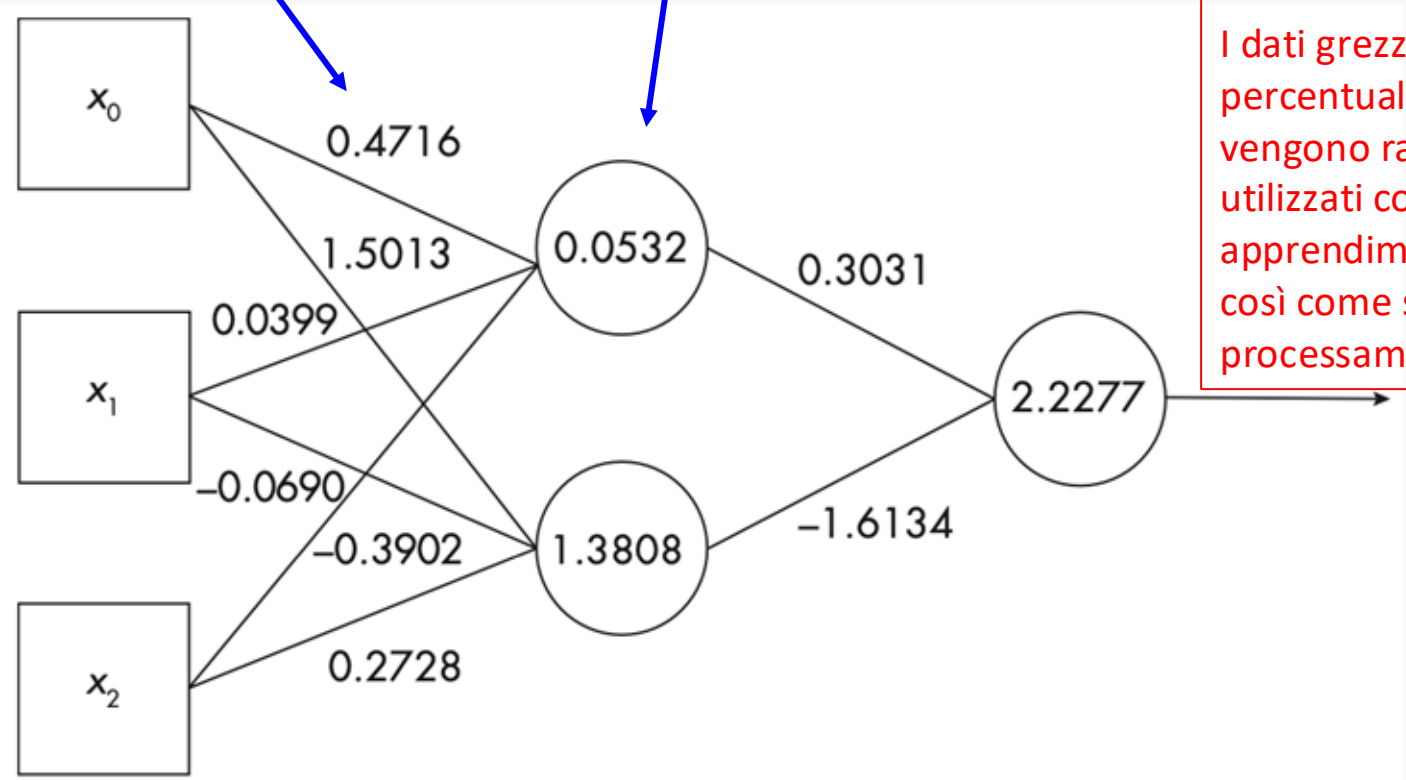


Figure 4-3: The two-neuron model trained on the wine dataset

I dati grezzi, come la percentuale di alcol, vengono raramente utilizzati con i modelli di apprendimento automatico così come sono!!! (Pre-processamento)

ESEMPIO

Dataset: Varietà di uva per un vino italiano

Pre-processing

- Ogni caratteristica viene *regolata* sottraendo il valore medio della caratteristica sul set di addestramento e dividendo quel risultato per una misura di quanto siano sparsi i dati attorno al valore medio (la deviazione standard).
- Il contenuto di alcol originale era del 12,29%, un valore ragionevole per il vino, ma dopo il ridimensionamento è diventato -0,7359.

$$z = \frac{x_i - \mu}{\sigma}$$

Dove:

- z è la **standardizzazione**
- Xi è il punto dati originale.
- μ è la media del set di dati.
- σ è la deviazione standard del set di dati.

