**IL SEMANTIC WEB**

E’ il momento di introdurre un altro sistema intelligente.

Abbiamo visto come funzionano i chatbot, che possiamo considerare modelli semplici di intelligenza simulata.

Un argomento che sta diventando sempre più importante è rappresentato dal **web semantico.** Argomento strettamente collegato sono **le ontologie** (provate a “giocare” con protegè http://protege.stanford.edu/).

Guardate prima questo video introduttivo, semplice ed interessante:
http://www.youtube.com/watch?v=rhgUDGtT2EM&feature=related

Completate la visione a casa… Non tratteremo il semantic web per motivi di tempo, ma vi consiglio di approfondire l’argomento.

*“Ho un sogno per il Web … ed esso è diviso in due parti. Nella prima parte, il Web diventa un mezzo molto più potente per la collaborazione tra le persone. Ho sempre immaginato lo spazio delle informazioni come qualcosa cui chiunque potesse avere accesso immediato ed intuitivo, e non solo di consultarlo, ma di crearlo. […]*

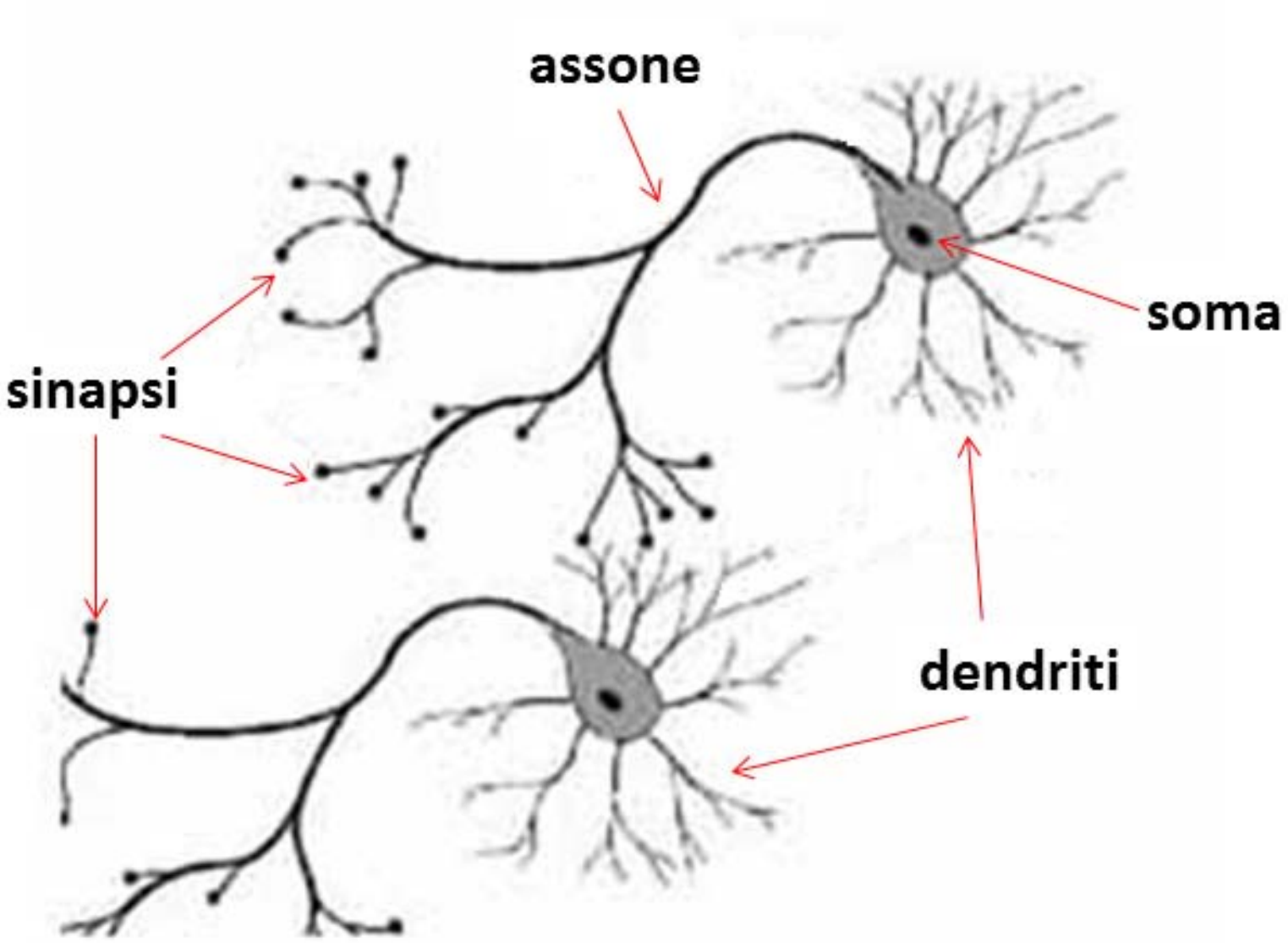
***Inoltre, il sogno della comunicazione tra le persone, attraverso la condivisione della conoscenza, deve essere possibile per gruppi di ogni dimensione, interagenti elettronicamente con la facilità con la quale essi comunicano di persona”.***

Tim Berners – Lee, Weaving the Web

## LE ARTIFICIAL NEURAL NETWORK (ANN)

Se nella prima parte del corso ci siamo occupati puramente di logica e di rappresentazione non biologica della conoscenza, ora ci addentreremo in un tipo di rappresentazione e di AI più strettamente legata all'intelligenza biologica da cui trae spunto: le reti neurali.

