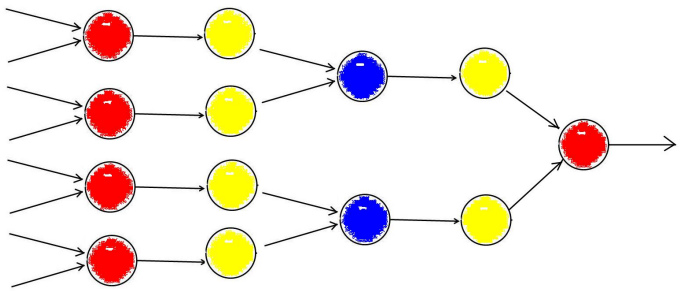
Reti neurali artificiali

Edmondo Trentin (DIISM)

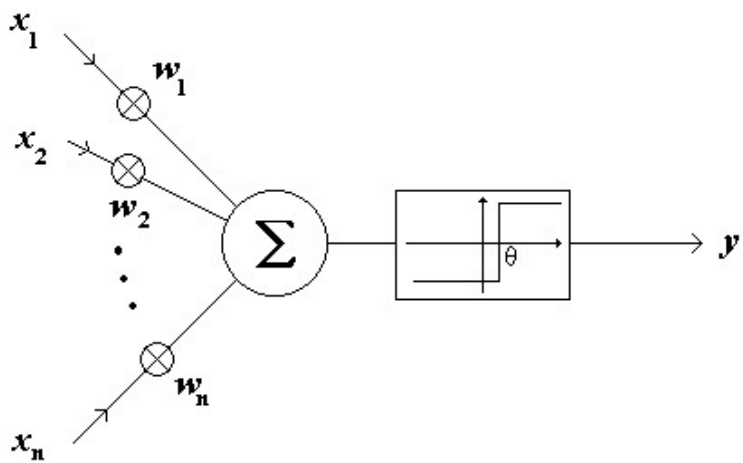
Stiamo accerchiando il concetto di "rete neurale"

Quasi senza rendercene conto, abbiamo già studiato degli **esempi** (ancorché molto semplici, particolari e parziali) di **rete neurale artificiale**.

• *Esempio 1*: le reti di TLU (unità logiche a soglia) (o di neuroni di McCulloch e Pitts)

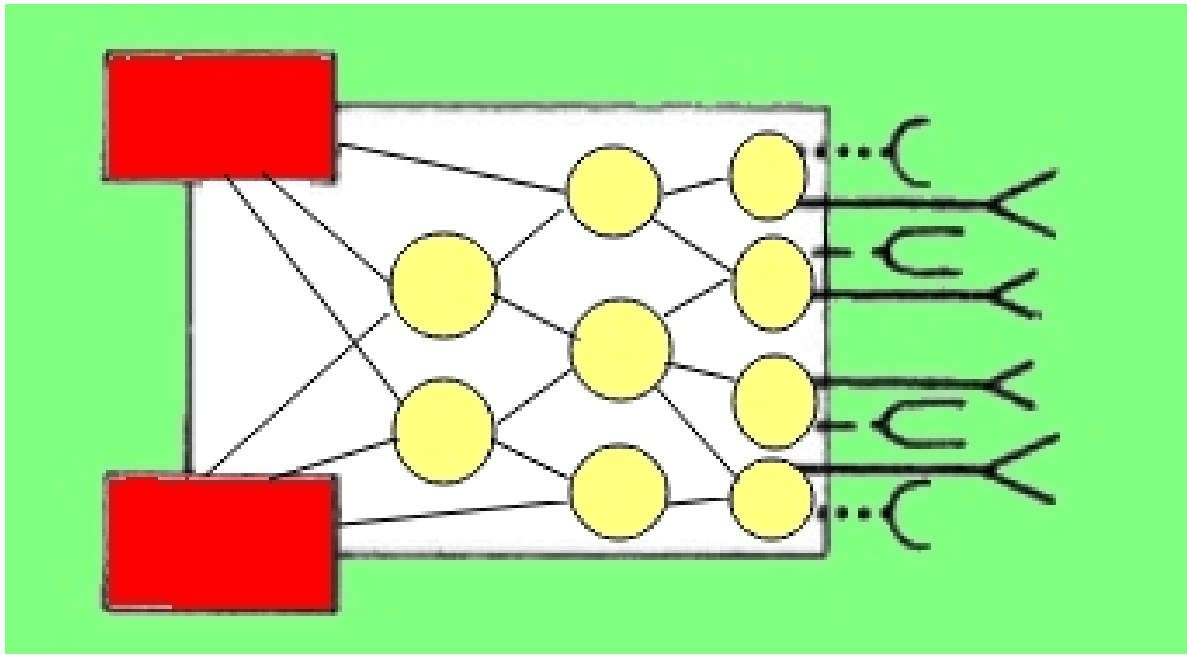


dove ogni cerchietto rappresenta un circuito del tipo



Stiamo accerchiando il concetto di "rete neurale"

- *Esempio 2*: il cervello dei veicoli pensanti di Braitenberg



(che è di fatto una rete di TLU)

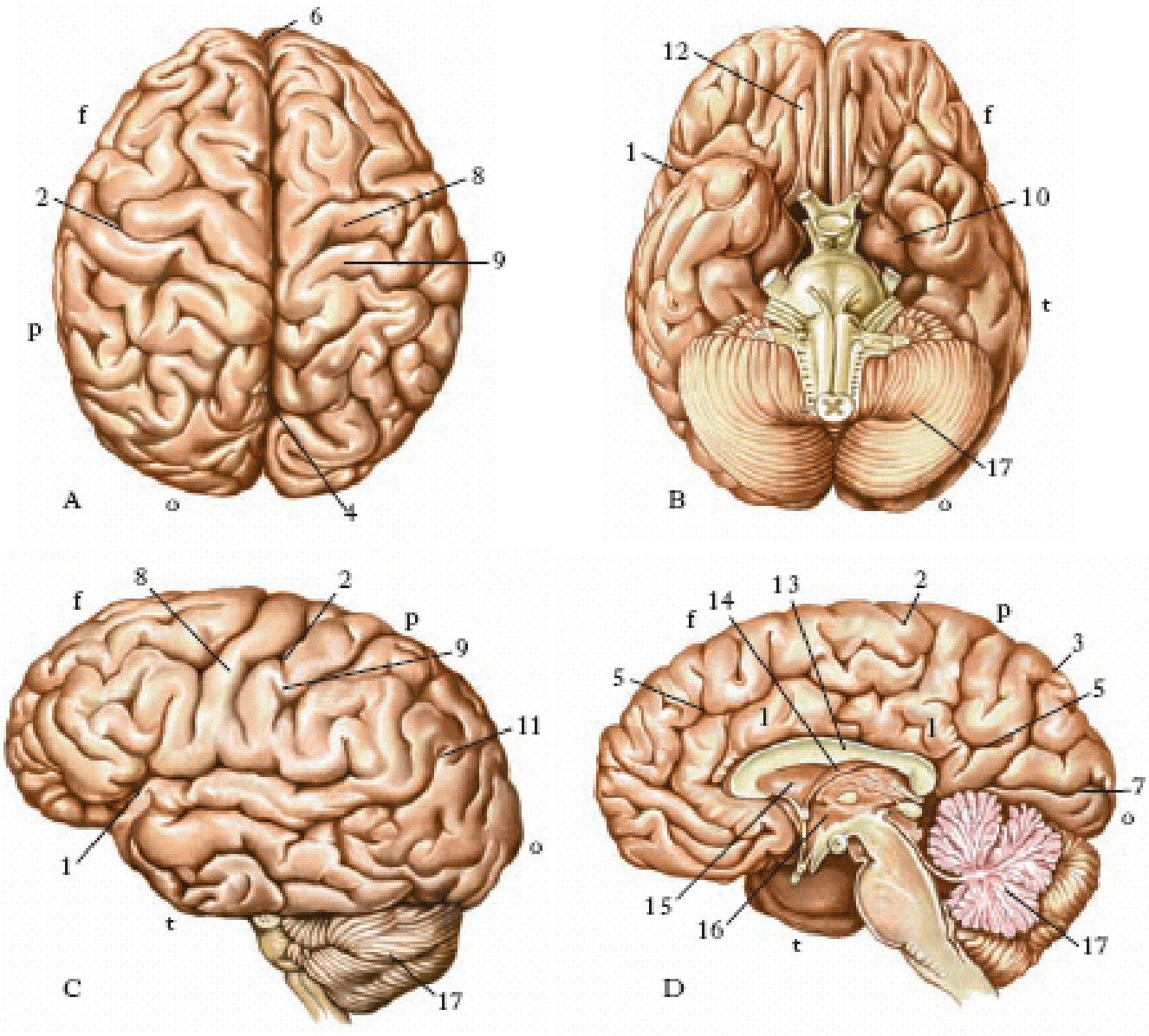
Stiamo accerchiando il concetto di "rete neurale"

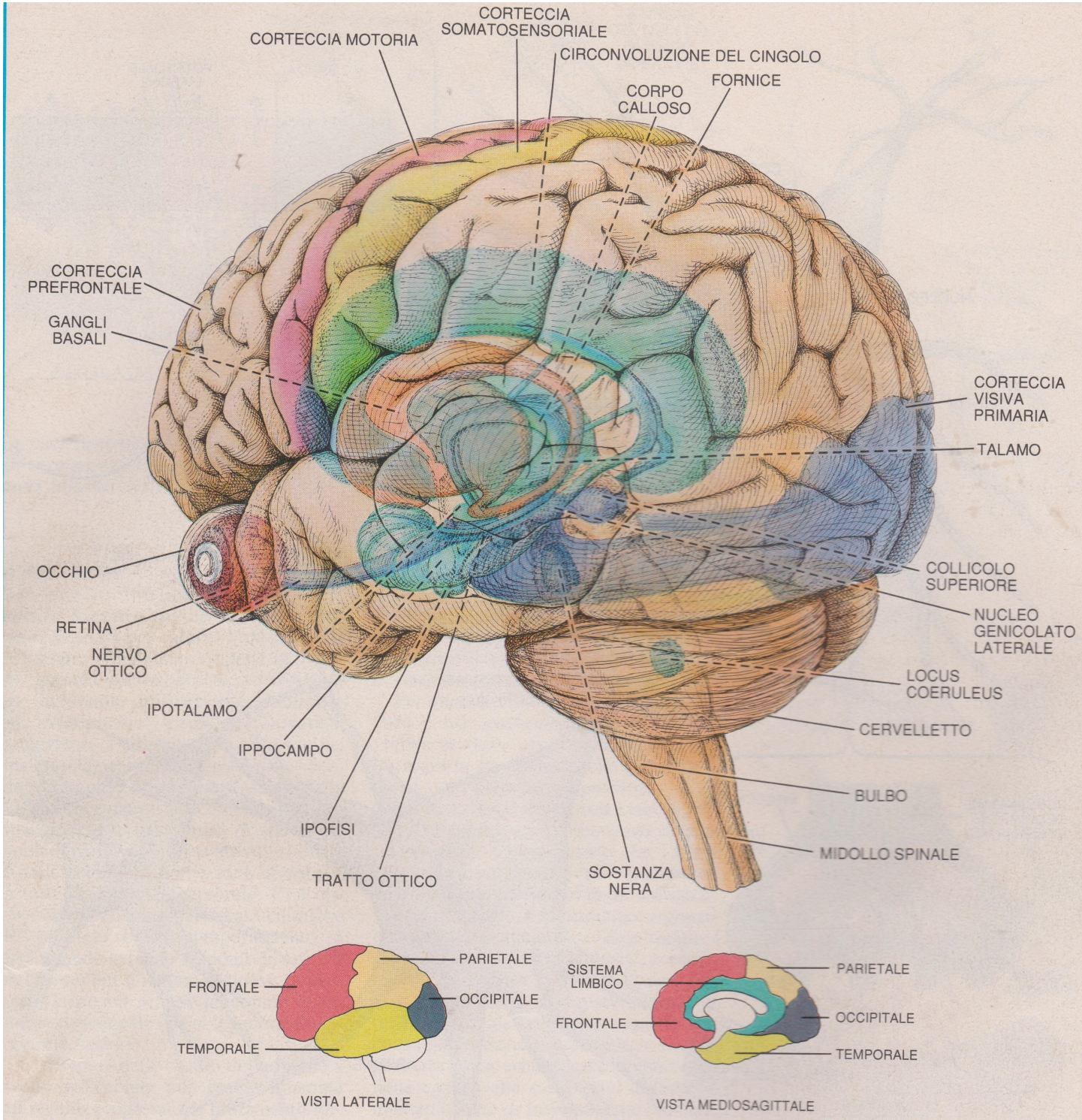
• *Esempio 3*: la palestra cinese di J. Searle è quasi una caricatura di rete neurale, eppure ne possiede molti attributi

• *Esempio 4*: i Churchland sostengono una IA forte che scaturisca non già dalla macchina di Turing ma da un **paradigma di calcolo distribuito e parallelo** ispirato alla macchina biologica intelligente per eccellenza: **il cervello**.

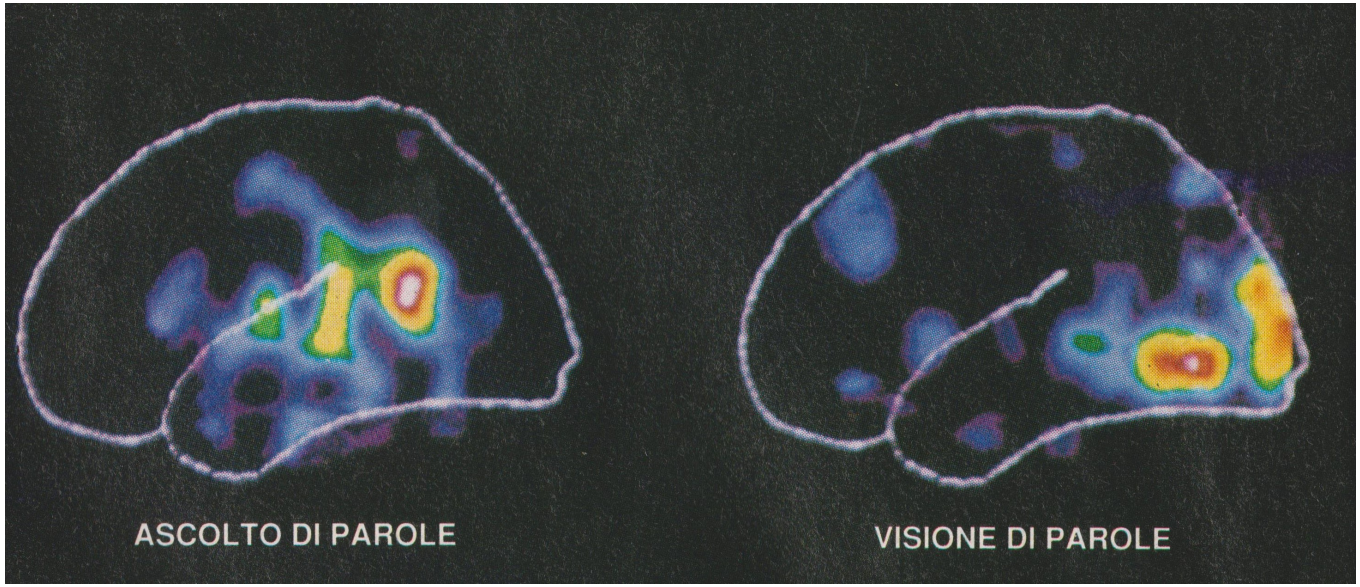
Le reti di TLU (o di neuroni di McCulloch e Pitts) si candidano ad essere i prototipi di tali paradigmi di calcolo, ma per
(1) apprezzarli,
(2) individuarne i limiti, e
(3) spingersi oltre
è *in primis* **necessario conoscere i tratti fondamentali del paradigma biologico di riferimento.**

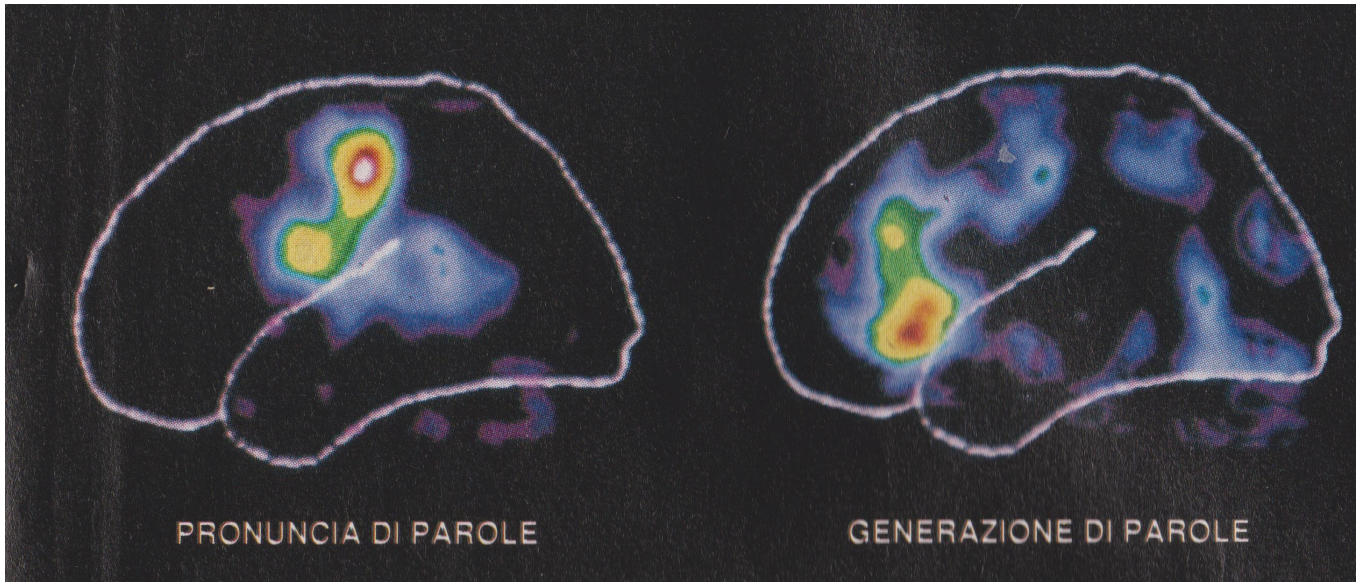
Il “paradigma biologico di riferimentio”


6
f
2
8
9
p
4
A
o
12
f
1
10
t
17
B
o
8
2
f
p
9
11
1
o
t
17
C
13
2
14
f
p
3
5
5
1
1
7
o
15
16
t
17
D

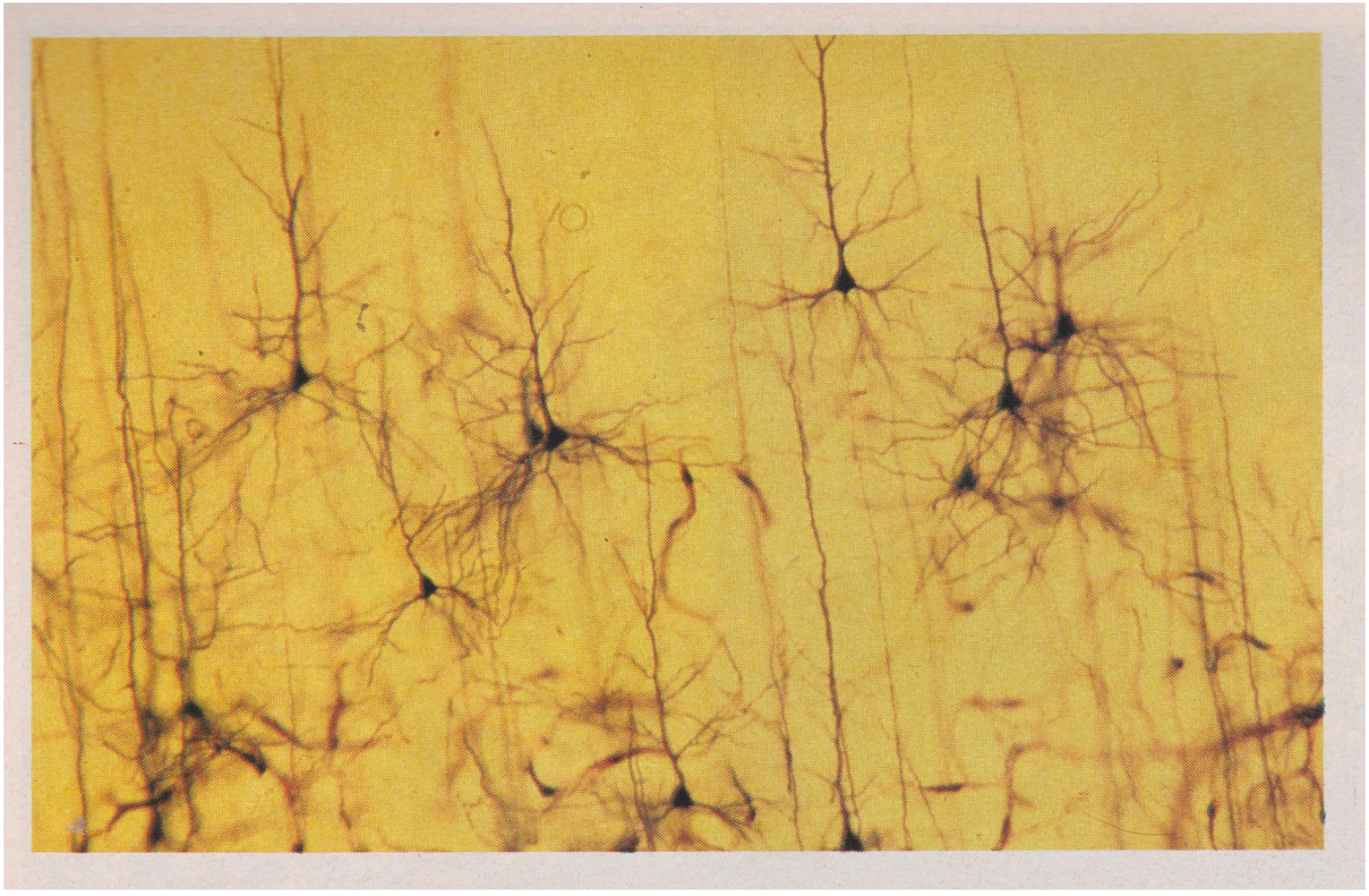

CORTECCIA MOTORIA
CORTECCIA SOMATOSENSORIALE
CIRCONVOLUZIONE DEL CINGOLO
FORNICE
CORPO CALLOSO
CORTECCIA PREFRONTALE
GANGLI BASALI
CORTECCIA VISIVA PRIMARIA
TALAMO
OCCHIO
COLLICOLO SUPERIORE
RETINA
NUCLEO GENICOLATO LATERALE
NERVO OTTICO
LOCUS COERULEUS
IPOTALAMO
CERVELLETTO
IPPOCAMPO
BULBO
IPOFISI
MIDOLLO SPINALE
TRATTO OTTICO
SOSTANZA NERA
PARIETALE
FRONTALE
OCCIPITALE
TEMPORALE
VISTA LATERALE
SISTEMA LIMBICO
PARIETALE
FRONTALE
OCCIPITALE
TEMPORALE
VISTA MEDIOSAGITTALE

Attività cerebrale e funzioni cognitive superiori: analisi tramite tomografia a emissione di positroni (PET)

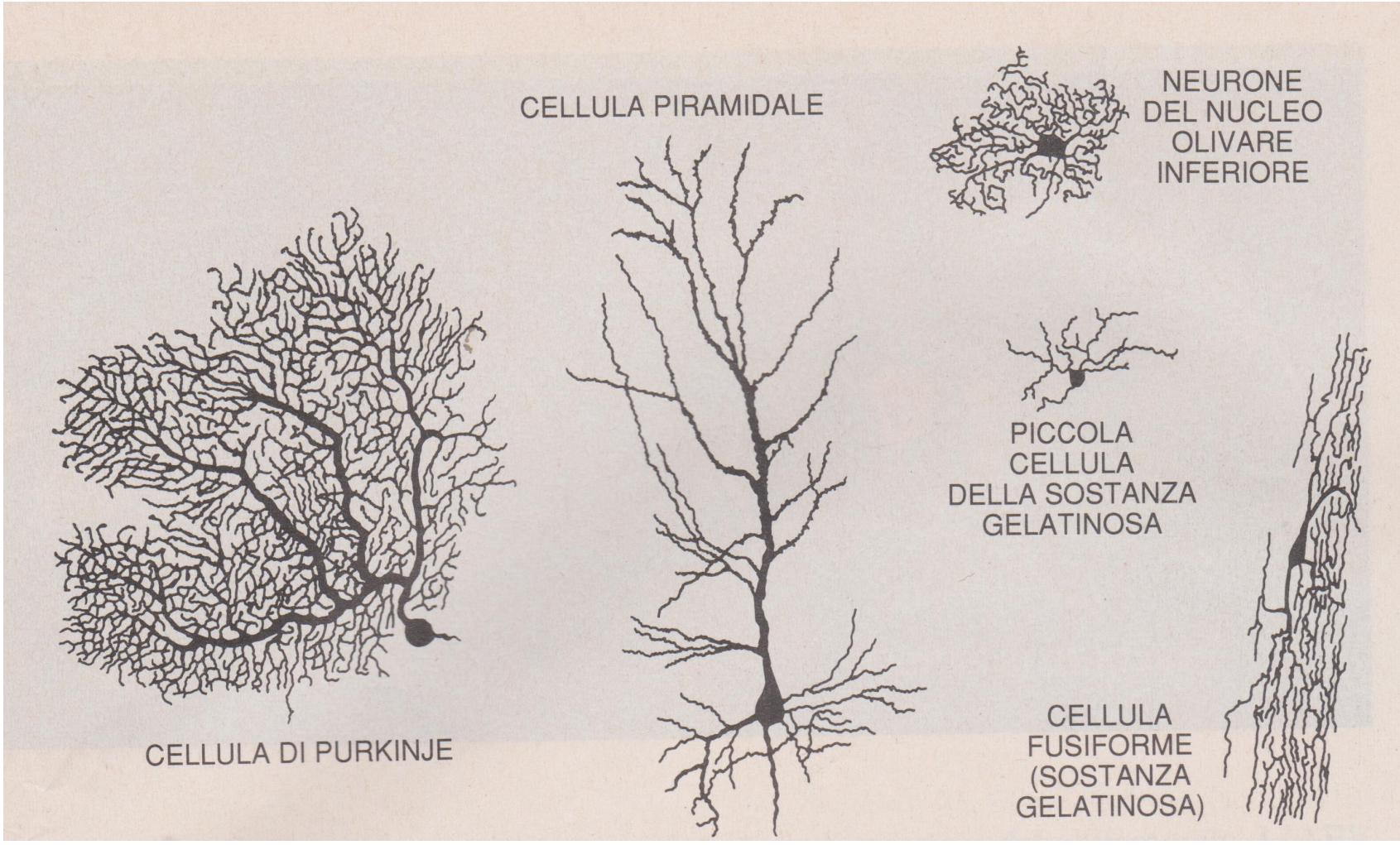




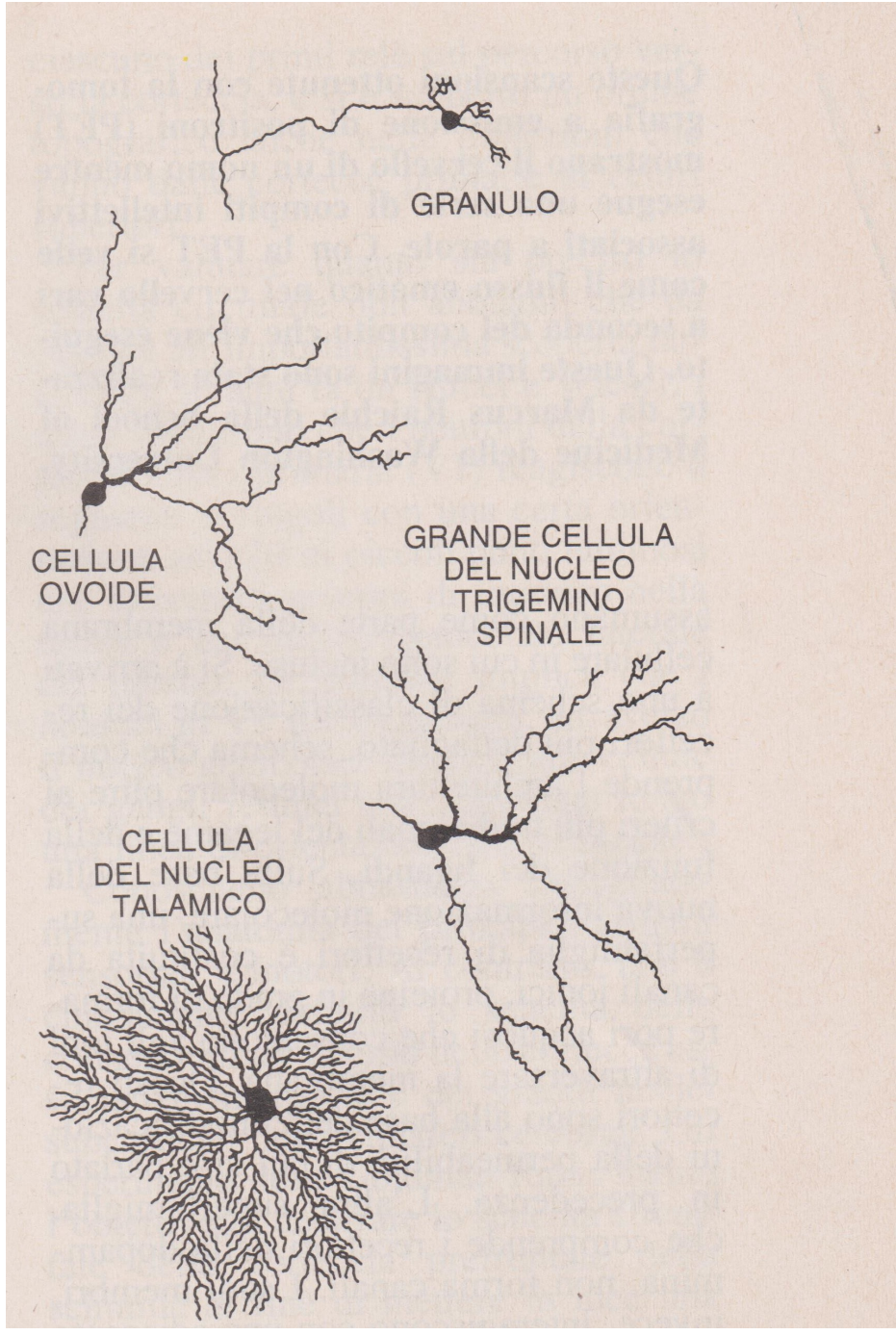
Colorazione con il metodo dei sali d'argento di Camillo Golgi (1873)



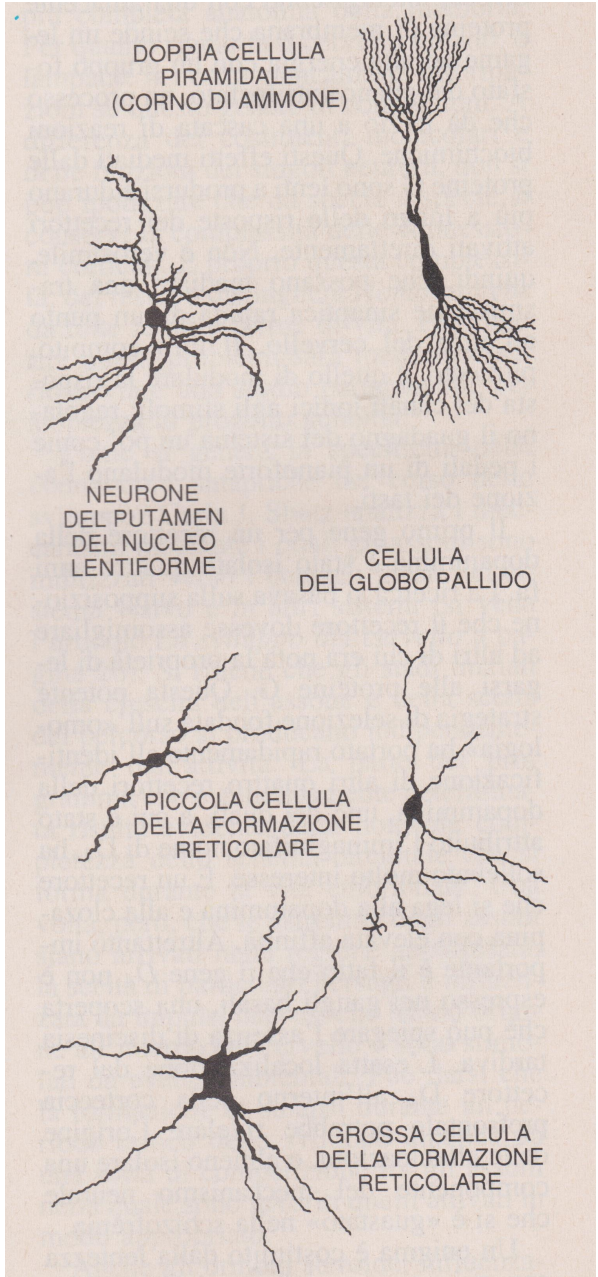
Alcuni tipi di neuroni #1:

