Artificial Neural Networks

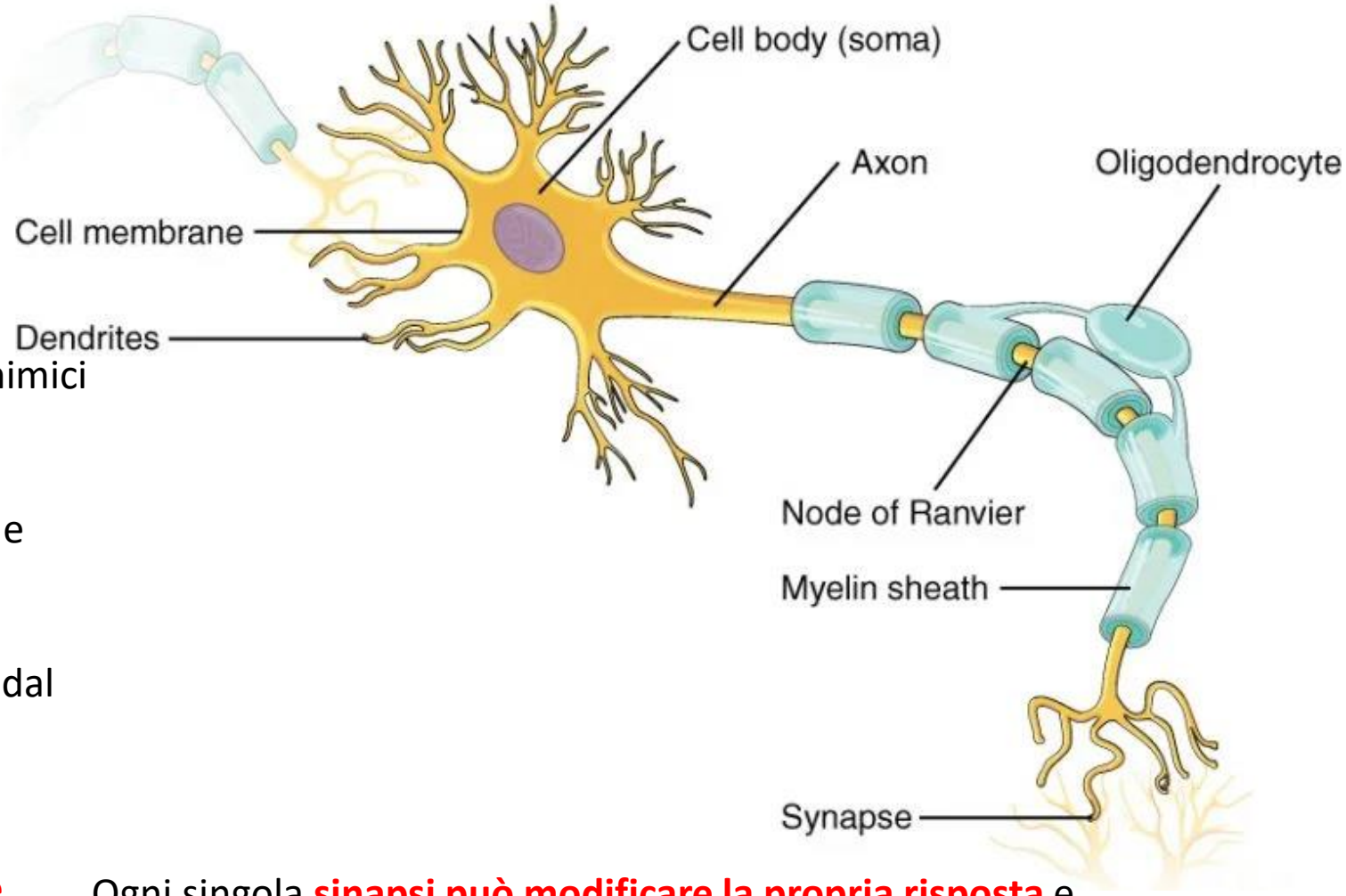
- Il paradigma delle reti neurali artificiali è **ispirato** al modo in cui il sistema **nervoso biologico elabora informazioni**.

- È composto da **un gran** numero di **elementi di elaborazione altamente interconnessi** che lavorano all'unisono per risolvere un problema specifico.

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**



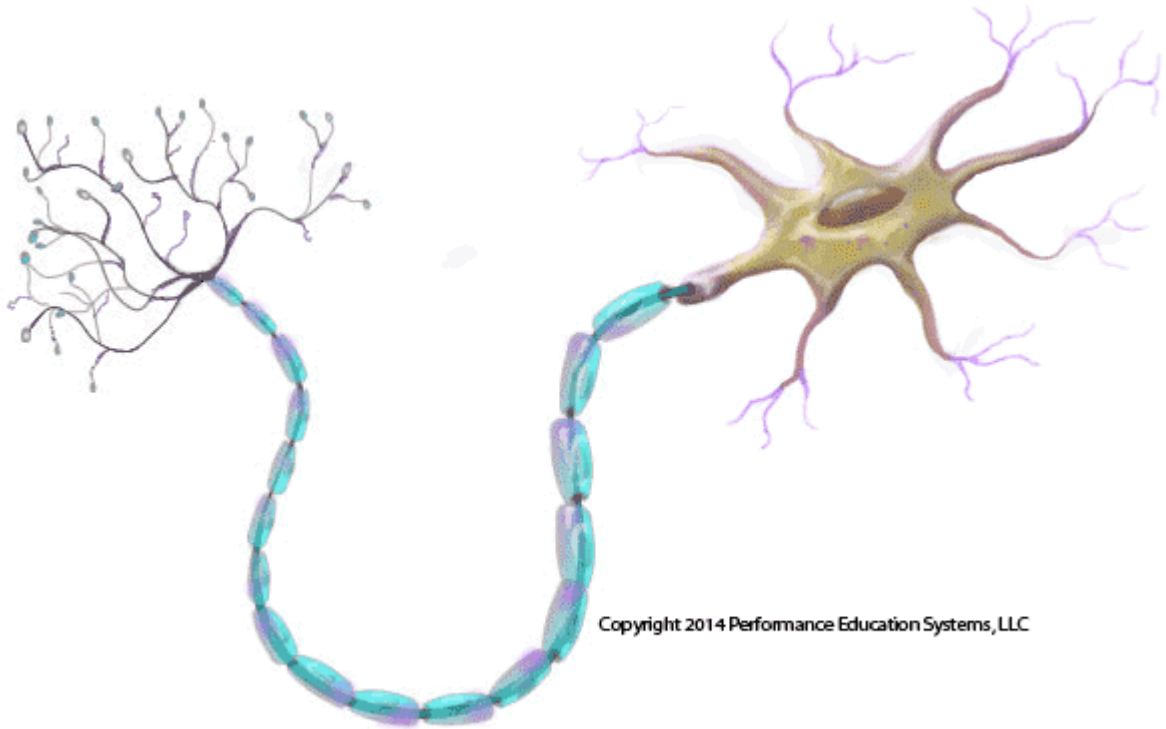
- I **dendriti** ricevono (*input*) segnali elettrici e chimici dagli altri neuroni e li trasmettono al **soma**.