Ogni trattazione sulle reti neurali inizia con un'immagine del neurone biologico, che ha ispirato i modelli di neurone artificiale.


assone
soma
sinapsi
dendriti

Un neurone (cellula nervosa) è l'unità fondamentale del sistema nervoso (tutto, cervello incluso). In un neurone si distinguono:

- Il corpo cellulare (soma) che contiene il nucleo cellulare.

- Dal corpo cellulare si ramifica un gran numero di fibre (i dendriti) ed una singola fibra lunga (l'assone).

- I dendriti si ramificano in una rete attorno alla cellula (simile ad un cespuglio) e l'assone si allunga in genere di circa un centimetro (ma in casi estremi moto di più).

- Verso la sua estremità l'assone si suddivide in ramificazioni che si connettono ai dendriti ed ai corpi cellulari di altri neuroni, attraverso una giunzione (collegamento) che prende il nome di sinapsi. Ogni neurone forma sinapsi con altri neuroni (da una dozzina ad un centinaio di migliaia).

I neuroni trasmettono un segnale elettrico lungo gli assoni.

Tra il terminale di un assone e la cellula che riceverà il segnale sussiste uno spazio (fessura sinaptica), che viene superato dai segnali grazie a sostanze chimiche dette neurotrasmettitori.

Quando un segnale elettrico arriva in prossimità, le sinapsi rilasciano i neurotrasmettitori.

Semplificando, possiamo dire che la quantità di neurotrasmettitore rilasciato determina la conduttività della sinapsi (cioè quanto attenui o esalti il segnale elettrico proveniente dall'assone).

Il neurone a valle della fessura sinaptica p dotato di recettori in grado di intercettare il neuromodulatore: si generano correnti locali nei pressi della sinapsi che possono sommarsi nello spazio e nel tempo in prossimità dei dendriti e del neurone.

Quando la somma delle correnti che arriva alla base dell'assone si rivela superiore ad una data soglia, viene generato un impulso (spike) di corrente di breve durata (da 2 a 5 millisecondi).

Lo spike si propaga attraverso l'assone verso le sinapsi e, quando le raggiunge, queste rilasciano i neurotrasmettitori, dando il via alla ripetizione dell'intero processo per i neuroni collegati a valle.

In una rete neurale artificiale i neuroni sono **unità** che elaborano in modo semplice i segnali ricevuti in ingresso da altri neuroni.

Le sinapsi sono **connessioni** tra le unità ed il **segnale elettrico** è un numero (solitamente compreso nell'intervallo [0,1]).

La conduttività delle sinapsi è data dal **peso della connessione** (solitamente indicato con **w**), un numero il cui effetto sul segnale si ottiene moltiplicando il segnale per il peso stesso, prima che sia raggiunta l'unità successiva.

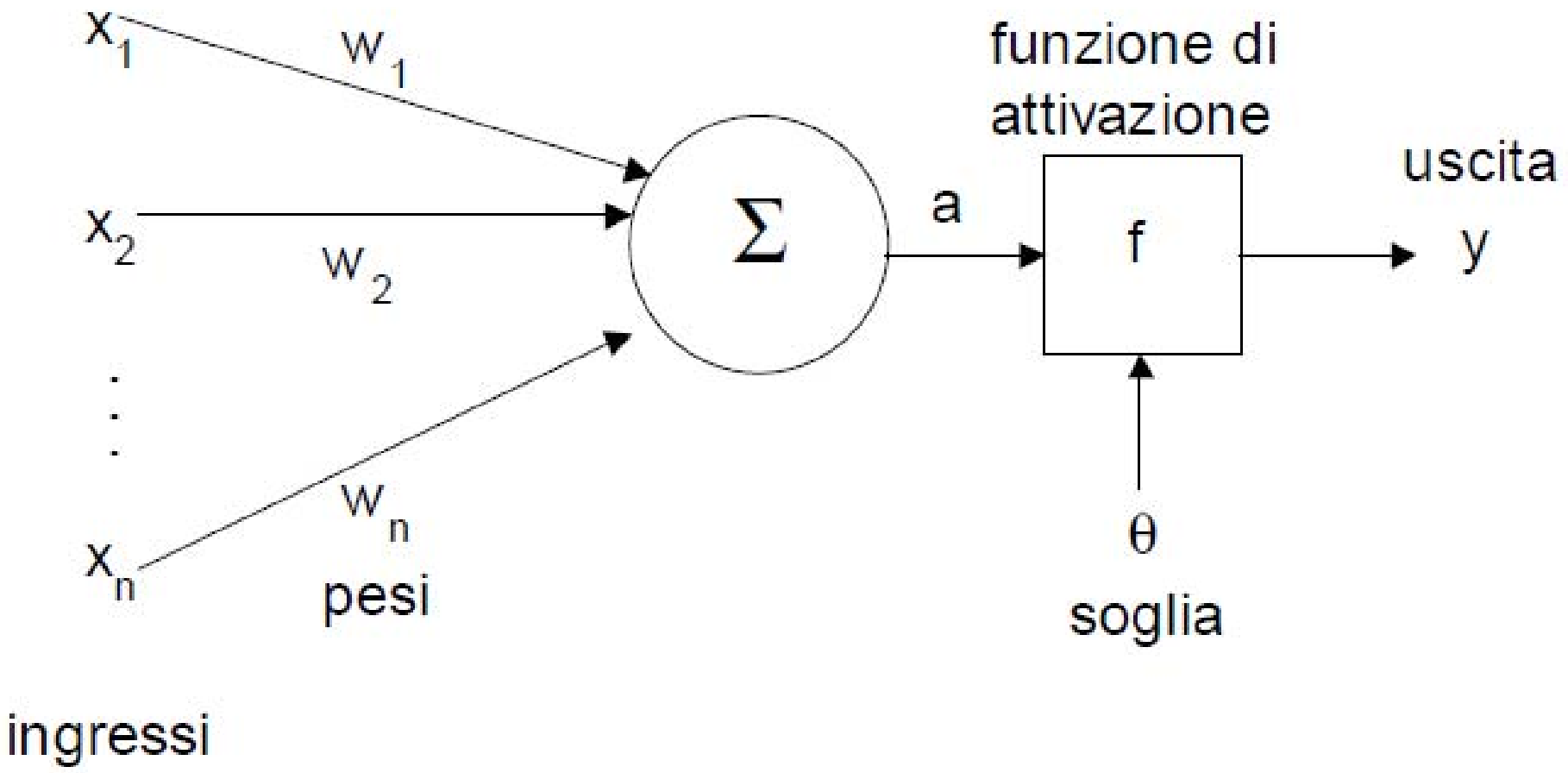
Per portare un esempio semplice, se x è il segnale che si muove lungo la connessione tra due neuroni e w è il peso della connessione, il segnale "pesato" sarà dato da w*x.
Le unità dunque calcolano il potenziale di attivazione e si attivano.

