

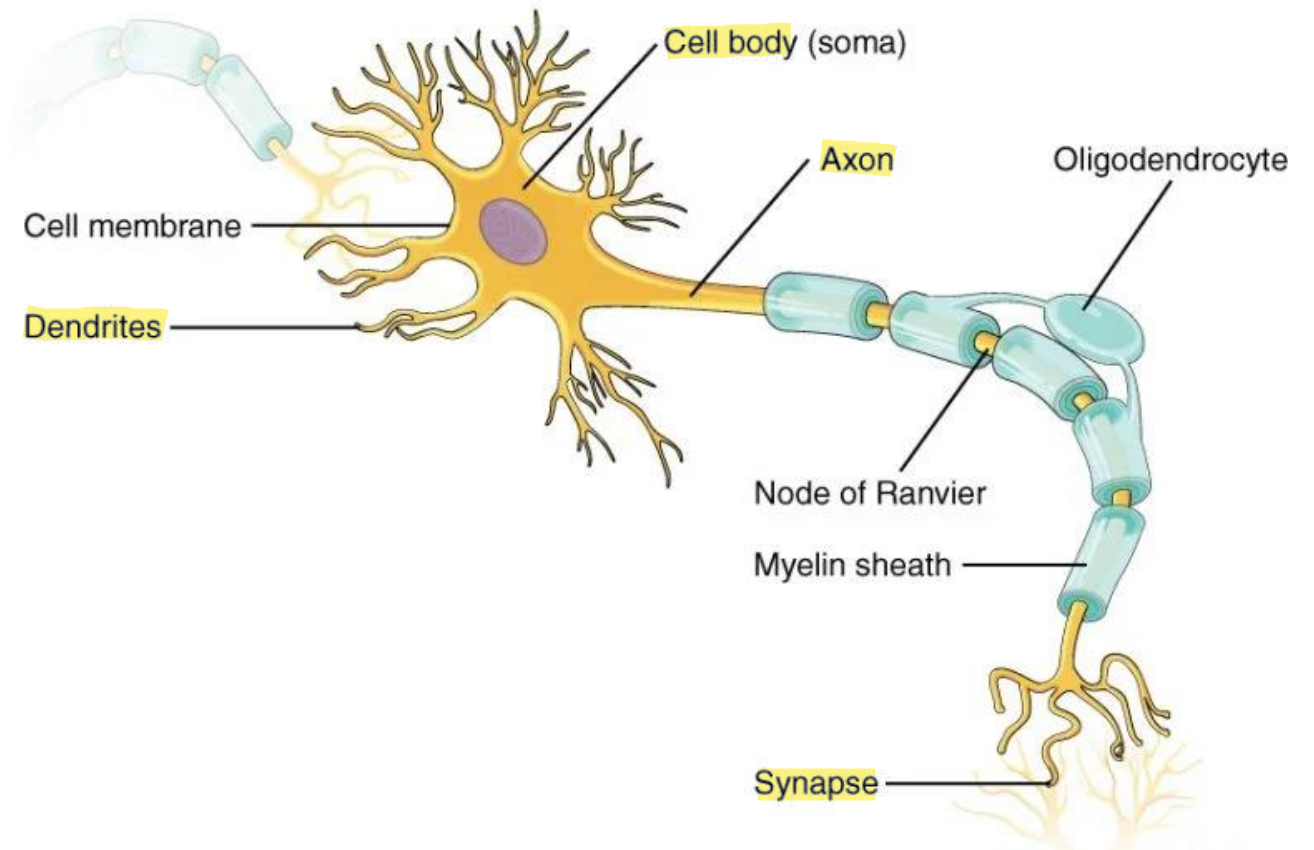


Artificial Neural Networks

Il neurone Biologico

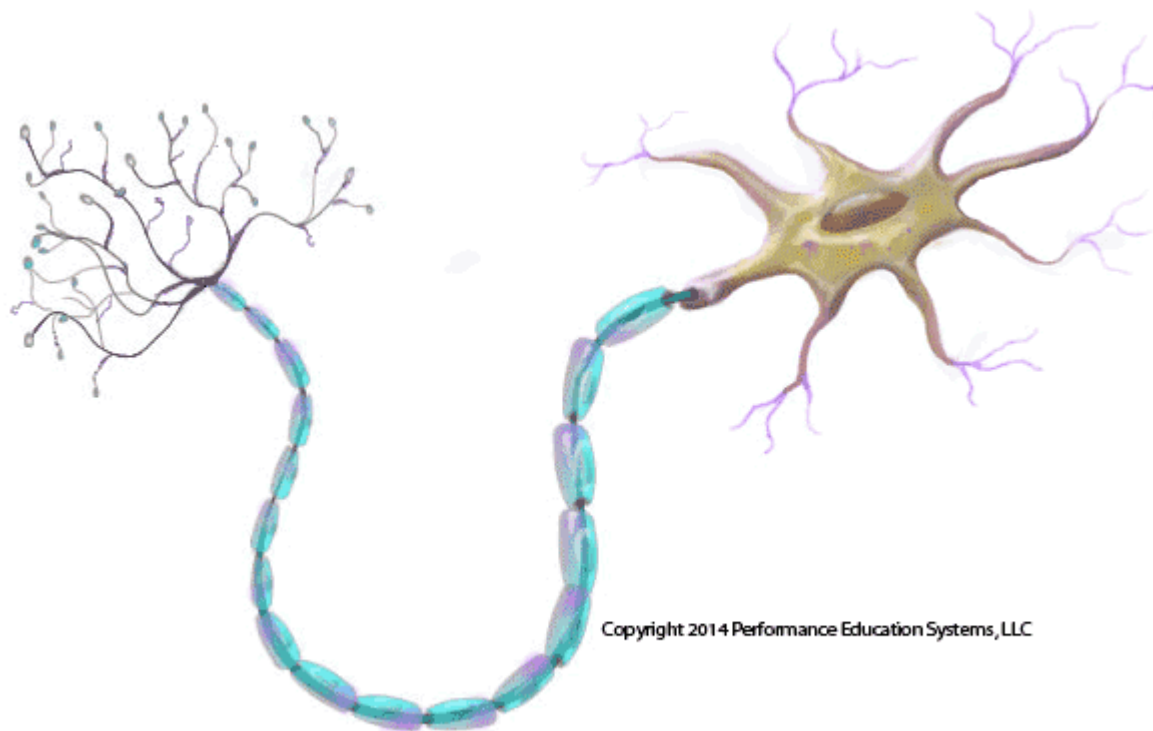
Il neurone biologico è composto da 4 parti principali:

- Il corpo cellulare o **soma**,
 - Le estensioni delle cellule, **dendriti**
 - Un'ulteriore estensione, **assone**
 - Le **sinapsi**
- I **dendriti** ricevono (*input*) segnali elettrici e chimici dagli altri neuroni e li trasmettono al **soma**.
 - il **soma** elabora i segnali in ingresso nel tempo e converte il valore elaborato in un output.
 - **L'assone**, invece, trasmette il segnale elettrico dal soma ad altri neuroni o a cellule muscolari o ghiandolari.
 - All'estremità dell'assone ci sono le **sinapsi**, che collegano il neurone ad altri neuroni per trasmettere il segnale in uscita (*output*).



Ogni singola **sinapsi** può modificare la propria risposta e variare, in questo modo, l'efficienza di trasporto dell'informazione.

Il neurone Biologico

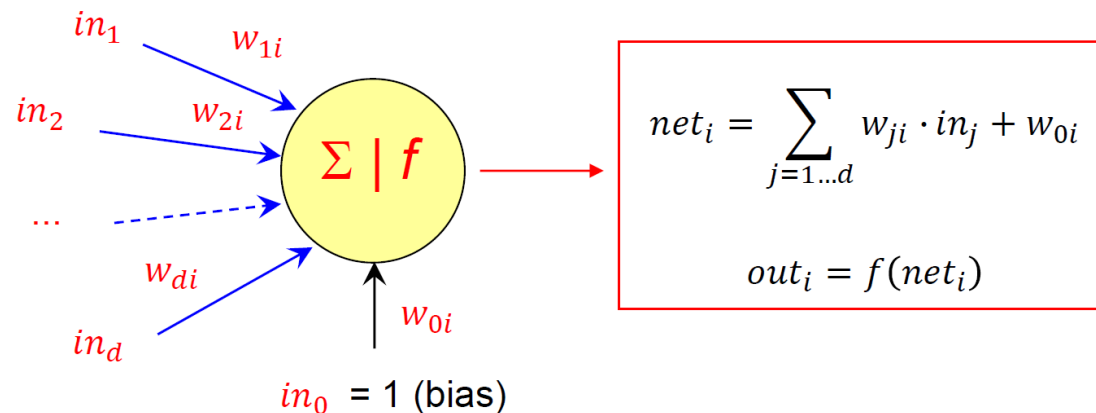
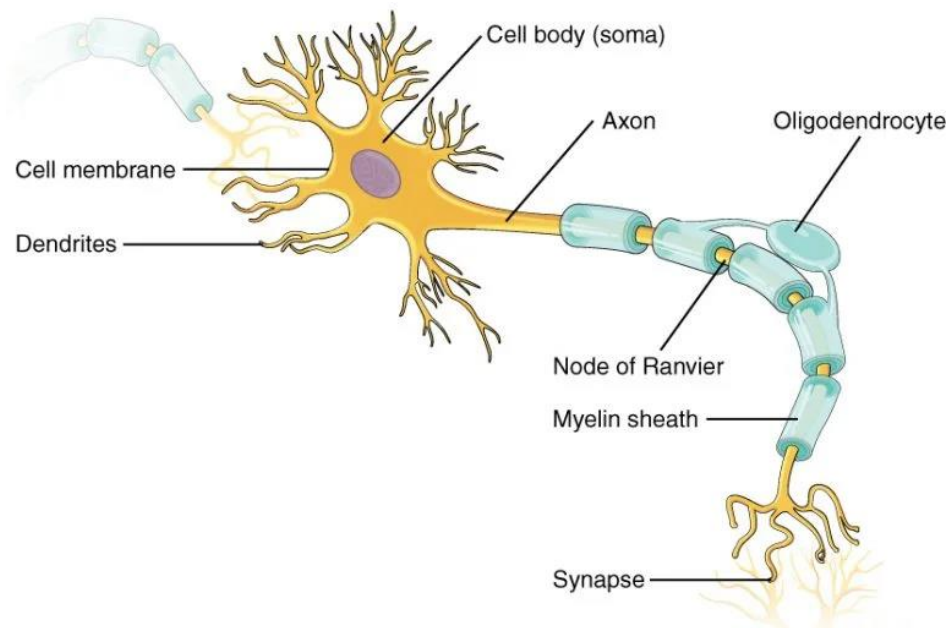


Quando il soma riceve i segnali in ingresso dai dendriti, esegue una “elaborazione”. **Se i segnali ricevuti superano una certa soglia**, viene prodotto un nuovo segnale di uscita sull’assone. Questo segnale si propagherà ad altri neuroni, anche molto distanti tra loro.