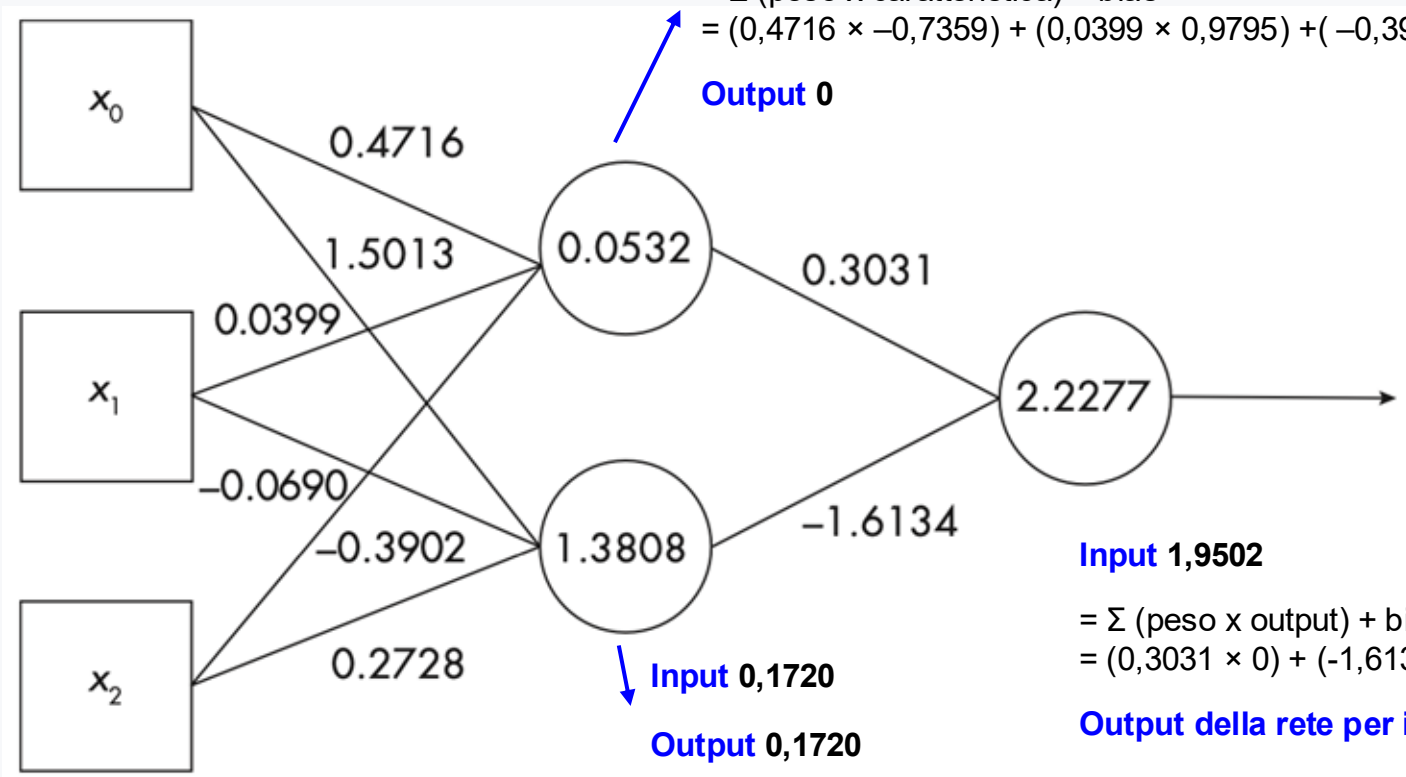
$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}}$$

Dove:

- σ è la **deviazione standard** della popolazione,
- N è il numero totale di dati nella popolazione,
- x_i è ogni singolo dato nella popolazione,
- μ è la media della popolazione.