Il calcolo del **potenziale di attivazione** è dato dalla somma dei segnali pesati che arrivano dagli altri neuroni.

Vediamo il processo in un modo più formale. Abbiamo n canali di ingresso $x_1, \ldots, x_n$ ad ognuno dei quali è associato un peso $w_i$, ovvero un numero reale che riproduce la sinapsi. Se $w_i > 0$, il canale è ***eccitatorio***, se $w_i < 0$, il canale si dice ***inibitorio***.

Il valore assoluto di di un peso rappresenta la forza della connessione.

L’output, cioè il segnale con cui il neurone trasmette la sua attività all’esterno, è calcolata applicando una ***funzione di attivazione*** alla somma pesata degli ingressi.



Indicando con $a = \sum_{i=1}^{n} w_i x_i$ la somma pesata degli ingressi, si ha

$$y = f(a) = f(\sum_{i=1}^{n} w_i x_i)$$

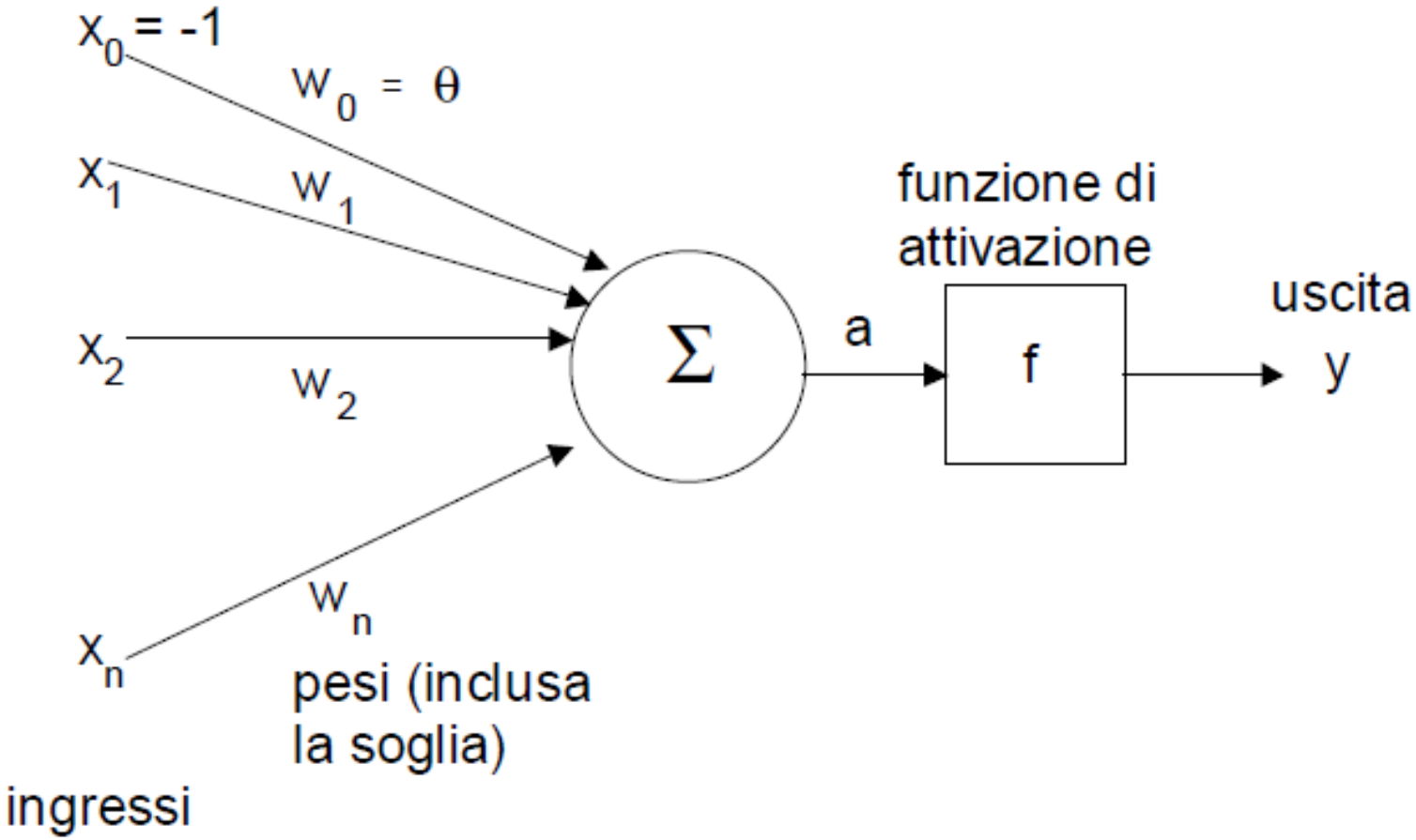
La somma pesata degli input in letteratura è spesso indicata con la parola ***net***.