- 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

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

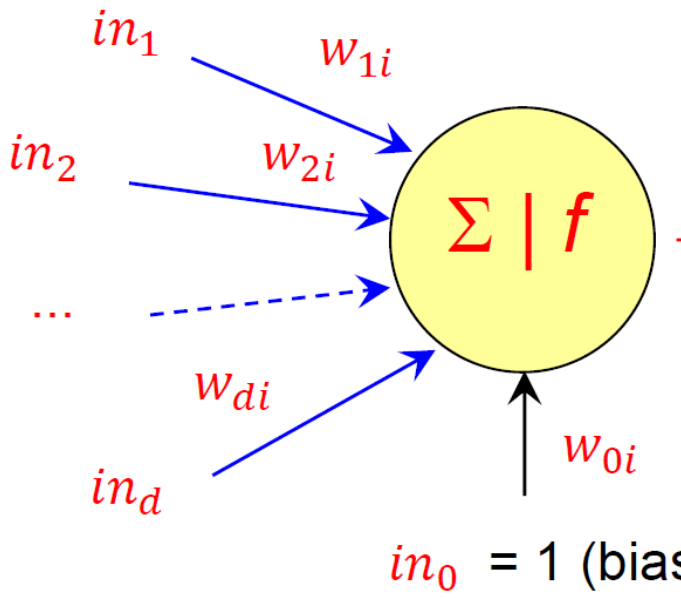


Quando il soma riceve i segnali in ingresso dai dendriti, esegue una “elaborazione”. **Se i segnali ricevuti superano una certa soglia**, viene prodotto un nuovo segnale di uscita sull’assone. Questo segnale si propagherà ad altri neuroni, anche molto distanti tra loro**. Il valore di questa soglia e l’efficienza di trasmissione elettrochimica delle sinapsi sono strettamente legate ai processi di apprendimento**.

Neurone Artificiale

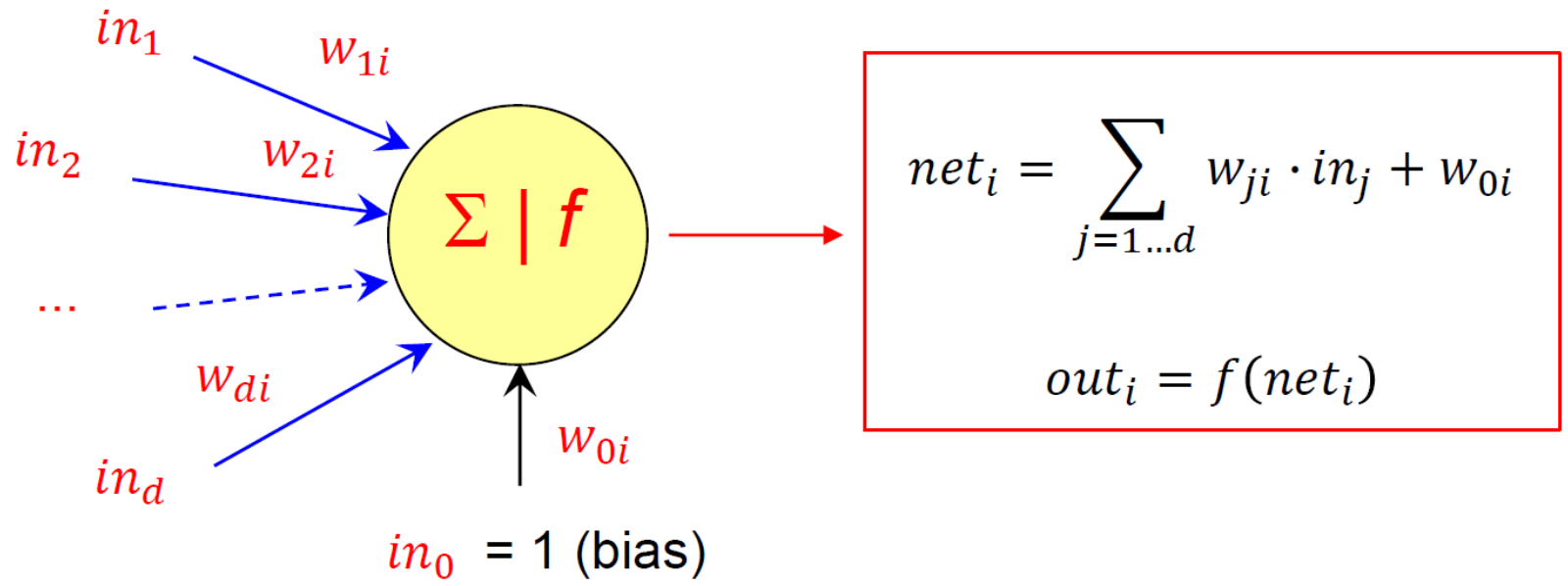
Primo modello del 1943 di McCulloch and Pitts. Con input e output binari era in grado di eseguire computazioni logiche.