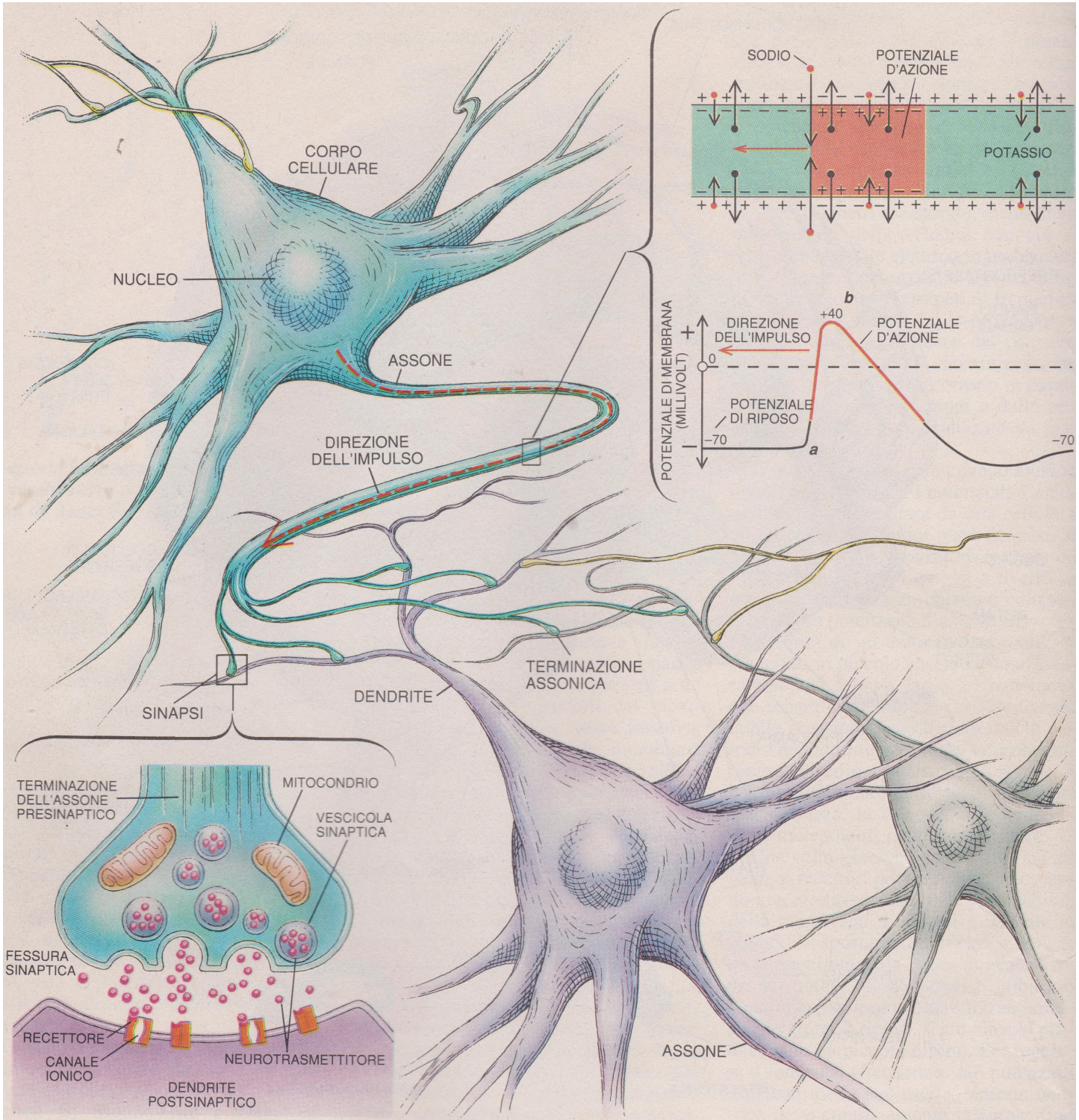


Alcuni tipi di neuroni #2:



Alcuni tipi di neuroni #3:




SODIO
POTENZIALE D'AZIONE
POTASSIO
CORPO CELLULARE
NUCLEO
ASSONE
DIREZIONE DELL'IMPULSO
POTENZIALE DI MEMBRANA (MILLIVOLT)
b
+40
DIREZIONE DELL'IMPULSO
POTENZIALE D'AZIONE
0
POTENZIALE DI RIPOSO
–70
a
–70
TERMINAZIONE ASSONICA
DENDRITE
SINAPSI
TERMINAZIONE DELL'ASSONE PRESINAPTICO
MITOCONDRIO
VESCICOLA SINAPTICA
FESSURA SINAPTICA
RECETTORE
CANALE IONICO
NEUROTRASMETTITORE
DENDRITE POSTSINAPTICO
ASSONE

Neuroplasticità: fenomeno di modifica e adattamento del cervello a fronte di sviluppo del soggetto, stimolazioni sensoriali, danni cerebrali, ecc. Appartiene a due principali tipologie:

1. *Neuroplasticità strutturale*: costruzione e alterazione dei "circuiti" (assoni che vanno a proiettarsi verso altri neuroni postsinaptici anche distanti), generazione di nuovi neuroni a seguito del danneggiamento di altri
2. *Neuroplasticità funzionale*: definizione e modifica delle funzioni di circuiti di neuroni. In particolare:
   - 2.1 plasticità dipendente dall'attività
   - 2.2 plasticità reattiva, es. la rimappatura di una funzione da una porzione all'altra della corteccia a fronte di una lesione.

   Si realizza sotto forma di:
   - **plasticità sinaptica**: una data sinapsi si rafforza o si indebolisce (es. "potenziamento/depressione a lungo termine")
   - plasticità omeostatica (un neurone auto-regola la propria eccitabilià in funzione dello stato complessivo di eccitazione dei circuiti in cui è inserito)
   - plasticità intrinseca (*idem* ma diversa per chimica e fisica)

Plasticità sinaptica: apprendimento hebbiano/anti-hebbiano

Donald O. Hebb (1904-1985)

The Organization of Behavior, J. Wiley & Sons, 1949: la sinapsi tra due neuroni che si attivano frequentemente assieme[1] si rafforza, si deprime invece se essi si attivano ripetutamente in modo scorrelato l'uno dall'altro.

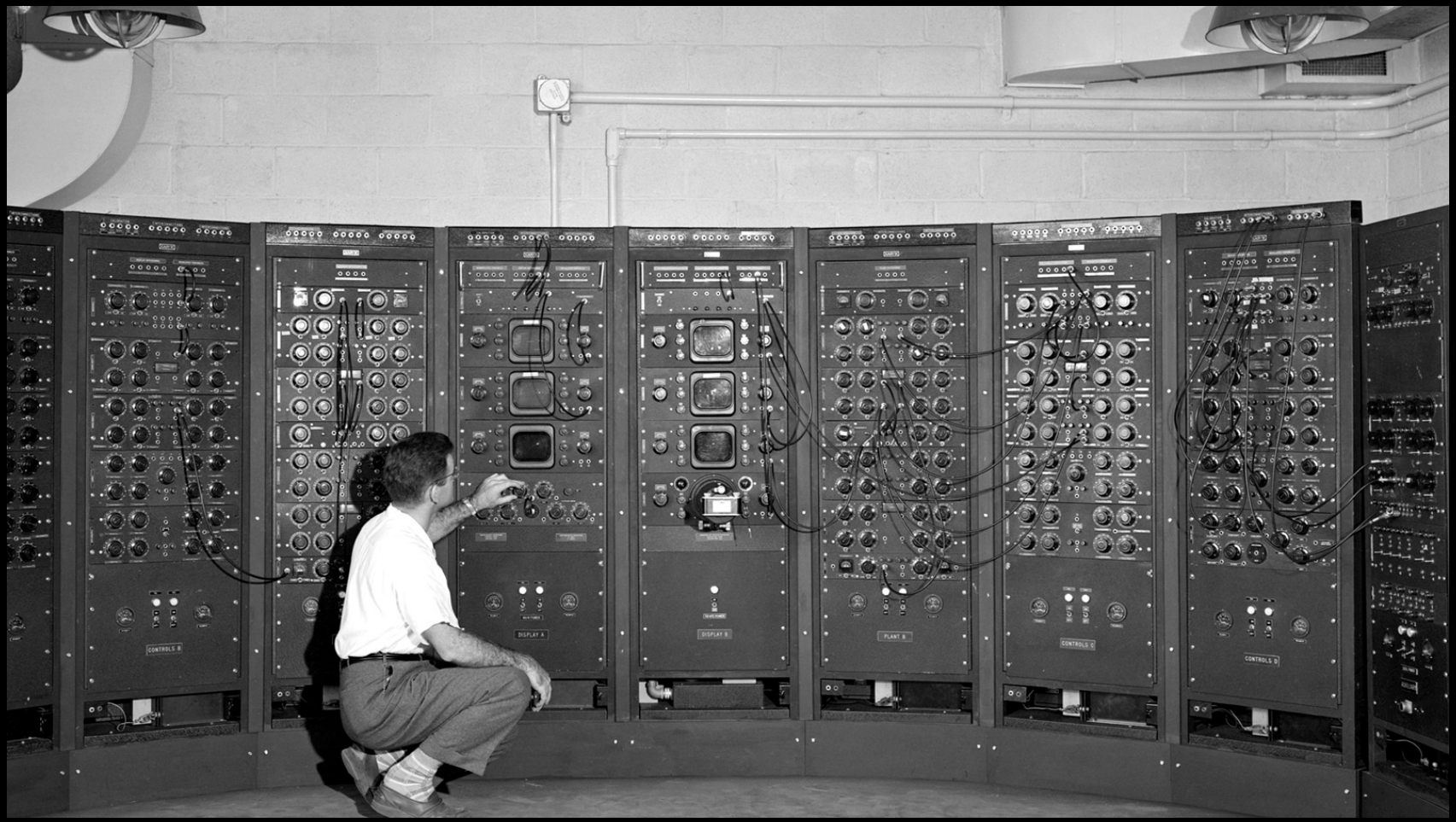
[1] L'uno concorre spesso ad eccitare l'altro.

The big Q

Q: cosa manca dunque a reti di TLU, di neuroni di McCulloch e Pitts, o ai veicoli pensanti dotati di "cervello" per mimare davvero il paradigma biologico di riferimento e consustanziarsi con la macchina intelligente dei Churchland?

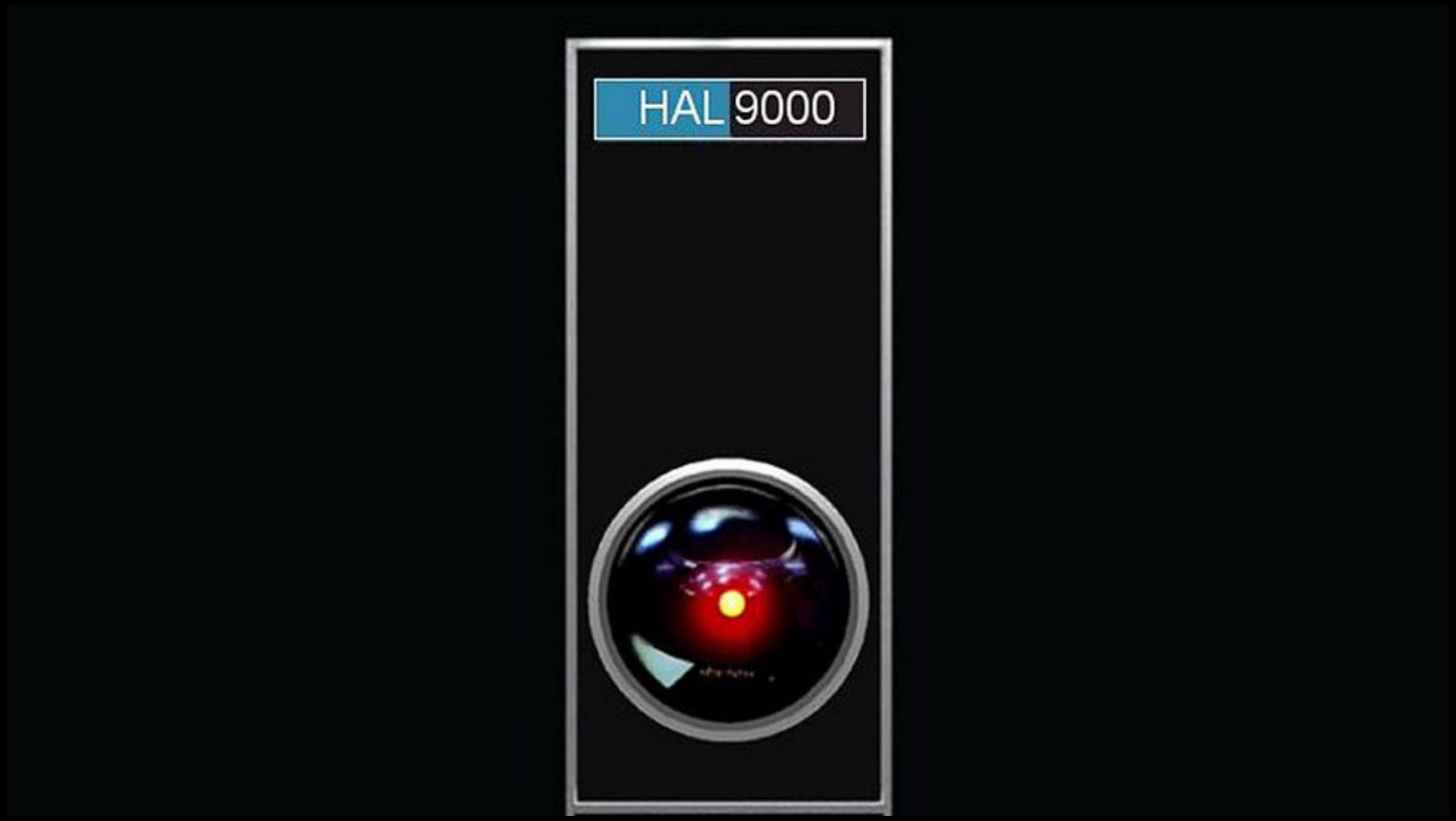
Siamo nella seconda metà del 1967. Questi due signori girano il Nord America passando da un laboratorio di IA all'altro, per capire cosa ci fosse e cosa ci si attendesse.

Videro i primi “cervelloni” in funzione, parlarono con gli scienziati, presso i laboratori della Rank Xerox sentirono parlare il primo sintetizzatore vocale della stoia, capace anche di cantare “Daisy”.

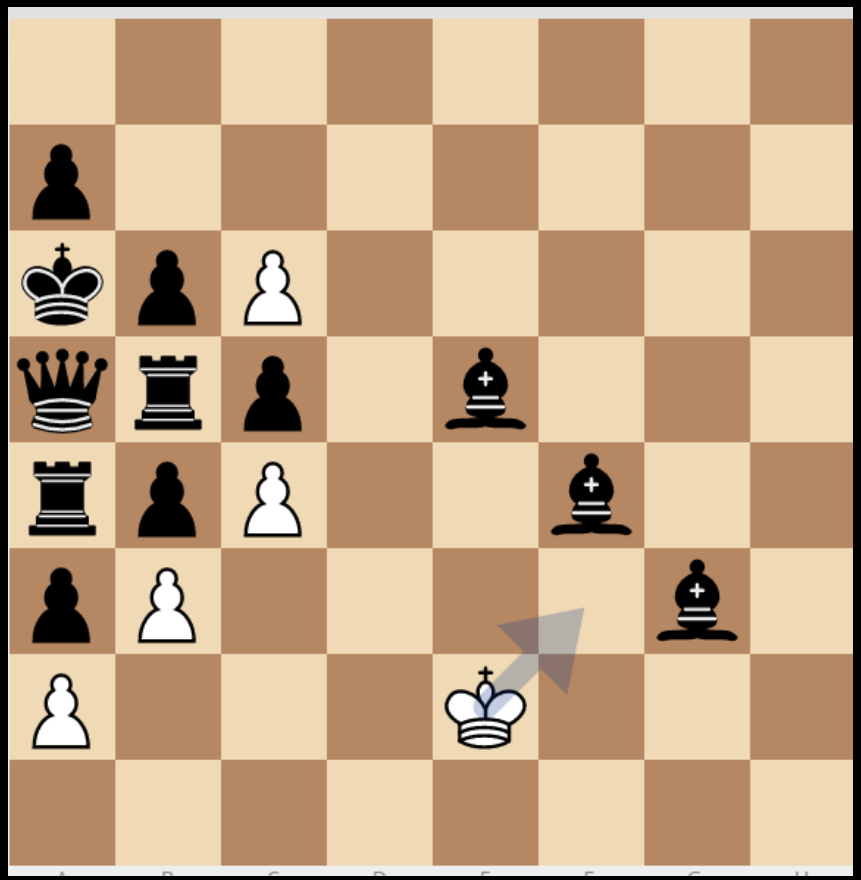


I signori in questione sono Stanley Kubrick e Arthur C. Clarke.

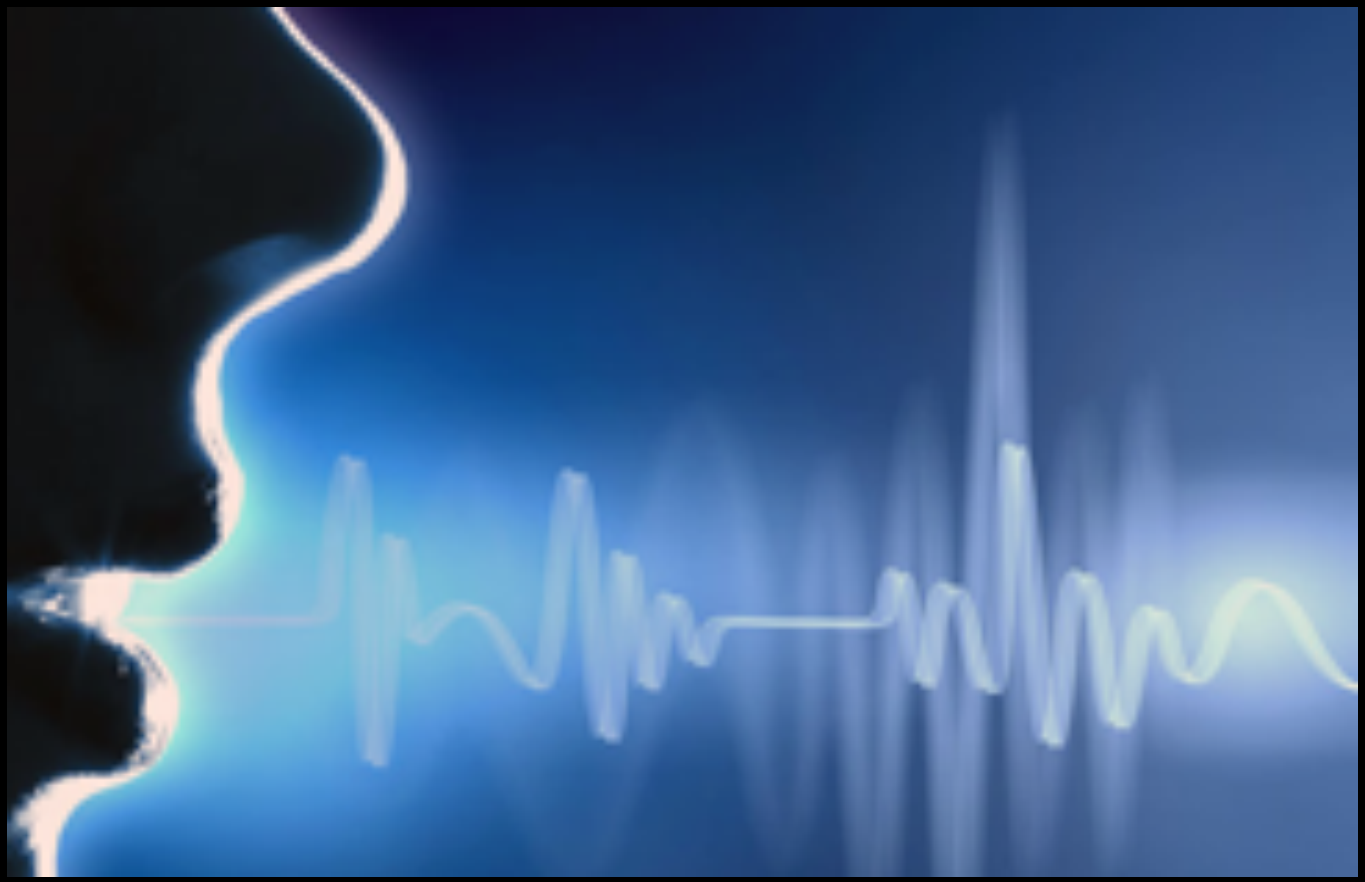
L'IA è incarnata dal supercomputer HAL 9000, di cui non si sa se abbia una coscienza ma che *si comporta* come se la avesse.



Kubrick e Clarke inscenano una teoria comportamentale volta a dimostrare l'intelligenza di HAL.
(1) Capacità di giocare e vincere una partita a scacchi contro un avversario umano:



(2) Capacità di parlare, di riconoscere il parlato, di leggere il labiale:



(2) Capacità di riconoscere i volti e di cogliere lo stato emotivo dalle espressioni facciali:

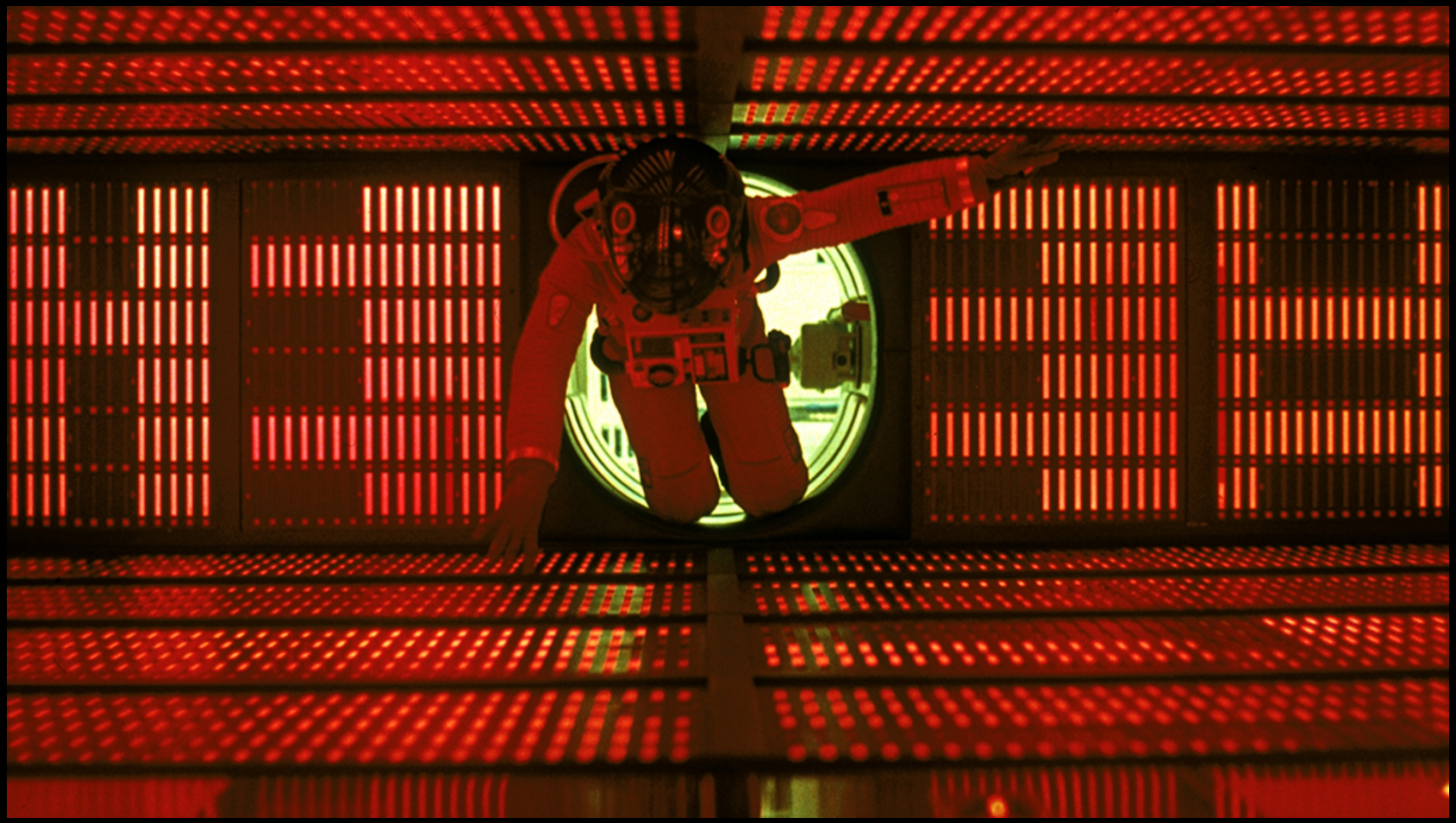
(2) Capacità di controllare sofisticate robotiche e “domotiche” (le funzioni operative dell’astronave):

HAL ha una controparte sulla Terra, ma i due gemelli vanno diversificandosi. Sull'astronave, HAL diventa un rischio per la missione e un pericolo per l'equipaggio.

HAL sul pianeta Terra

HAL sull'astronave

Non resta che disattivarlo. L'astronauta David smonta uno ad uno i blocchi della **memoria** di HAL.



HAL regredisce allo stadio infantile, e canta "Daisy" (Girotondo) che il suo istruttore gli aveva insegnato da piccolo, infine si smorza.

• Ma quale computer modifica le proprie funzionalità a seconda del proprio vissuto?
• Quale computer non cessa di funzionare quando se ne staccano uno alla volta i pezzi?
• Quale computer non è stato programmato ma addestrato da un istruttore?

HAL 9000 è una rete neurale artificiale:
è una macchina distribuita e parallela (quindi, *fault tolerant*)
costituita da unità semplici e cooperanti
dalla cui attività simultanea e interrelata
scaturiscono comportamenti intelligenti.
In particolare, essa **è capace di apprendere**,
la qual cosa la rende adattativa.

Il comportamento intelligente di HAL fa emergere dunque la risposta alla nostra "big Q"

Q: cosa manca dunque a reti di TLU, di neuroni di McCulloch e Pitts, o ai veicoli pensanti dotati di "cervello" per mimare davvero il paradigma biologico di riferimento e consustanziarsi con la macchina intelligente dei Churchland?

che è dunque la seguente:

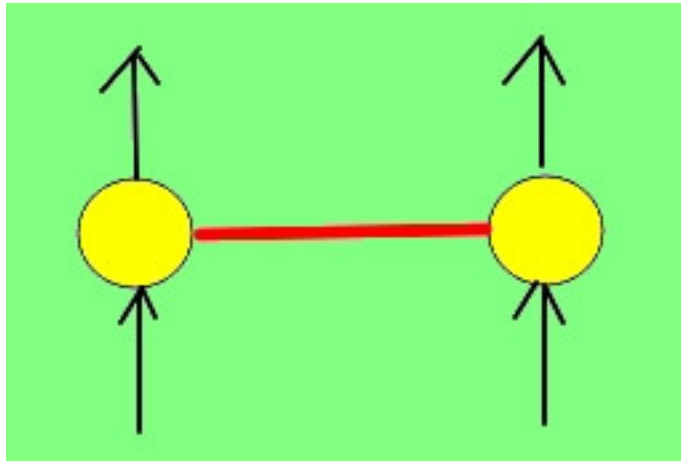
A: un meccanismo di apprendimento.

Le reti di TLU possono calcolare qualsiasi funzione logica/digitale, ma è il progettista che deve definirne i pesi $w_1, \ldots, w_n$ e le soglie θ in modo rigido acciocché esse la realizzino.
Goal: vogliamo modelli capaci di **apprenedere** $w_1, \ldots, w_n$ e θ (per ogni neurone/unità della rete) dall'osservazione dell'ambiente.

Lo faremo in due passi:
(1) introducendo l'apprendimento nei Veicoli pensanti;
(2) definendo formalmente le reti neurali artificiali e sviluppando per esse un preciso **algoritmo di apprendimento** (*backpropagation*).

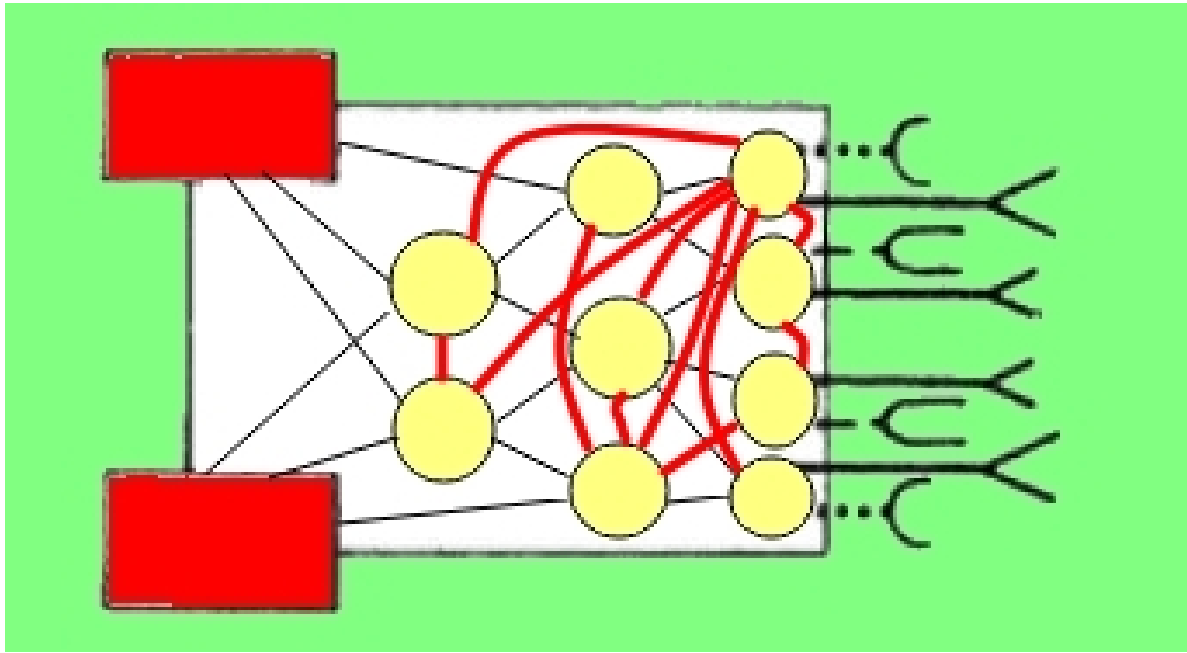
Veicolo 7: apprendimento e concetti

Lo ***Mnemotrix*** è un materiale immaginario che a riposo si comporta come una resistenza perfetta. Il cavo rosso di mnemotrix qui sotto non consente dunque il transito di corrente elettrica tra gli elementi gialli da esso collegati:



Se ai suoi estremi transita simultaneamente corrente elettrica, la sua conduttanza aumenta (in funzione di quanto intensa e persistente è quella corrente) e la corrente inizia a passare da un estremità all'altra del cavo. Lo Mnemotrix rimane un buon conduttore tanto più a lungo quanto più frequentemente e persistentemente le sue estremità sono state simultaneamente stimolate, per poi tornare a riposo al cessare della "stimolazione".