Il calcolo del segnale da inviare alle altre unità sulla base del segnale di attivazione è effettuato mediante funzioni matematiche dette ***funzioni di trasferimento***.

La soglia di fatto abbassa il valore di ingresso della funzione di attivazione, quindi può essere vista come un peso associato ad un ulteriore canale di ingresso $x_0$, che ha un valore costante pari a -1.

La formula allora diventa $y = f(a) = f(\sum_{i=0}^{n} w_i x_i)$ in cui $w_0 = \theta$

cui corrisponde il modello illustrato nella figura che segue:



A volte non si considera la soglia ma il suo opposto (***bias***) che può essere visto come il peso di un canale di input con valore costante pari a 1. In questo caso, indicando il bias con la lettera b, la formula diventa

$$y = f(a) = f(\sum_{i=0}^{n} w_i x_i)$$

in cui $x_0=1$ e $w_0 = b$

È possibile adottare diversi tipi di funzioni di trasferimento, che caratterizzano il tipo di unità che ne fa uso. Vedremo le più utilizzate.

## Le funzioni di attivazione

Ricordiamo che abbiamo definito con **a** il livello di attivazione e quindi con f(a) la funzione di attivazione che entra in gioco a seconda del valore del livello.

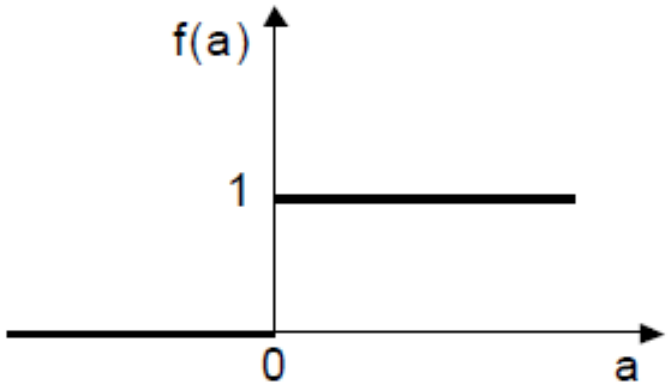
**a** può essere un numero reale che può anche apparterene ad un dato intervallo (per esempio [0,1]) oppure può essere un numero appartenente ad un insieme discreto (di solito {0,1} oppure {-1,+1}).

**Funzione binaria a soglia**

L'uscita (y) assume i valori:

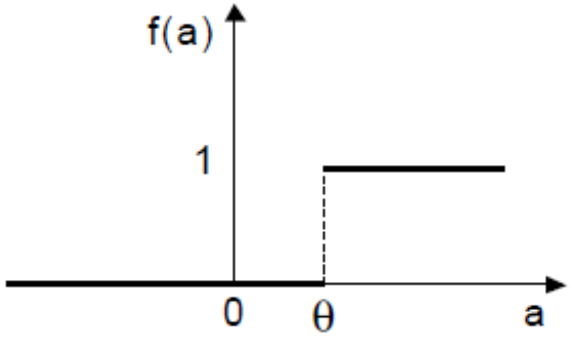
$$y = \begin{cases} 1 & se \quad a \geq 0 \\ 0 & se \quad a < 0 \end{cases}$$

Il grafico corrispondente




è:

Se vogliamo evidenziare il contributo x0, scorporandolo dal livello di attivazione, il grafico diventa:

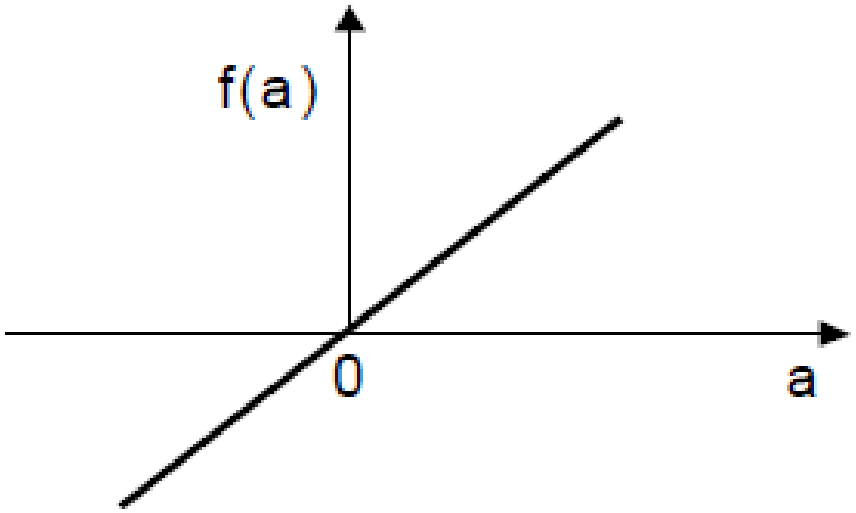



e, corrispondentemente, la formula sarà:

$$y = \begin{cases} 1 & se \quad \sum_{i=1}^{n} w_i x_i \geq \vartheta \\ 0 & se \quad a < \vartheta \end{cases}$$