Dataset: Varietà di uva per un vino italiano

Sample 1 (−0.7359, 0.9795, −0.1333)
Sample 2 (0.0967, −1.2138, −1.0500)



Input –0,2028

= Σ (peso x caratteristica) + bias
= (0,4716 × –0,7359) + (0,0399 × 0,9795) +(–0,3902 × –0,1333) + 0,0532

Output 0

Input 0,1720

Output 0,1720

Input 1,9502

= Σ (peso x output) + bias
= (0,3031 × 0) + (-1,6134 × 0,1720) + 2,2277

Output della rete per il 1° campione 0,8755

Come dovremmo interpretare questo output?

Output delle Reti Neurali

Le reti neurali non ci dicono l'etichetta di classe effettiva per l'input, ma solo la loro confidenza in un'etichetta rispetto a un'altra.

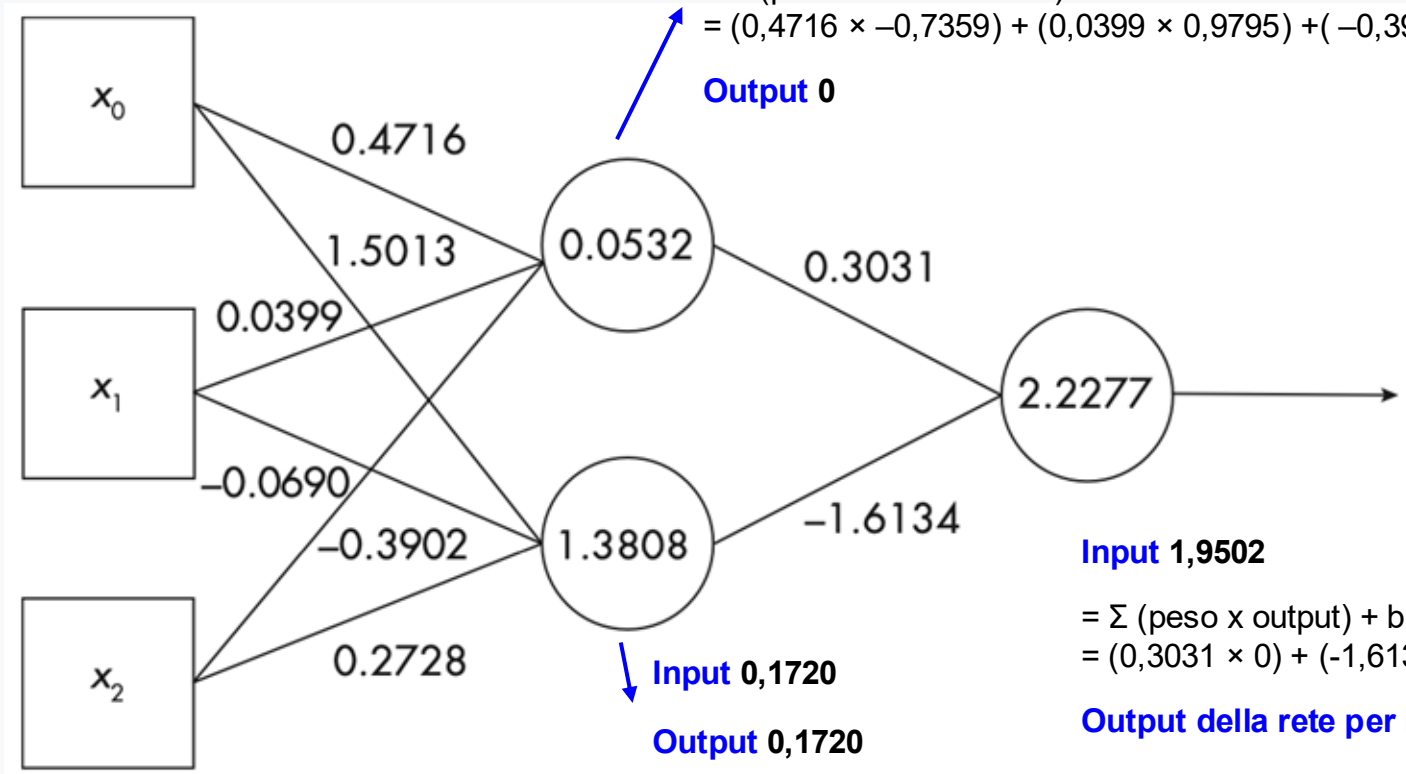
Output delle Reti Neurali (2)