Il Veicolo 7 è costruito da un Veicolo 5 nel quale ogni coppia di elementi a soglia è collegata con un pezzo di cavo Mnemotrix (la figura mostra solo un certo numero di collegamenti):



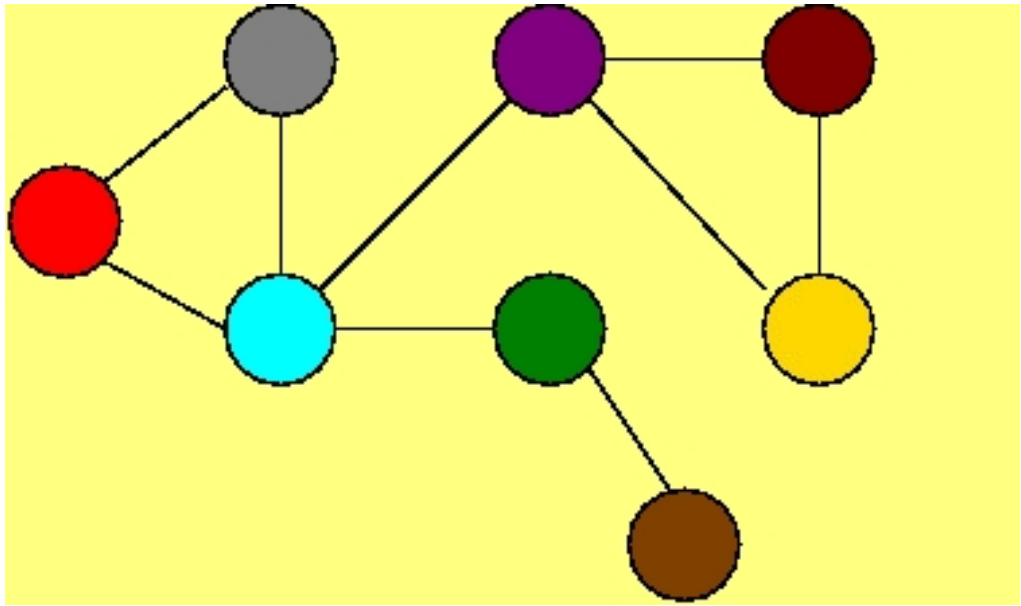
All'inizio il Veicolo 7 si comporta esattamente come il Veicolo 5. Con il tempo e l'esperienza, però, il Veicolo 7 inizia a cambiare comportamenti e mostra una certa **adattabilità** all'ambiente (che i diversi Veicoli finora non avevano).

Se i veicoli aggressivi presenti nell'ambiente sono prevalentemente rossi, lo Mnemotrix che collega l'elemento a soglia che rappresenta il colore rosso con l'elemento a soglia che controlla il comportamento di fronte a un'aggressione aumenterà la propria conduttanza, e i due elementi a soglia si stimoleranno a vicenda (*Nota: pensa all'apprendimento Hebbiano!*).

- Di conseguenza, ogni volta che il Veicolo 7 si imbatterà in veicoli rossi esso reagirà come di fronte a tipi pericolosi: esso ha realizzato un'**associazione** (*Nota: pensa ai cani di Pavlov!*).
- Viceversa, ogni volta che si imbatterà in veicoli aggressivi, a prescindere dal loro colore esso "vedrà rosso": il Veicolo 7 ha dunque appreso un **concetto**.
- Se i veicoli pacifici sono tutti grigi e quelli colorati sono aggressivi, il Veicolo 7 reagirà a una potenziale aggressione di fronte alla presenza del colore, di un qualsiasi colore diverso dal grigio: esso ha compiuto un'**astrazione**, generalizzando gli esempi incontrati fino a quel momento tanto da estrapolarne il concetto di *colore*.

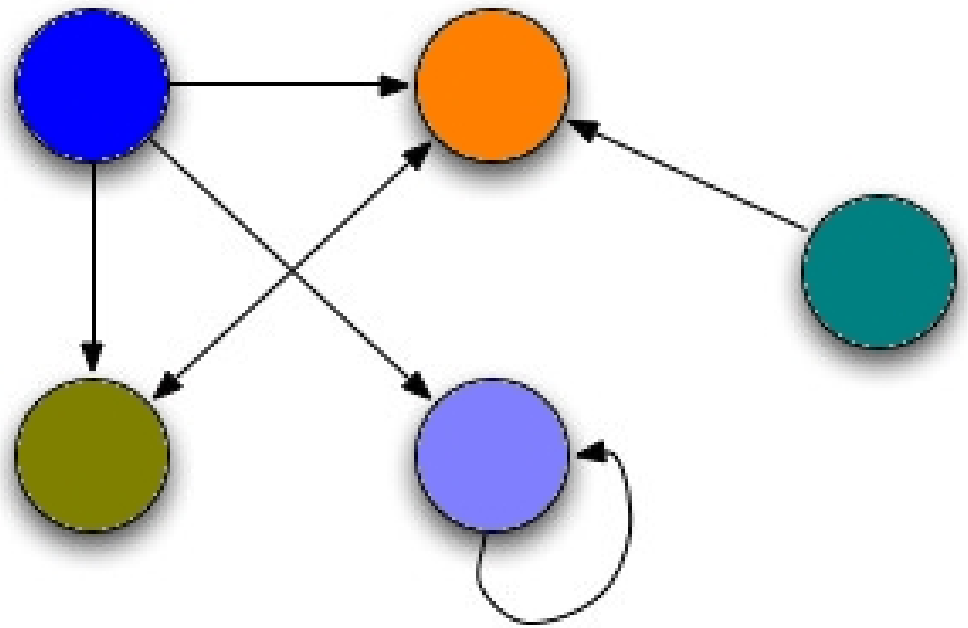
Grafi

Per arrivare a "formalizzare" cosa intendiamo per rete neurale, ci servono alcuni elementi di teoria dei grafi.



Un **grafo** $\mathcal{G}$ è una coppia $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ dove $\mathcal{V}$ è un insieme di **vertici** o *nodi* (i cerchietti nella figura) e $\mathcal{E}$ è un insieme di **archi** o *link* tra coppie di vertici (i segmenti tra nodi nella figura).
Quello della figura è un grafo **non orientato** (o *non diretto*) in quanto la relazione tra vertici rappresentata dagli archi è simmetrica (non c'è un verso univoco di percorrenza dell'arco da un vertice all'altro).

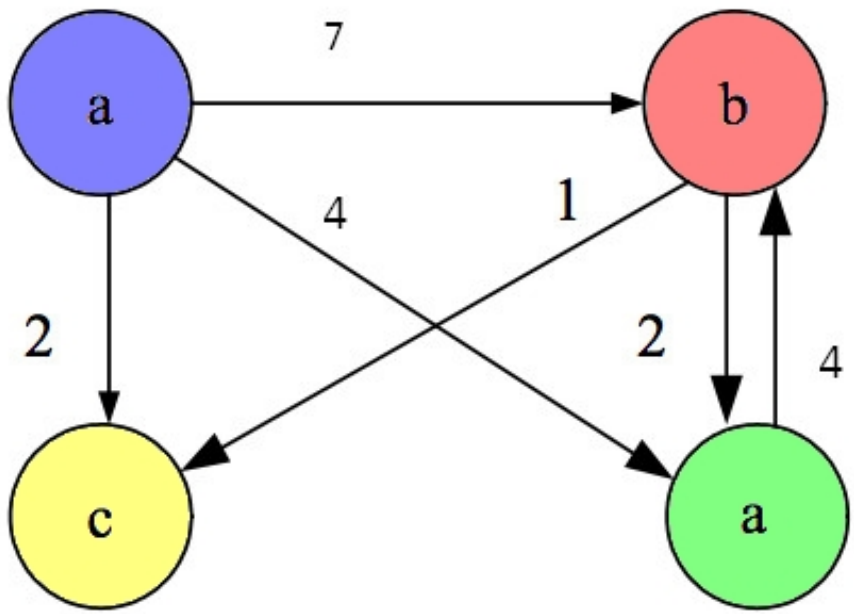
Nei grafi **orientati** (o *diretti*) gli archi hanno invece un ben preciso verso di percorrenza, nella figura è indicato dalla freccia:



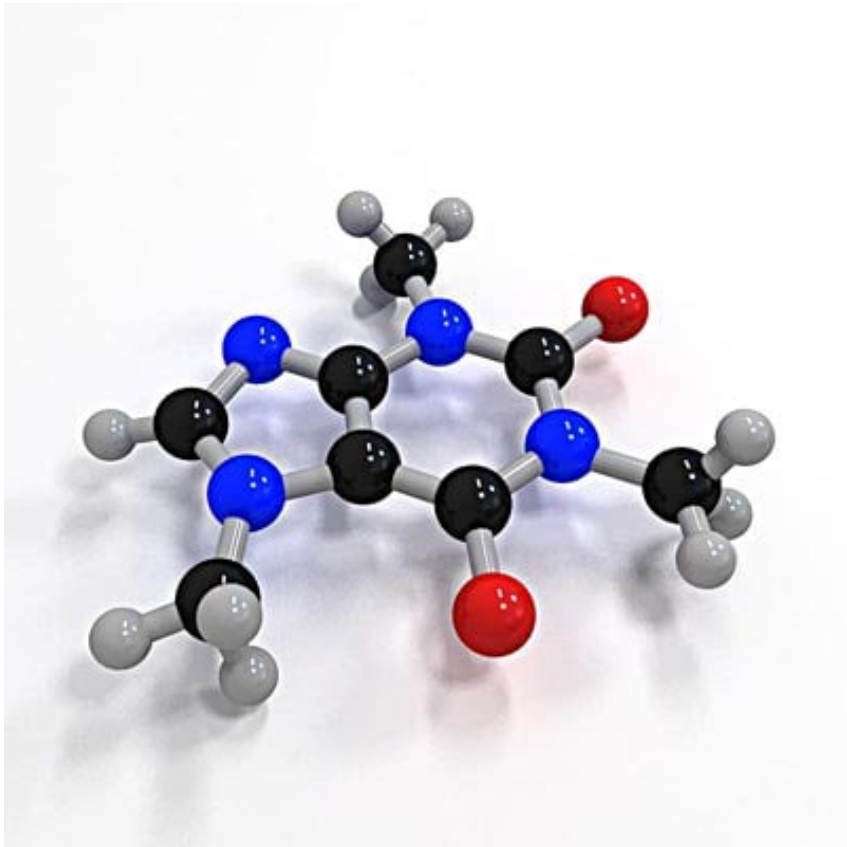
La relazione tra vertici rappresentata dagli archi è un *ordinamento parziale* dei vertici (individua una *precedenza di percorrenza* del grafo). Possono esserci riflessività (un nodo è in relazione con se stesso) e simmetrie (doppio verso di percorrenza, doppia freccia). Un grafo non orientato è implicitamente un caso particolare di grafo diretto dove tutti gli archi hanno doppia freccia.

Grafi etichettati: a ogni vertice e/o a ogni arco può essere associata un'*etichetta*.

- Etichettatura dei vertici $\ell_{\mathcal{V}} : \mathcal{V} \to \mathcal{L}_{\mathcal{V}}$ dove $\mathcal{L}_{\mathcal{V}}$ è l'insieme delle possibili etichette per i nodi
- Etichettatura degli archi $\ell_{\mathcal{E}} : \mathcal{E} \to \mathcal{L}_{\mathcal{E}}$ dove $\mathcal{L}_{\mathcal{E}}$ è l'insieme delle possibili etichette per gli archi

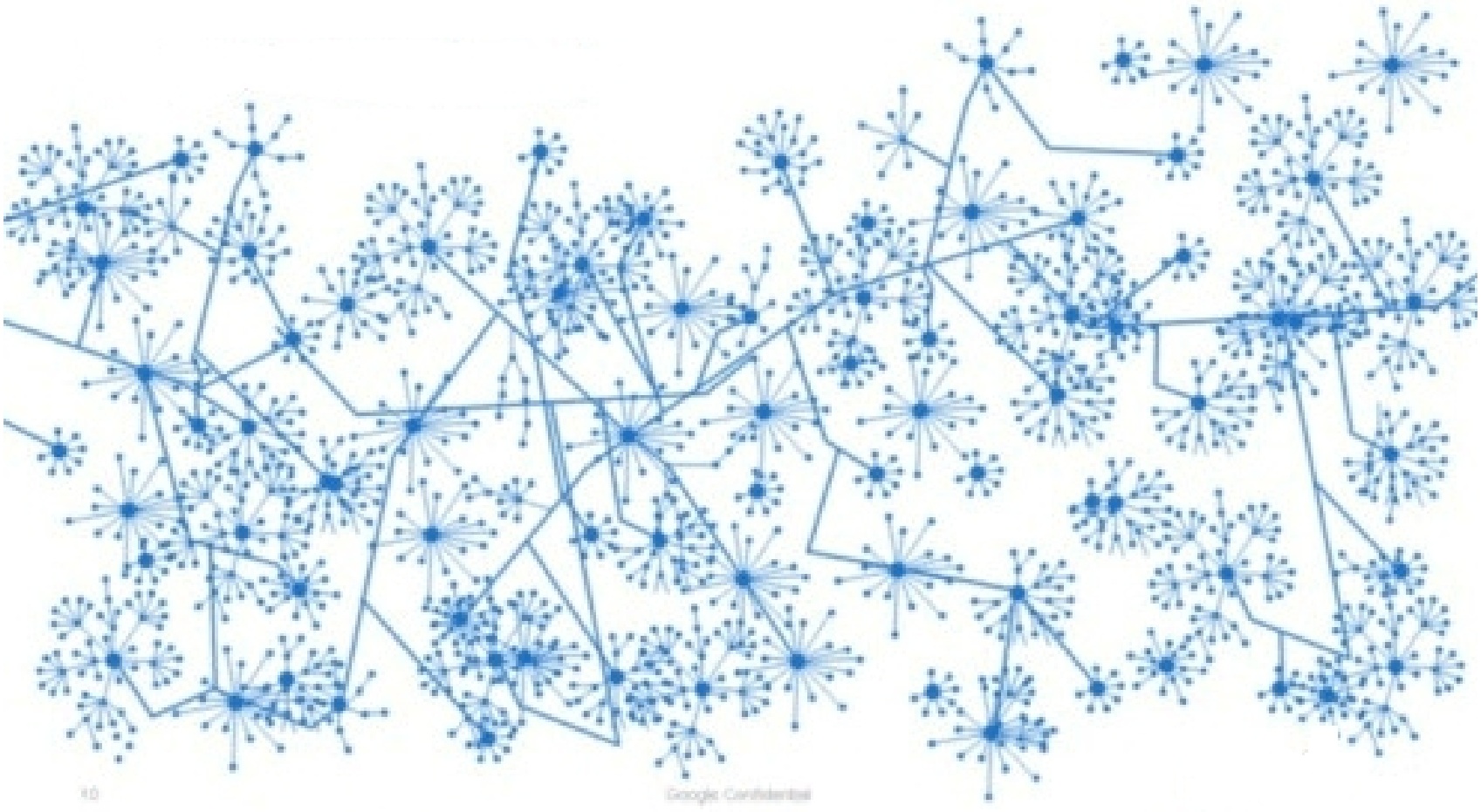


Esempio 1: molecole (ad es. in bioinformatica)



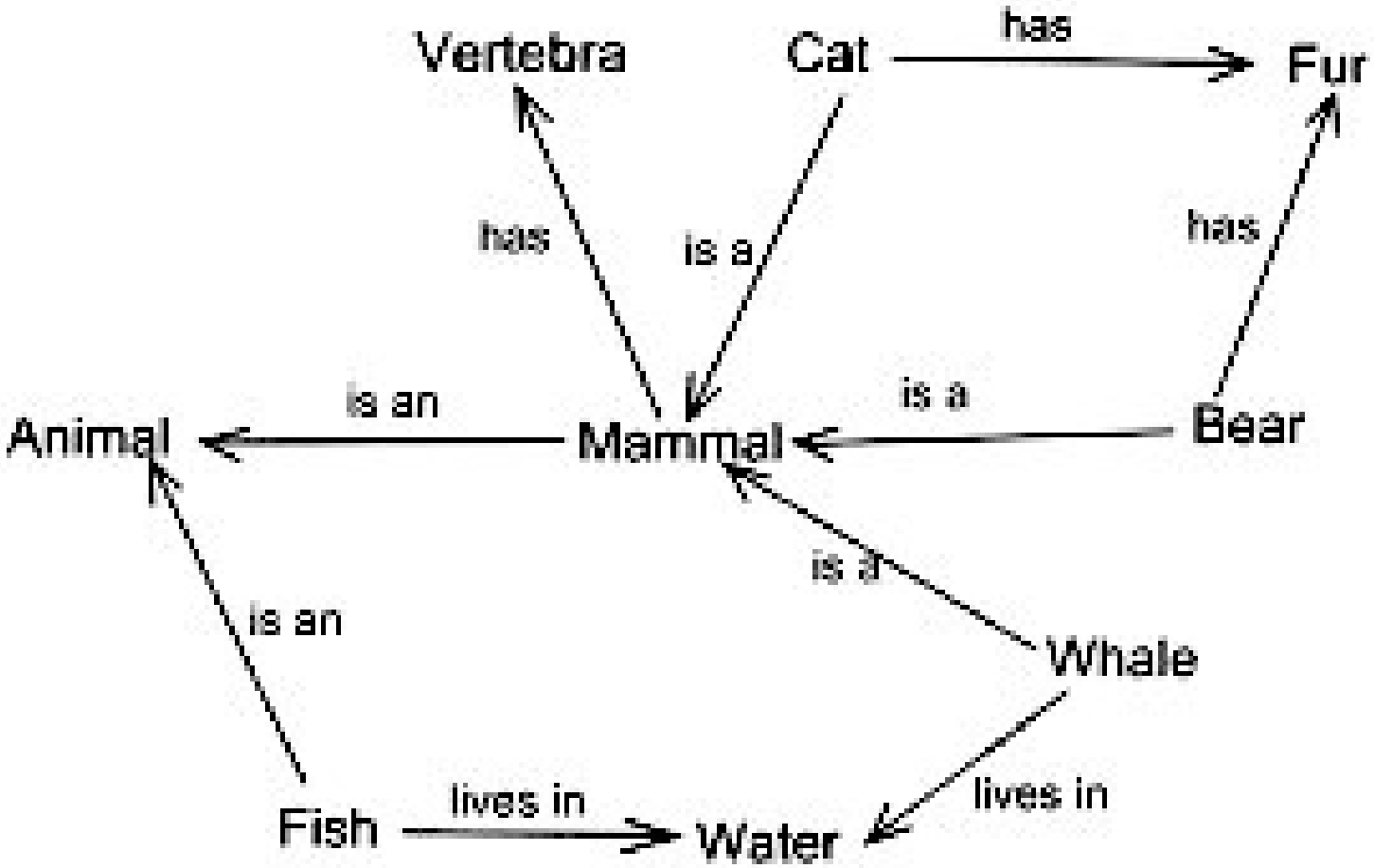
Gli atomi sono i vertici del grafo, i legami chimici sono gli archi. Le etichette dei nodi specificano il tipo di atomo e le sue caratteristiche chimico-fisiche. Le etichette degli archi specificano se il legame è semplice o doppio.

Esempio 2: il *world wide web*



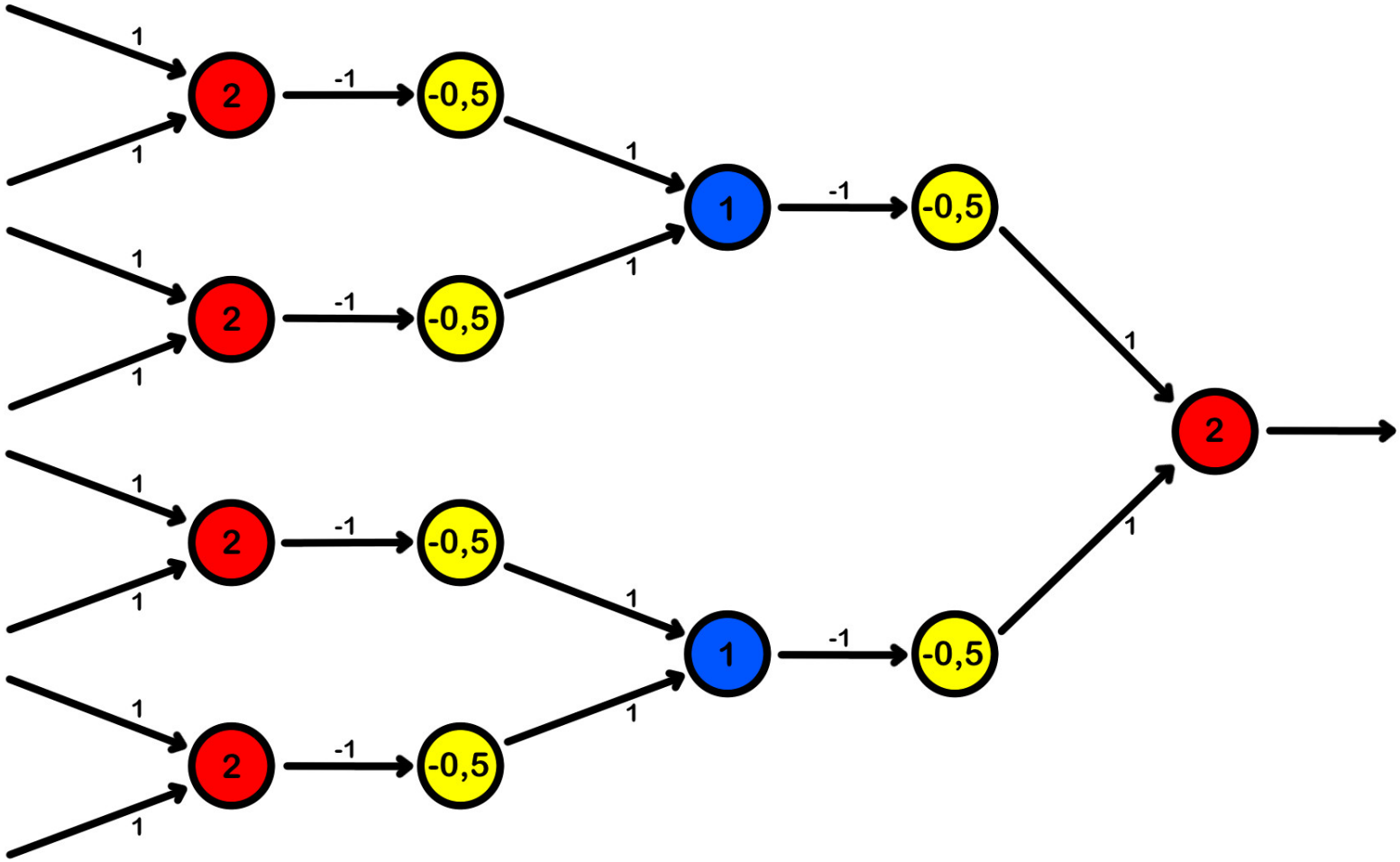
I vertici sono le pagine web, le cui etichette ne rappresentano il contenuto. Gli *hyperlink* fungono da archi, senza etichette.
Per designare grafi molto grandi si usa anche il termine "*reti*".
Esercizio: il WWW è un grafo orientato o non orientato?

Esempio 3: **reti semantiche** per rappresentare *basi di conoscenza* (come *WordNet* per l'elaborazione del linguaggio naturale)



I vertici sono i "concetti", gli archi rappresentano le relazioni semantiche tra concetti.

Esempio 4: reti di TLU



Le etichette degli archi sono i pesi delle connessioni, le etichette dei vertici sono i valori delle rispettive soglie θ.

Una **rete neurale artificiale (ANN)** è una tripla

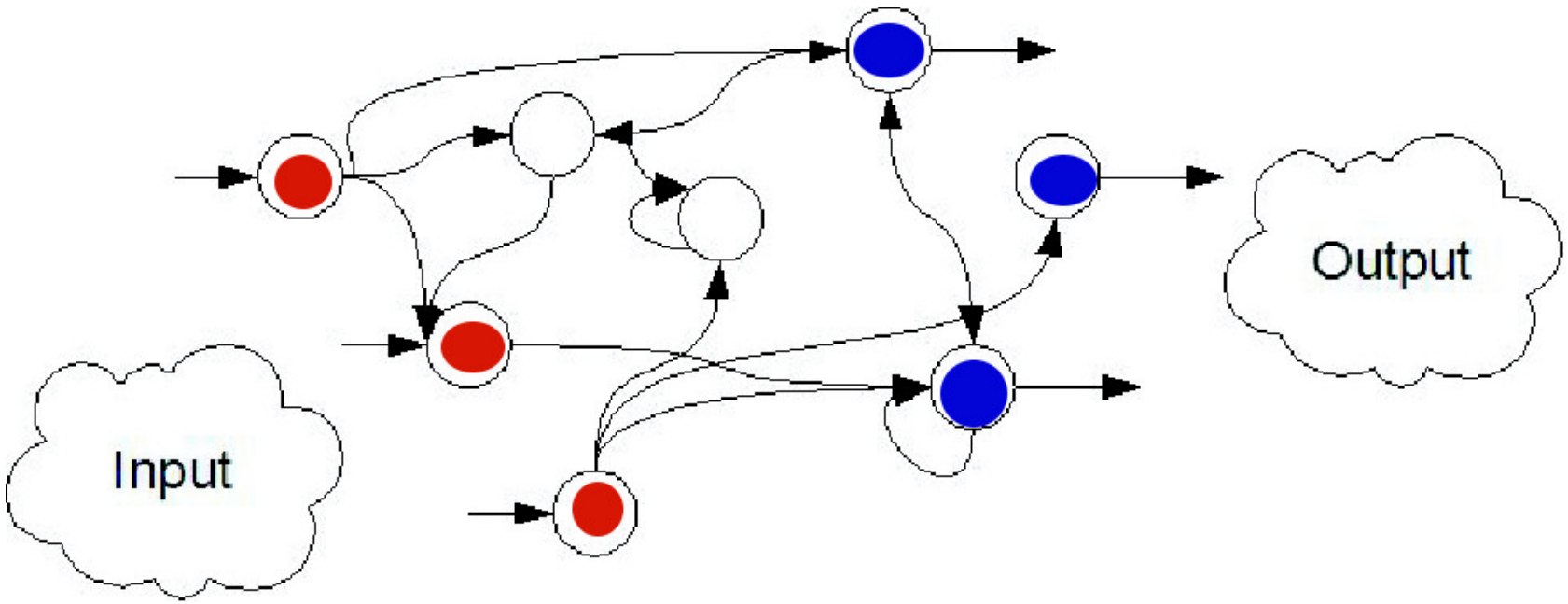
$$\mathscr{N} = (\mathscr{A}, \mathscr{D}, \mathscr{L})$$

dove

- $\mathscr{A}$ è l'**architettura** o topologia di $\mathscr{N}$ ("struttura fisica")
- $\mathscr{D}$ è la **dinamica** (regole di propagazione del segnale attraverso $\mathscr{N}$ e funzione calcolata dalla ANN)
- $\mathscr{L}$ è un **algoritmo di apprendimento** adatto a $\mathscr{N}$

Architettura

L'architettura è un grafo $\mathscr{A} = (\mathcal{V}, \mathcal{E})$ orientato o non orientato i cui nodi sono detti **neuroni** (o **unità**) e i cui archi sono detti **connessioni** (o *connessioni sinaptiche*).

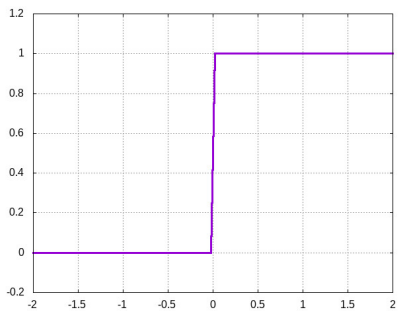
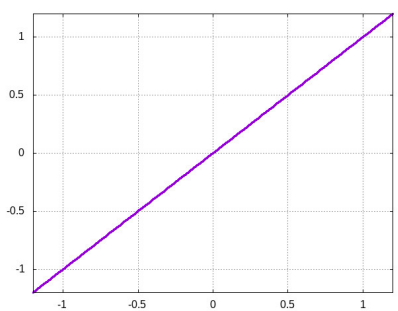
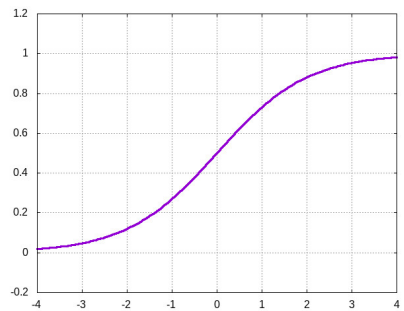
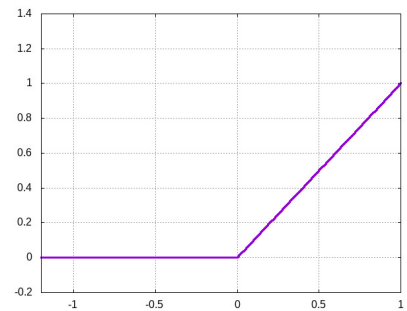


$\mathcal{V}$ è partizionato in 3 sottoclassi: **neuroni di input**, **neuroni di output** e **neuroni nascosti**.

Il grafo $\mathscr{A}$ è etichettato.

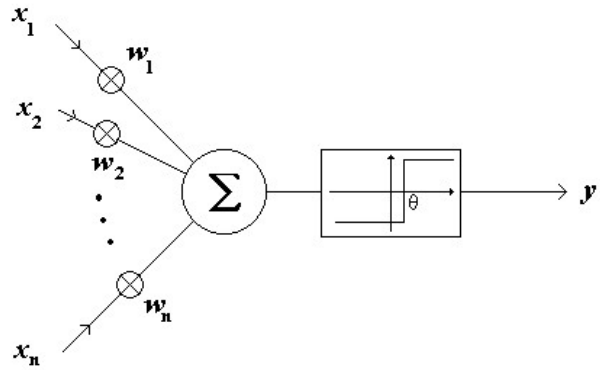
- Etichettatura delle connessioni: $\ell_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}$ (a ogni connessione ϵ è associato un numero reale $w = \ell_{\mathcal{E}}(\epsilon)$ detto **peso** della connessione ϵ, come nelle TLU).
- Etichettatura dei neuroni: $\ell_{\mathcal{V}} : \mathcal{V} \to \mathcal{F}$ dove $\mathcal{F}$ è l'insieme delle funzioni reali di variabile reale, $\mathcal{F} = \{f | f : \mathbb{R} \to \mathbb{R}\}$. A ogni unità v è quindi associata una funzione $f_v(\cdot) = \ell_{\mathcal{V}}(v)$ detta **funzione di attivazione** di v (come nelle TLU).

Esempi di funzioni di attivazione:

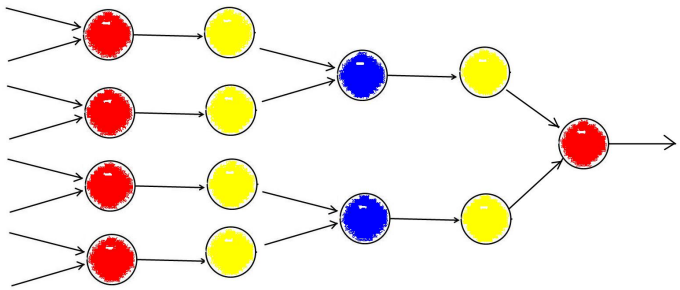
A gradino	Lineare	Sigmoide (logistica)	ReLU (*Rectifier*)
			
(Come TLU)	$y = x$	$y = \frac{1}{1+e^{-x}}$	$y = \max(0, x)$

Dinamica

Ricorda il funzionamento delle TLU ...



... ergo la "dinamica" di propagazione del segnale nelle reti di TLU:

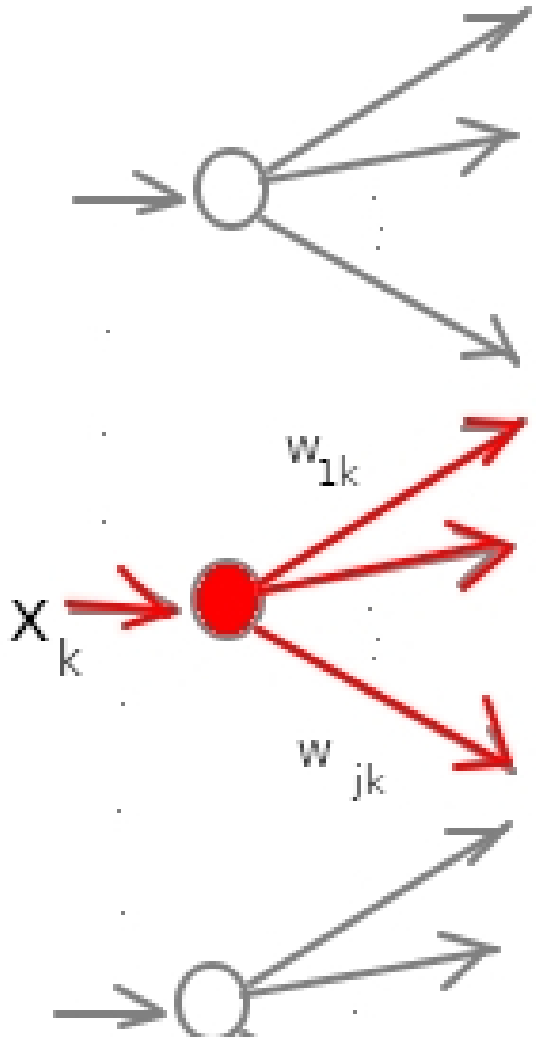


La dinamica $\mathscr{D}$ di una ANN $\mathscr{N}$ è il processo di propagazione (ovvero, trasformazione) dell'input attraverso la rete fino a ottenerne un output corrispondente. È l'equivalente della **computazione** in una MdT.