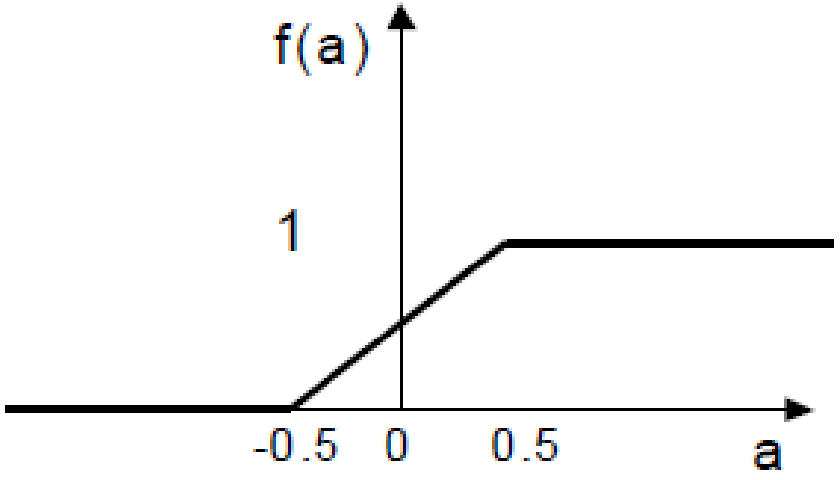
# Funzione lineare

f(a) = a



# Funzione lineare a tratti

$$y = \begin{cases} 0 & se \quad a \leq -0.5 \\ a + 0.5 & se \quad -0.5 < a < 0.5 \\ 1 & se \quad a \geq 0.5 \end{cases}$$

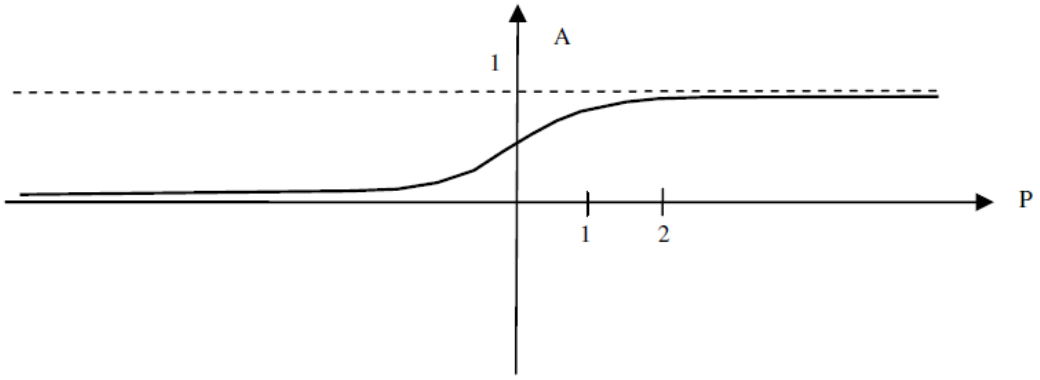


## Funzione sigmoidale

La funzione ha la seguente equazione:

$$a = \frac{1}{1+e^{-p}} = \frac{1}{1+\frac{1}{e^{p}}}$$

***e*** è il numero di Nepero (vale circa 2,718281828) che gode di una serie di proprietà che potete approfondire facilmente in Rete.
Il grafico della funzione è una versione senza discontinuità della funzione binaria:



**Funzione segno**

Le funzioni di attivazione viste, eccezion fatta per la funzione lineare, assumono valori compresi tra 0 e +1. È tuttavia possibile ridefinire la funzione a soglia in modo che possa assumere valori compresi tra–1 e +1.

La funzione a soglia viene ridefinita così:

$$y = \begin{cases} +1 & se \quad a > 0 \\ 0 & se \quad a = 0 \\ -1 & se \quad a < 0 \end{cases}$$

Qui: http://www.docstoc.com/docs/93689916/Artificial-Intelligence-CIS-342 una bella e semplice introduzione generale (un riassunto veloce di quello che abbiamo visto)

## ANN: cenni storici

Abbiamo visto, per ora, i principi generali che occorrono per poter comprendere le ANN.
La bellezza delle reti neurali risiede nella semplicità del meccanismo.
Ma come si è arrivati a queste implementazioni in AI?

Abbiamo visto che la rete neurale artificiale è un tentativo di riprodurre la struttura di base di un rete neurale del cervello umano.

A differenza dei modelli dell'AI classica, si assiste ad un vero e proprio modello che simula i meccanismi del cervello umano.

Nel modello di AI classica si ha un motore centrale (è l'analogia con il cervello) che analizza le diverse strategie possibili e decide quale sia la migliore; in una rete neurale non vi è un decisore centrale, ma la decisione è il risultato di un lavoro cooperativo e distribuito tra tutte le unità di base (i neuroni).

Questo modello è chiaramente applicabile non solo alle decisioni, ma anche a tutti i tipi di rappresentazione della conoscenza.

Dopo un iniziale periodo di entusiasmo, la ricerca in questo campo ha attraversato un periodo di difficoltà, durante il quale il supporto professionale ed economico fu minimo, ed i progressi più importanti furono realizzati da una relativamente ristretta cerchia di ricercatori.

**Agli inizi degli anni ’40, il neurofisiologo Warren McCulloch ed il logico Walter Pitts elaborano un modello formale dell’attività dei neuroni** e ne riproducono, per la prima volta, uno artificiale: un dispositivo binario semplice con soglie elettriche fissate che, per lo scarso livello tecnologico del tempo, producevano risultati che non andavano oltre semplici funzioni logiche quali "a o b" piuttosto che "a e b".

McCullogh & Pitts in un loro paper, presentano il modello semplificato di neurone a soglia, asserendo che la semplificazione è efficiente perché può riprodurre il meccanismo generale di funzionamento del neurone umano e, quindi, del cervello.
M&P dimostrano anche che si può costruire una rete di neuroni in grado di svolgere tutte le operazioni del calcolatore universale di Turing. Inizialmente, dunque, non c'è novità rispetto all'AI classica: il neurone di M&P è un calcolatore universale.