- I modelli binari emettono un valore di confidenza che interpretiamo come la probabilità che l'input appartenga alla classe 1.
- Le probabilità sono numeri compresi tra 0 (nessuna possibilità) e 1 (assolutamente certo).
- Utilizziamo una soglia, un valore di *cutoff*, per decidere quale etichetta assegnare.
- L'approccio più comune per i modelli binari è una soglia del 50 %. Se l'output supera il 50 % (probabilità 0,5), assegniamo l'input alla classe 1.

ESEMPIO

Dataset: Varietà di uva per un vino italiano

Sample 1 (−0.7359, 0.9795, −0.1333)
Sample 2 (0.0967, −1.2138, −1.0500)



Input –0,2028

= Σ (peso x caratteristica) + bias
= (0,4716 × –0,7359) + (0,0399 × 0,9795) +(–0,3902 × –0,1333) + 0,0532

Output 0

Input 0,1720

Output 0,1720

Input 1,9502

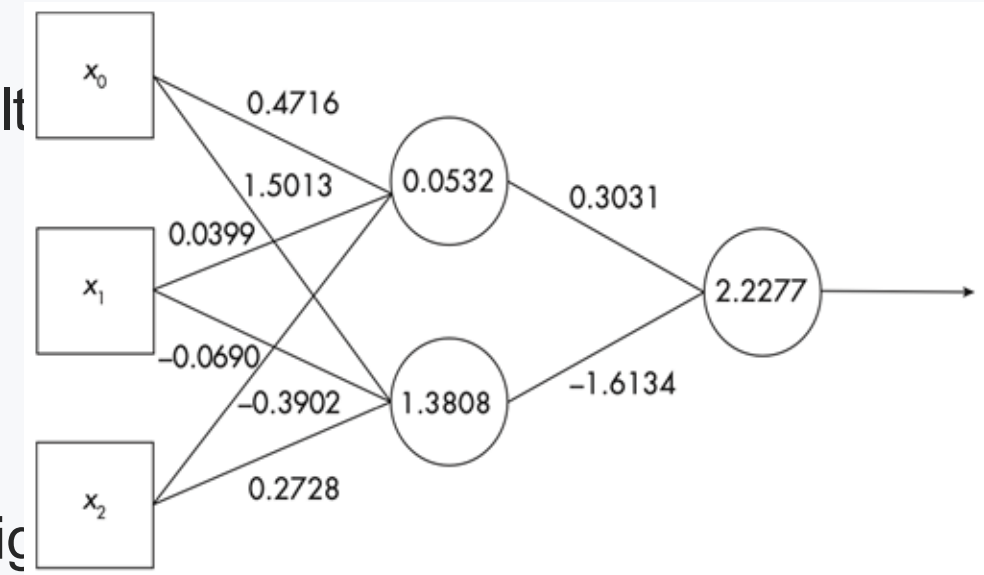
= Σ (peso x output) + bias
= (0,3031 × 0) + (-1,6134 × 0,1720) + 2,2277

Output della rete per il 1° campione 0,8755

Questo output è superiore al 50 %, quindi assegniamo "classe 1" come etichetta

Dataset: Varietà di uva per un vino italiano

- Addestriamo tutti i modelli utilizzando gli stessi set di addestramento e test
- Ripetiamo il processo 240 volt per ciascuno dei tre modelli
- Ecco le accuratezze medie:
  - **2 nodi** 81,5 %
  - **3 nodi** 83,6 %
  - **8 nodi** 86,2 %
- Le prestazioni del modello mig di nodi nello strato nascosto
- *Un modello più complesso (più nodi) implica la capacità di apprendere associazioni più complesse nascoste all'interno del set di addestramento*



Inizializzazione delle Reti Neurali

Le reti neurali vengono inizializzate in modo casuale, in modo tale che l'addestramento ripetuto porti a modelli con prestazioni diverse anche quando si utilizzano gli stessi dati di addestramento.