Il neurone artificiale i-esimo

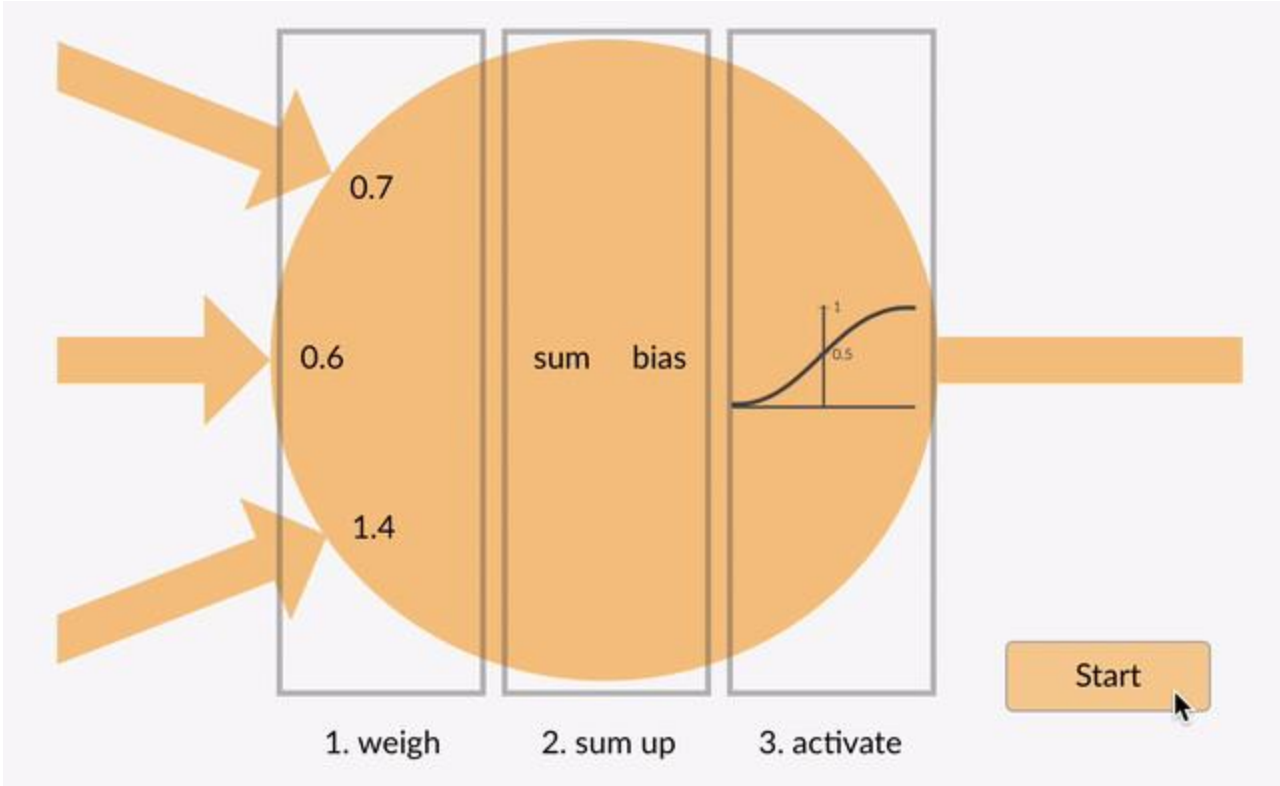


$$net_i = \sum_{j=1\ldots d} w_{ji} \cdot in_j + w_{0i}$$

$$out_i = f(net_i)$$



- $\boldsymbol{in_1, in_2, \ldots, in_d}$ sono i d ingressi che il neurone i riceve da assoni di neuroni afferenti
- $\boldsymbol{w_{1i}, w_{2i}, \ldots, w_{di}}$ 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.
- $\boldsymbol{w_{0i}}$(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.
- $\boldsymbol{net_i}$ è il livello di eccitazione globale del neurone (potenziale interno).
- $\boldsymbol{f(\cdot)}$ è la funzione di attivazione che determina il comportamento del neurone (ovvero il suo output out_i in
- funzione del suo livello di eccitazione net_i). La funzione di attivazione simula il comportamento del neurone biologico di attivarsi solo se i segnali in ingresso superano una certa soglia.

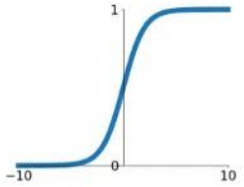

0.7
0.6
1.4
sum
bias
1
0.5
Start
1. weigh
2. sum up
3. activate

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.**

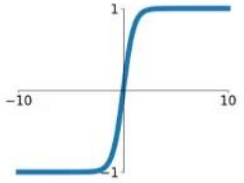
Sigmoid

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



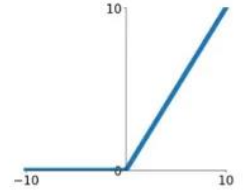
tanh

$\tanh(x)$



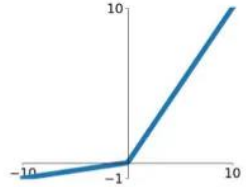
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

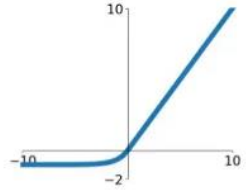


Maxout

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

ELU

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



Separabilità lineare di un percettrone a soglia

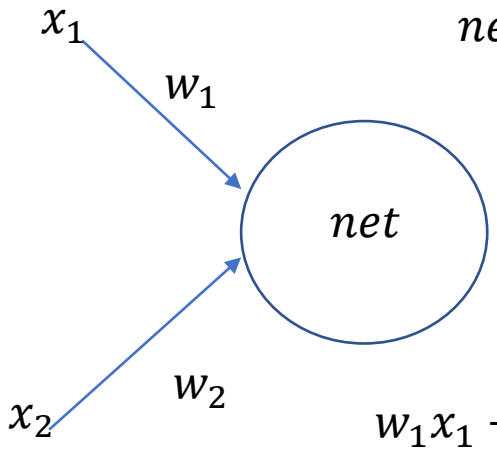
Perceptron

Modello single−neuron con funzione di attivazione a soglia . **Effettua una separazione lineare tra due classi**

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

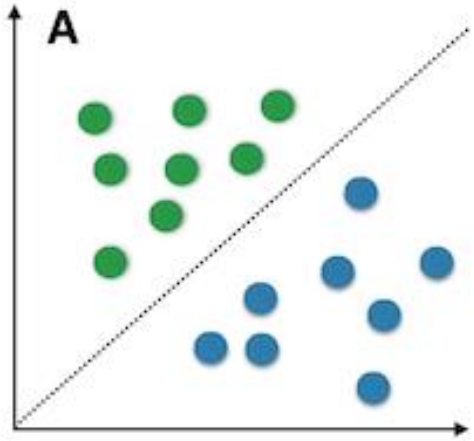
$$f(net) = \begin{cases} -1 & \text{se } net < \theta \\ 1 & \text{se } net > \theta \end{cases}$$

Funzione di attivazione a soglia θ



$$net = w_1 x_1 + w_2 x_2$$

$$f$$



$$w_1 x_1 + w_2 x_2 < \theta \longrightarrow w_1 x_1 + w_2 x_2 - \theta < 0 \qquad (x_1, x_2) \ sta \ sopra \ la \ retta$$

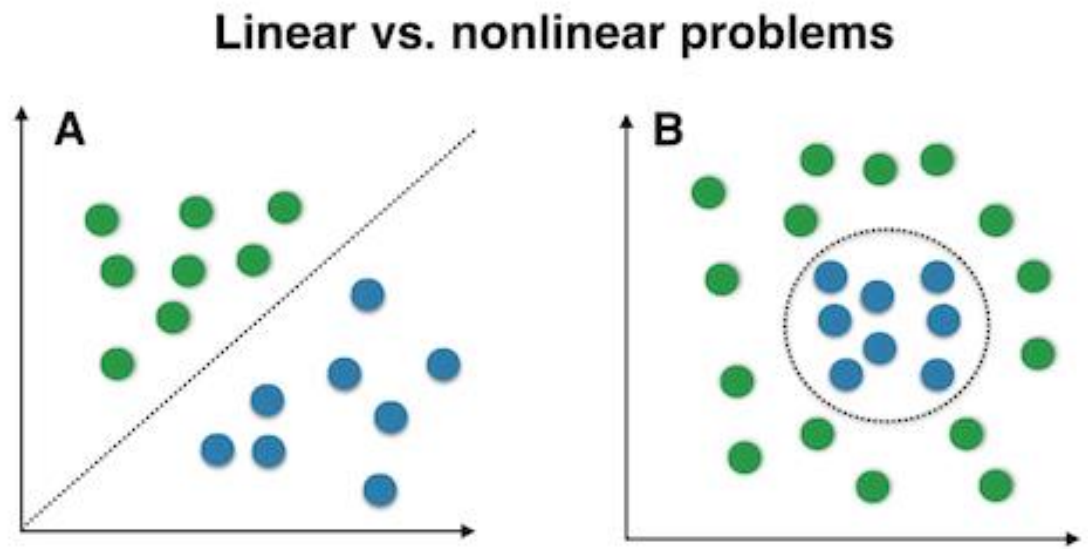
$$w_1 x_1 + w_2 x_2 > \theta \longrightarrow w_1 x_1 + w_2 x_2 - \theta > 0 \qquad (x_1, x_2) \ sta \ sotto \ la \ retta$$

Equazione implicita della retta $w_1 x_1 + w_2 x_2 - \theta = 0$

$$x_2 = -\frac{-\omega_1}{\omega_2} x_1 + \frac{\theta}{\omega_2}$$

Per poter avere separazioni più complicate rispetto a quelle lineari:

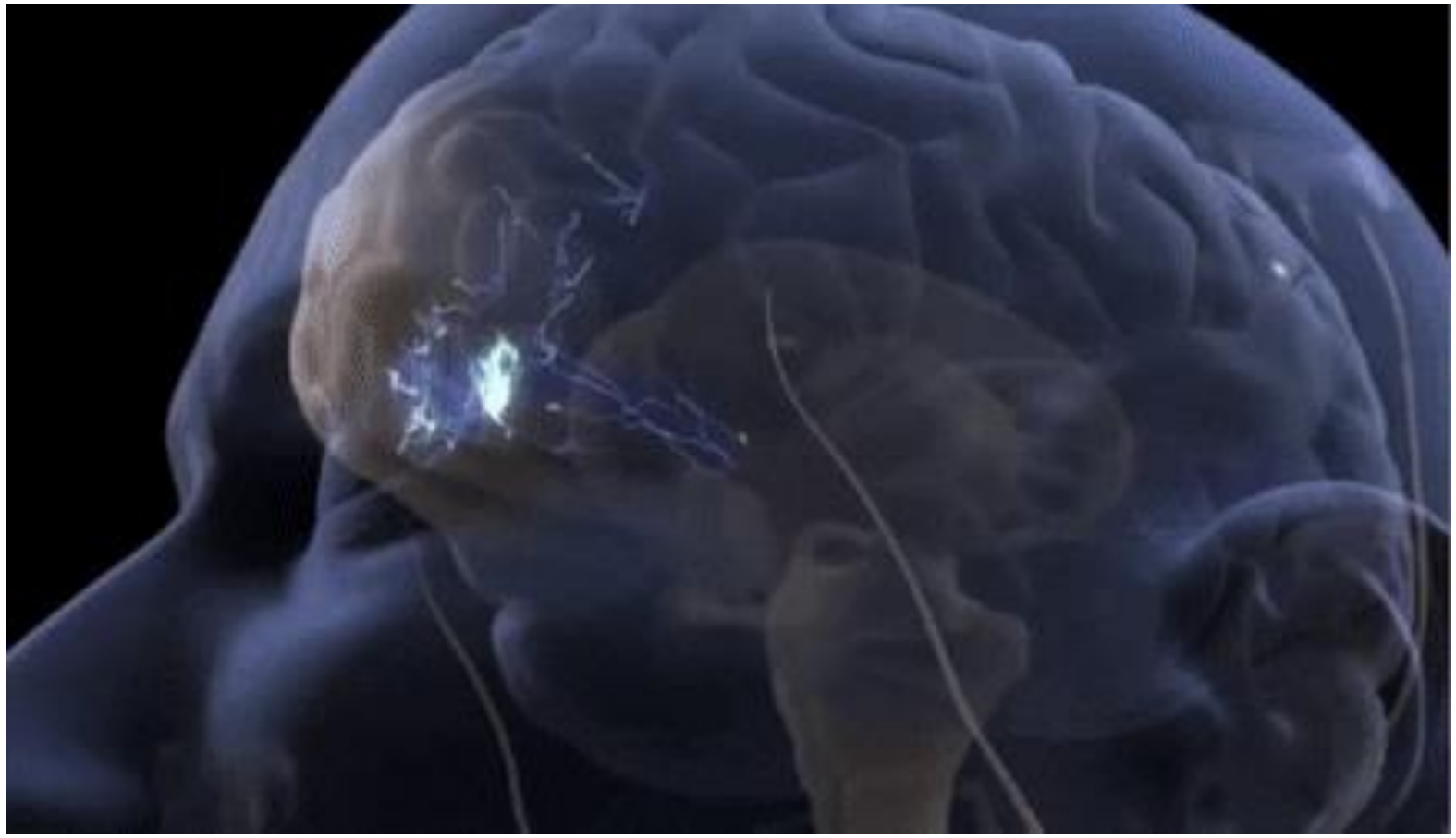
la soluzione è utilizzare più Neuroni artificiali organizzati su diversi strati Multilayer Perceptron (MLP)



Cervello umano

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.



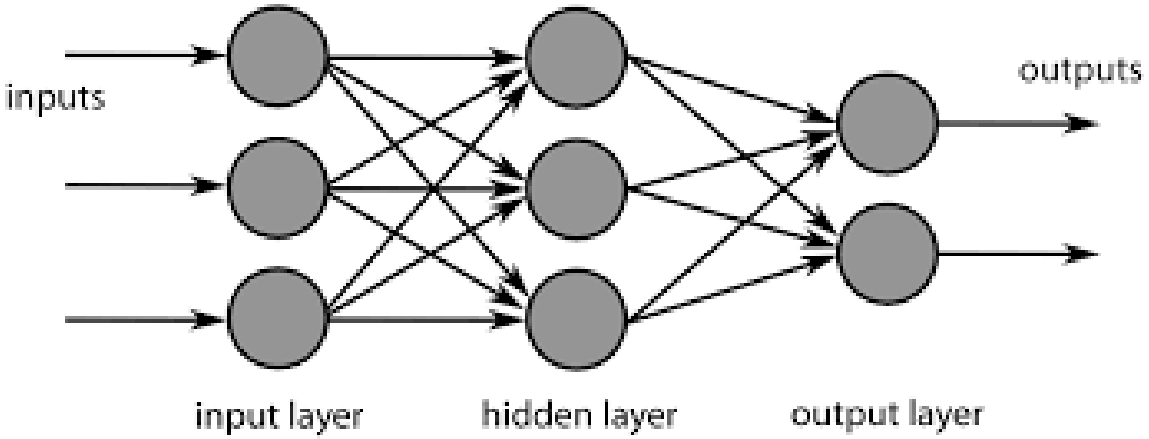
Reti neurali artificiali (ANN)

- Similmente al cervello, una rete neurale artificiale ANN è costituita da neuroni artificiali collegati tra loro.
- Ogni **connessione** (chiamata **edge**), come le sinapsi in un cervello biologico, può trasmettere un segnale ad altri neuroni.
- Il **peso associato** a ciascuna connessione **aumenta o diminuisce la forza del segnale**.
- Tipicamente, i neuroni sono aggregati in livelli. Diversi livelli possono eseguire diverse trasformazioni sui loro input.
- I segnali viaggiano dal primo livello (il livello di input), all'ultimo livello (il livello di output), possibilmente dopo aver attraversato i livelli più volte.

Tipologie di reti neurali

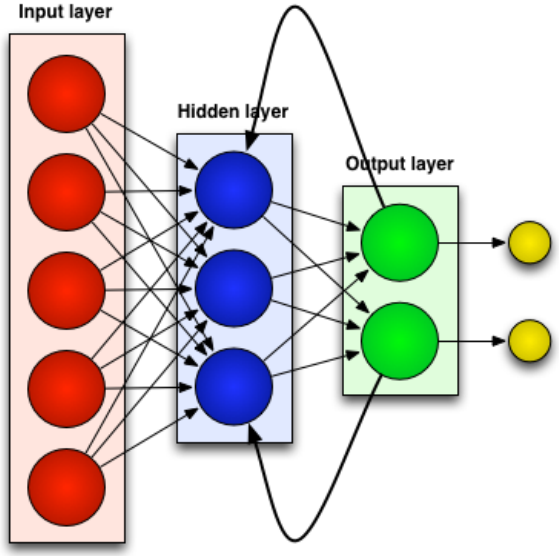
Le reti neurali sono composte da gruppi di neuroni artificiali organizzati in livelli.
Tipicamente sono presenti un livello di input, un livello di output, e uno o più livelli intermedi.
Ogni livello contiene uno o più neuroni. I **layer intermedi** sono chiamati **hidden layer** in quanto restano "invisibili" 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

nelle reti 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



Multi Layer Perceptron (MLP)

Multi Layer Perceptron (MLP) è l'FFNN più comune costituito da tre o più layer (strati):

- un layer di input
- uno o più layer nascosti
- un layer di output

Le MLP sono fully-connected: ogni neurone in uno strato è connesso con ogni neurone nello strato successivo

Un teorema noto **come universal approximation theorem** asserisce che ogni funzione continua che mappa intervalli di numeri reali su un intervallo di numeri reali può essere approssimata da un MLP con un solo hidden layer. Questa è una delle motivazioni per cui per molti decenni (fino all'esplosione del deep learning) ci si è soffermati su reti neurali a 3 livelli.