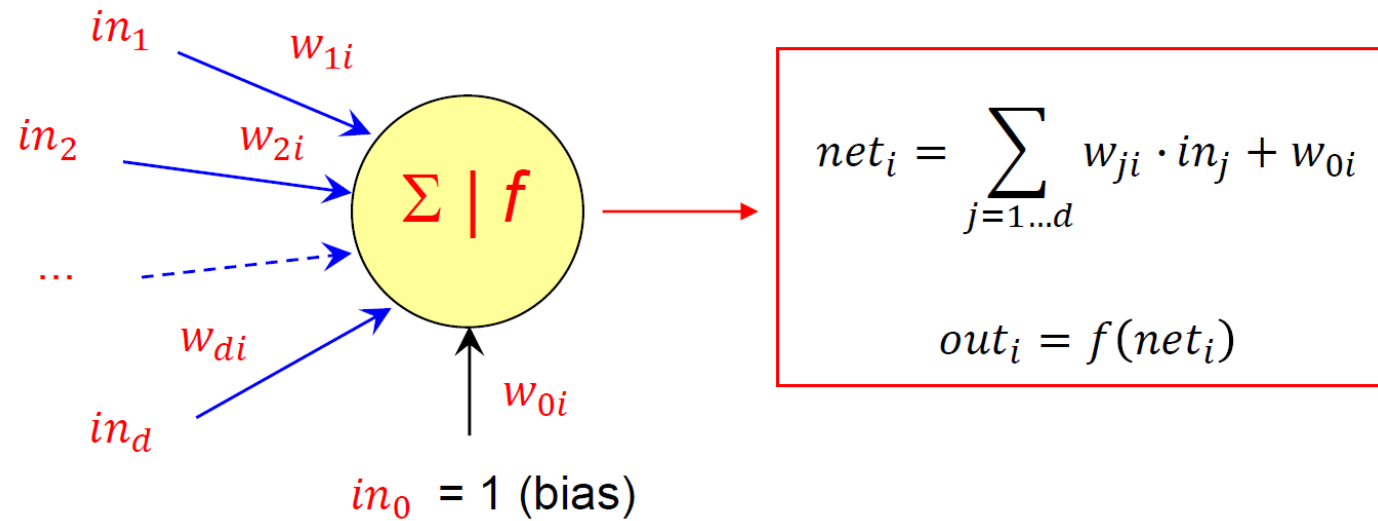
→ **Il valore di questa soglia e l’efficienza di trasmissione elettrochimica delle sinapsi sono strettamente legate ai processi di apprendimento.**

Neurone Artificiale

- che descriveva il primo modello matematico di un neurone artificiale
- Primo schema di funzionamento del neurone biologico ad opera di McCulloch and Pitts (1943)
- 1957: Rosenblatt inventa il percettrone (**perceptron**), primo esempio di rete neurale:
 - Formulazione matematica** del neurone artificiale del 1943
 - Idea della possibilità di **addestrarlo** in base all'errore dell'output



Neurone Artificiale



- in_1, in_2, \dots, in_d sono i **d ingressi** che il neurone i riceve da assoni di neuroni afferenti
- $w_{1i}, w_{2i}, \dots, w_{di}$ sono i pesi (**weight**) che determinano l'efficacia delle connessioni sinaptiche dei dendriti (agiremo su questi valori durante l'apprendimento), l'importanza dell'input i -esimo sull'output.
- w_{0i} (detto **bias**) è un ulteriore peso che si considera collegato a un input fittizio con valore sempre 1 questo peso è utile per «tarare» il punto di lavoro ottimale del neurone.
- $f(\cdot)$ è la **funzione di attivazione** simula il comportamento del neurone biologico di attivarsi solo se i segnali in ingresso **superano una certa soglia**. È una sorta di **interruttore** della cellula.

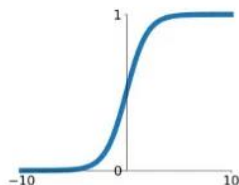


Funzioni di attivazione

- Una funzione di attivazione determina se un neurone deve essere **attivato o meno**.
- Si tratta di alcune semplici **operazioni matematiche** per determinare se l'input del neurone alla rete è rilevante o meno nel processo di previsione.
- Le funzioni di attivazione possono essere di diversi tipi, ma in generale devono essere **non lineari** per consentire alla rete di apprendere relazioni complesse tra le sue variabili di input, e **derivabili**.

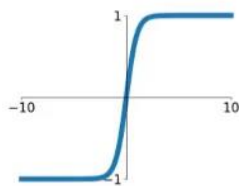
Sigmoid

$$f(x) = \frac{1}{1+e^{-x}}$$



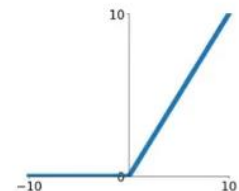
tanh

$$\tanh(x)$$



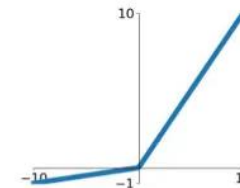
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

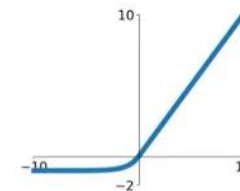


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

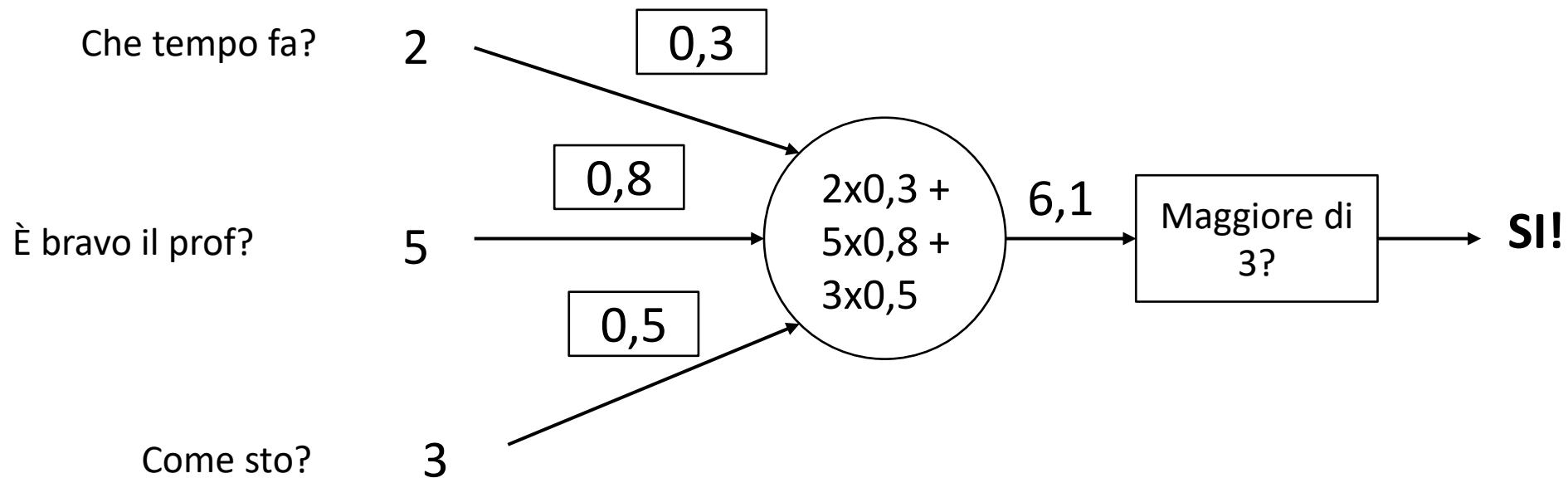
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