Inizializzazione delle Reti Neurali (2)

- Le NN vengono inizializzate casualmente
  - L'addestramento ripetuto porta a modelli con prestazioni diverse anche utilizzando gli stessi dati.
  - Questa casualità è necessaria per garantire l'apprendimento.
- L'importanza dell'inizializzazione casuale
  - Inizializzare i pesi con lo stesso valore forza i pesi ad apprendere caratteristiche simili.
  - Se tutti i pesi iniziali sono zero, il modello non apprende affatto.
  - Una raccolta casuale di valori iniziali permette al processo iterativo di affinare i pesi e bias.

Inizializzazione delle NN (3)

- Perché **l'inizializzazione casuale è necessaria**
  - Inizializzare i pesi con lo stesso valore forza i pesi ad apprendere caratteristiche simili
  - Se tutti i pesi iniziali sono zero, il modello non apprende affatto
  - **Una raccolta casuale di valori iniziali permette al processo iterativo di affinare i pesi e bias**
- Effetti dei pesi e bias iniziali
  - Ogni set di pesi e bias iniziali porta a risultati finali diversi
- Esempio precedente: Accuratezze di una rete su dati uguali
  - 89%, 85%, 73%, 81%, 85% (variazione dovuta all'inizializzazione casuale)
- **Le reti dovrebbero essere addestrate più volte**
  - Le reti più piccole mostrano una maggiore variazione rispetto ai modelli più grandi

Riepilogo:

- Neurone (Nodo):
  - Moltiplica input per pesi, somma i prodotti, aggiunge un bias e applica la funzione di attivazione.
- Strati di Rete:
  - Reti formate da neuroni disposti in strati.
  - L'output di uno strato diventa l'input del successivo.
- Addestramento:
  - Regola iterativamente i pesi e bias iniziali casuali.
- Reti Neurali Binarie:
  - Producono output che rappresenta la probabilità che l'input appartenga alla classe 1.
  - Utilizzano una funzione di attivazione per generare una stima probabilistica.

Da dove provengono pesi e bias?

- I pesi e i bias sono appresi tramite due algoritmi fondamentali:
  - Backpropagation
  - Gradient Descent
- L'addestramento è un processo di ottimizzazione:
  - Trova i pesi e i bias che meglio si adattano ai dati di addestramento.
- Obiettivo:
  - Adattarsi alle tendenze generali nei dati.
  - Evitare di apprendere dettagli troppo specifici.
- Maggiori dettagli emergeranno esplorando il processo di addestramento.

Algoritmo di training generale

1. Selezione dell'Architettura:
   - Numero di layer nascosti, nodi per layer e funzione di attivazione.
2. Inizializzazione:
   - Pesi e bias inizializzati casualmente ma in modo intelligente.
3. Forward Pass:
   - Esegui i dati di training attraverso il modello.
   - Calcola l'errore medio.
4. Backward Pass:
   - Usa la backpropagation per stimare l'impatto di ogni peso e bias sull'errore.
   - Aggiorna i pesi con il gradient descent.
5. Iterazione:
   - Ripeti i passaggi dal 3 al 5 finché il modello non è “abbastanza buono”.

Nota Importante:

- **Architettura:** Include layer nascosti e vettori di input/output.
- **Errore Medio:** Misura quanto l'output previsto differisce dai dati reali.
- **Training Iterativo:** Simile a camminare dal punto A al punto B: passi incrementali verso un modello accurato.