Training di una rete neurale

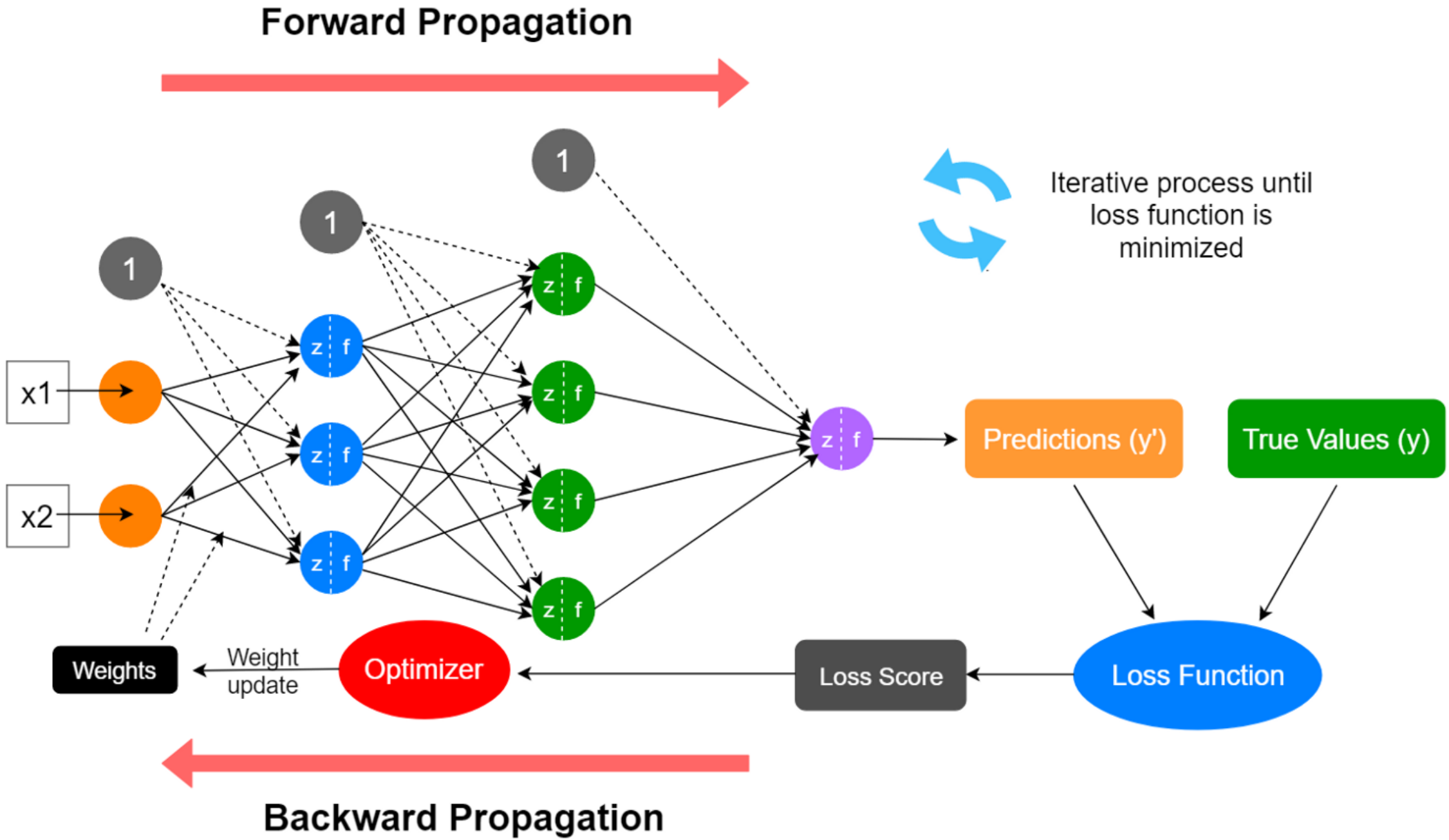
Il processo di apprendimento è una caratteristica chiave delle ANN ed è strettamente correlato al modo in cui il cervello umano apprende:

- eseguiamo un'azione
- siamo approvati o corretti da un trainer o coach per migliorare in un determinato compito.

Iterativamente

- i dati di addestramento vengono presentati alla rete **(forward),**
- quindi i pesi vengono aggiustati **(backward)** sulla base di quanto sono simili i valori restituiti dalla rete rispetto a quelli desiderati **(loss)**
- Dopo che tutti i casi sono stati presentati, il processo spesso ricomincia da capo

Durante la fase di apprendimento, i pesi vengono regolati per migliorare la performance sui dati di allenamento