Esempio pratico

Stamattina esco per andare alla lezione?





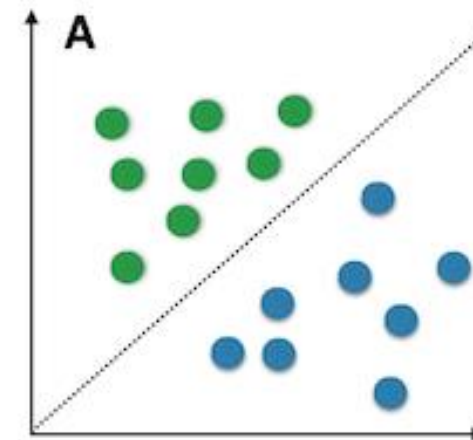
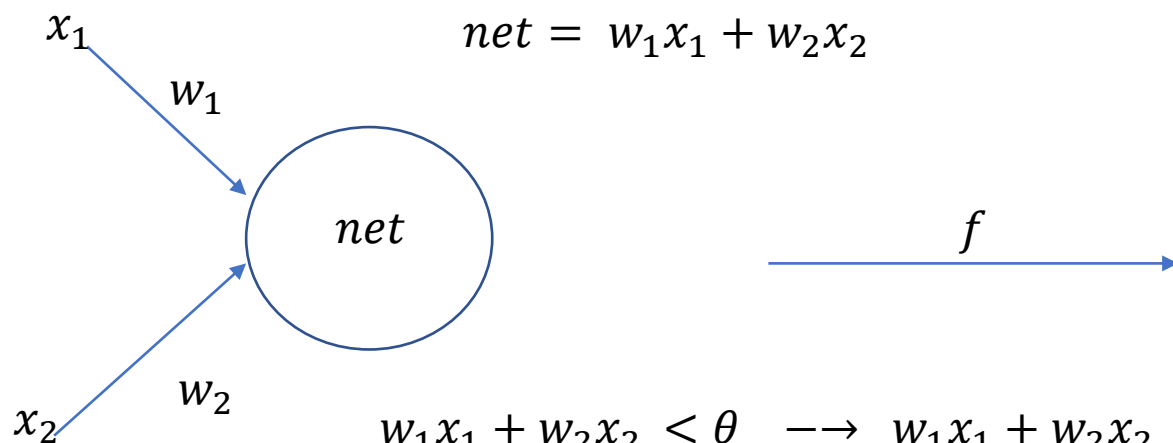
Esempio pratico (più generale)

Singolo neurone artificiale che effettua una separazione lineare tra due classi

Per semplicità consideriamo
il caso di 2 soli input, con
bias=0

$$f(net) = \begin{cases} -1 & \text{se } net < \theta \\ 1 & \text{se } net > \theta \end{cases}$$

Funzione di attivazione a soglia θ



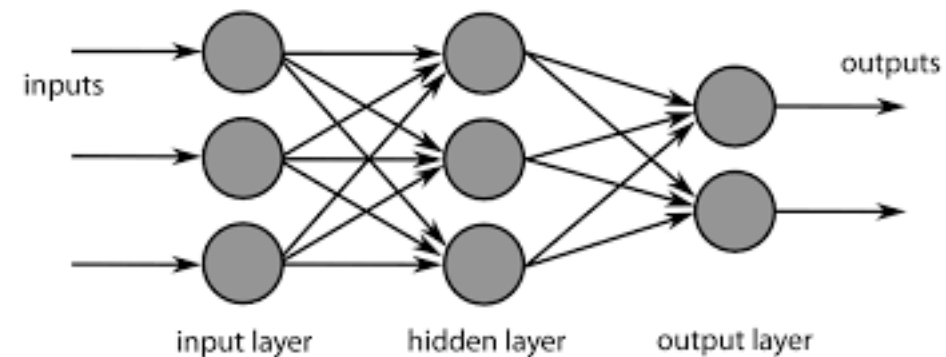
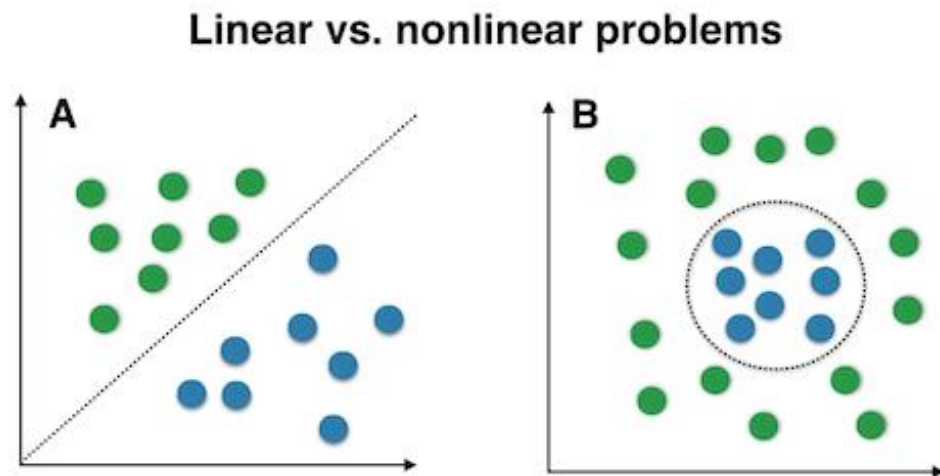
$$w_1x_1 + w_2x_2 < \theta \quad \rightarrow \quad w_1x_1 + w_2x_2 - \theta < 0 \quad (x_1, x_2) \text{ sta sopra la retta}$$

$$w_1x_1 + w_2x_2 > \theta \quad \rightarrow \quad w_1x_1 + w_2x_2 - \theta > 0 \quad (x_1, x_2) \text{ sta sotto la retta}$$