Gestione degli errori e overfitting

- Calcolo dell'errore:
  - Differenza tra l'output previsto e quello reale.
  - Esempio: Output previsto 0,44, reale 1 → errore = 0,56.
- Obiettivo del training:
  - Ridurre l'errore medio su tutti i dati di training.
  - Errore prossimo a zero → rete performante sui dati di training.
- Overfitting:
  - Rete apprende dettagli specifici dei dati → scarsa generalizzazione.
  - Soluzioni per l'overfitting:
    - » Più dati di training.
    - » Decadimento del peso: Penalizza pesi elevati.
    - » Aumento dei dati: Modifica dati esistenti per generarne di nuovi.

Inizializzazione dei pesi e algoritmi di training

- Inizializzazione intelligente dei pesi:
  - Basata su funzione di attivazione, fan-in e fan-out.
  - Bias inizializzati a zero, pesi scelti con formule specifiche.
- Vantaggi dell'inizializzazione avanzata:
  - Migliore performance rispetto ai vecchi metodi casuali.
- Algoritmi di training:
  - Gradient Descent:
    - » Riduce l'errore aggiornando pesi e bias.
  - Backpropagation:
    - » Determina il contributo di ogni peso/bias all'errore.
  - Entrambi lavorano insieme per ottimizzare il modello.

Applichiamo le reti neurali tradizionali al dataset delle impronte di dinosauro

Selezione dell'architettura della rete neurale



Output e Funzione di Attivazione:

- Nodo di output → Sigmoide (probabilità per classe 1: teropode).
- Decisione: probabilità > 50% → classe 1; altrimenti classe 0.

- **Strati Nascosti:**
  - Funzioni di attivazione: Unità Lineari Rettificate (ReLU).
  - Numero di strati nascosti limitato: dataset piccolo con 1.336 campioni.

Nodi per Layer:

- Primo layer: da piccolo a circa il doppio delle 1.600 feature di input.
- Secondo layer (se presente): al massimo metà del primo layer.

ESEMPIO

Applichiamo le reti neurali tradizionali al dataset delle impronte di dinosauro



Risultati delle Architetture Testate

Approccio ai test:

- Raccolta di architetture a uno e due layer.
- Migliore architettura testata 100 volte → accuratezza media.

Risultati:

- Miglior modello: due layer nascosti, rispettivamente con:
  - 800 nodi (primo layer) e 100 nodi (secondo layer).
- Accuratezza media: 77,4% (min 69,3%; max 81,5%).

Table 4-1: Trial Architectures with the Dinosaur Footprint Dataset

Accuracy (%)	Architecture	Weights and biases
59.4	10	16,021
77.0	400	640,801
76.7	800	1,281,601
81.2	**2,400**	**3,844,801**
75.8	100, 50	165,201
81.2	**800, 100**	**1,361,001**
77.9	2,400, 800	5,764,001

Applichiamo le reti neurali tradizionali al dataset delle impronte di dinosauro

Addestramento

- Addestramento del modello 100 volte sul dataset delle impronte
  - accuratezza media del 77,4 %, con un minimo del 69,3 % e un massimo dell'81,5 %.
- Mettiamo questo risultato nella sua corretta relazione con quelli del ML classico



Table Dinosaur Footprint Models

Model	Accuracy (%)
RF300	83.3
RBF SVM	82.4
7-NN	80.0
3-NN	77.6
MLP	**77.4**
1-NN	76.1
Linear SVM	70.7

La nostra rete neurale non è stata la migliore in questo set di dati. In effetti, è stata una delle peggiori... dobbiamo aspettare la rivoluzione del deep learning per vedere qualcosa di più speciale?

MLP (perceptron multistrato) modello di rete neurale artificiale che consiste in più strati di neuroni collegati in cascata

Ricapitolando:

- Le reti neurali sono raccolte di nodi (**neuroni**) che accettano input multipli e producono un singolo numero come output.
- Le reti neurali sono spesso disposte in **strati** in modo che l'input dello strato corrente sia l'output dello strato precedente.
- Le reti neurali sono inizializzate in modo **casuale**, quindi un addestramento **ripetuto** porta a modelli con prestazioni diverse.
- Le reti neurali sono addestrate tramite **discesa del gradiente**, utilizzando la direzione del gradiente fornita dalla **backpropagation** per aggiornare i pesi e i bias in modo iterativo.