(1) Sia $\boldsymbol{x} = (x_1, \ldots, x_n)$ il vettore di feature numeriche in input. La ANN deve avere n **neuroni di input**, il k-imo dei quali riceve in input la componente x_k di $\boldsymbol{x}$.

- Per convenzione, ad ogni unità di input è associata la funzione di attivazione lineare (che realizza la trasformazione identica $y = x$).
- A fronte dell'input x_k, il neurone di input k-imo produce un output o_k dato dall'applicazione della propria funzione di attivazione, cioé $o_k = x_k$.
- Questo segnale di output o_k viene propagato lungo tutte le connessioni in uscita dal neurone di input k-imo e dirette a neuroni nascosti o di output della rete.
- Nell'attraversare la generica connessione j-ima, avente peso w_{jk}, il segnale o_k viene moltiplicato per w_{jk}. Il neurone cui la connessione j-ima punta (il neurone "destinazione") riceve dunque attraverso questa connessione un contributo $w_{jk} o_k$ alla propria attivazione.

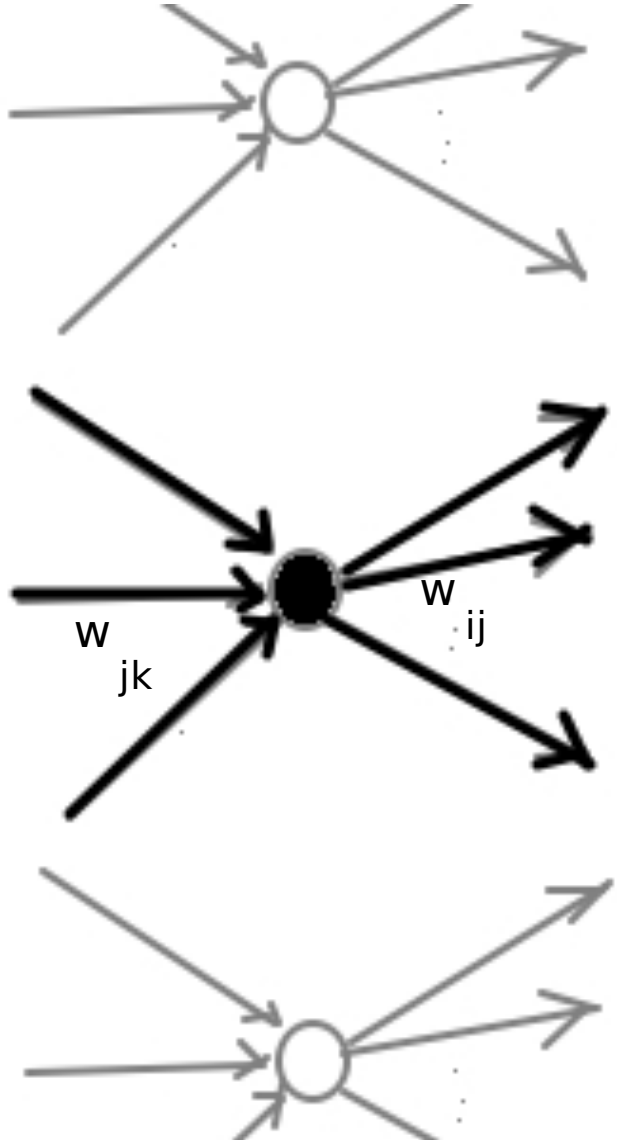
Graficamente:



(2) Consideriamo poi il j-imo **neurone nascosto**. Esso riceve in input una molteplicità di "segnali" $a_{j1}, a_{j2}, \ldots, a_{jn_j}$ provenienti da altri neuroni a monte (neuroni di input o altri neuroni nascosti) del tipo $a_{jk} = w_{jk} o_k$, dove k è un generico neurone "sorgente" collegato a j tramite una connessione avente peso w_{jk}.

- Al neurone j è associata la funzione di attivazione $f_j(.)$
- A fronte degli input $a_{j1}, \ldots, a_{jn_j}$ il neurone nascosto j-imo ne effettua la somma, ottenendo un unico valore $a_j \in \mathbb{R}$ detto **attivazione**: $a_j = \sum_{l=1}^{n_j} a_{jl}$, cioé $a_j = \sum_{l=1}^{n_j} w_{jl} o_l$
- il neurone j produce quindi un output o_j dato dall'applicazione ad a_j della propria funzione di attivazione, cioé $o_j = f_j(a_j)$
- Come per le unità di input, questo segnale o_j viene propagato lungo tutte le connessioni in uscita dal neurone nascosto j-imo e dirette a altri neuroni nascosti o di output della rete
- Ancora, nell'attraversare la generica connessione i-ima, avente peso w_{ij}, il segnale o_j viene moltiplicato per w_{ij}. Il neurone i cui la connessione i-ima punta (il neurone "destinazione") riceve a sua volta, attraverso questa connessione, un contributo $w_{ij} o_j$ alla propria attivazione.

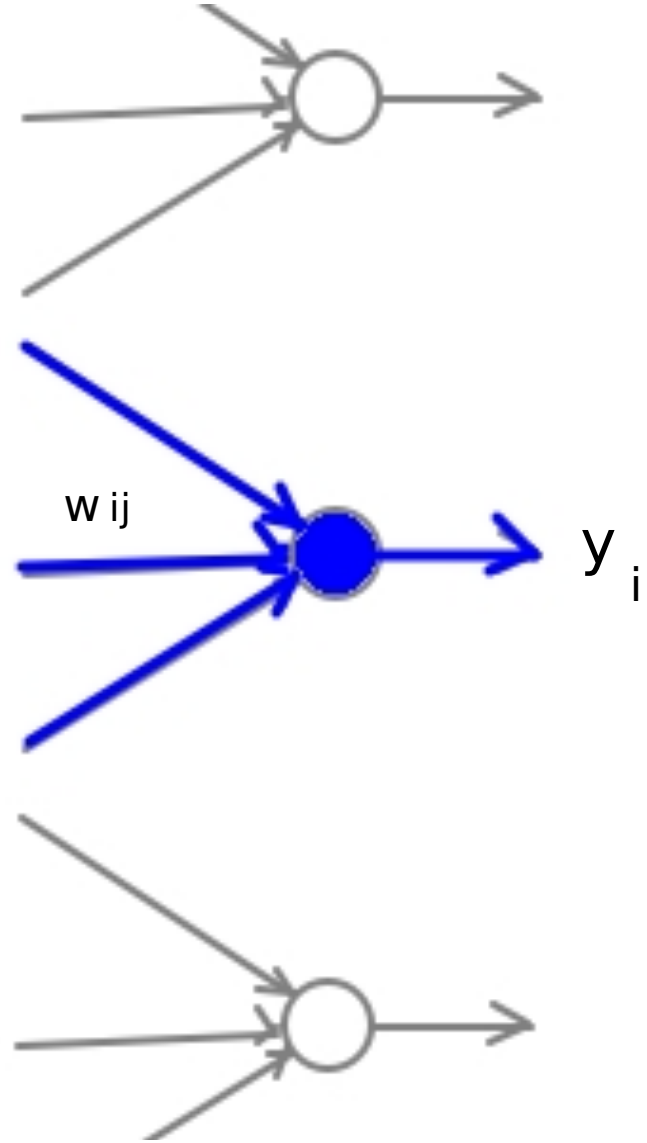
Graficamente:



(3) Per il generico **neurone di output** i-imo la dinamica è simile a quella dei neuroni nascosti. Esso riceve in input un'analoga molteplicità di "segnali" $a_{i1}, \ldots, a_{in_i}$ provenienti da altri neuroni a monte (neuroni di input o, più spesso, nascosti) del tipo $a_{ij} = w_{ij} o_j$, dove j è un generico neurone "sorgente".

- Al neurone i è associata la funzione di attivazione $f_i(.)$
- Esso fa dapprima la somma dei propri input $a_{ij}, \ldots, a_{in_i}$, ottenendone un'*attivazione* $a_i = \sum_{l=1}^{n_i} a_{il}$, cioé $a_i = \sum_{l=1}^{n_i} w_{il} o_l$
- il neurone i restituisce quindi un output $y_i = f_i(a_i)$
- y_i viene assunto come output i-imo della ANN $\mathscr{N}$
- Sia m il numero di neuroni di output di $\mathscr{N}$. Il vettore $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ costruito sui valori restituiti dalle funzioni di attivazione dei neuroni di output (presi nell'ordine) così ottenuti è l'**output della ANN** $\mathscr{N}$ calcolato sull'input attuale $\boldsymbol{x}$.
- La funzione $\varphi_{\mathscr{N}}: \mathbb{R}^n \to \mathbb{R}^m$ che ad ogni input $(x_1, \ldots, x_n)$ associa l'output $(y_1, \ldots, y_m)$ generato da $\mathscr{N}$ come appena descritto, si dice **funzione calcolata** da $\mathscr{N}$.

Graficamente:



Algoritmo di apprendimento

$\mathscr{L}$ è un *algoritmo* che ha per **INPUT**

- un'architettura iniziale $\mathscr{A}_0$ (con una ben precisa etichettatura iniziale delle connessioni, cioé un'assegnazione di valori iniziali ai pesi w di $\mathscr{A}_0$)
- una collezione di **esempi** $\mathscr{T} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots\}$ detta **training set**
- un numero reale positivo $\eta \in \mathbb{R}^+$ detto **learning rate** che controlla la velocità di apprendimento
- un numero $T \in \mathbb{N}$ di **epoche** di apprendimento (cioé, iterazioni sull'intero $\mathscr{T}$ di una **regola di apprendimento** caratteristica di $\mathscr{L}$)

e che restituisce come **OUTPUT**

- un'architettura addestrata $\mathscr{A}_T$ con una ben precisa etichettatura finale delle connessioni, cioé un'assegnazione di valori finali ai pesi w di $\mathscr{A}_0$.

Noi ci limiteremo ad algoritmi di apprendimento che modificano i pesi w lasciando inalterata la topologia del grafo $(\mathcal{V}, \mathcal{E})$.

- Ogni algoritmo di apprendimento $\mathscr{L}$ mira a migliorare un ben preciso **criterio di apprendimento** (*criterion function* o *loss function*) che è una misura quantitativa del "grado di adattamento" della rete $\mathscr{N}$ agli esempi in $\mathscr{T}$.
- Il cuore di $\mathscr{L}$ è la **regola di apprendimento**, che viene applicata iterativamente su tutti gli esempi $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots$:
  1. considera il prossimo esempio $\boldsymbol{z}_t$ in $\mathscr{T}$
  2. loop: per ogni connessione ϵ di $\mathscr{A}$
     - 2.1 sia $w = \ell_{\mathcal{E}}(\epsilon)$ il peso associato a ϵ
     - 2.2 determina una variazione Δw di w che produca un miglioramento del criterio di apprendimento valutato su $\boldsymbol{z}_t$
  3. al peso w di ogni connessione ϵ in $\mathscr{A}$ applica la corrispondente variazione Δw, ottenendo per ϵ una nuova etichetta $\ell_{\mathcal{E}}(\epsilon) = w'$ data da: $w' = w + \Delta w$

  L'intero processo è ripetuto, ripartendo dal primo esempio $\boldsymbol{z}_1$, tante volte ("*epoche*") quanto è il valore di T.

Nota: non esiste un algoritmo di apprendimento universale, applicabile a qualsiasi rete neurale. Esistono invece diversi algoritmi adatti a specifiche "famiglie" di ANN.

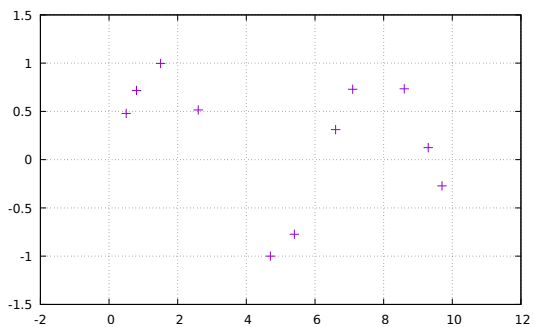
La natura del training set $\mathscr{T}$ dipende dal **tipo di apprendimento**.

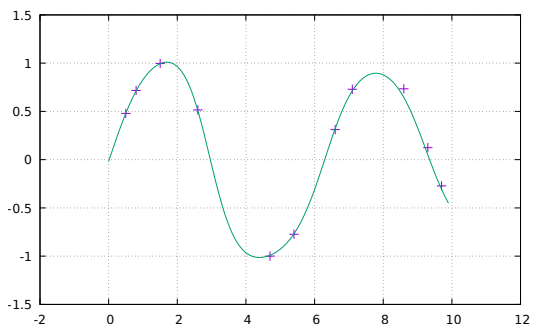
Tipi di apprendimento

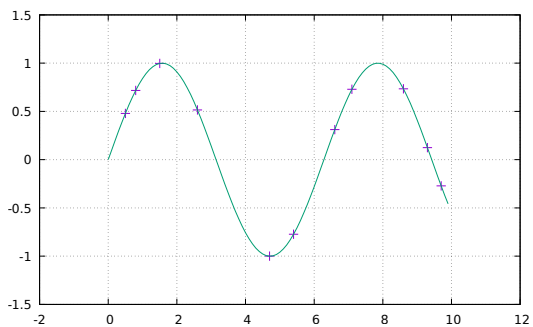
(i) Apprendimento supervisionato: $\boldsymbol{z}_i = (\boldsymbol{x}_i, \hat{\boldsymbol{y}}_i)$, quindi $\mathscr{T} = \{(\boldsymbol{x}_1, \hat{\boldsymbol{y}}_1), (\boldsymbol{x}_2, \hat{\boldsymbol{y}}_2), \ldots\}$ dove il generico $\boldsymbol{x}_i \in \mathbb{R}^n$ è un vettore di feature di input e $\hat{\boldsymbol{y}}_i \in \mathbb{R}^m$ è il corrispondente vettore di output "desiderato" (**target output**).

- Un esperto umano (il **supervisore**) ha condotto sul campo delle osservazioni, estraendone una rappresentazione in termini di feature, e a ciascuna di esse ha associato l'output corretto (il target per l'addestramento, osservato o altrimenti noto).
- Partendo dagli esempi, la ANN deve **inferire induttivamente** (cioé, apprenedere) la legge "universale" che spiega la relazione tra input e output.
- Assumendo che esista una relazione input-output (ovvero, stimolo-risposta) del tipo $\boldsymbol{y} = \phi(\boldsymbol{x}) + \sigma$ dove σ è un rumore aleatorio, ci si aspetta che la ANN impari a calcolare una funzione $\varphi(\cdot)$ tale per cui $\varphi(\boldsymbol{x}) \approx \phi(\boldsymbol{x})$ per ogni generico input $\boldsymbol{x}$, anche se $\boldsymbol{x}$ non è mai stato osservato in precedenza (i.e., non è incluso in $\mathscr{T}$).
- Quest'ultima proprietà è detta **capacità di generalizzazione** della ANN.

Lo scenario è quello visto nella demo di inizio corso: da una collezione $\mathscr{T}$ di esempi di training del tipo $(x_i, \hat{y}_i)$, ovvero $\mathscr{T} = \{(0.5, 0.48), (0.8, 0.72), (1.5, 1.00), (2.6, 0.52), (4.7, -0.99), (5.4, -0.77), (6.6, 0.31), (7.1, 0.73), (8.6, 0.74), (9.3, 0.12), (9.7, -0.27)\}$ (*fig. di sinistra*), la ANN apprende una legge $y = \varphi(x)$ (*fig. di centro*) in grado di *spiegare* gli esempi e generalizzare a nuovi input mai osservati in precedenza. La legge $\varphi(x)$ è un buon modello della vera legge $\phi(x)$ soggiacente alla generazione dei valori osservati in $\mathscr{T}$ (*fig. di destra*).







Esempi applicativi:

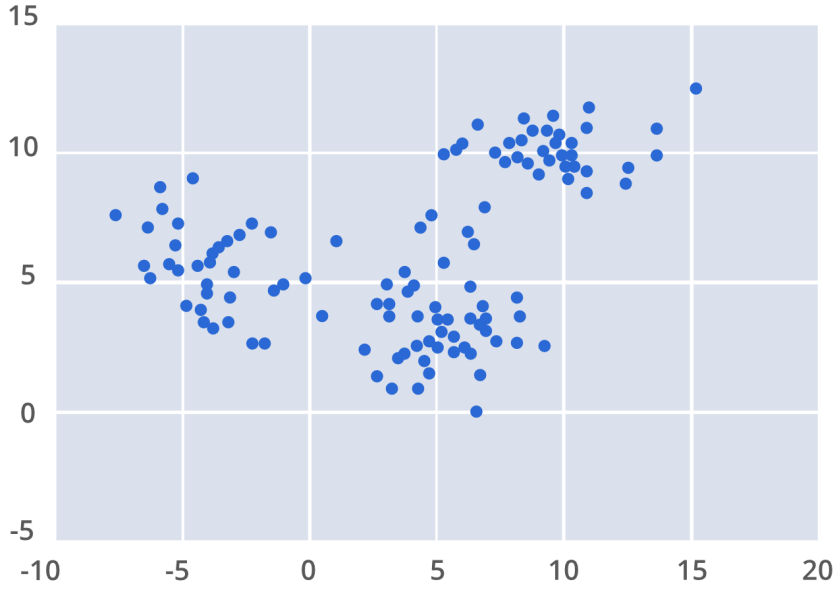
- Un **satellite geostazionario** acquisisce immagini di un lago. Da ciascuna di esse vengono estratti dei vettori di feature visive $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$. Contemporaneamente, dal lago vengono prelevati campioni d'acqua che sono analizzati in laboratorio ottenendo i rispettivi quantitativi di clorofilla $\hat{y}_1, \hat{y}_2, \ldots$. Una ANN viene addestrata su $(\boldsymbol{x}_1, \hat{y}_1), (\boldsymbol{x}_2, \hat{y}_2), \ldots$ perché apprenda un *modello di regressione* $y \approx \varphi(\boldsymbol{x}) \pm \sigma$ che da una qualsiasi immagine satellitare determini la corrispondente concentrazione di clorofilla nel lago.
- Nel **riconoscimento di caratteri manoscritti** si digitalizza una grande quantità di testo manoscritto. Per ogni carattere si estraggono feature $\boldsymbol{x}_i$. Un supervisore umano prepara un *dataset* accoppiando a ciascun vettore di feature $\boldsymbol{x}_i$ un vettore $\hat{\boldsymbol{y}}_i$ a componenti booleane 0/1 le quali sono tutte nulle tranne la k-ima allorquando $\boldsymbol{x}_i$ rappresenti un esempio di carattere k-imo (secondo l'ordinamento lessicografico: "A" è il carattere 1, "B" è il carattere 2, ecc.). Su questo dataset, una ANN viene addestrata a classificare le immagini di caratteri.

- I medici di una struttura specialistica raccolgono dati da test eseguiti sui pazienti. Ogni paziente è rappresentato da un vettore di feature $\boldsymbol{x}$ le cui componenti sono, ordinatamente, i risultati (numerici) dei test. I medici associano a $\boldsymbol{x}$ un'etichetta $\hat{y}$ binaria: se al paziente è stata diagnosticata una specifica malattia rara allora $\hat{y} = 1$, altrimenti $\hat{y} = 0$. Una ANN viene addestrata su una molteplicità di siffatte coppie $(\boldsymbol{x}, \hat{y})$ perché impari la relazione tra l'esito dei test e la presenza della patologia. La ANN viene infine usata come **sistema di supporto alla diagnostica**.
- Sia $x_1, x_2, \ldots, x_t, \ldots$ una **serie temporale finanziaria** osservata nel corso di 5 anni. I dati $x_1, \ldots, x_5$ sono i valori di un titolo nei rispettivi giorni della prima settimana della serie (da lunedì a venerdì), i dati $x_6, \ldots, x_{10}$ rappresentano la seconda settimana, e così via. Si "può" addestrare una ANN come *strumento predittivo* preparando un training set costituito da coppie $(\boldsymbol{x}, \hat{y})$ dove il target $\hat{y}$ della predizione è il valore che il titolo aveva il giorno t, un lunedì (alla riapertura dei mercati), basandosi sui valori osservati nel mese precedente, cioé sul vettore di feature

$$\boldsymbol{x} = (x_{t-20}, x_{t-19}, \ldots, x_{t-2}, x_{t-1})$$

(ii) Apprendimento non supervisionato (*unsupervised learning*): $\boldsymbol{z}_i = \boldsymbol{x}_i$, quindi $\mathscr{T} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots\}$ dove il generico $\boldsymbol{x}_i \in \mathbb{R}^n$ è un vettore di feature di input.

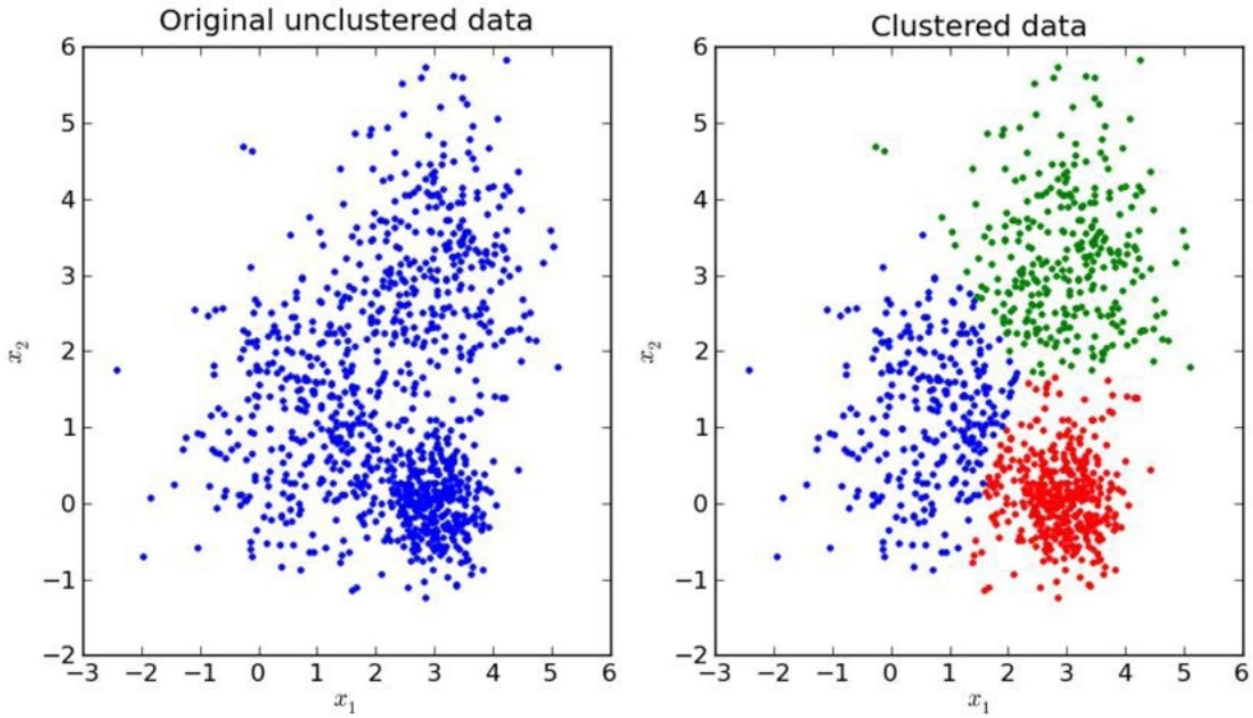
- Un esperto umano (o la macchina stessa) ha condotto sul campo delle osservazioni, estraendone le feature.
- Partendo dagli esempi, la ANN deve **inferire induttivamente** (cioé, apprendere) una legge in grado di "spiegare" i dati e/o di estrarre da essi l'informazione rilevante in essi incapsulata.
- Esempio grafico di $\mathscr{T}$ non supervisionato con feature bidimensionali:



- Esistono **tre scenari principali**:

Scenario 1: clustering

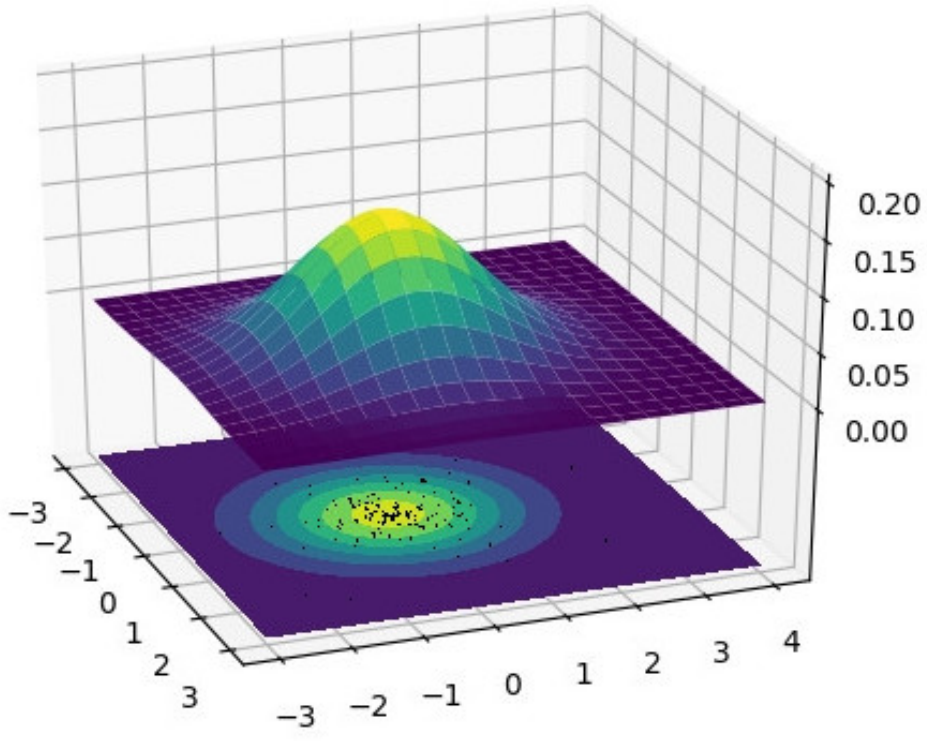
I dati vengono automaticamente partizionati dalla ANN in "nubi" (i *cluster*) ben separate le une dalle altre e ad alta coesione interna.



Se le feature sono state estratte in modo significativo, si può ipotizzare che il clustering **categorizzi** gli esempi $\mathscr{T}$ (*fig. di sinistra*) in base a un **criterio di similarità** (*fig. di destra*), oppure che un problema iniziale complesso venga scomposto in sotto-problemi più semplici (approccio **divide et impera**).

Scenario 2: stima probabilistica

I vettori di feature vengono pensati come variabili aleatorie generate da un certo processo stocastico latente, la ANN apprende (i.e., *stima*) la **legge di probabilità** che sta dietro la distribuzione dei dati.

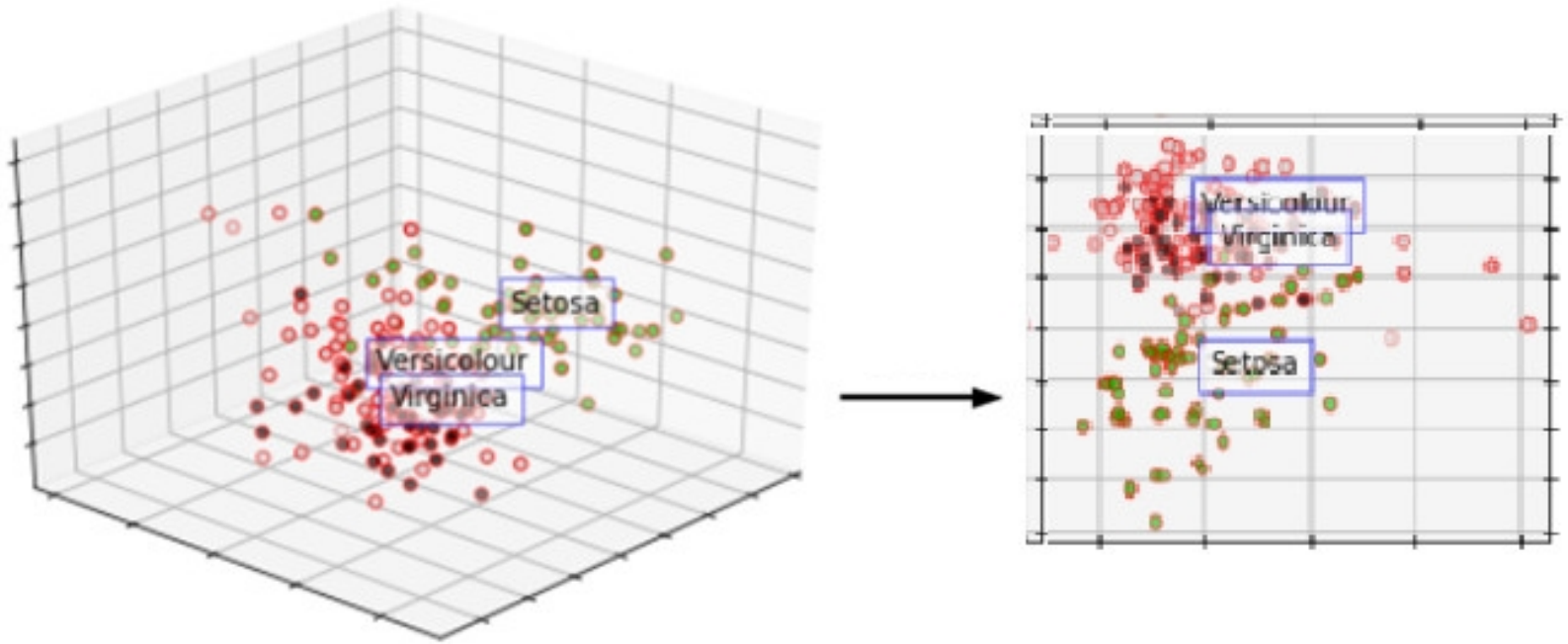


Probabilisticamente, **conoscere la legge di probabilità significa sapere "tutto" del fenomeno** di cui $\mathscr{T}$ è dunque un campione: come si distribuisce nello spazio delle feature, come si caratterizza rispetto ad altri fenomeni, come si possano generare nuovi campioni del medesimo fenomeno estraendoli a caso dalla legge stessa.

Scenario 3: estrazione di feature/riduzione della dimensionalità

La ANN apprende da $\mathscr{T}$ a realizzare una trasformazione $\boldsymbol{x}' = \varphi(\boldsymbol{x})$ in cui il vettore di feature $\boldsymbol{x} \in \mathbb{R}^n$ viene proiettato in un sotto-spazio a dimensionalità m ridotta ($m < n$) e la sua immagine $\boldsymbol{x}' \in \mathbb{R}^m$ preserva la maggior parte dell'informazione "utile" presente originariamente in $\boldsymbol{x}$.

• Ad esempio, per un certo task di classificazione a 3 categorie con vettori di feature tridimensionali si ottiene una rappresentazione bidimensionale:



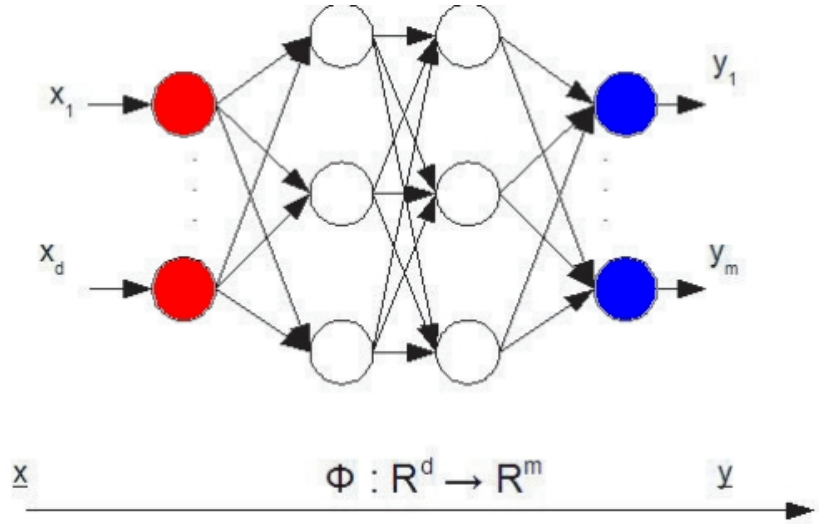
(iii) Apprendimento semi-supervisionato (o **parzialmente supervisionato**): il training set contiene sia esempi del tipo $\boldsymbol{z}_i = (\boldsymbol{x}_i, \hat{\boldsymbol{y}}_i)$ che del tipo $\boldsymbol{z}_j = \boldsymbol{x}_j$, dove $\boldsymbol{x}_i \in \mathbb{R}^n$ e $\boldsymbol{x}_j \in \mathbb{R}^n$ sono vettori di feature, mentre $\hat{\boldsymbol{y}}_i \in \mathbb{R}^m$ è il target output per $\boldsymbol{x}_i$.