**Nel 1949 lo psicologo Donald Hebb** introduce il principio del rinforzo delle sinapsi nel tempo, determinato dal loro utilizzo. (apprendimento non supervisionato)

**Nel 1954 Belmont Farley e Wesley Clark**, del MIT, si concentrarono su modelli computerizzati di simulazione del neurone, coinvolgendo anche alcuni neurologi, dando il via ad una tendenza alla multidisciplinarità ancora molto forte in AI.

**Nel 1958 l'informatico Frank Rosenblatt sviluppa il perceptrone** (apprendimento supervisionato), una rete a tre strati (layer), input, strato associativo, output, in grado di imparare a connettere o ad associare un input ad un output casuale.

Il perceptrone è ispirato agli studi sulla visione del cervello (campo in cui si hanno le conoscenze più approfondite).

Il funzionamento del perceptrone di Rosenblatt si basa sul concetto che lo strato di neuroni di input (di stato 0 o 1) dipenda dai segnali che vi giungono.

Rosenblatt ipotizza che esistano dei feature detector (simili ai neuroni center-surround, che rispondono a determinati stimoli), ognuno collegato a pochi recettori o sensori. Dunque ogni feature detector è collegato in modo casuale ad un numero di neuroni di input, con connessioni dai valori assolutamente casuali.

Tutti i feature detector sono collegati ad un neurone di output con connessioni non casuali.

L'output di questo neurone è calcolato sommando gli input moltiplicati per i rispettivi pesi: se il risultato è superiore alla soglia tale neurone "spara", se è inferiore alla soglia, no.

La grande innovazione di Rosenblatt è che ha definito una procedura per modificare i pesi in base all'esperienza: la perceptron learning procedure.

Cosa faccio? Prendo un input e lo utilizzo per fissare il valore di tutti neuroni di input.

A questo punto calcolo l'output di tutti gli altri neuroni del sistema e lo confronto con il risultato che avrei voluto (immaginate che tra gli input dati – es. lettere dell'alfabeto -, mi aspetti in uscita una specifica lettera dell'alfabeto).

In pratica se in uscita ottengo quello che desidero, lascio i pesi così come sono, altrimenti li vario leggermente (W piccolo), finché non ottengo l'output che desidero.

In altre parole se l'output è on quando doveva essere off, riduco la forza delle connessioni verso i feature detector che sono on, quindi la prossima volta questi saranno meno importanti, viceversa se l'output è off quando doveva essere on, aumento la forza delle connessioni verso i feature detector che sono on.

Tutti questi aumenti sono W molto piccoli.

Posso dare ad un perceptrone compiti auto-associativi (fornisco in input un'immagine e deve generare in output la stessa immagine), oppure etero-associati o ancora di categorizzazione.

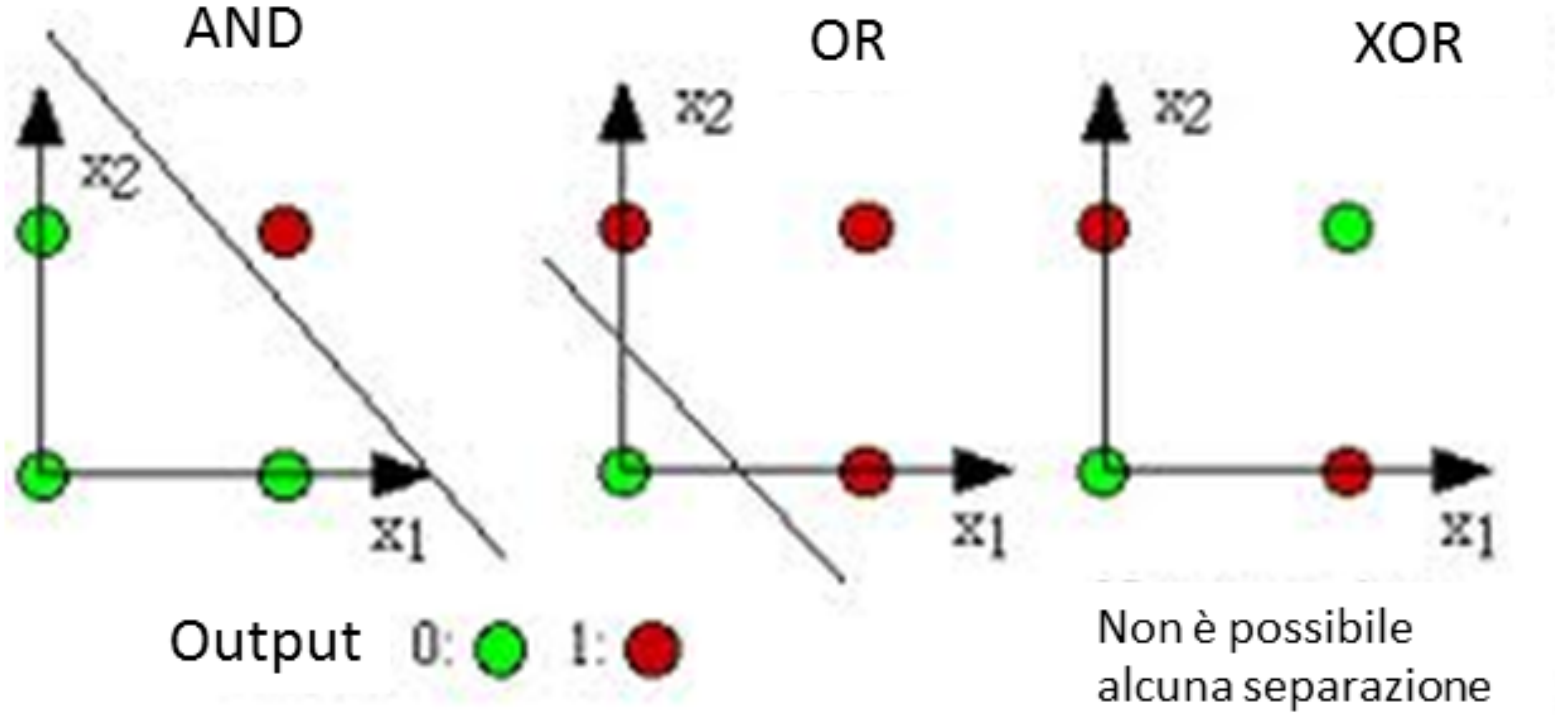
Il Perceptrone ha comunque, oltre alla semplificazione eccessiva, altri limiti già noti anche a McCollough, Pitts e Rosenblat, quale quello della separabilità lineare, che approfondiremo in seguito.