Equazione implicita della retta $w_1x_1 + w_2x_2 - \theta = 0$

Per poter avere separazioni più complicate rispetto a quelle lineari:

- la soluzione è utilizzare più neuroni artificiali organizzati su diversi (*tanti*) strati (layer) **Multilayer Perceptron (MLP)**





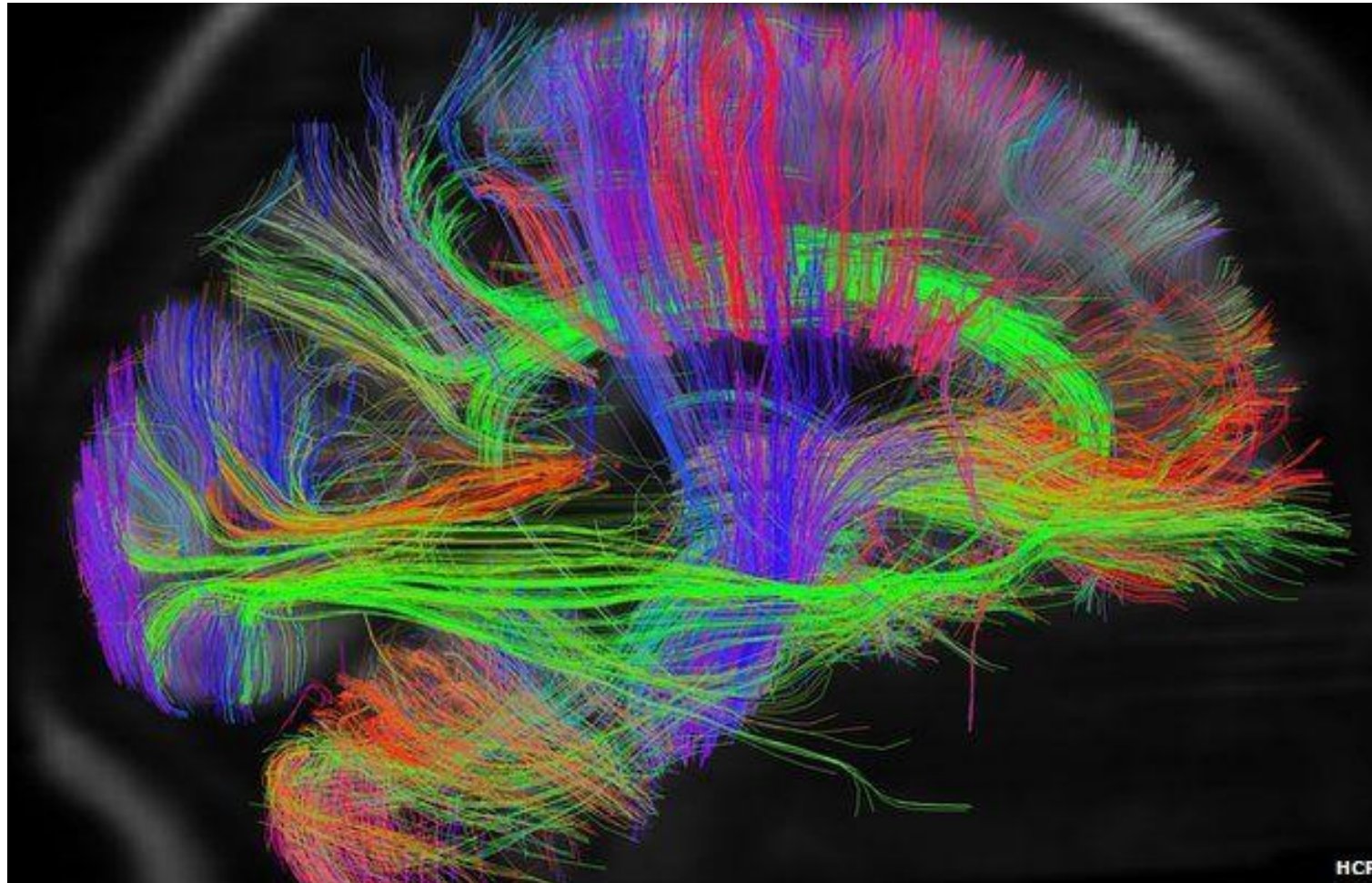
Cervello umano

- **È quanto accade anche in natura!**

- Un singolo neurone biologico è un elemento debole, ma **connesso con miliardi di altri neuroni** diventa una potente **rete**, chiamata cervello.
- Il cervello umano contiene circa 100 miliardi di neuroni (10^{11}) che comunicano tramite segnali elettrici e chimici attraverso più di **100 trilioni (10^{14}) di sinapsi**.

- **Nuove sinapsi vengono costantemente create in modo casuale.**
 - Se due neuroni connessi da una sinapsi si attivano **nello stesso momento**, la sinapsi che li connette è **rinforzata → abbiamo imparato**
 - Se due neuroni connessi da una sinapsi **non si attivano mai insieme**, la sinapsi si **indebolisce** finché viene rimossa **→ abbiamo dimenticato**
- È il motivo per cui anche se nasciamo con già tutti i neuroni, non sappiamo già tutto.