• Sia $\mathscr{S} = \{(\boldsymbol{x}_1^{(s)}, \hat{\boldsymbol{y}}_1), (\boldsymbol{x}_2^{(s)}, \hat{\boldsymbol{y}}_2) \ldots\}$ un training set supervisionato e sia $\mathscr{U} = \{\boldsymbol{x}_1^{(u)}, \boldsymbol{x}_2^{(u)}, \ldots\}$ un training set unsupervised. La loro unione $\mathscr{T} = \mathscr{S} \cup \mathscr{U}$ è un training set semi-supervisionato.

• In genere si ha $|\mathscr{U}| \gg |\mathscr{S}|$, cioé ci sono moltissimi esempi non supervisionati a disposizione (esempio: pagine *web*) ma le risorse a disposizione consentono di procedere alla supervisione da parte degli esperti di una frazione limitata dell'intera messe di dati.

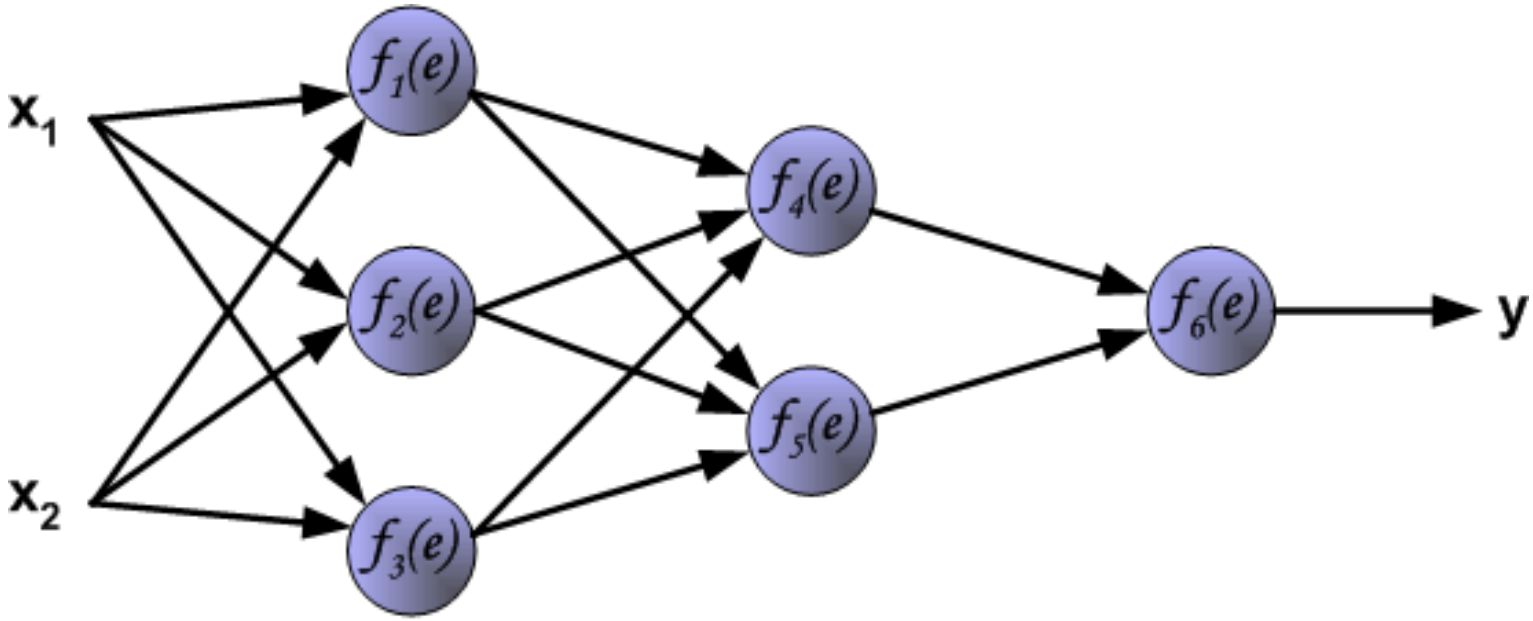
• In genere si inizia ad addestrare la ANN su $\mathscr{S}$ perché apprenda un modello $\boldsymbol{y} = \varphi(\boldsymbol{x})$, poi si applica $\varphi(.)$ ad alcuni dati di $\mathscr{U}$ particolarmente "vicini" a quelli di $\mathscr{S}$ (ovvero, sui quali la ANN si sente particolarmente "confidente") e li si etichetta con l'output restituito dalla ANN. I nuovi dati così etichettati possono essere inseriti in $\mathscr{S}$, ottenendone un nuovo training set supervisionato $\mathscr{S}'$ più esteso, e da esso un nuovo modello $\boldsymbol{y} = \varphi'(\boldsymbol{x})$ viene appreso dalla ANN, e così via.

Multilayer Perceptron (MLP)

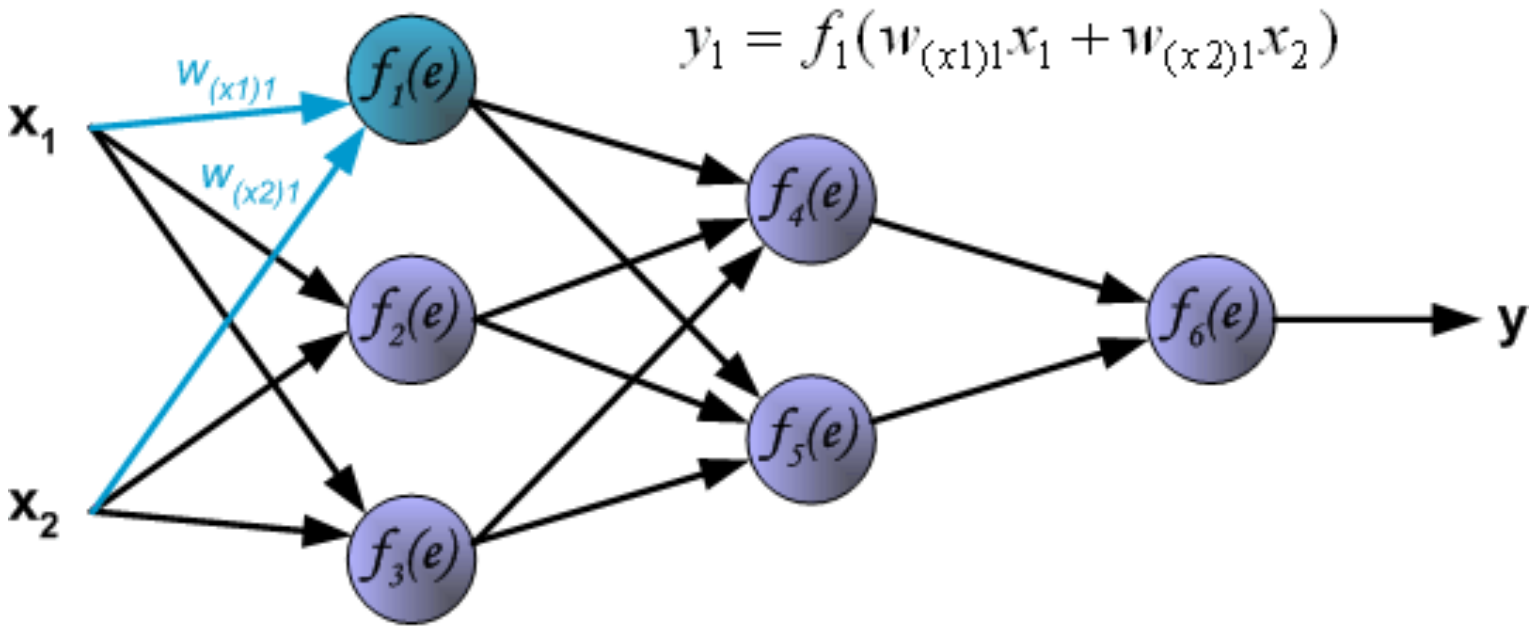


- Architettura a **strati** (*layer*) e *fully connected* a propagazione "in avanti" (**feed-forward**)
- Funzioni di attivazione sigmoidi e/o lineari. Tutti i neuroni di un certo strato hanno la medesima funzione di attivazione
- Dinamica: **propagazione simultanea del segnale** (senza ritardi) dal layer di input a quello di output (attraversando zero, uno, o più strati nascosti)
- **Apprendimento supervisionato** tramite l'**algoritmo di apprendimento** $\mathscr{L}$ detto "**backpropagation**".

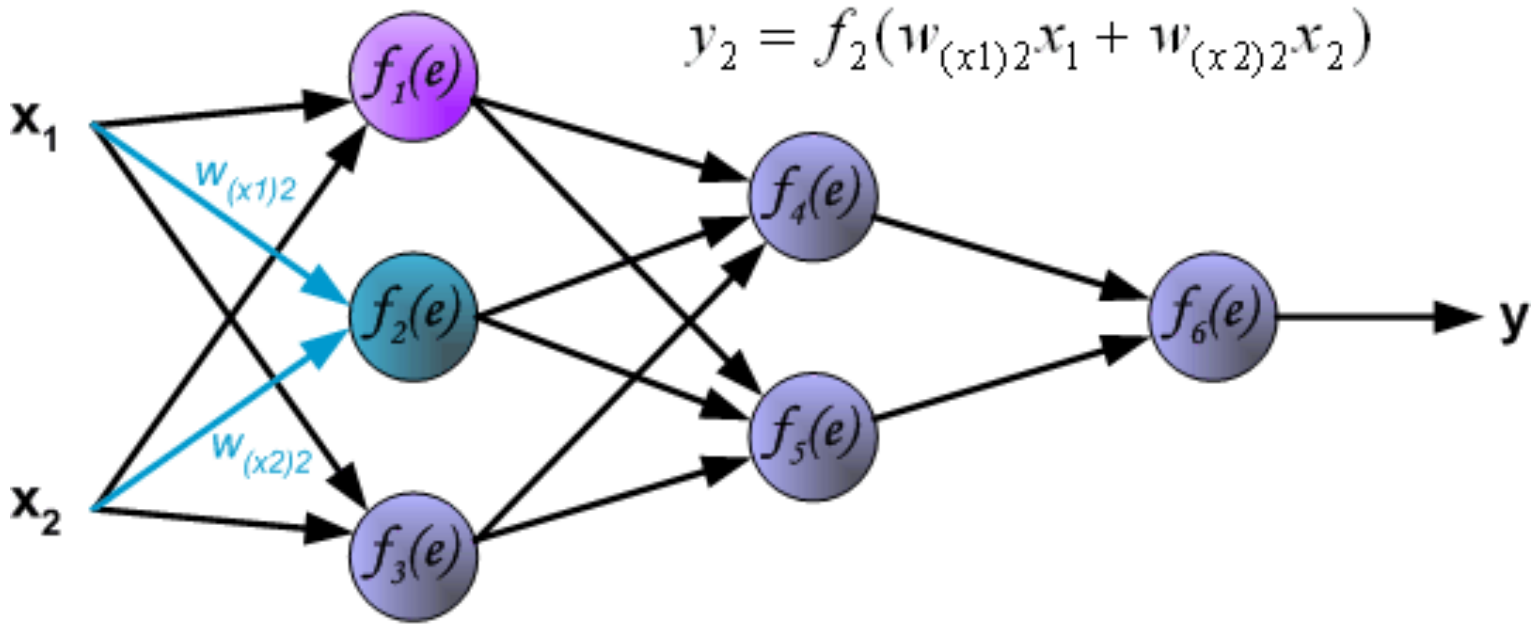
MLP: dinamica



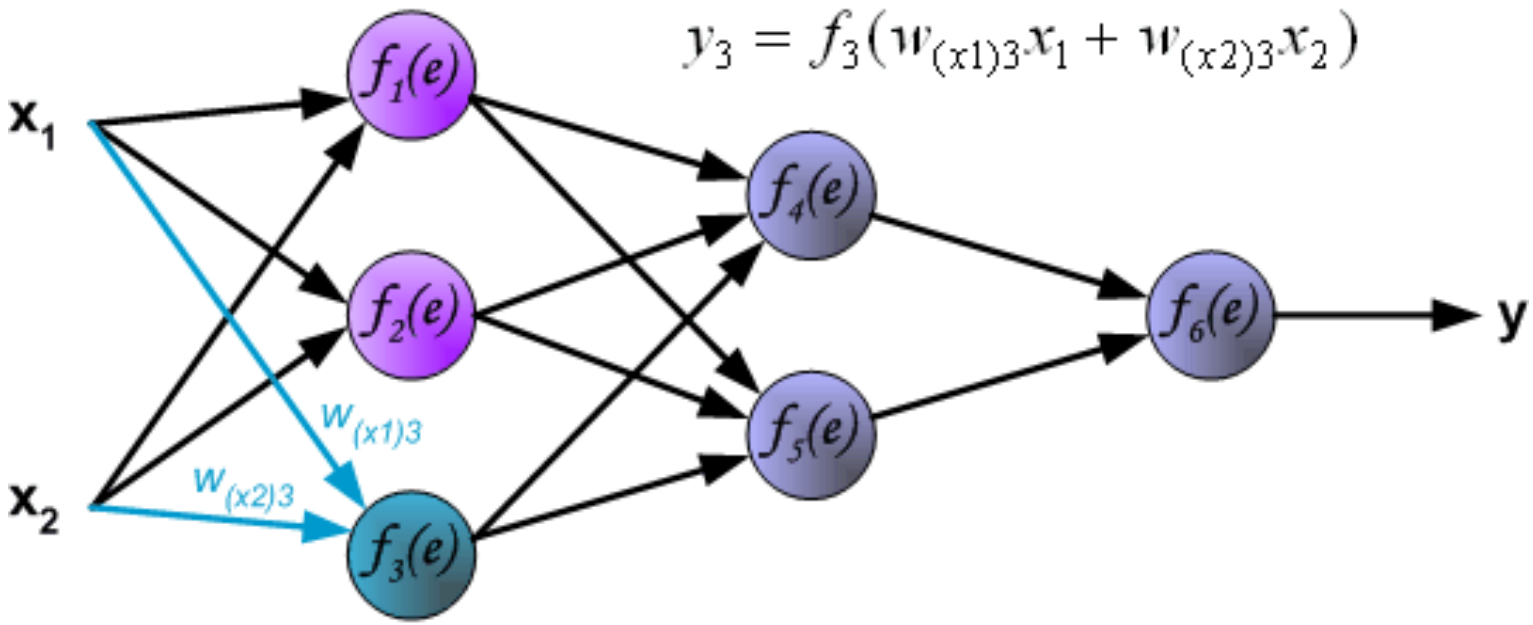
MLP: dinamica



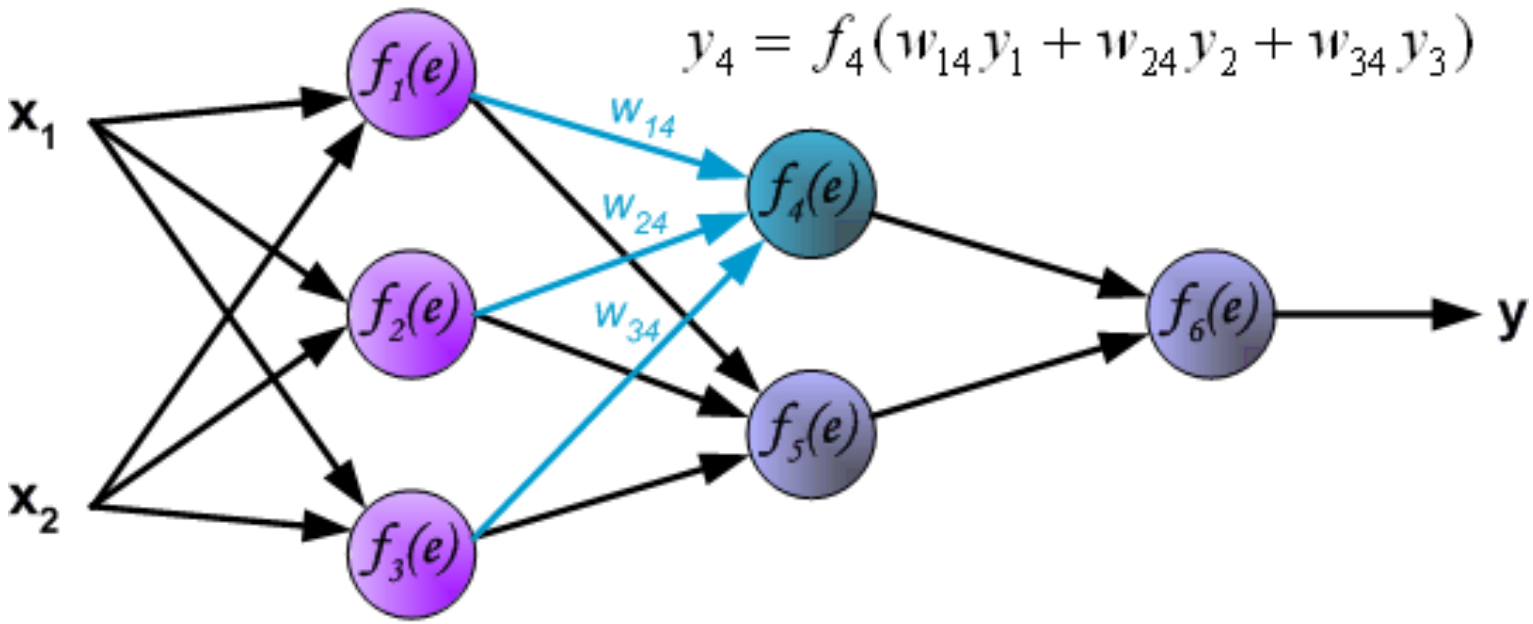
MLP: dinamica



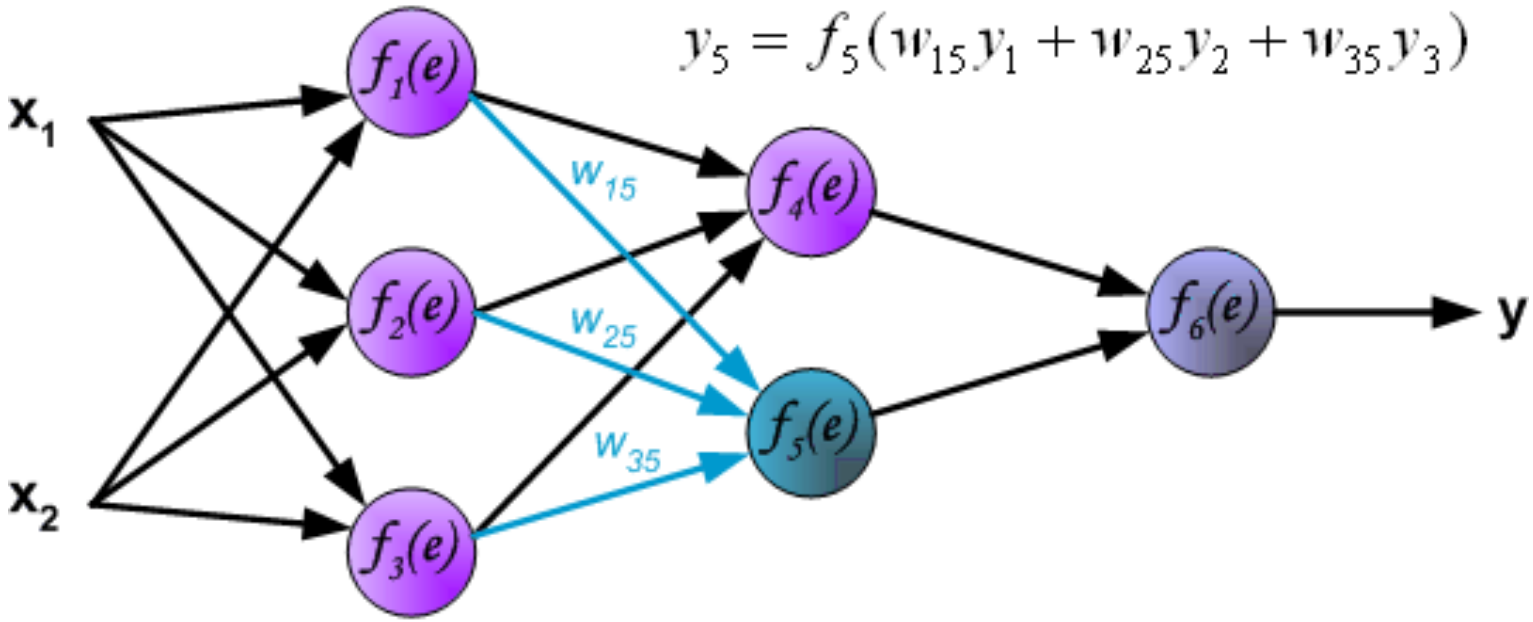
MLP: dinamica



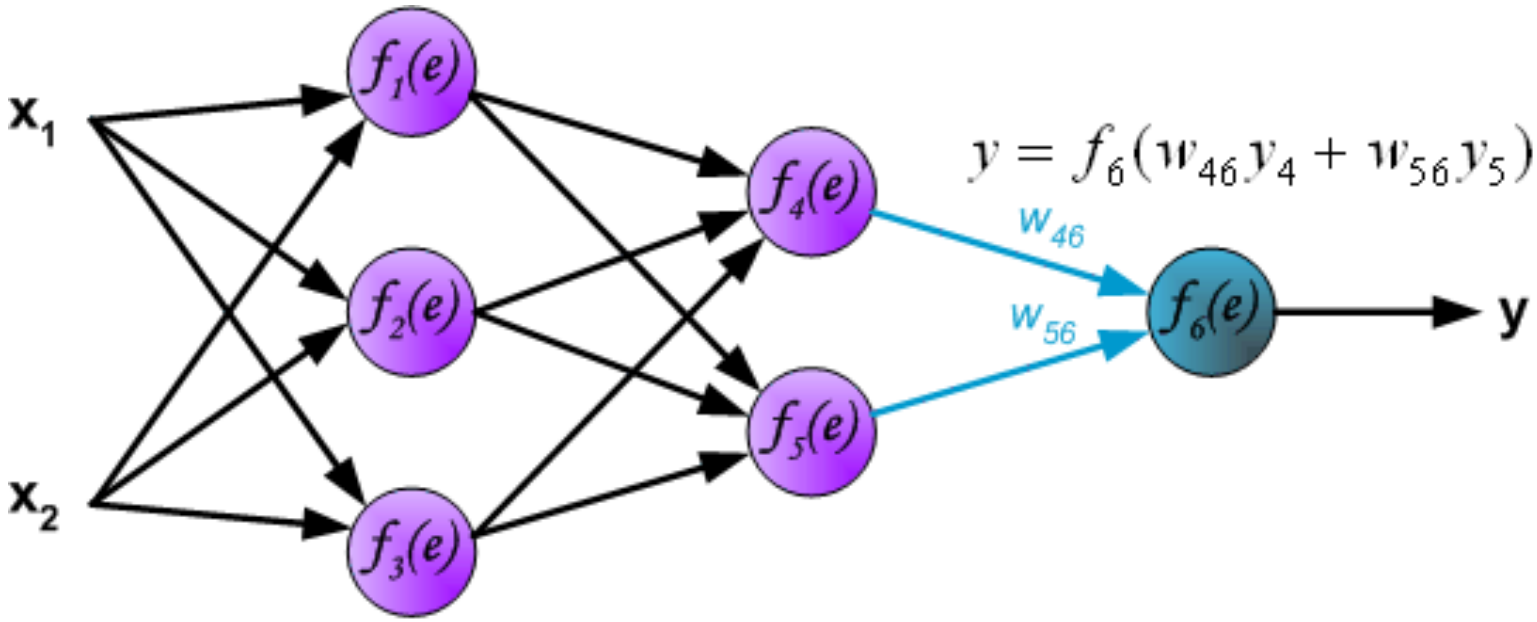
MLP: dinamica



MLP: dinamica

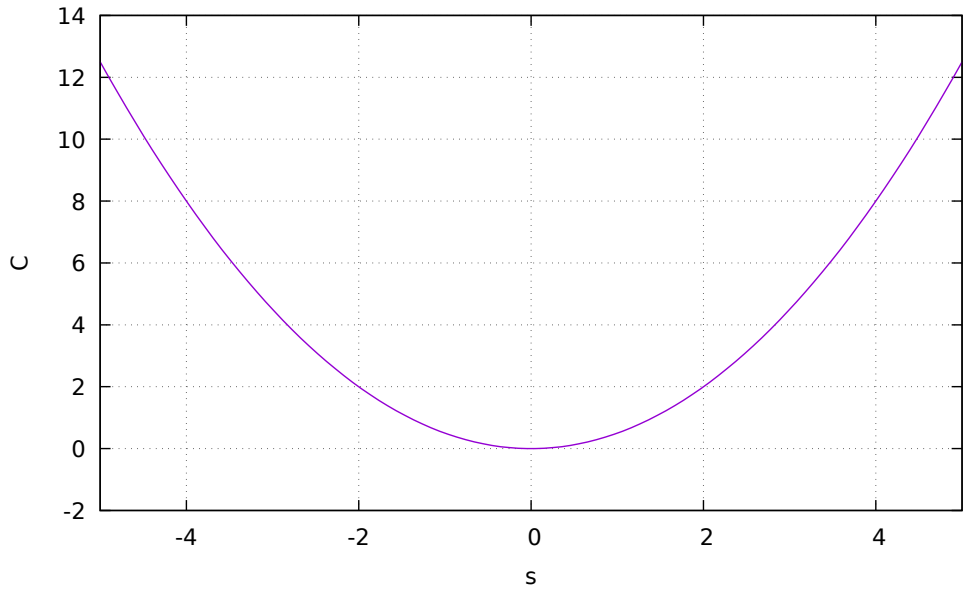


MLP: dinamica



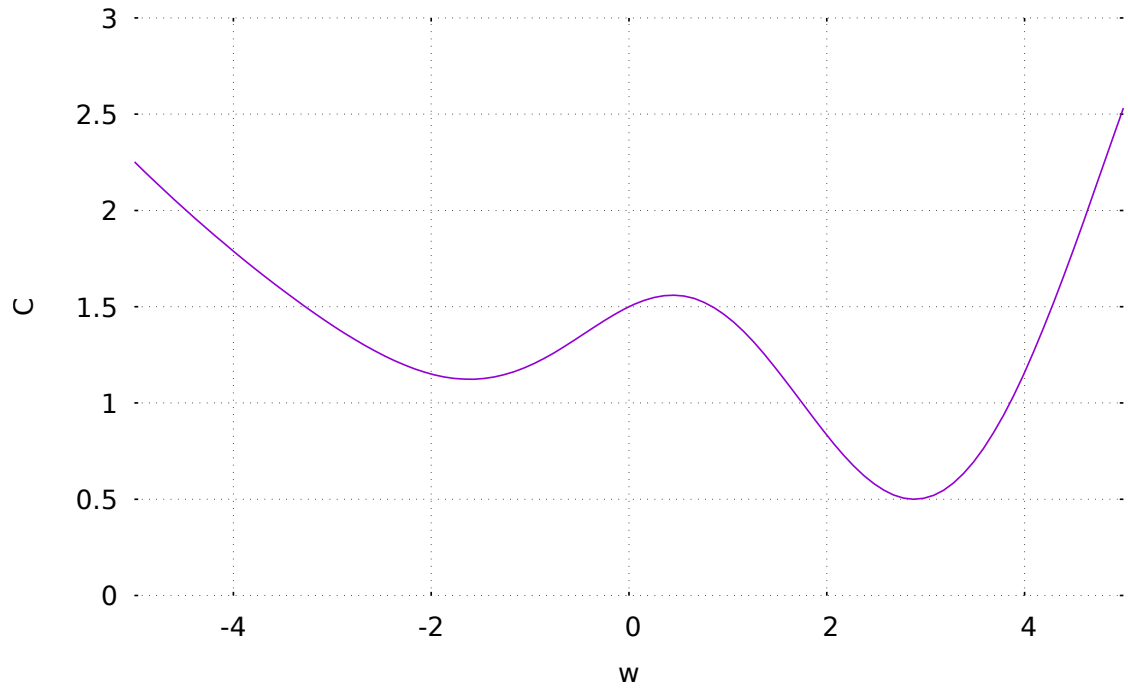
MLP: apprendimento tramite Backpropagation (BP)

- Algoritmo supervisionato, $\mathscr{T} = \{(\boldsymbol{x}_1, \hat{\boldsymbol{y}}_1), (\boldsymbol{x}_2, \hat{\boldsymbol{y}}_2), \ldots\}$
- **Criterio**: minimizzazione della distanza quadratica C tra l'output del MLP e il target output, $C = \frac{1}{2} \sum_{i=1}^{m} (\widehat{y}_i - y_i)^2$
- C **è funzione dello** scarto $s = \widehat{y}_i - y_i$, che è funzione dello **specifico valore dei pesi**:



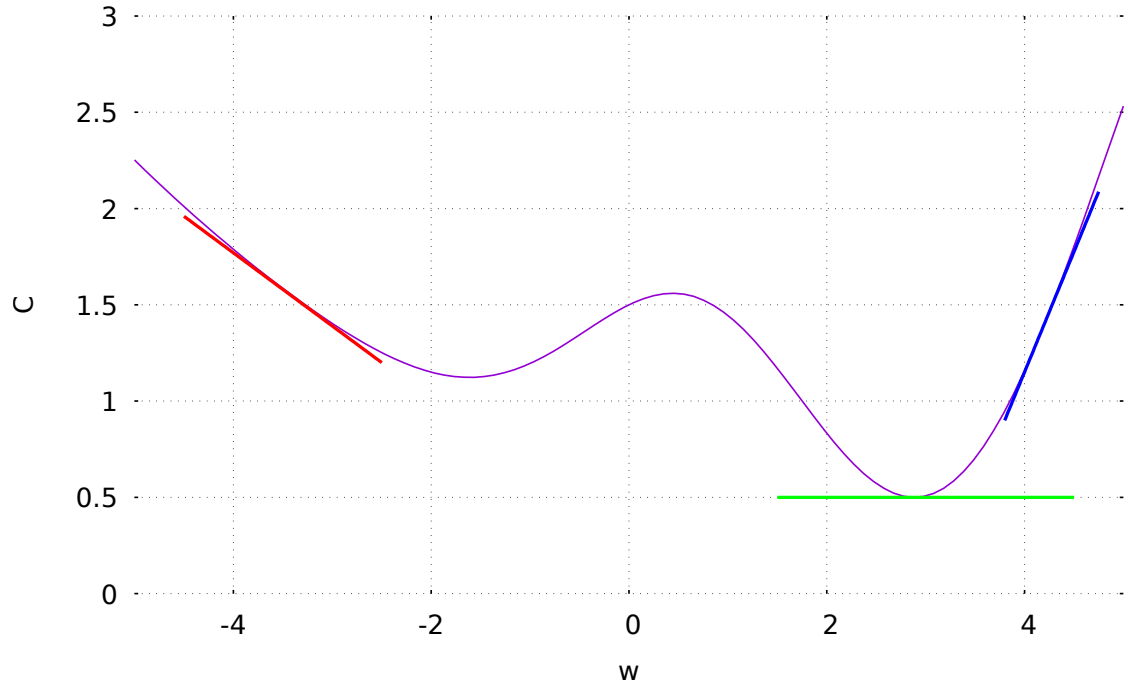
- C ha i seguenti vantaggi:
  1. È sempre non-negativo, a prescindere dal valore dei pesi w
  2. Ha uno e un solo minimo ($s = 0$)
  3. Ha una forma *morbida*, **continua e derivabile**
  4. Penalizza distanze grandi tra y_i e $\widehat{y}_i$, mentre è "tollerante" quando le distanze sono piccole (vicine a 0).