Intanto, **nel 1960, Bernard Widrow e Marcian Hoff** (Stanford University) sviluppano la rete **ADALINE** (ADAptive Linear Element), che adotta una nuova regola di apprendimento: la Least-Mean-Squares (LMS).

**Nel 1969 Marvin Minsky e Seymour Paper** criticano aspramente le ricerche di Rosenblatt, dimostrando che le capacità del perceptrone sono molto limitate perché non in grado di risolvere nemmeno problemi semplici come lo XOR (O esclusivo).
Le conseguenze sono l'abbandono generale degli studi sulle reti neurali.

Vediamo velocemente in cosa consiste il problema della separabilità lineare attraverso il caso dello XOR.
Il perceptrone di Rosenblatt apprende funzioni calcolabili, cioè linearmente separabili.
Vediamo tre funzioni semplici:
AND: vero (pallino rosso) se si ha x=1 e y=1; falso (pallino verde) in tutti gli altri casi.
OR: vero (pallino rosso) se si ha x=1 oppure y=1; falso (pallino verde) in caso contrario.
XOR: falso (pallino verde) solo se x=y=1 oppure se x=y=0; vero (pallino rosso) negli altri casi.

Fate riferimento alla figura che segue:




il neurone deve decidere se “sparare” o no in base al fatto che il risultato si trovi sopra o sotto la soglia, quindi (nel nostro caso in cui i valori sono due, cioè l’1 o lo 0), i risultati (vero=1 e falso=0) devono poter essere separati in due classi, divise dall’opportuna soglia.

Graficamente, devo poter dividere in due lo spazio con una linea (è la funzione della soglia) in modo che i pallini rossi si trovino tutti da una parte e i pallini verdi in un'altra parte.

La conseguenza di quello che abbiamo visto è che non esiste alcuna soglia che possa essere fissata per dividere lo spazio in due e la funzione XOR non è tra quelle apprendibili da un perceptrone.

Il caso dello XOR è il più semplice, utilizzato per farvi comprendere il problema e utilizzato da Minsky e Papert per criticare il perceptrone.

La realtà è che un numero molto elevato di funzioni matematiche risultano non separabili e quindi non possono essere calcolate dal perceptrone.
Cade la tesi del calcolatore universale.

Non è la sola critica di Minsky e Papert, che adducono altri casi critici per il perceptrone, quali il "problema della parità" e del "credit assignement" (approfondire).

L'interesse per le reti neurali resta sopito fino agli anni '80, quando Grossberg e Carpenter svilupparono Art (Adptive Resonance Theory) basata su modelli biologici.

Nel fattempo erano state condotte altre ricerche, tra cui va ricordato sicuramente Klopf, che lavora all'apprendimento atrificiale ispirandosi ai neuroni biologici (eterostasi) e precedentemente Werbos, nel 1974, che inizia a studiare i metodi di apprendimento back-propagation, ovvero un perceptrone a più layer, con migliori funzioni e regole di apprendimento. Ricordiamo ancora gli studi di Shun-Ichi Aari sul metodo error-correction, apprendimento basato su classificazione tramite esempi adattabili, e la rete Neocognitron, di Kunihiko (1975), che permette di interpretare caratteri scritti a mano.

Dunque negli anni ’80 si assiste ad un ritorno di interesse per le reti neurali, principalmente per due fattori: nuovi risultati teorici che consentono di superare le limitazioni del Perceptrone; elevata potenza di calcolo dei nuovi sistemi di elaborazione.

**Nel 1982 Hopfield** analizza le reti costituite da unità binarie e connessioni simmetriche. È possibile introdurre una funzione energia che determina lo stato della rete (che tende a minimizzare la sua energia). Questo modello dinamico porta una rivoluzione nello sviluppo delle reti neurali.
**Nel 1986 viene proposto l’algoritmo di backpropagation**, che è una generalizzazione dell’algoritmo di apprendimento del perceptrone alle reti multi-strato (multi-layer) e consente di superare i problemi individuati da Minsky e Papert.

L’introduzione di livelli intermedi permette alla rete di creare una rappresentazione interna dell’ambiente esterno, quindi la rete acquisisce la capacità di interpretare e generalizzare. Oggi la backpropagation è l’algoritmo più utilizzato.

Un altro modello da ricordare è il ***competitive learning*** di **Rumelhart e Zipster**: ogni unità apprende come rispondere ad una configurazione di ingresso specifica, entrando in competizione con altre unità attraverso mutue inibizioni.