Connettoma

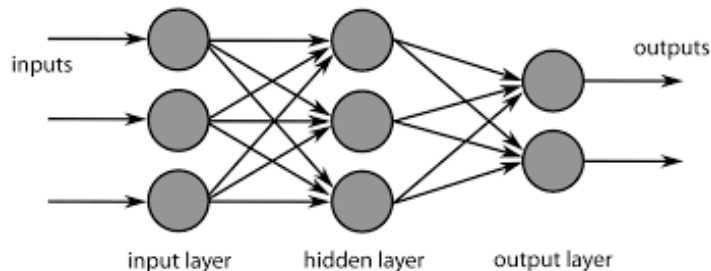


Tipologie di reti neurali

- Le reti neurali sono composte da gruppi di neuroni artificiali organizzati in **livelli**.
- Tipicamente sono presenti un livello di **input**, un livello di **output**, e uno o più **livelli intermedi**.
- Ogni livello contiene uno o più neuroni. I **layer intermedi** sono chiamati **hidden layer** in quanto restano “invisibili” dall'esterno della rete, la quale si interfaccia all'ambiente solo tramite il layer di ingresso e quello di uscita.

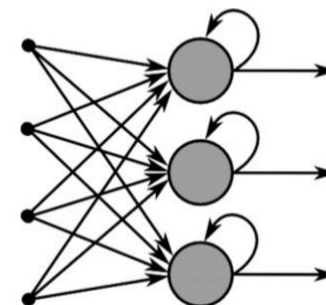
Feedforward (FFNN)

Nelle reti **feedforward** («alimentazione in avanti») le connessioni collegano i neuroni di un livello con i neuroni di un livello successivo. **Non** sono consentite connessioni all'indietro o connessioni verso lo stesso livello. È di gran lunga il tipo di rete più utilizzata.



Ricorrenti

Sono previste connessioni di **feedback**, in genere verso **neuroni dello stesso livello**, ma anche **all'indietro**. Questo tipo di rete crea una sorta di **memoria** di quanto accaduto in passato e rende quindi l'uscita attuale **non solo dipendente dall'ingresso attuale**, ma anche da tutti gli **ingressi precedenti**.





Come imparano le Reti Neurali?

- Abbiamo visto che le macchine possono apprendere dai **dati**
- Imparare dai dati spesso è simile a imparare a suonare la chitarra
 - *Uno* dei modi con cui l'essere umano impara

- Elementi principali:

- Devo osservare **come** si fa un accordo → dati annotati
 - Devo **tentar e ripetere** diverse volte → apprendimento iterativo
 - Non imparo **tutti** gli accordi subito → apprendimento per batch
 - Qualcuno mi deve **correggere** → funzione obiettivo/costo/loss
-



Addestramento di una rete

Devo ripetere tante volte
per diventare bravo

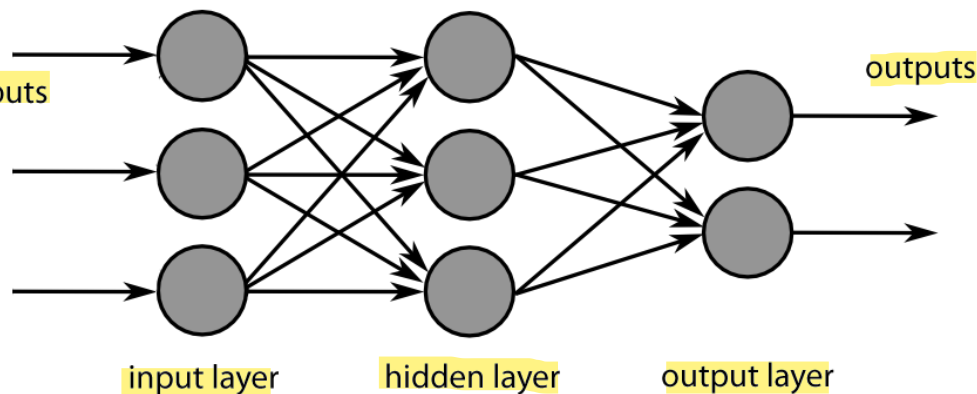
Ripetizione

1.

Dati di input divisi in
piccolo gruppi
(mini batch)

Ho tanti dati
annotati!

inputs



2.

Output
e
label

3.

Funzione di loss

Qualcuno mi deve dire
come sto andando.
È l'obiettivo del mio
apprendere!

4.

Errore

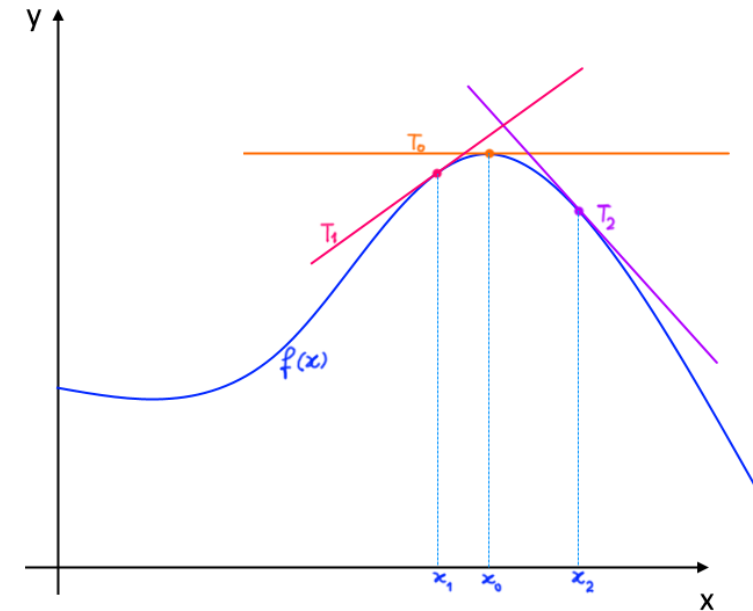
Cambio la posizione
delle mie dita per
ottenere accordi diversi