Esempio di come potrebbe variare la distanza quadratica C al variare di un generico peso w:



- Il nostro criterio è la **minimizzazione** di C.
- Q: come trovare un valore di w che minimizzi C?
- Se $C(w)$ fosse una funzione reale, continua e derivabile della variabile reale w, avente forma nota e chiusa, il minimo si troverebbe tra le possibili soluzioni dell'equazione $\frac{\mathrm{d}C(w)}{\mathrm{d}w} = 0$.
- Per gli MLP (a meno di non predefinire un'architettura specifica) non esiste tale forma nota e chiusa per $C(w)$.

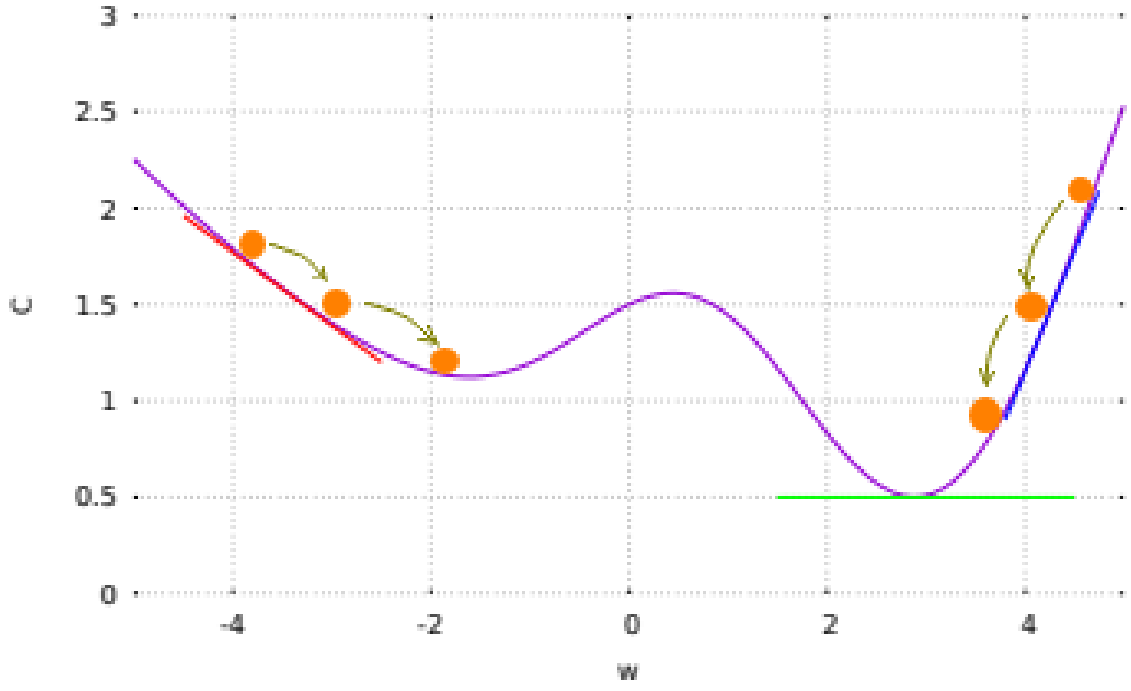
Ricorda però l'interpretazione geometrica della derivata: la derivata di $C(w)$ in un punto w_0 è la "pendenza" (*coefficiente angolare*) della retta tangente a $C(w)$ nel punto w_0.



Pendenza negativa, $C'(w_0) < 0$, pendenza nulla, $C'(w_0) = 0$, e pendenza positiva, $C'(w_0) > 0$.

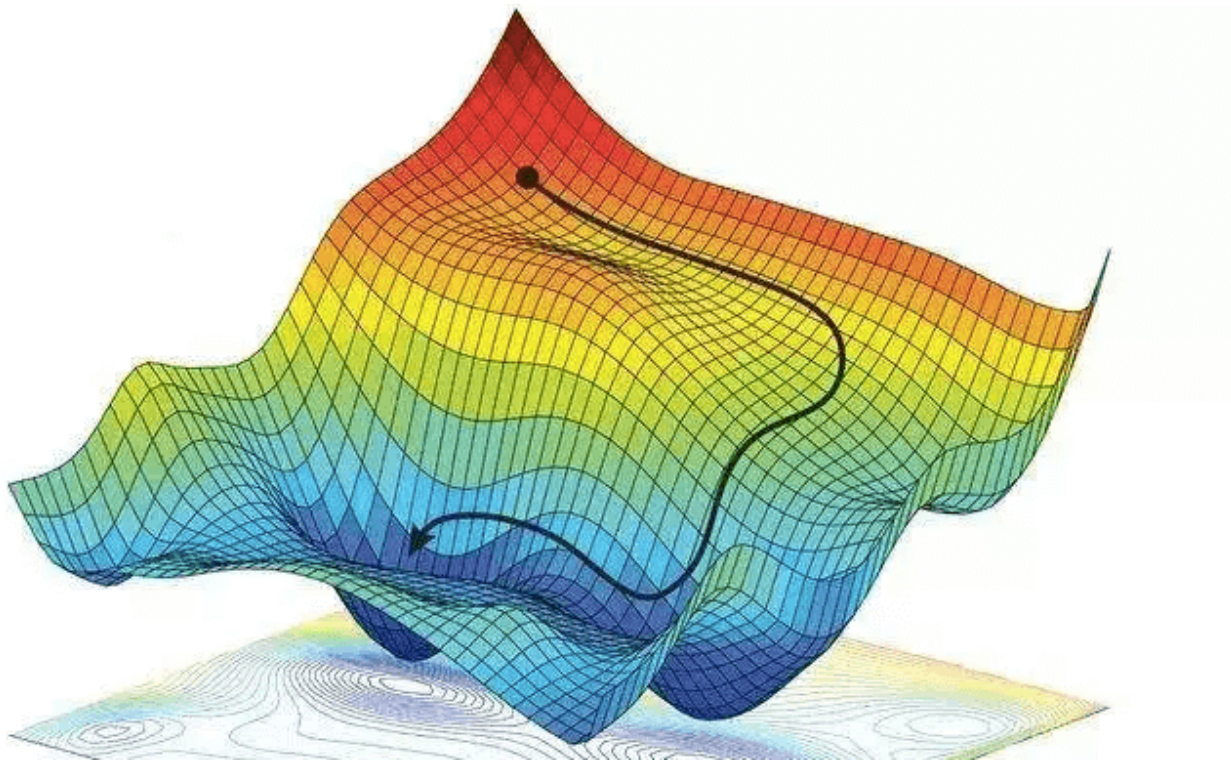
- **Metodo "della derivata"** per cercare il minimo: se w_0 è in "zona rossa", fai un passo verso destra; se è in "zona blu", fai un passo a sinistra; se è in quella verde, non muoverti.
- I movimenti avvengono **in verso opposto al segno della derivata**.

Partendo da un valore iniziale casuale w_0 die pesi, l'apprendimento procede discendendo a piccoli passi Δw lungo la china della superficie di C:



- Formalmente: $\Delta w = -\eta \frac{\mathrm{d}C(w)}{\mathrm{d}w}$
- La lunghezza dei passi è controllata dal learning rate η
- Il metodo garantisce un progressivo miglioramento del criterio C
- Si può rimanere **intrappolati in un minimo locale** di C
- Diverse inizializzazioni casuali w_0 dei pesi portano ad apprendere differenti valori finali (più o meno buoni) dei pesi.

In generale lo spazio dei pesi dello MLP è multi-dimensionale (la dimensionalità è uguale al numero di connessioni nella rete: $|\mathcal{E}|$) e il metodo "della derivata" prende il nome di **metodo del gradiente** (discesa del gradiente o *steepest descent*):



Se lo MLP ha p pesi $W = (w_1, \ldots, w_p)$, cioé se $|\mathcal{E}| = p$, allora il *gradiente* di C rispetto ai pesi W è il vettore $\nabla_W C = \left(\frac{\partial C}{\partial w_1}, \ldots, \frac{\partial C}{\partial w_p}\right)$ delle derivate ("parziali") di C rispetto a ciascuno dei pesi.

- Per un generico peso w dello MLP, $w \in W$, la **regola di apprendimento** rimane come prima: $\Delta w = -\eta \frac{\partial C}{\partial w}$
- Come abbiamo visto, tale derivata **non ha forma nota e chiusa** indipendente dall'architettura del MLP
- Per rendere il metodo del gradiente un vero e proprio algoritmo, implementabile in software, nel presente contesto serve dunque **una procedura che consenta la computazione** di $\frac{\partial C}{\partial w}$ per qualsivoglia peso w dello MLP.
- È proprio ciò che fa l'algoritmo di **Backpropagation (BP)**

BP: notazione

- Scrivo $f_i(a_i)$ per denotare la funzione di attivazione del neurone i-imo
- Assumo che lo MLP abbia ℓ strati: L_0 (*layer di input*), $L_1, ..., L_{\ell-1}$ (*layer nascosti*) e L_ℓ (*layer di output*). Scrivo $k \in L_j$ per riferirmi al k-imo neurone nel j-imo strato.

BP, caso 1: w è nello strato di output

Sia $w = w_{ij}$, dove $j \in L_{\ell-1}$ e $i \in L_\ell$. Si ha:

$$\begin{aligned}
\frac{\partial C}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \left\{ \frac{1}{2} \sum_{k=1}^{m} (\widehat{y}_k - y_k)^2 \right\} \\
&= \frac{1}{2} \sum_{k=1}^{m} \frac{\partial}{\partial w_{ij}} (\widehat{y}_k - y_k)^2 \\
&= \frac{1}{2} \frac{\partial}{\partial w_{ij}} (\widehat{y}_i - y_i)^2 \\
&= -(\widehat{y}_i - y_i) \frac{\partial y_i}{\partial w_{ij}}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial y_i}{\partial w_{ij}} &= \frac{\partial f_i(a_i)}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} \\
&= f_i'(a_i) \frac{\partial}{\partial w_{ij}} \sum_k w_{ik} o_k \\
&= f_i'(a_i) o_j
\end{aligned}$$

Riassumendo, finora abbiamo:

- $\frac{\partial C}{\partial w_{ij}} = -(\widehat{y}_i - y_i)\frac{\partial y_i}{\partial w_{ij}}$
- $\frac{\partial y_i}{\partial w_{ij}} = f_i'(a_i)o_j$

da cui:

$$\Delta w_{ij} = \eta(\widehat{y}_i - y_i)f_i'(a_i)o_j$$

Quindi, se $i \in L_\ell$ e $j \in L_{\ell-1}$, definendo

$$\delta_i = (\widehat{y}_i - y_i)f_i'(a_i)$$

si ottiene la seguente **regola delta** (*delta rule*):

$$\Delta w_{ij} = \eta\delta_i o_j \tag{1}$$

dove $o_j = f_j(a_j)$.

BP, caso 2: w è in uno strato nascosto

Per fissare le idee, sia $w = w_{jk}$ dove $j \in L_{\ell-1}$ e $k \in L_{\ell-2}$ (nondimeno, i risultati che otterremo varranno anche per tutti gli altri layer nascosti $L_{\ell-3}, ..., L_0$). Inoltre, sia $\Delta w_{jk} = -\eta \frac{\partial C}{\partial w_{jk}}$. Si ha:

$$
\begin{aligned}
\frac{\partial C}{\partial w_{jk}} &= \frac{\partial}{\partial w_{jk}} \left\{ \frac{1}{2} \sum_{i=1}^{m} (\widehat{y}_i - y_i)^2 \right\} \\
&= \frac{\partial}{\partial o_j} \left\{ \frac{1}{2} \sum_{i=1}^{m} (\widehat{y}_i - y_i)^2 \right\} \frac{\partial o_j}{\partial w_{jk}} \\
&= \left\{ \frac{1}{2} \sum_{i=1}^{m} \frac{\partial}{\partial o_j} (\widehat{y}_i - y_i)^2 \right\} \frac{\partial o_j}{\partial w_{jk}} \\
&= \left\{ - \sum_{i=1}^{m} (\widehat{y}_i - y_i) \frac{\partial y_i}{\partial o_j} \right\} \frac{\partial o_j}{\partial w_{jk}}
\end{aligned}
$$

Calcoliamo $\frac{\partial y_i}{\partial o_j}$.

$$\begin{aligned}
\frac{\partial y_i}{\partial o_j} &= \frac{\partial f_i(a_i)}{\partial a_i}\frac{\partial a_i}{\partial o_j} \\
&= f_i^{'}(a_i)\frac{\partial}{\partial o_j}\sum_j w_{ij}o_j \\
&= f_i^{'}(a_i)w_{ij}
\end{aligned}$$

da cui, poiché $\frac{\partial C}{\partial w_{jk}} = \left\{-\sum_{i=1}^{m}(\widehat{y}_i - y_i)\frac{\partial y_i}{\partial o_j}\right\}\frac{\partial o_j}{\partial w_{jk}}$, si ottiene:

$$\begin{aligned}
\frac{\partial C}{\partial w_{jk}} &= \left\{-\sum_{i=1}^{m}(\widehat{y}_i - y_i)f_i^{'}(a_i)w_{ij}\right\}\frac{\partial o_j}{\partial w_{jk}} \\
&= -\left(\sum_{i=1}^{m} w_{ij}\delta_i\right)\frac{\partial f_j(a_j)}{\partial a_j}\frac{\partial a_j}{\partial w_{jk}} \\
&= -\left(\sum_{i=1}^{m} w_{ij}\delta_i\right) f_j^{'}(a_j)o_k
\end{aligned}$$

Finora, $\Delta w_{jk} = -\eta \frac{\partial C}{\partial w_{jk}}$ e $\frac{\partial C}{\partial w_{jk}} = -\left(\sum_{i=1}^{m} w_{ij}\delta_i\right) f_j^{'}(a_j) o_k$.
Possiamo quindi scrivere:

$$\Delta w_{jk} = \eta \left(\sum_{i=1}^{m} w_{ij}\delta_i\right) f_j^{'}(a_j) o_k \tag{2}$$

Definendo

$$\delta_j = \left(\sum_{i=1}^{m} w_{ij}\delta_i\right) f_j^{'}(a_j) \tag{3}$$

otteniamo la seguente *regola delta*:

$$\Delta w_{jk} = \eta \delta_j o_k \tag{4}$$

che è nella stessa forma ottenuta nel caso 1.

Generalized delta rule

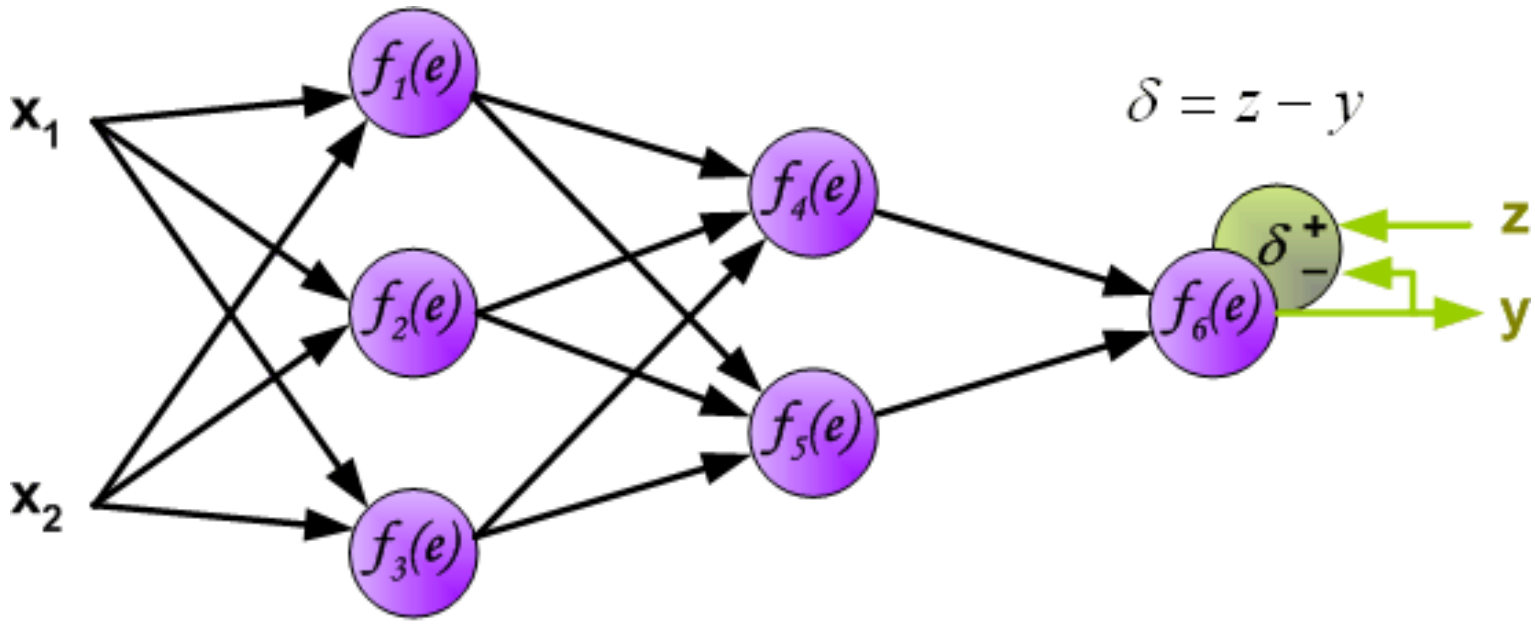
Riassumendo, dato un qualsiasi peso w_{jk} in qualsiasi strato, può essere applicata la **regola delta generalizzata** (*generalized delta-rule*):

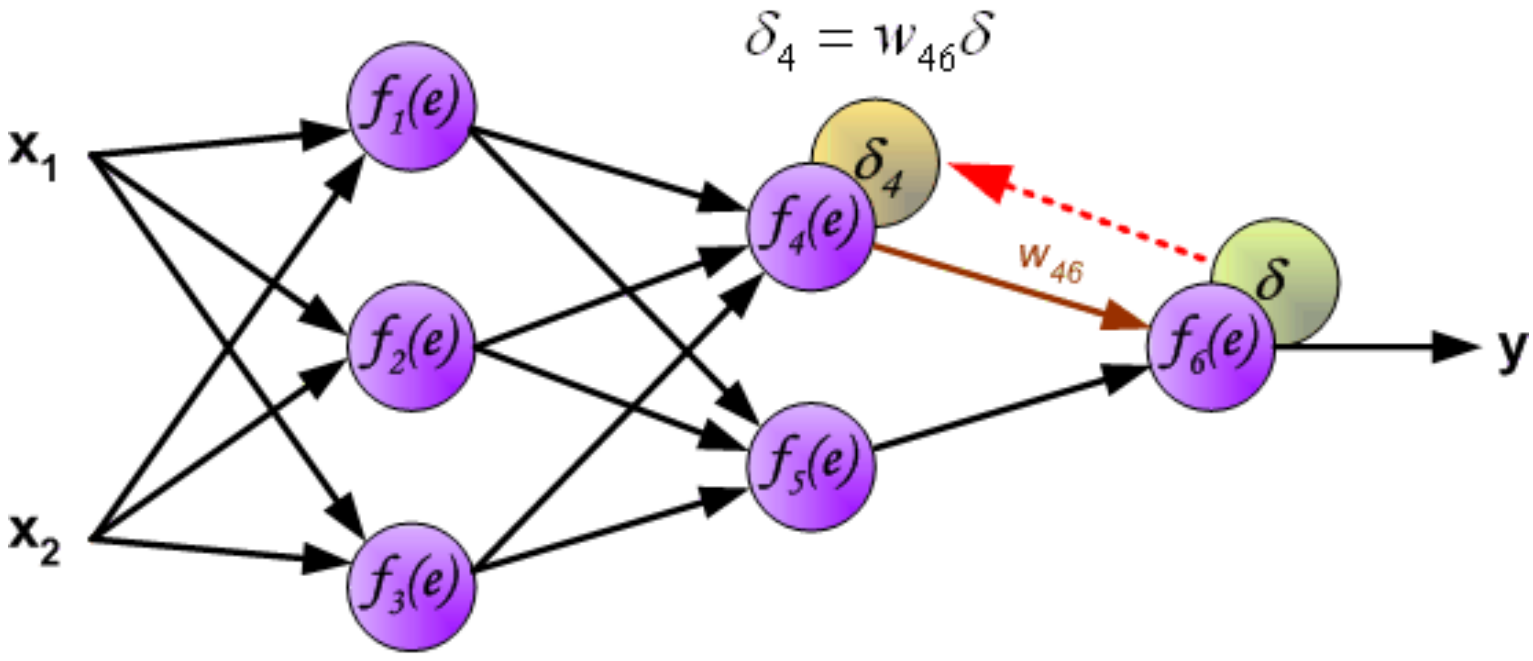
$$\Delta w_{jk} = \eta \delta_j o_k \tag{5}$$

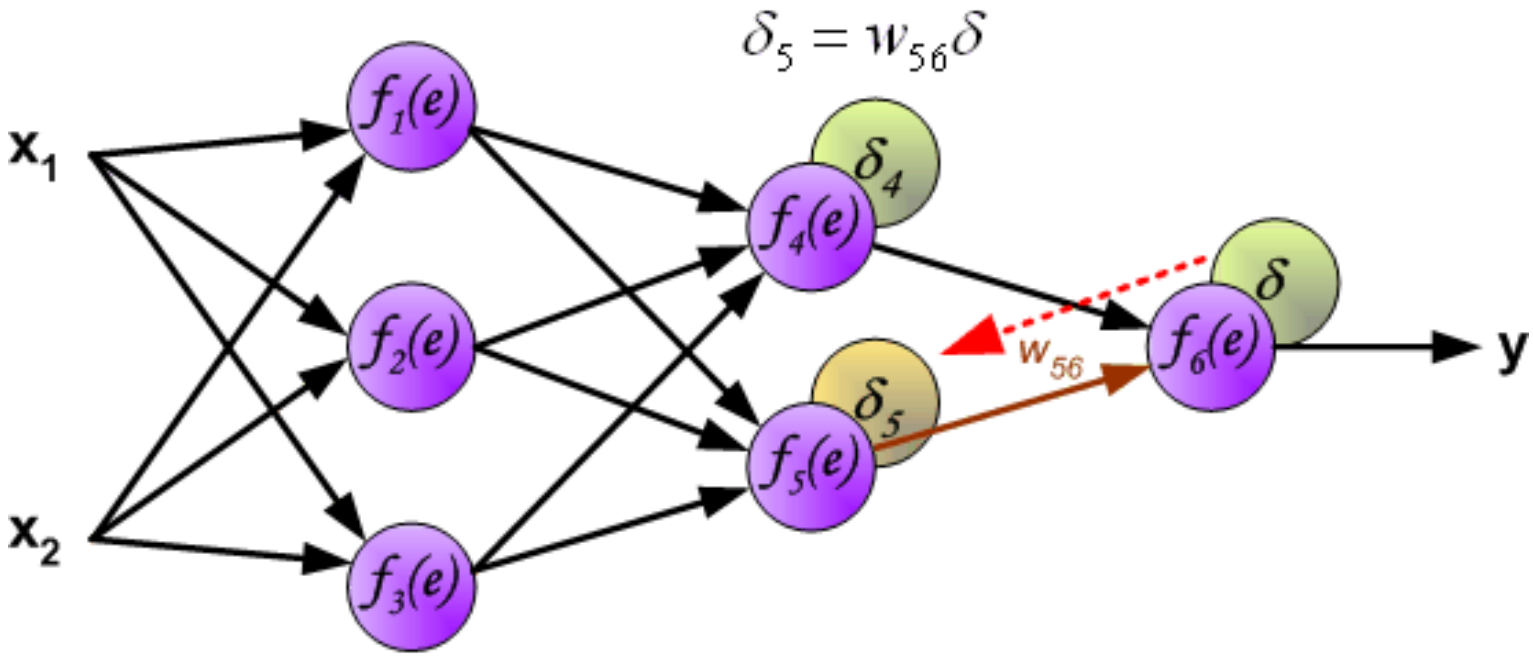
dove $\delta_j o_k$ è definita come segue:

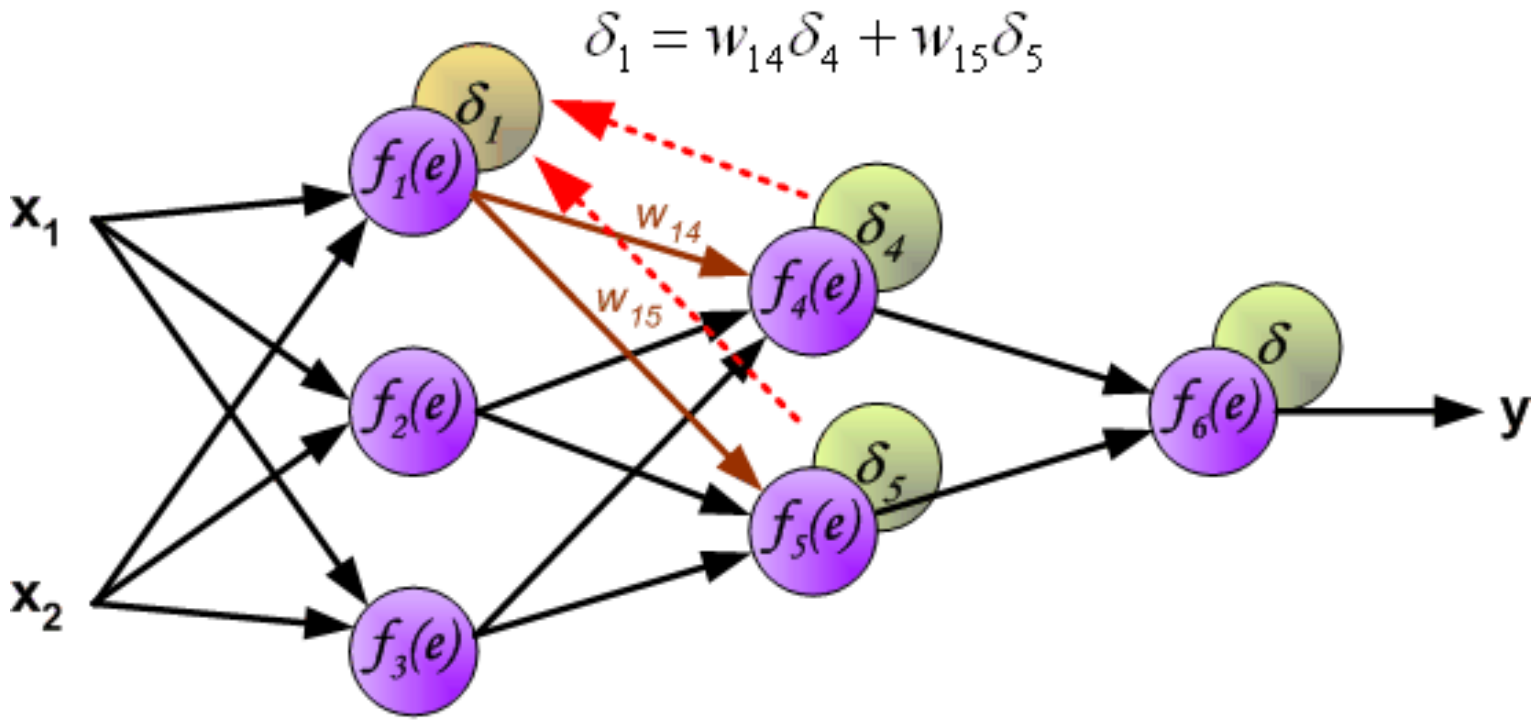
$$\delta_j = \begin{cases} (\widehat{y}_j - y_j) f_j'(a_j) & \text{se } j \in L_\ell \\ (\sum_{i \in L_{k+1}} w_{ij} \delta_i) f_j'(a_j) & \text{se } j \in L_k \text{ dove } k = \ell - 1, ..., 0 \end{cases}$$

Il termine $\widehat{y}_j f_j'(a_j) o_k$ che si ottiene facendo i prodotti nel caso in cui $j \in L_\ell$ è detto di *apprendimento hebbiano*.