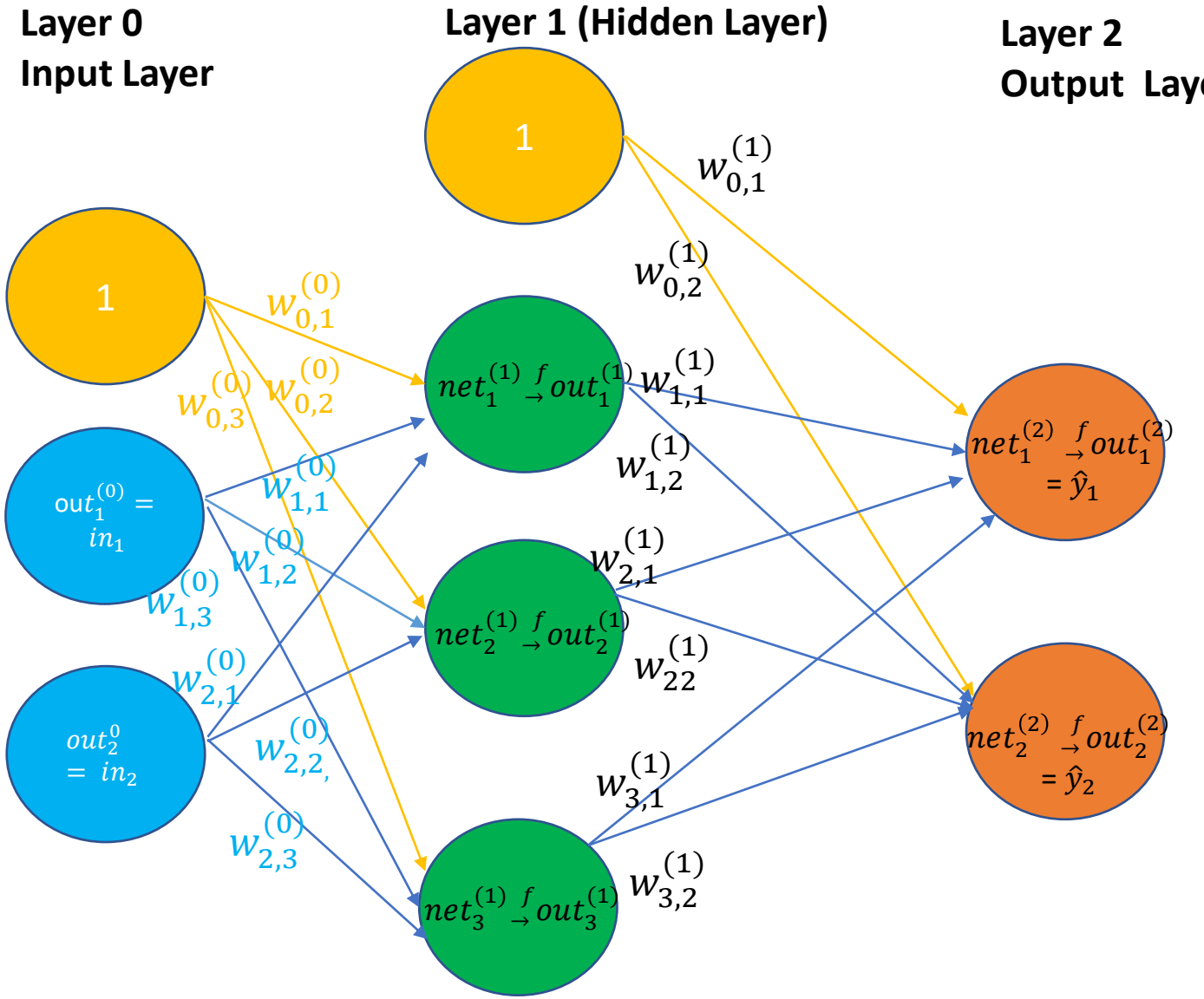

Forward Propagation
1
1
1
x1
x2
z
f
Iterative process until loss function is minimized
Predictions (y')
True Values (y)
Loss Function
Loss Score
Optimizer
Weight update
Weights
Backward Propagation

Forward propagation

- Con forward propagation (o inference) si intende la propagazione delle informazioni in avanti dal livello di input a quello di output.
- Una volta addestrata, una rete neurale può semplicemente
- processare pattern attraverso forward propagation

Forward Propagation

nell'esempio una rete a **3** livelli $d{:}nH{:}s$
input d=2 neuroni, livello nascosto **hidden nH=3** neuroni**, output s =2 neuroni**



$$out_j^{(0)} = in_j \quad j=1,\ldots,d$$

$$net_i^{(h)} = \sum_{j=1}^{d} out_j^{(h-1)} w_{j,i}^{(h-1)} + w_{0,i}$$

$$out_i^{(h)} = f(net_i^{(h)})$$

$w_{j,i}^{(h-1)}$= peso che l'input del j-esimo neurone del layer $h-1$ ha sull'i-esimo neurone del layer h

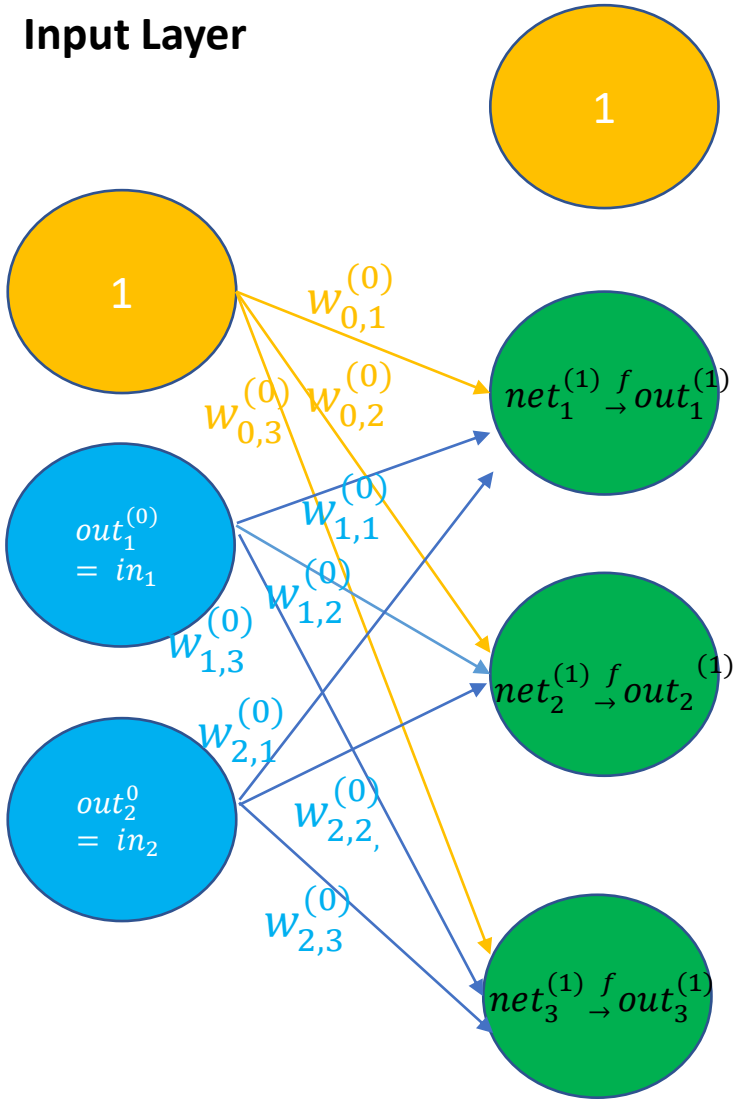
Il **numero totale di pesi** (o parametri) **è $(d+1)\times nH+(nH+1)\times s$**

Forward Propagation

Layer 0
Input Layer

Layer 1 (Hidden Layer)

Layer 2
Output Layer



$$net_1^{(1)} = w_{0,1}^{(0)} \cdot 1 + w_{1,1}^{(0)} \cdot out_1^{(0)} + w_{2,1}^{(0)} \cdot out_2^{(0)}$$

$$net_2^{(1)} = w_{0,2}^{(0)} \cdot 1 + w_{1,2}^{(0)} \cdot out_1^{(0)} + w_{2,2}^{(0)} \cdot out_2^{(0)}$$