Problema

- Ma come faccio a capire in automatico se sto **incrementando o diminuendo l'errore**?
 - Quindi se mi sto avvicinando o allontanando dalla soluzione desiderata
- Soluzione: utilizzo la **matematica!**
 - Il mio obiettivo è una **funzione**
 - Calcolo la **derivata** di questa funzione
 - Capisco da che parte sto andando!

indica la variazione
della funzione in quel
punto





Addestramento di una rete

- Per una rete neurale

APPRENDERE

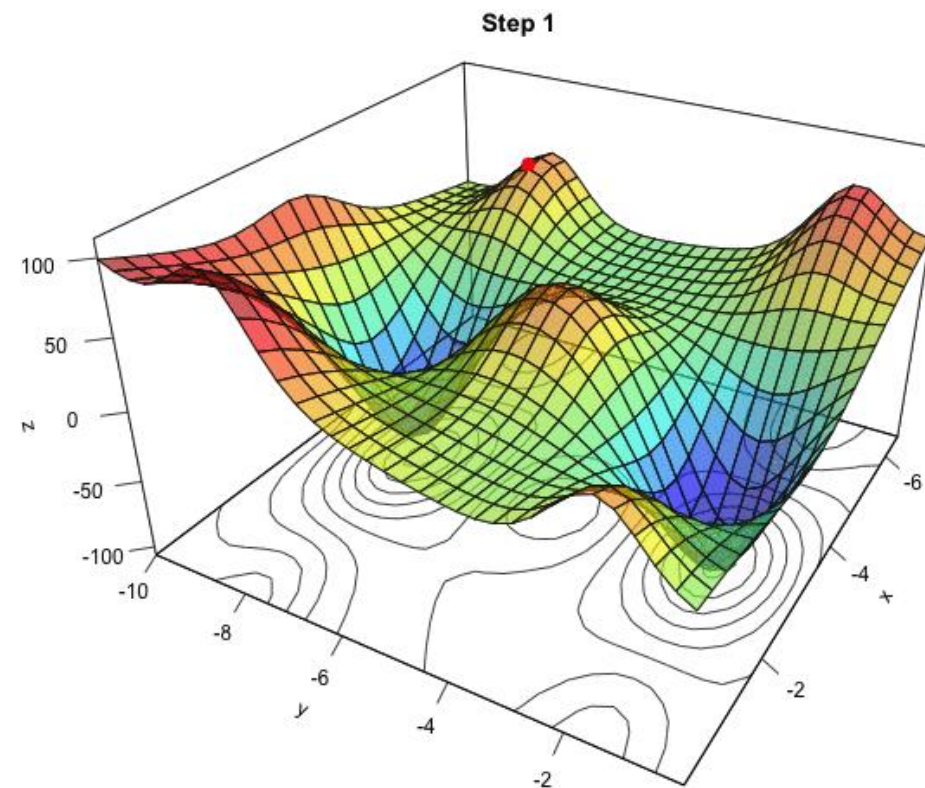
=

MINIMIZZARE LA FUNZIONE OBIETTIVO

Discesa del gradiente

Architettura Rete Neurale

- Fissata la topologia (numero di livelli e neuroni), l'addestramento di una rete neurale consiste nel determinare il valore dei pesi w che determinano il **mapping desiderato** tra input e output.
- Passo dopo passo, tramite la **discesa del gradiente**, mi avvicino all'obiettivo.





Riassumendo l'addestramento

Loss function → output desiderato della rete

↓ Come?

Minimizzando la loss function → mi avvicino all'obiettivo

↓ In pratica, come la realizzo?

Aggiustando i pesi (e bias) della rete

↓ tramite

Discesa del gradiente → Ottimizzatore

↓ Basato su

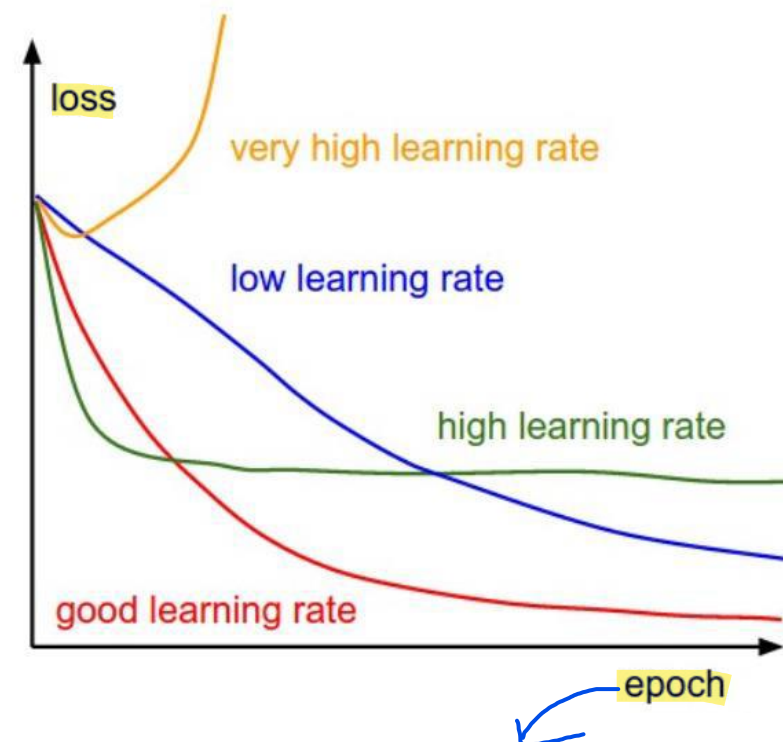
Backpropagation + Regola della catena (*chain rule*)

↓ Quindi

La funzione di loss deve essere derivabile!

Parametri:

- Learning rate
- Batch size
- ...



Un'epoca è un termine che si riferisce a un ciclo completo di addestramento attraverso l'intero set di dati di addestramento.