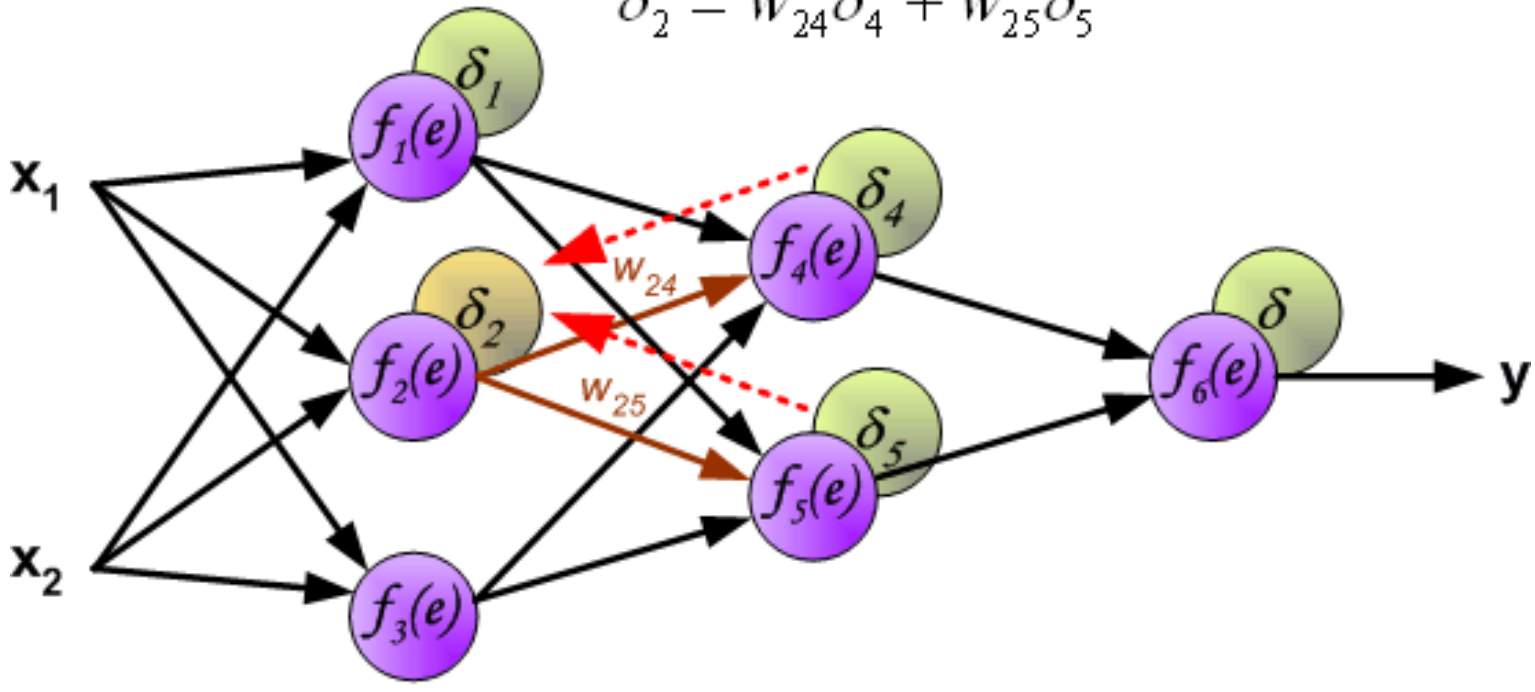

x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
$\delta = z - y$
δ + −
z
y


$\delta_4 = w_{46}\delta$
x1
x2
f1(e)
f2(e)
f3(e)
f4(e)
f5(e)
f6(e)
δ4
δ
w46
y

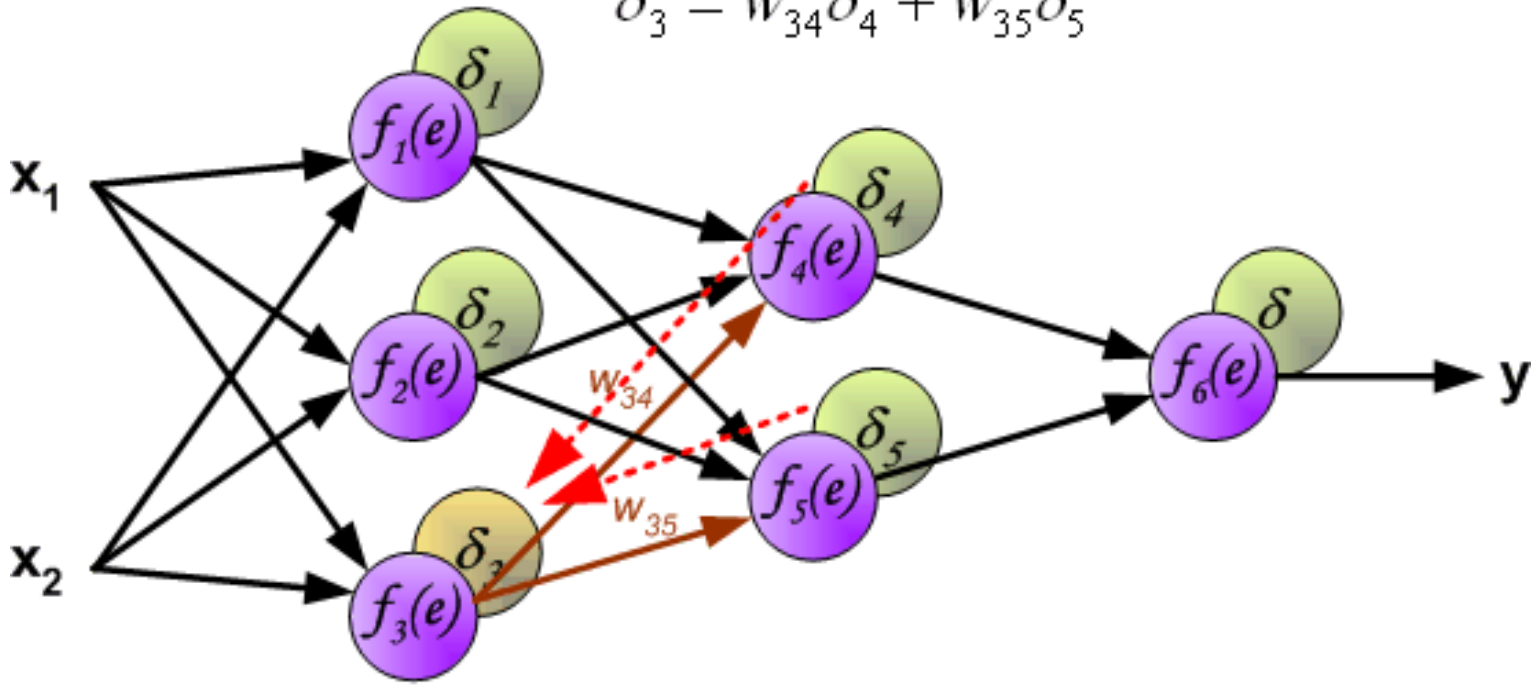

$\delta_5 = w_{56}\delta$
x1
x2
f1(e)
f2(e)
f3(e)
f4(e)
f5(e)
f6(e)
δ4
δ5
δ
w56
y


$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5$
δ_1
$f_1(e)$
w_{14}
w_{15}
δ_4
$f_4(e)$
x_1
$f_2(e)$
δ_5
$f_5(e)$
δ
$f_6(e)$
y
x_2
$f_3(e)$

$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$

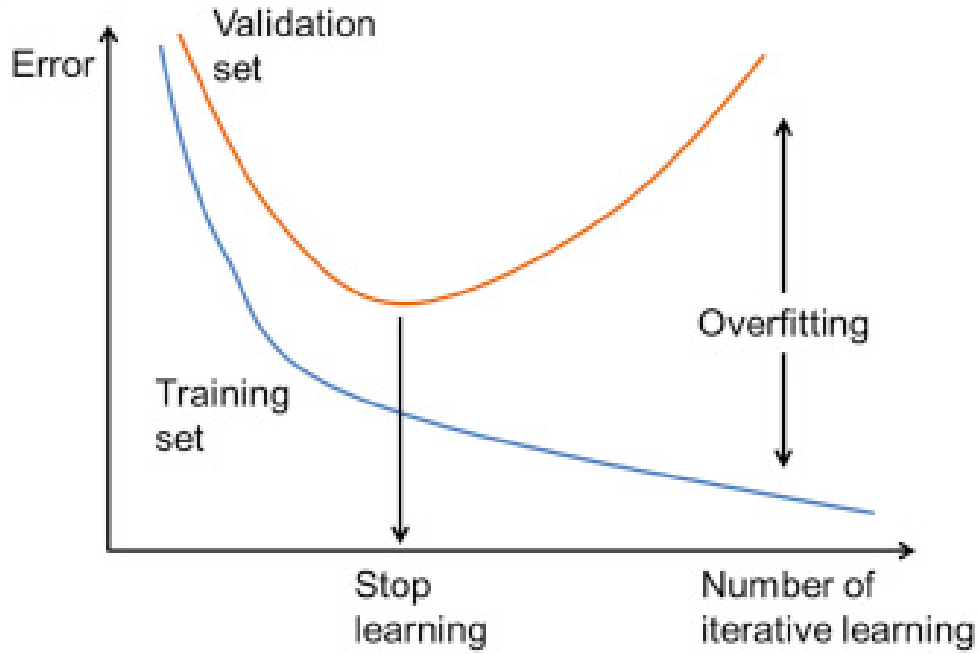


$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$



BP: criteri di arresto

1. dopo un numero prefissato di epoche
2. una volta che C sia sceso al di sotto di una data soglia: $C < \theta$
3. una volta che il ΔC tra epoche consecutive sia sceso sotto una certa soglia: $\Delta C < \theta$
4. valutando empiricamente la capacità di generalizzazione:

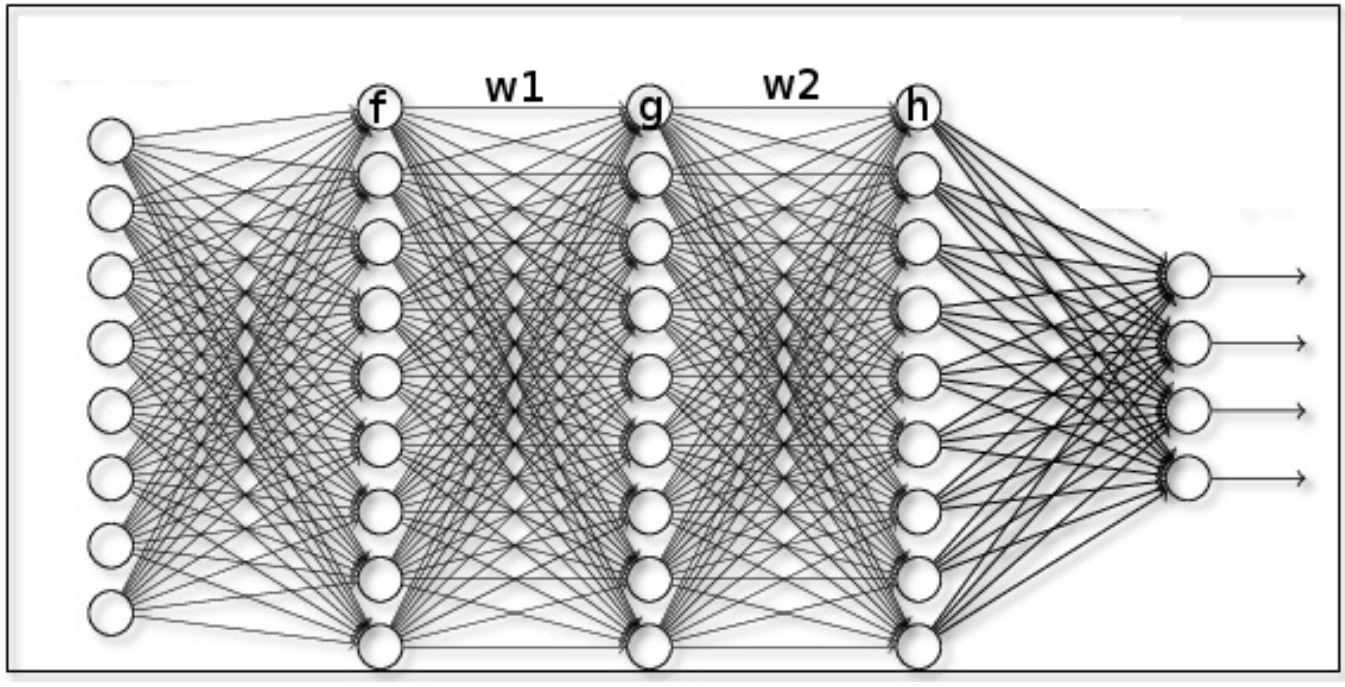


Curva di apprendimento: distanza quadratica C misurata sul training set.

Curva di generalizzazione: distanza quadratica C misurata su un *validation set* disgiunto e indipendente dal training set.

Deep learning

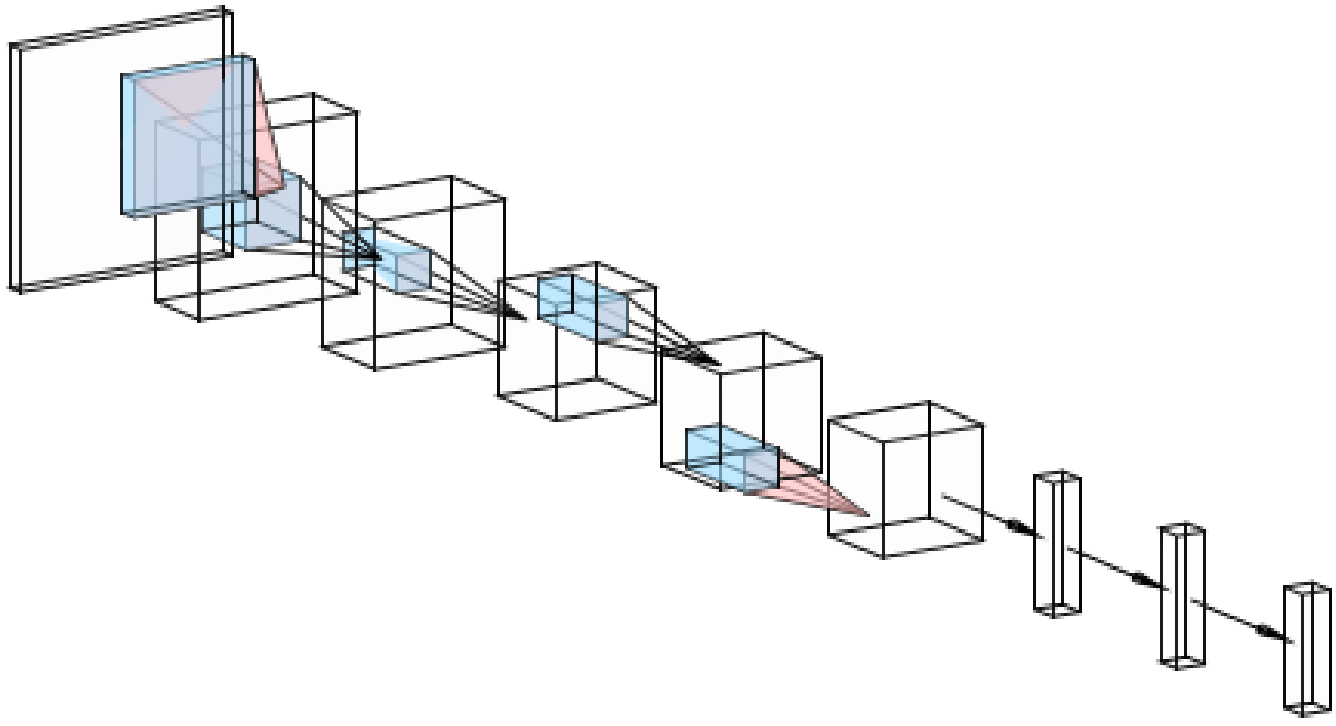
Una **Deep neural network** (DNN) è una ANN la cui architettura sia "profonda" (con molti strati):



• L'avvento del deep learning è stato reso possibile dai vertiginosi progressi nella **velocità di calcolo** dei microprocessori (CPU) e delle **graphics processing unit (GPU)**

• Il deep learning ha portato allo sviluppo di numerosi *trucchi del mestiere*, cioé migliorie algoritmiche volte a rendere l'apprendimento delle DNN più rapido, più stabile (numericamente) e più efficace.

Le *reti neurali convolutive* (**convolutional neural network, CNN**) sono le DNN più popolari allo stato dell'arte:

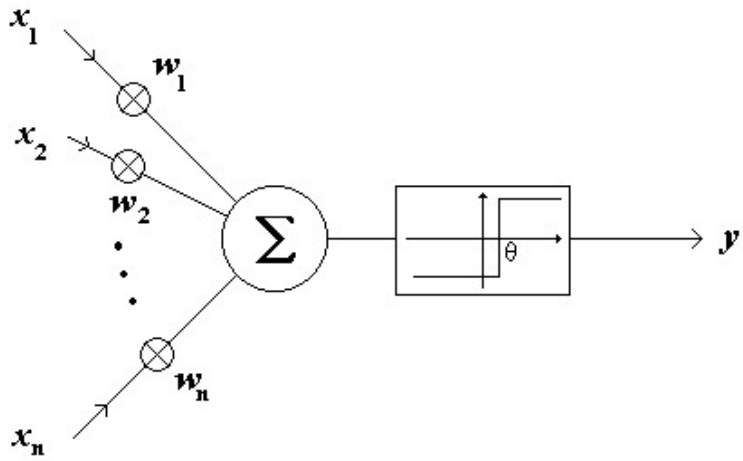


Sono reti feed-forward non fully-connected con speciali funzioni di attivazione che realizzano "convoluzioni" su porzioni dell'input. Gli algoritmi di apprendimmento sono ad hoc.

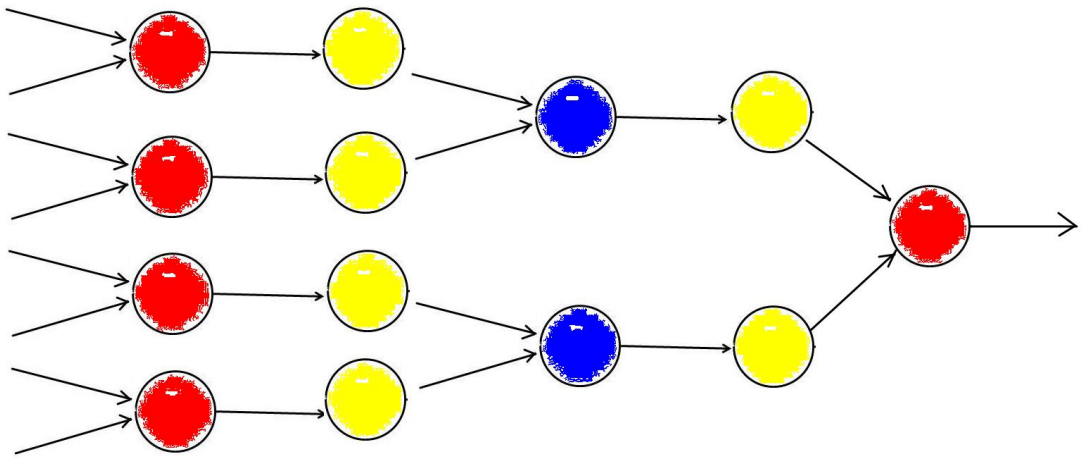
• Sono usate per **visione artificiale** e apprendimento a partire da **immagini** (es. riconoscimento di caratteri, di oggetti, di volti, ...). L'input è in genere una *bitmap*.

Meta-reti e altri mix

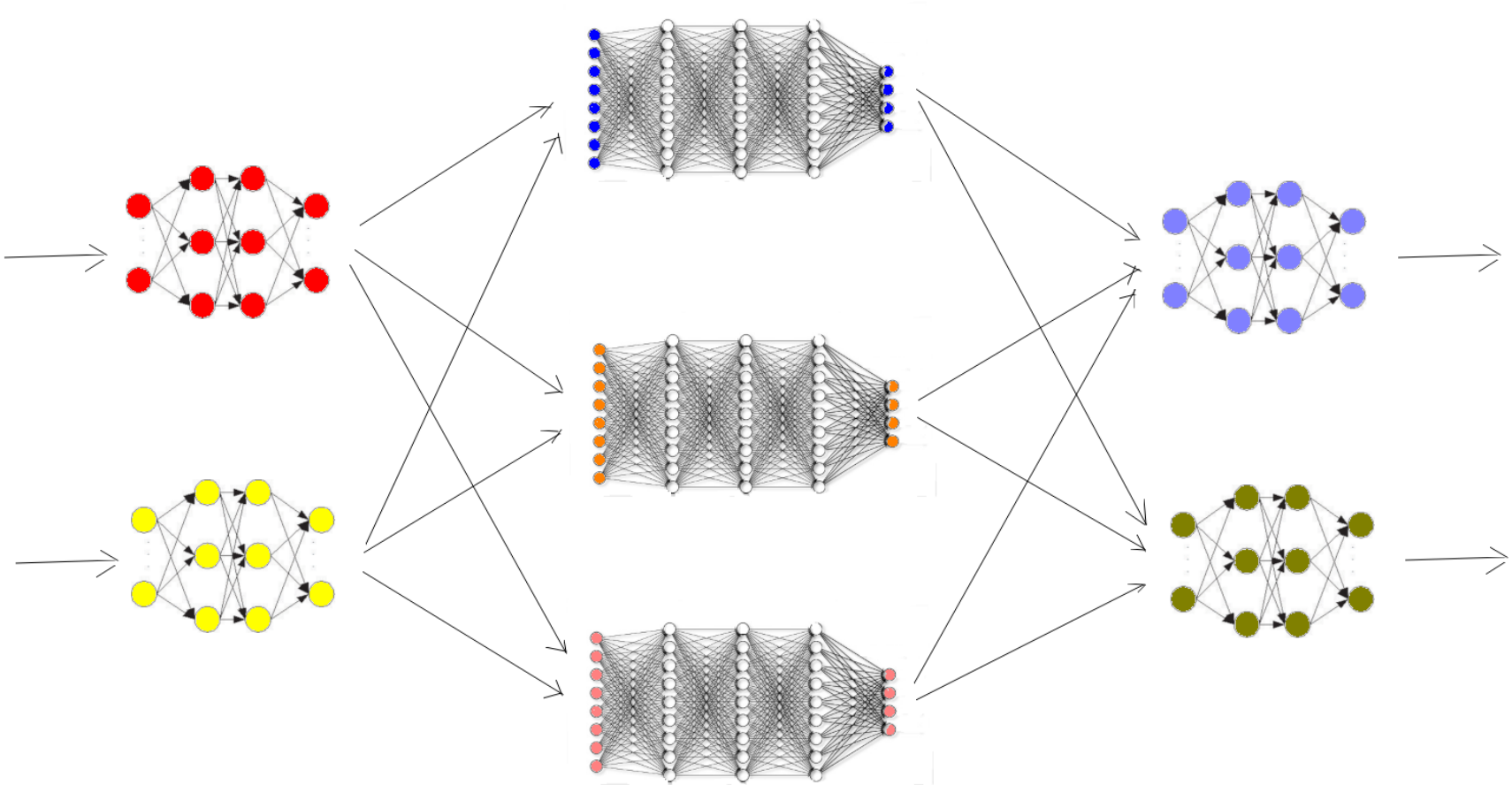
Come da singole TLU ...



... abbiamo costruito reti di TLU ...

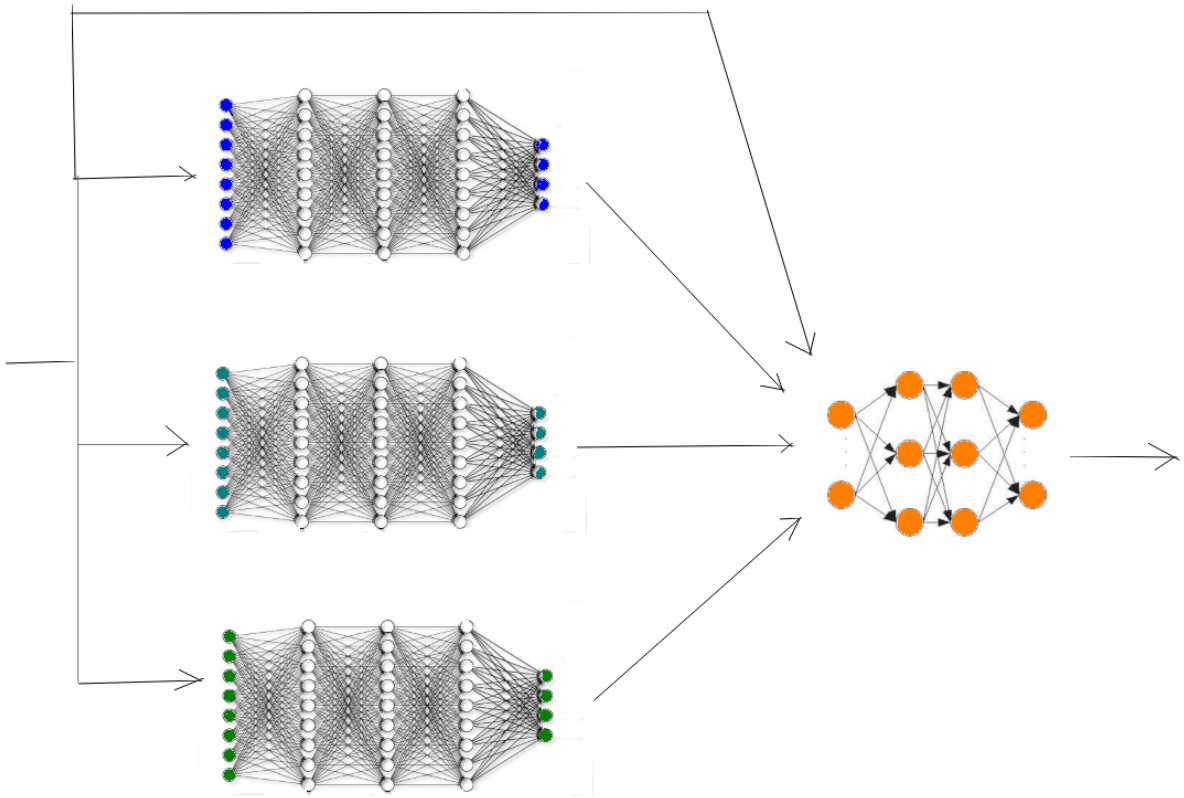


... così da singole ANN possiamo costruire (meta-)reti di ANN:



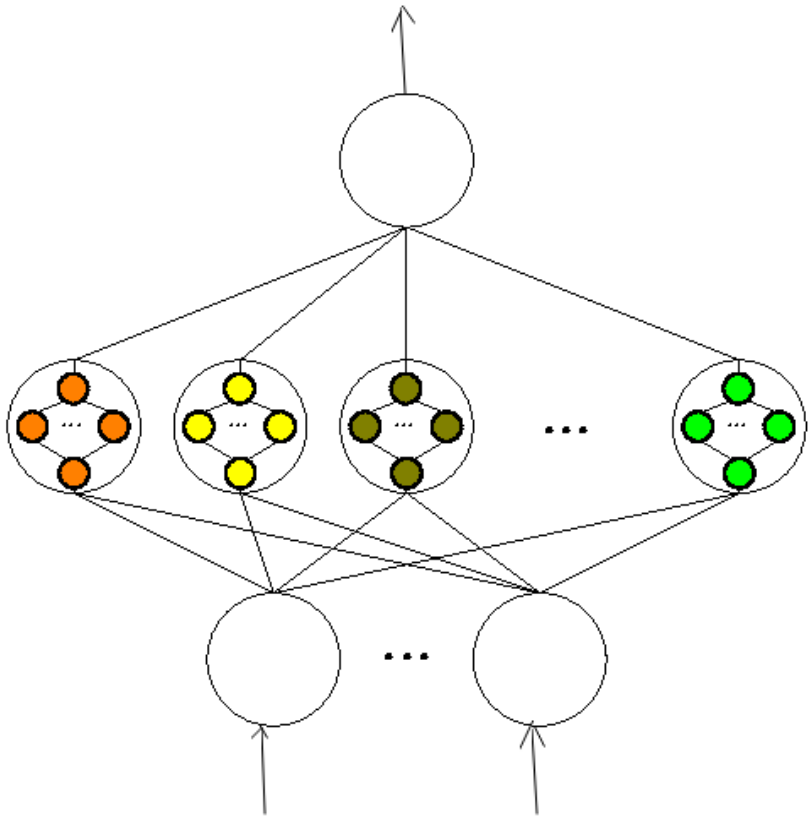
Si parla di "**Ensemble**" di ANN. Possono essere viste come DNN.

Mistura di esperti (*Mixture of experts*)



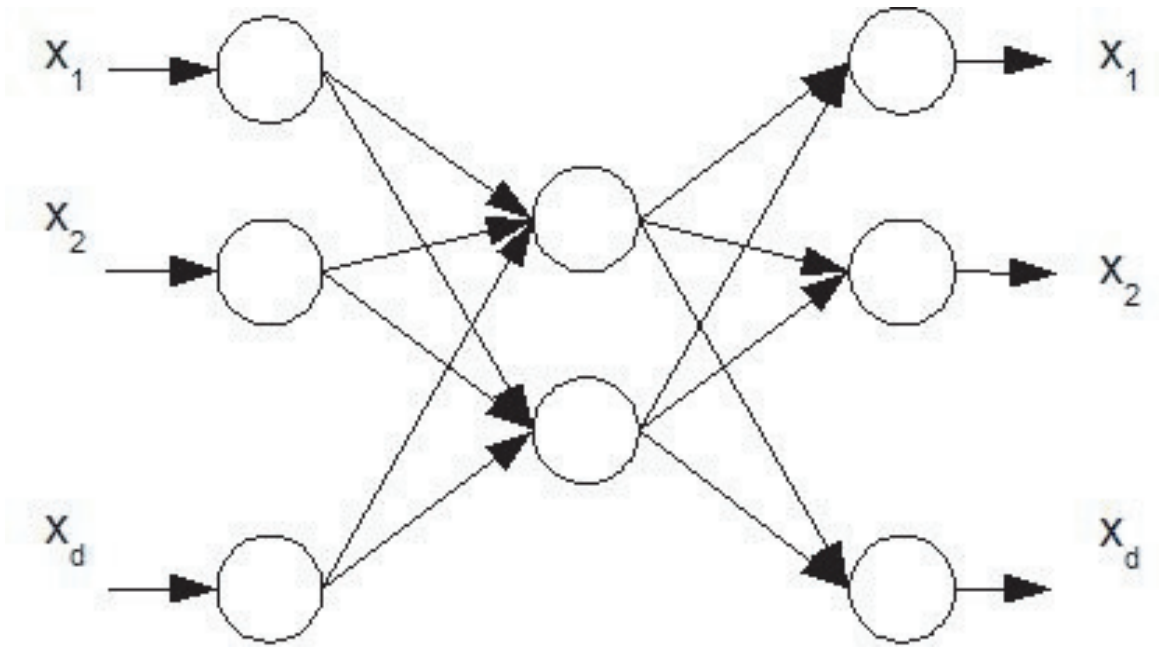
- È un caso particolare e popolarissimo di ensemble
- È un approccio *divide et impera*
- L'output della mistura è restituito da una ANN "miscelatore" (*gating network*) che riceve in input sia il vettore di feature in ingresso al sistema (come tutte le altre ANN, dette *esperto 1*, *esperto 2*, *esperto 3*, ...) sia l'ipotesi di output calcolata da ciascun esperto in base alla propria specializzazione.

Funzioni di attivazione adattative



La sotto-rete (colorata, nella figura) associata al generico neurone v della ANN "esterna" calcola la corrispondente funzione di attivazione $f_v : \mathbb{R} \to \mathbb{R}$, adattativa e specifica per v. La $f_v(\cdot)$ viene appresa durante la fase di training mediante algoritmi ad hoc.

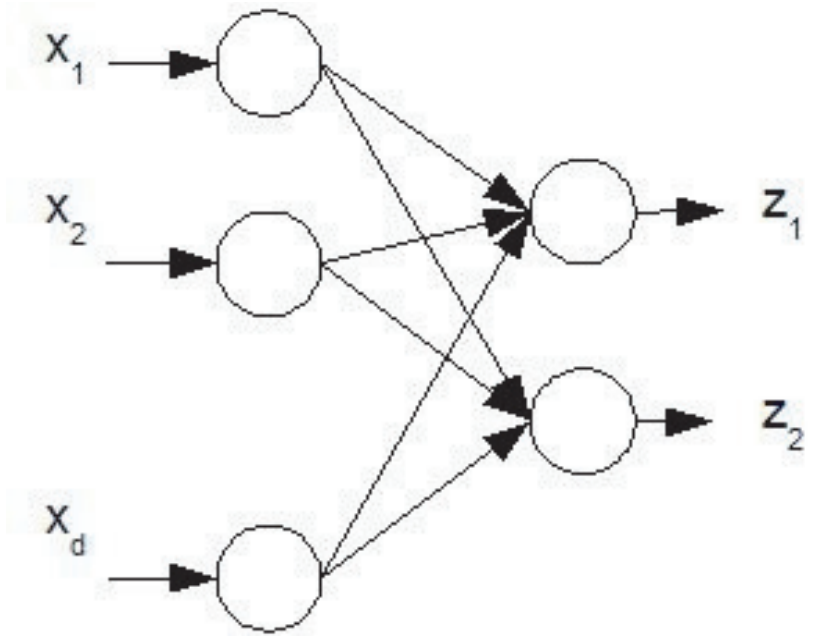
Autoassociatore (*autoencoder*)



È un MLP che tramite BP apprende una trasformazione autoassociativa: gli input sono trasformati in se stessi. Training set: $\mathscr{T} = \{(\underline{x}, \underline{x})\}$. Formalmente, siamo di fronte a un esempio di **apprendimento non supervisionato**.

Training

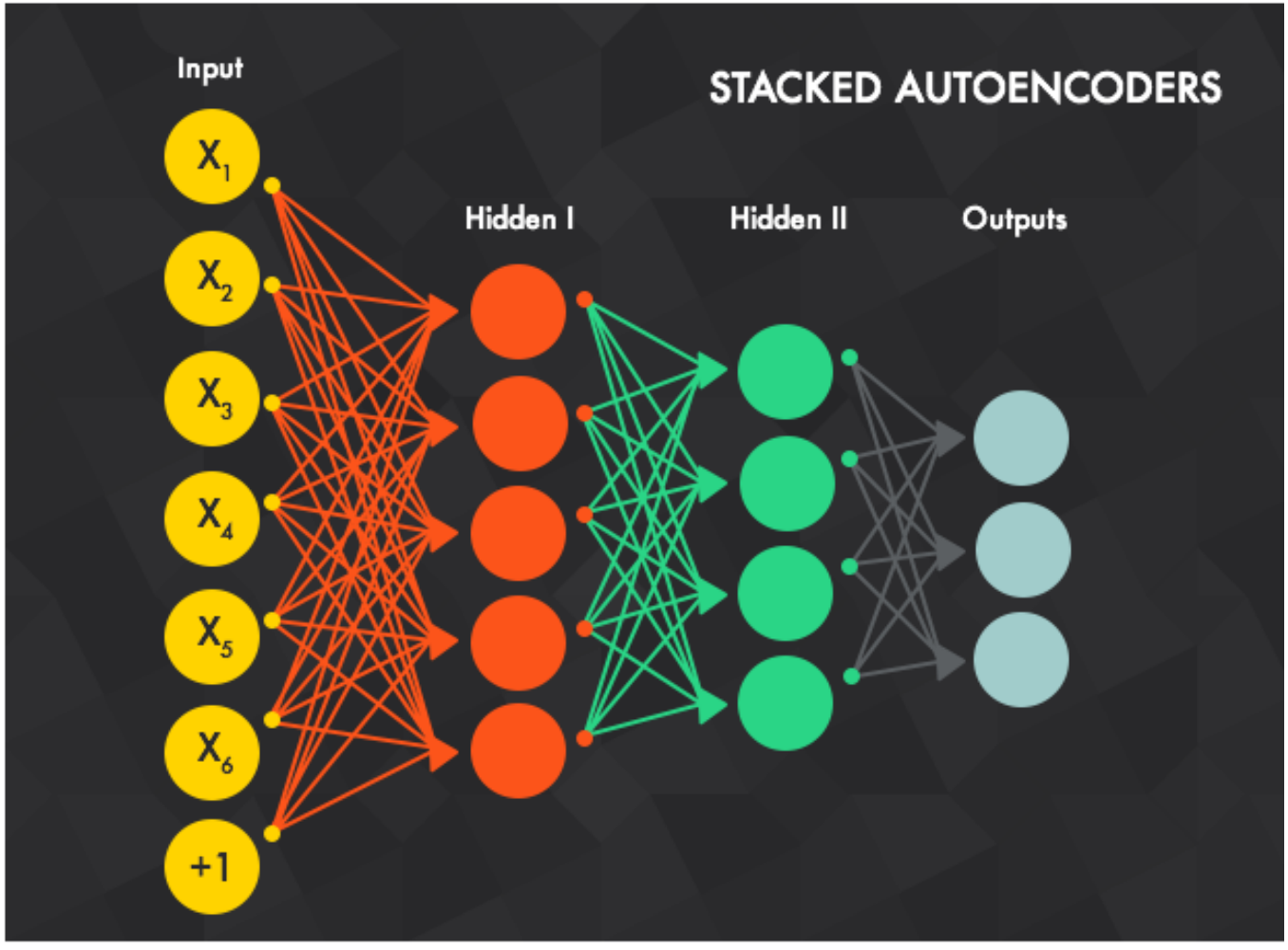
1. La ANN viene addestrata a "mappare" $\underline{x}$ su $\underline{x}$ via BP.
2. Almeno uno dei layer nascosti è un "collo di bottiglia" e apprende una **rappresentazione** a dimensionalità ridotta dei vettori delle feature.
3. Finito il training, rimuoviamo il/i layer superiore/i:



Così facendo si ottiene un "estrattore di feature" che realizza una **riduzione di dimensionalità** dell'originario spazio delle feature.