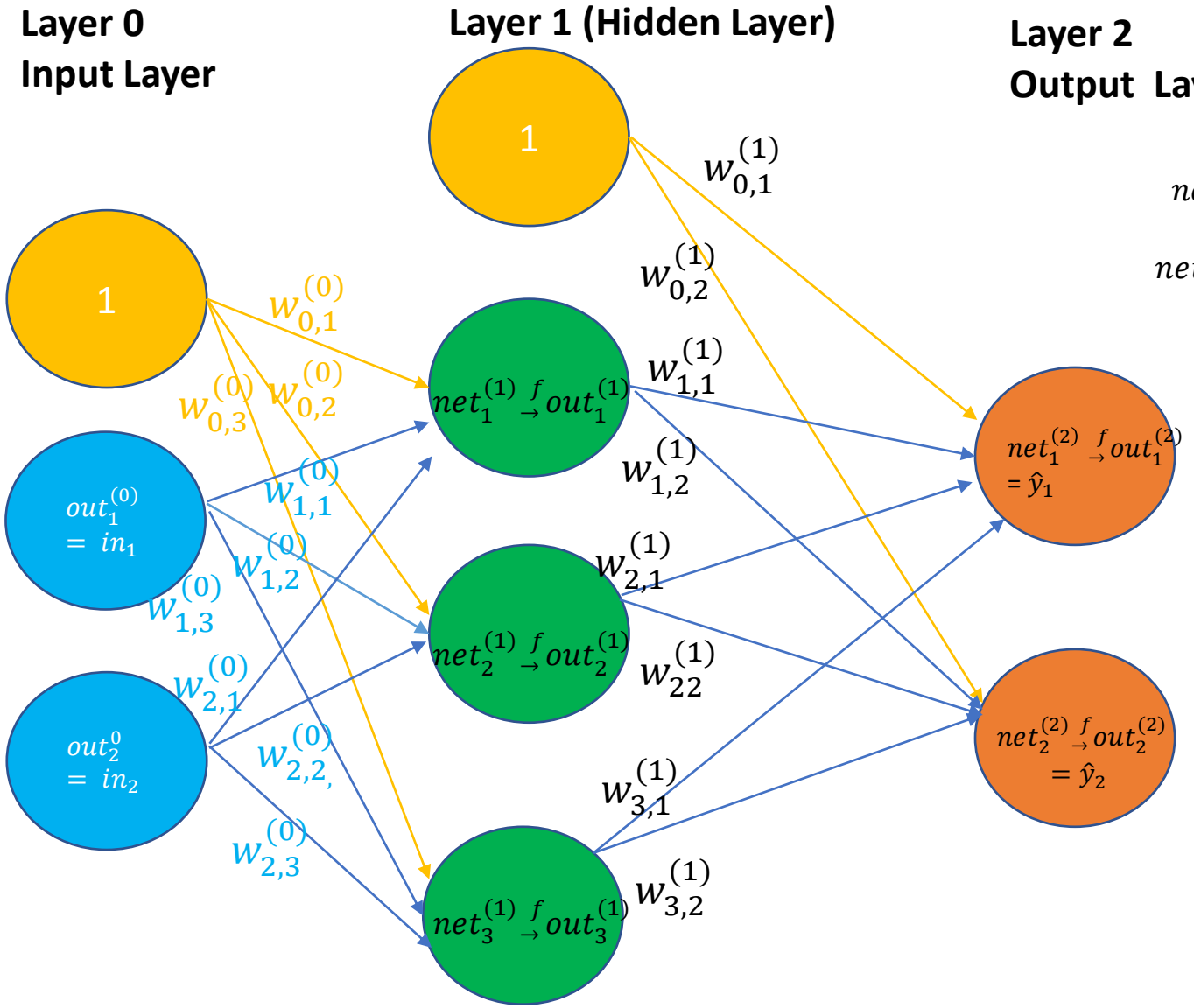
$$net_3^{(1)} = w_{0,3}^{(0)} \cdot 1 + w_{1,3}^{(0)} \cdot out_1^{(0)} + w_{2,3}^{(0)} \cdot out_2^{(0)}$$

$$\underbrace{\begin{bmatrix} net_1^{(1)} \\ net_2^{(1)} \\ net_3^{(1)} \end{bmatrix}}_{net^{(1)}} = \underbrace{\begin{bmatrix} w_{0,1}^{(0)} & w_{1,1}^{(0)} & w_{2,1}^{(0)} \\ w_{0,2}^{(0)} & w_{1,2}^{(0)} & w_{2,2}^{(0)} \\ w_{0,3}^{(0)} & w_{1,3}^{(0)} & w_{2,3}^{(0)} \end{bmatrix}}_{W^{(0)}} \underbrace{\begin{bmatrix} 1 \\ out_1^{(0)} \\ out_2^{(0)} \end{bmatrix}}_{out^{(0)}}$$

$W^{(0)}$ ha dimensioni $(d+1) \times nH$

$net^{(1)}$ ha dimensioni $nH \times 1$

Forward Propagation



$$net_1^{(2)} = w_{0,1}^{(1)} \cdot 1 + w_{1,1}^{(1)} \cdot out_1^{(1)} + w_{2,1}^{(1)} \cdot out_2^{(1)} + w_{3,1}^{(1)} \cdot out_3^{(1)}$$

$$net_1^{(2)} = w_{0,2}^{(1)} \cdot 1 + w_{1,2}^{(1)} \cdot out_1^{(1)} + w_{2,2}^{(1)} \cdot out_2^{(1)} + w_{3,2}^{(1)} \cdot out_3^{(1)}$$

$$\begin{bmatrix} net_1^{(2)} \\ net_2^{(2)} \end{bmatrix} = \begin{bmatrix} w_{0,1}^{(1)} & w_{1,1}^{(1)} & w_{2,1}^{(1)} & w_{3,1}^{(1)} \\ w_{0,2}^{(1)} & w_{1,2}^{(1)} & w_{2,2}^{(1)} & w_{3,2}^{(1)} \end{bmatrix} \begin{bmatrix} 1 \\ out_1^{(1)} \\ out_2^{(1)} \\ out_3^{(1)} \end{bmatrix}$$

$net^{(2)}$ $W^{(1)}$ $out^{(1)}$

$W^{(1)}$ ha dimensioni $(nH + 1) \times s$

$net^{(2)}$ ha dimensioni $s \times 1$

MLP: Training

- Fissata la topologia (numero di livelli e neuroni), l'addestramento di una rete neurale consiste nel determinare il valore dei pesi $\mathbf{w}$ che determinano il mapping desiderato tra input e output

Loss 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.**

- L'obiettivo del modello di machine learning è quello di minimizzare la loss function, ossia di trovare i valori dei parametri del modello che producono la previsione migliore possibile sui 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),** mentre se si sta risolvendo un problema di **classificazione multiclasse**, la loss function più comune è la funzione di **entropia incrociata categorica** (categorical cross-entropy).

- In generale, la scelta della loss function può influenzare significativamente le prestazioni del modello di machine learning, e pertanto è una scelta importante da fare durante la progettazione del modello.

Una **loss function** è per un **singolo esempio di addestramento**

Una **cost function** è la perdita media sull'intero set di dati di addestramento. Le strategie di ottimizzazione mirano a **minimizzare la funzione di costo.**

Generalmente, **la funzione di costo** viene calcolata come

$$C = \frac{\sum_{i=1}^{n} L(y_i, \hat{y}_i)}{n}$$

dove

- n è il numero di esempi di training
- y_i è l'osservazione effettiva dell'esempio di training i-esimo
- $\hat{y}_i$ la previsione dell'esempio di training i-esimo

L è la loss-function

- Nel caso di regressione, **la cost function** più comune è l'errore quadratico medio (mean squared error o MSE). L'errore quadratico medio è 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).

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

Il training di una rete consiste **nel risolvere per ogni epoca dell'allenamento *k* il seguente problema di ottimizzazione,** per calcolare i pesi che rendono minimo al passo k la funzione costo.