Cost Function

- La loss function, o funzione di perdita, è **una misura dell'errore della previsione prodotta da un modello di machine learning rispetto ai dati di training**. Essa rappresenta la **discrepanza tra l'output previsto dal modello e l'output reale associato ai dati di training**.
- In pratica, la scelta della loss function dipende dal tipo di problema di machine learning che si vuole risolvere.
- Ad esempio, se si sta risolvendo un **problema di classificazione binaria**, la loss function più comune è la funzione di entropia incrociata binaria **(Binary Cross-Entropy)**:

$$BCE = -y_i \cdot \log \hat{y}_i - (1 - y_i) \cdot \log(1 - \hat{y}_i)$$

- Se si sta risolvendo un problema di **classificazione multiclasse**, la loss function più comune è la funzione di **entropia incrociata categorica (Categorical Cross-Entropy)**:

$$CCE = - \sum_i y_i \cdot \log \hat{y}_i$$

In entrambi i casi, \hat{y}_i è lo i -esimo valore scalare emesso dal modello (*predizione*), y_i è la corrispondente label (*etichetta*).



Cost Function

- Per usare la Cross Entropy loss, nel layer di output dobbiamo avere delle **probabilità**
- Solitamente si utilizza il **softmax layer** per trasformare l'output della rete (*logits*) in probabilità (valori nel range [0, 1] che sommano a 1)
- Il softmax è una funzione continua e differenziabile

Example :

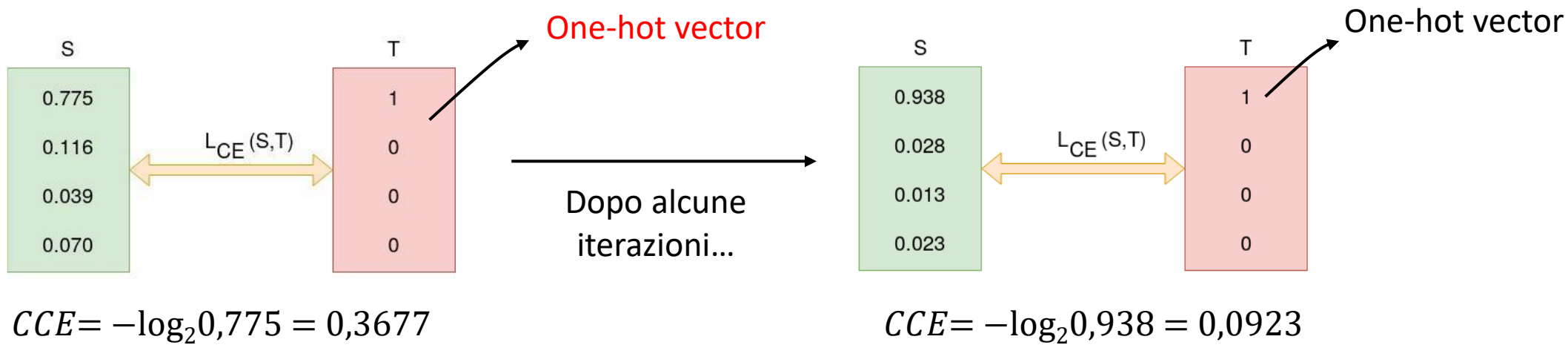
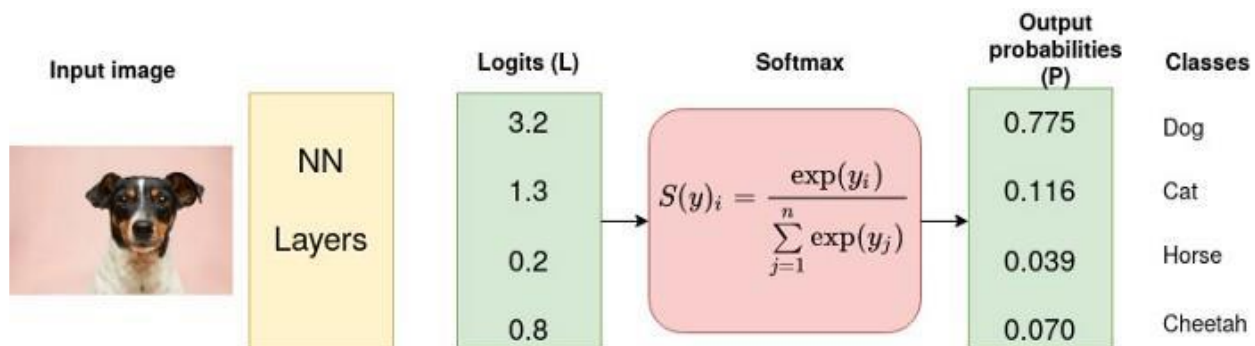
$$2.33 \rightarrow P(\text{Class 1}) = \frac{\exp(2.33)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.83827314$$

$$-1.46 \rightarrow P(\text{Class 2}) = \frac{\exp(-1.46)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.01894129$$

$$0.56 \rightarrow P(\text{Class 3}) = \frac{\exp(0.56)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.14278557$$

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^n e^{a_k}}$$

Addestramento di una rete



La loss è **scesa**, la rete ha **imparato**!



Cost Function

- Nel caso di un task di regressione, **la cost function** più comune:
 - **Errore quadratico medio (Mean Squared Error o MSE)**: definito come la media dei quadrati delle differenze tra l'output previsto dal modello e l'output reale associato ai dati di training.
In altre parole, l'MSE è calcolato come:

$$C(W) = \frac{\sum_{i=1}^n (y_i - \hat{y}_i(W))^2}{n}$$

dove n è il numero di esempi di training, y_i è l'**output reale associato a ciascun esempio di training** e $\hat{y}_i(W)$, che dipende dai parametri della rete che indichiamo con W , è l'output previsto dal modello per l'input corrispondente.

- Esistono anche altre cost function utilizzate in problemi di regressione, come ad esempio la cost function **di errore assoluto medio (Mean Absolute Error o MAE)**.

$$C(W) = \frac{\sum_{i=1}^n |y_i - \hat{y}_i(W)|}{n}$$

Esempio: MLP per il riconoscimento di cifre scritte a mano