- *Nota*: è un esempio dello **scenario 3** dell'unsupervised learning.

Il procedimento si può reiterare, riducendo progressivamente e gradualmente la dimensionalità dello spazio delle feature, ottenendo una DNN di **autoassociatori impilati** l'uno sopra l'altro (*stacked autoencoders*). Si tratta di una popolare tecnica di inizializzazione delle DNN.



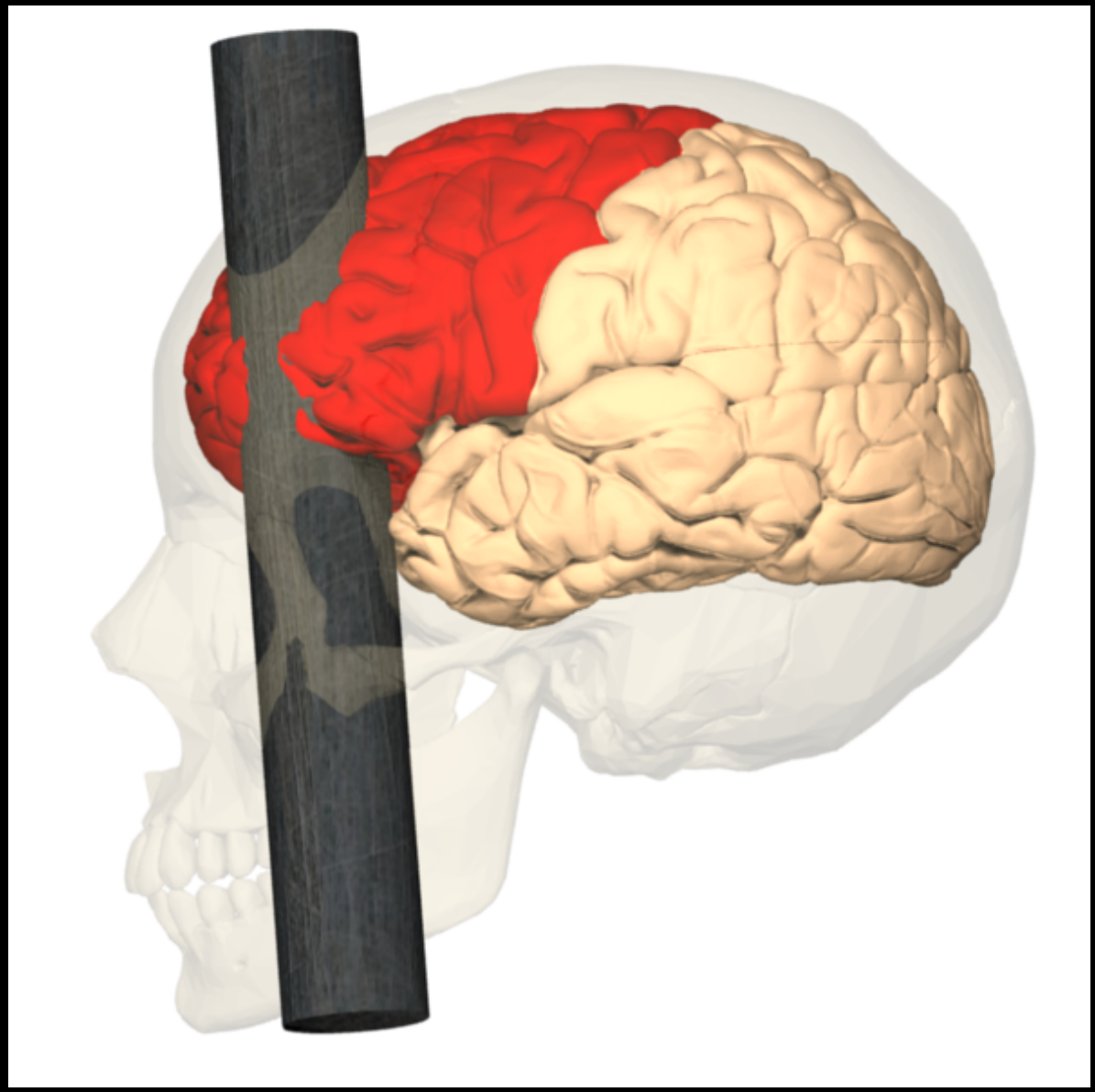
Esempio di applicazione delle ANN: *affective computing*

Facciamo un piccolo viaggio indietro nel tempo ...

Phineas P. Gage (1823-1860)

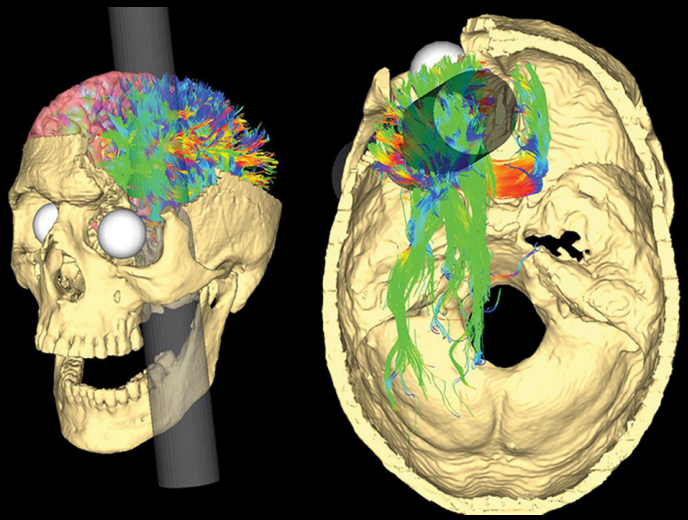
Il cranio di P. Gage presso il Warren Anatomical Museum

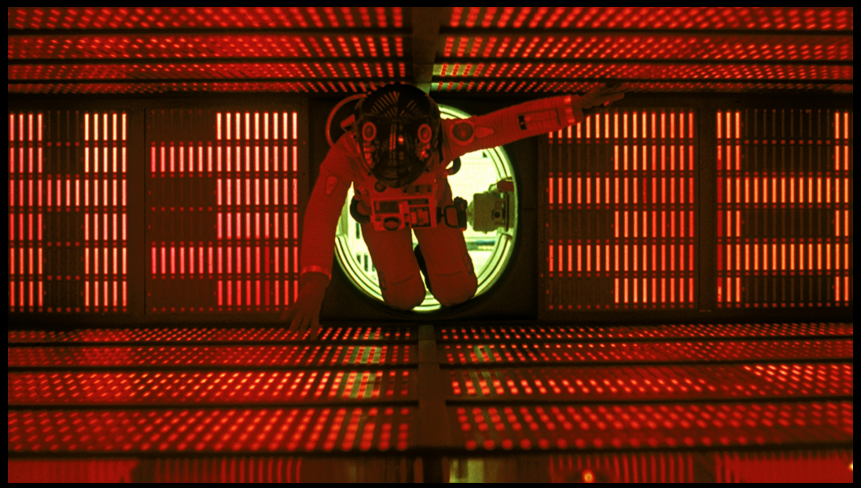
(Harvard Medical School, Boston)



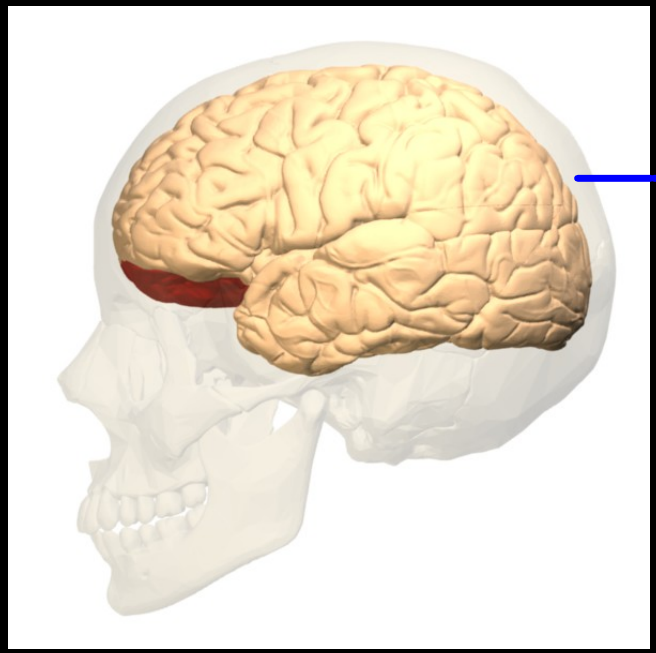
Esercizio: quali potrebbero essere gli “errori di Cartesio”?

Intelligenza, emozioni, cervelli danneggiati

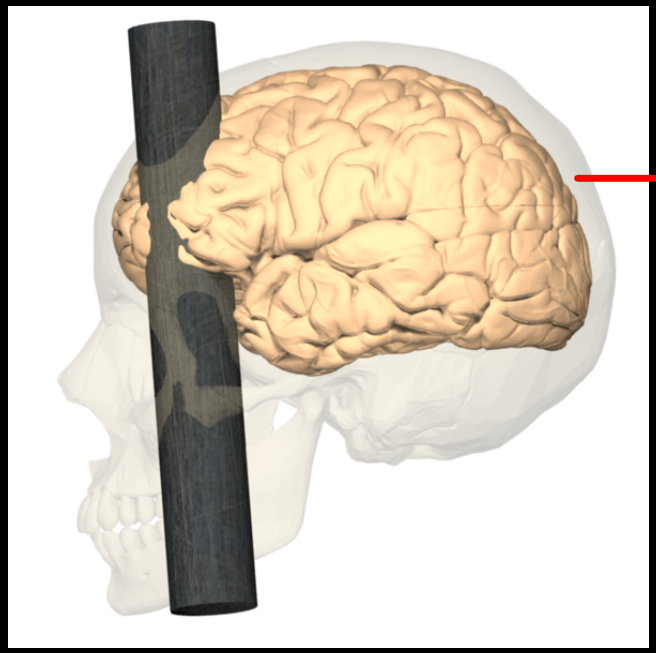


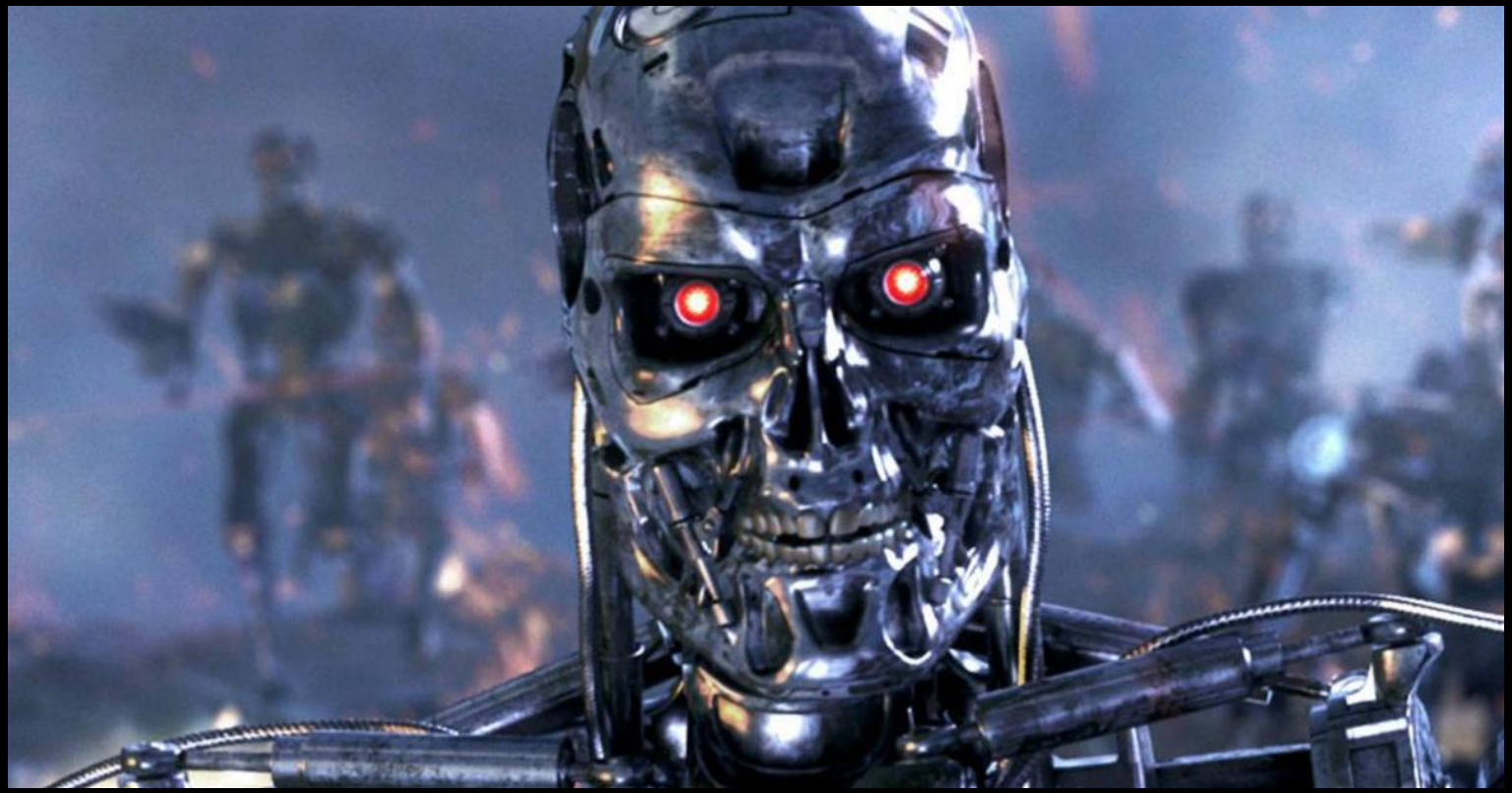


UPLOAD



UPLOAD





Rosalind Picard, “Affective Computing” (MIT Press, 1997)

AFFECTIVE COMPUTING

Elaborazione del parlato e del linguaggio naturale

AFFECTIVE COMPUTING

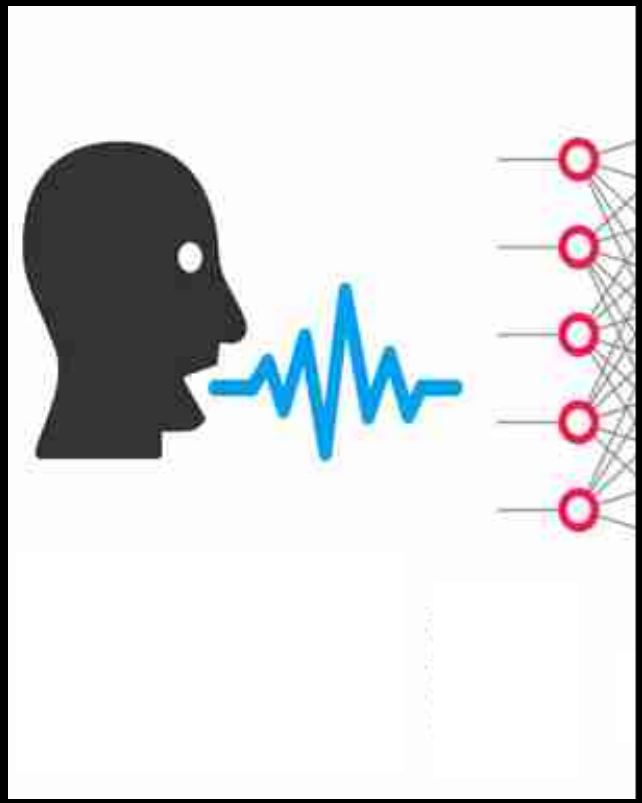
Computer vision: riconoscimento della mimica facciale

AFFECTIVE COMPUTING

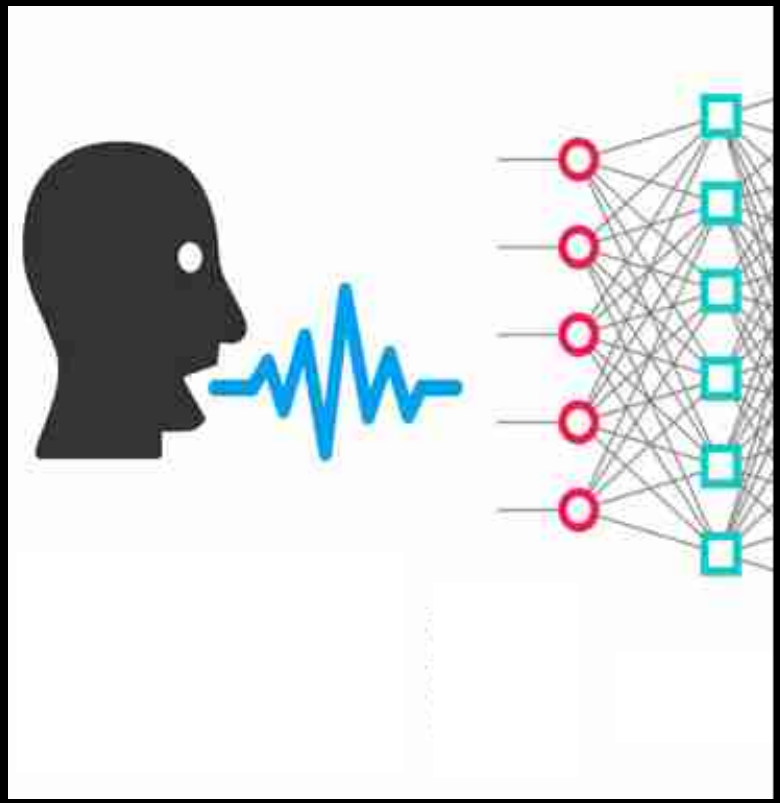
Computer vision: riconoscimento del linguaggio del corpo

Una persona parla: di che umore è?

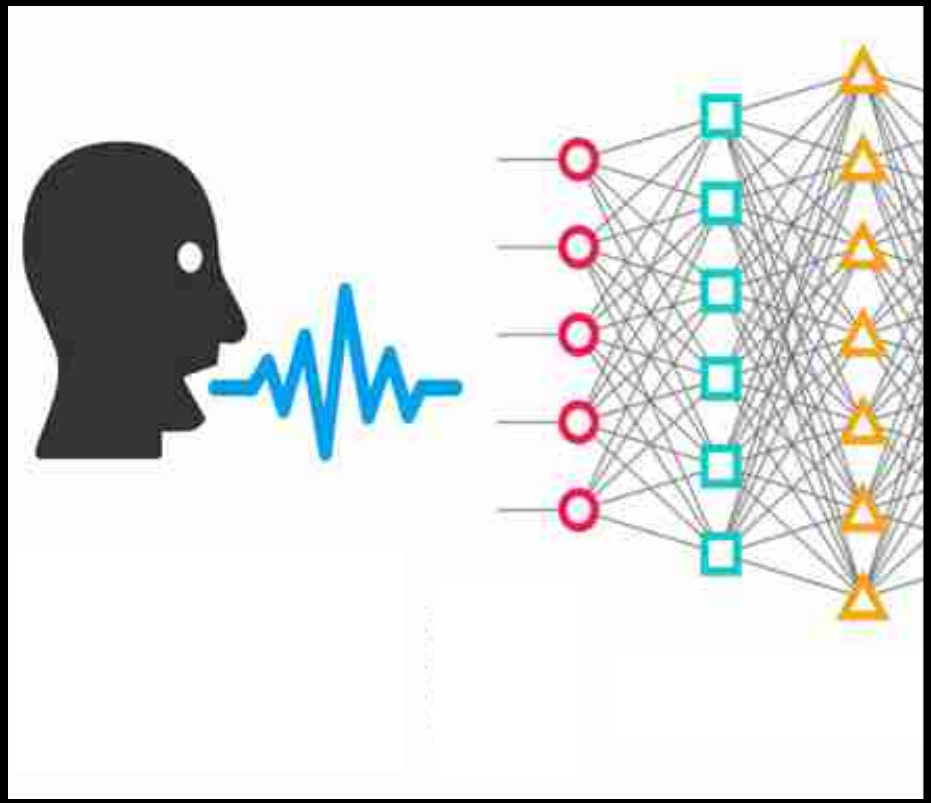
Digitalizzazione della voce



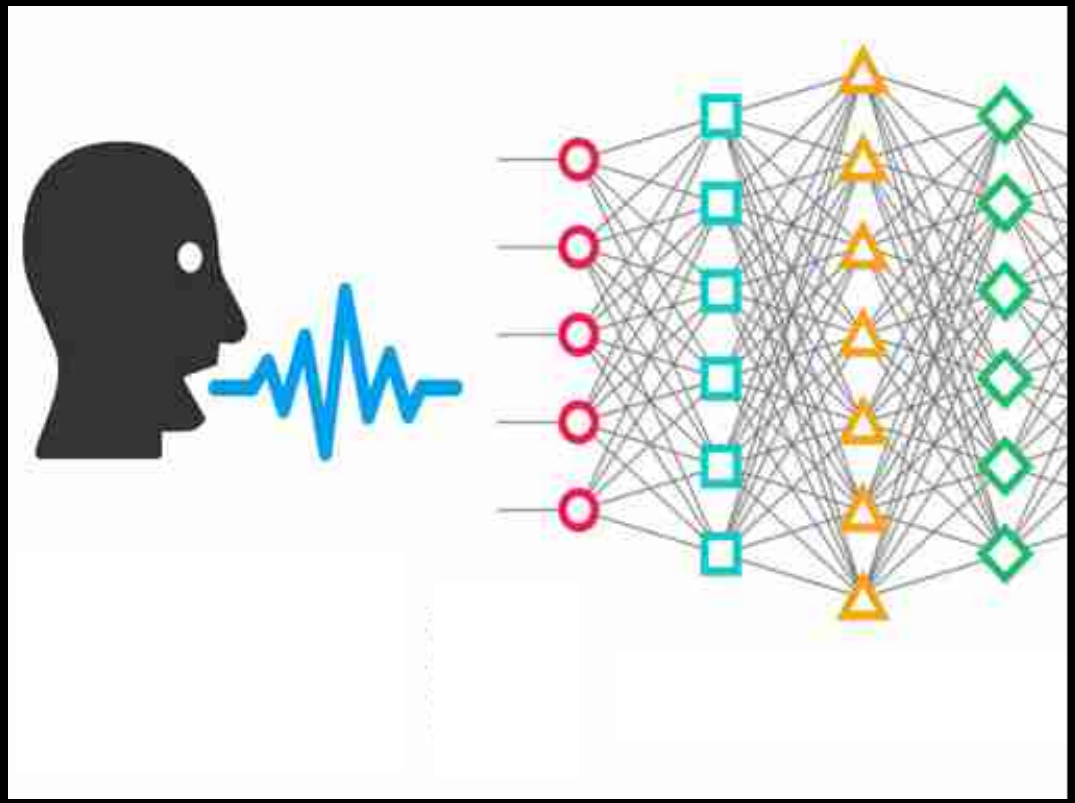
Rappresentazione neurale di basso livello



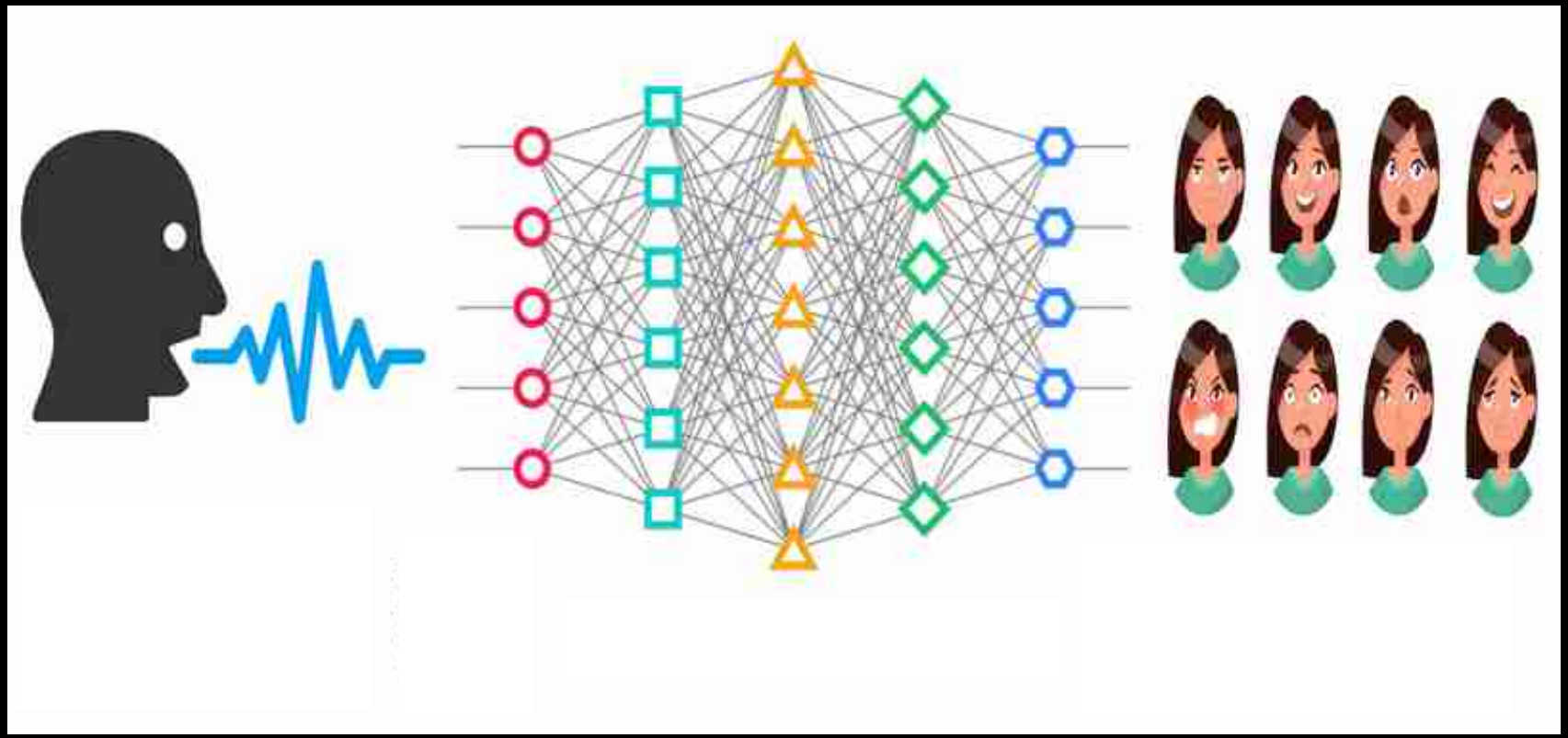
Rappresentazione neurale di alto livello



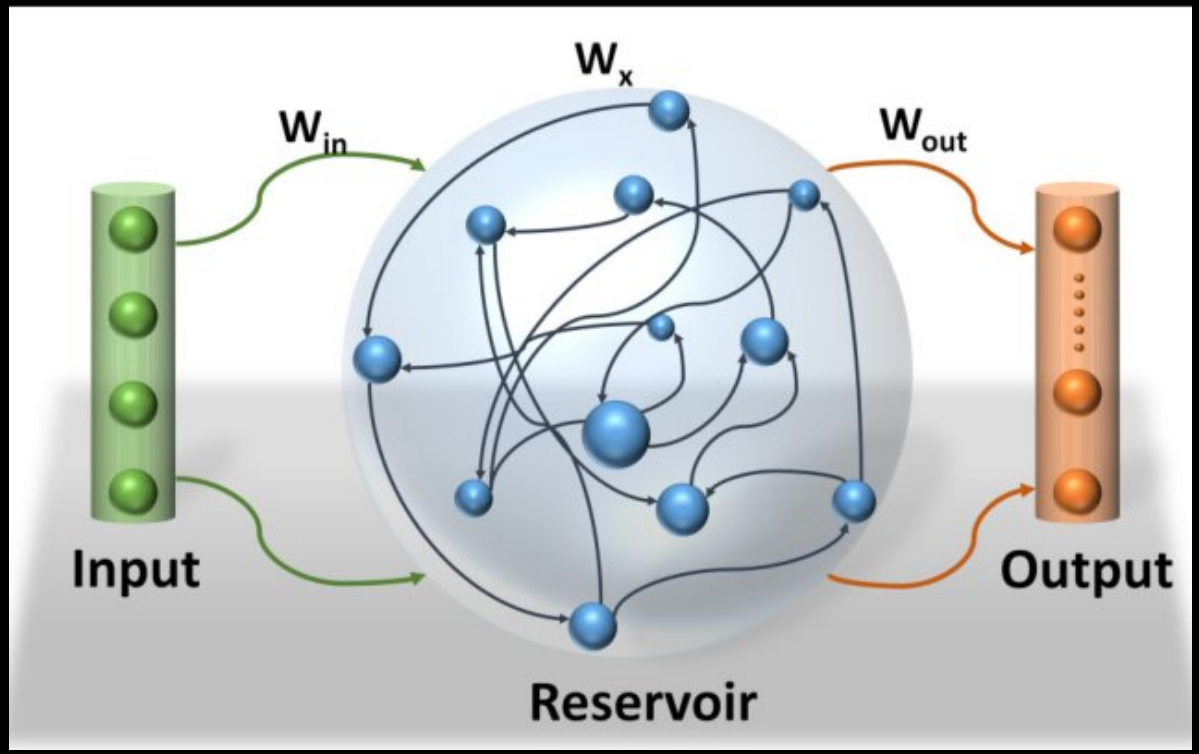
Interpretazione



Decisione: l'umore del parlatore è ...



Echo state network (ESN)



- In questo esempio, la ESN è applicata al riconoscimento di emozioni a partire dal segnale vocale.
- La ESN riceve sequenze di feature in input, che mettono in agitazione il *reservoir*. L'output è calcolato sullo stato di attivazione dei neuroni del reservoir a fine sequenza.
- Solo i pesi w_{out} vengono appresi. Gli altri sono generati a caso.

Dati usati per gli esperimenti

- Dataset: "*Corpus of spoken words for studies of auditory speech and emotional prosody processing*" (WaSeP©), Wendt e Scheich (2002).
- 222 "pseudo-parole" foneticamente bilanciate.
- Ogni parola è pronunciata più volte (per un totale di 4714 segnali acustici) da attori (maschi e femmine) in 6 diverse prosodie corrispondenti alle seguenti emozioni (*the big sux*):

  rabbia, disgusto, paura, tristezza, gioia, "neutra".

Risultati

- Un test percettivo basato su un **campione di 74 uditori umani** ha mostrato una percentuale media di classificazioni corrette delle emozioni presenti nei singoli segnali audio pari al **78.53%** dei casi (*Scherer et al., 2003*).
- L'accuratezza (percentuale di classificazioni corrette) fornita da un **MLP** è risultata[2] pari al **39.32%** (*Trentin, Scherer e Schwenker, 2015*). Il MLP non è adeguato a elaborare sequenze di vettori di feature, per applicarlo al presente *task* si è dovuto rappresentare un'intera sequenza come un unico vettore di feature a dimensionalità molto alta e prefissata.
- L'accuratezza della **ESN** (in realtà, di una sua variante probabilistica) è risultata pari al **96.69%** (*Trentin, Scherer e Schwenker, 2015*).

[2] Limitatamente ai segnali acustici pronunciati da voci femminili.

Esercizio(/regalo): a proposito di reti neurali, intelligenze artificiali e Internet, leggi lo splendido e visionario racconto *A fin di bene* di Primo Levi (scritto nel 1967?), pubblicato in *Vizio di forma* (1971) e oggi in *Tutti i racconti*, Einaudi (2015).