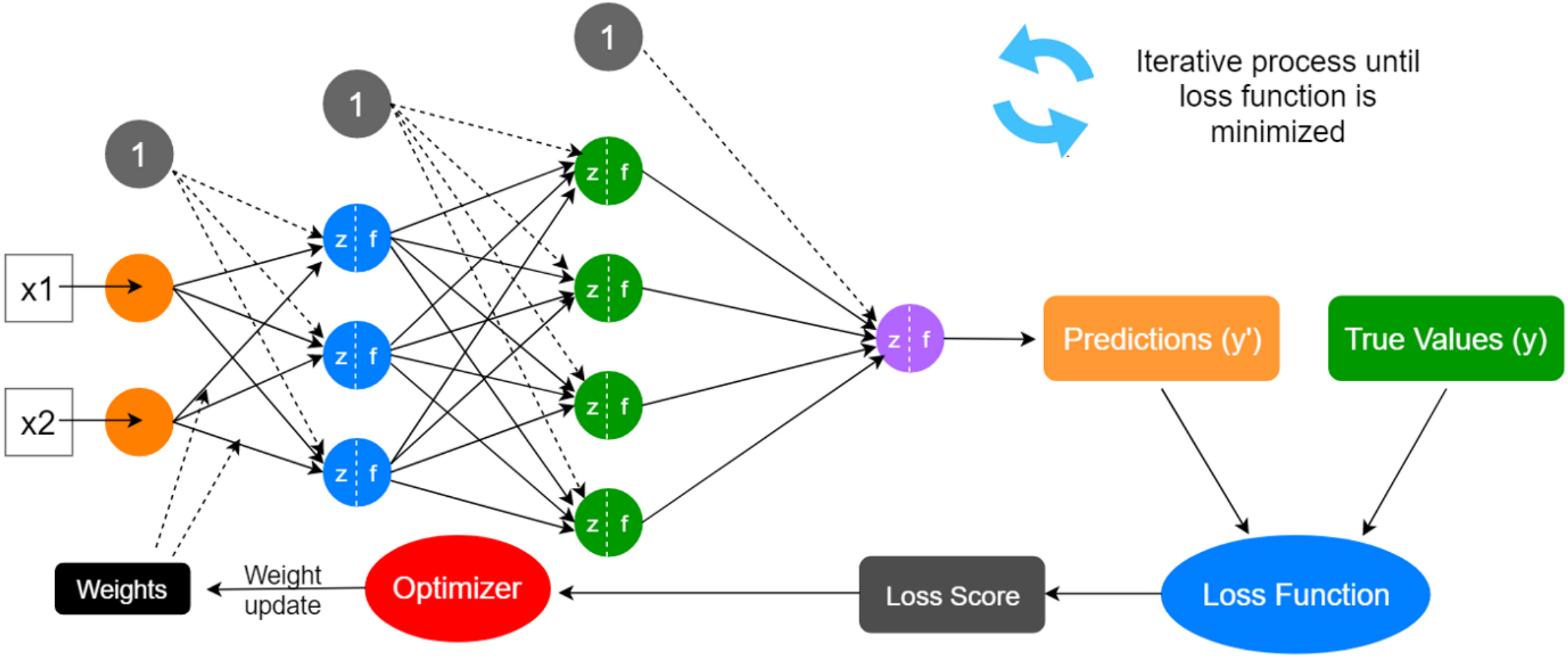
$$\arg\min_{W_k} C(W_k)$$

dove W_k rappresentano i pesi della rete alla epoca di allenamento *k*

- **Backpropagation dell'errore**
- **Metodo di discesa del gradiente**

Forward Propagation

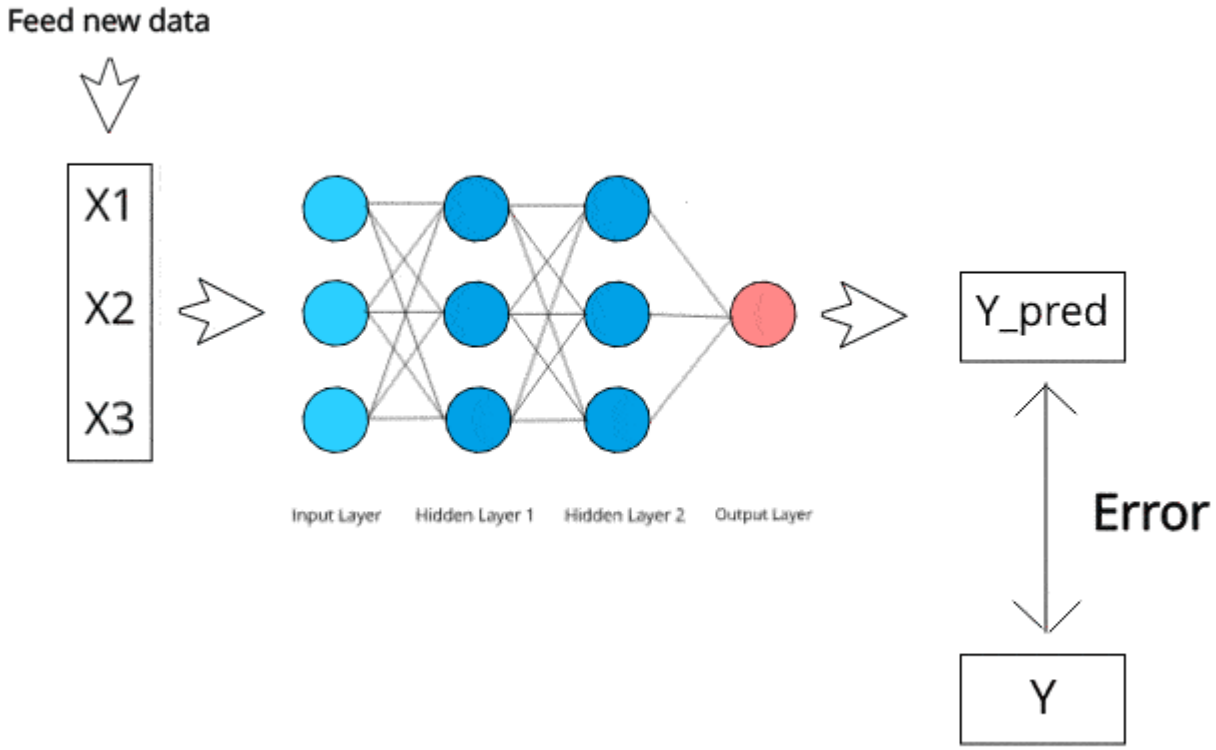
nell'esempio una rete a **4 livelli** $d{:}nH_1{:}nH_2{:}s$
input d**=2** neuroni, **livello nascosto 1** nH_1 **=3 neuroni**,
livello nascosto 2 nH_2 **=4 neuroni, output** s **=1 neurone**



Backward Propagation

Il numero totale di pesi (o parametri)
è $(d{+}1){\times}nH_1 + (nH_1 {+}1){\times}\, nH_2 + (nH_2{+}1)$ ×s=
3×3+4×4+5×1=30

nell'esempio una rete a **4 livelli** $d{:}nH_1{:}nH_2{:}s$
input d=3 neuroni, **livello nascosto 1 nH_1 =3 neuroni**,
livello nascosto 2 nH_2 =3 neuroni, output s =1 neurone.
In questo esempio viene trascurato il bias.



Il numero totale di pesi (o parametri), in questo caso, senza tener conto del bias è dato da:
è d×nH_1 +nH_1 × nH_2+ nH_2 × s= 3×3+3×3+3×1=**21**

Esempio: MLP per il riconoscimento di cifre scritte a mano

MNIST è un set di dati ampiamente utilizzato per l'attività di classificazione delle cifre scritte a mano. Consiste di 70.000 immagini di cifre scritte a mano in scala di grigi di 28x28 pixel etichettate. Il set di dati è suddiviso in 60.000 immagini di addestramento e 10.000 immagini di test. Ci sono 10 classi (una per ciascuna delle 10 cifre). Il compito è addestrare un modello utilizzando le 60.000 immagini di addestramento e successivamente testarne l'accuratezza della classificazione sulle 10.000 immagini di prova.