**Kohonen** propone un algortmo di competitive learning che mette in relazione la densità spaziale dei cluster alla distribuzione di probabilità delle possibili configurazioni di ingresso.

Oggi le reti neurali sono ampiamente ed efficacemente utilizzate ed i modelli, dinamici e multi-strato, sono in continua evoluzione.

Approfondire: chi era Herbert Alexander Simon?

I diversi modelli di reti neurali si differenziano a seconda di alcune caratteristiche:
tipo di utilizzo
- architettura dei collegamenti
- modalità di apprendimento
- algoritmo di apprendimento

Parleremo di tutto nel seguito, ma innanzitutto distinguiamo le reti per tipo di utilizzo, per cui si distinguono tre categorie principali:
- **memorie associative**. Possono apprendere associazioni tra pattern, ovvero insiemi complessi di dati, quali ad esempio i pixel di un'immagine, in modo che la presentazione di un pattern specifico fornisca come output un pattern obiettivo anche se il primo risulta impreciso o parziale (resistenza al rumore). Una memoria associativa può essere anche utilizzata per fornire in output il pattern completo in risposta ad un pattern parziale in input.

- **Simulatori di funzioni matematiche**. Comprendono la funzione che lega l'input all'output in basse ad esempi forniti in fase di training. Dopo l'addestramento la rete è in grado di fornire output anche in risposta ad input diversi da quelli usati negli esempi di addestramento. La rete dunque interpola ed estrapola dai dai dati del training set. Tali capacità sono verificabili addestrando una rete con una sequenza di dati input/output proveniente da una funzione nota. La rete si comporta come una black box, poiché la funzione di trasferimento interna non è svelata completamente. La rete a retropropagazione dell'errore (error back propagation) fa parte di questa categoria.
- **Classificatori**. Si tratta di reti costruite con lo scopo di classificare i dati in categorie in base a caratteristiche di similitudine. Possono essere sia di tipo supervisionato che auto-organizzante, quindi senza categorie predefinite.

## Classificazione delle modalità di apprendimento

Abbiamo detto che Rosenblatt introdusse il concetto di apprendimento nelle reti neurali e, incidentalmente, abbiamo nominato, ad esempio, il termine “apprendimento supervisionato”.

Prima di proseguire, sarà bene chiarire pochi semplici concetti su tale argomento.

L’apprendimento si può classificare adottando criteri diversi a seconda che sia

- Basato sull’informazione a disposizione, ovvero:
  - Supervisionato
  - Con rinforzo
  - Non supervisionato
- Basato sul ruolo del sistema che apprende, e cioè
  - Apprendimento passivo
  - Apprendimento attivo

**Nell'apprendimento supervisionato** si ha a disposizione un insieme di esempi classificati $x:<v_1, v_2, \ldots, v_n, o>$, in cui $\mathbf{v_i}$ rappresenta i valori delle variabili di ingresso ed **o** è l'uscita.

L'apprendimento supervisionato implica necessariamente l'esistenza di un istruttore che conosce la risposta corretta attesa.

In questo caso si vuole apprendere una data funzione obiettivo $f: V_1 \times V_2 \times V_3 \ldots \rightarrow O$

Un possibile algoritmo consiste nello scegliere, all'interno delle classi di ipotesi H, quella che rende minimo l'errore di approssimazione della funzione obiettivo sugli esempi a disposizione.

L'apprendimento supervisionato avviene secondo meccanismi induttivi, a partire cioè da esempi. Consiste dunque nella descrizione di una funzione a partire da un insieme di coppie ingresso/uscita dette ***esempi***.

In generale possono esservi molte ipotesi alle quali è possibile assegnare un grado di preferenza, detto ***inclinazione***, al di là del semplice criterio di consistenza con gli esempi.

Ogni ipotesi tale da approssimare il comportamento della funzione obiettivo su un numero sufficientemente grande di esempi lo approssimerà anche sui campioni non osservati (dalla teoria statistica del campionamento e dalla computational learning theory)

**Nell'apprendimento per rinforzo**, l'apprendista interagisce con l'ambiente e riceve una ricompensa (numerica) positiva o negativa a seconda del risultato delle azioni conseguenza dell'apprendimento.

Si apprende così una strategia di comportamento, ovvero un "piano" o sequenza di azioni. In questo caso la misura della prestazione consiste nel rendere massima "a lungo termine" la ricompensa complessiva ricevuta dal sistema.

**L’apprendimento non supervisionato** prevede che l’esperienza di apprendimento sia rappresentata da dati non classificati (ad esempio cartelle cliniche, pagine web). “Apprendere” in questo caso significa rilevare ridondanze o correlazioni statistiche dei dati.

Occorre un ultimo inciso sulle modalità di apprendimento dal punto di vista del learner.
Nell'**apprendimento passivo** il sistema che apprende può imparare solo dai dati che vengono messi a disposizione (esempi).
**Nell'apprendimento attivo**, invece, il learner può fare domande e condurre esperimenti. In questo caso sorge, però, la necessità di limitare l'intrusività del learner, stabilendo un modo ottimale di indagine autonoma.
Tra i modelli supervisionati occorre ricordare i MultiLayer Perceptron (vedremo meglio questi ultimi), i Neurofuzzy Models.

Tra quelli non supervisionati (unsupervised) i più utilizzati sono le SOM (Self-Organizing Map).

Nel seguito, quando parleremo dei modelli e delle architetture, approfondiremo i meccanismi di addestramento supervisionato e gli algoritmi adottabili.

## Modelli e architetture di rete neurale

Vi sono diversi modelli e diverse architetture adottabili per l'implementazione di una rete neurale.

La scelta ovviamente dipende dal tipo di problema che si vuole trattare.

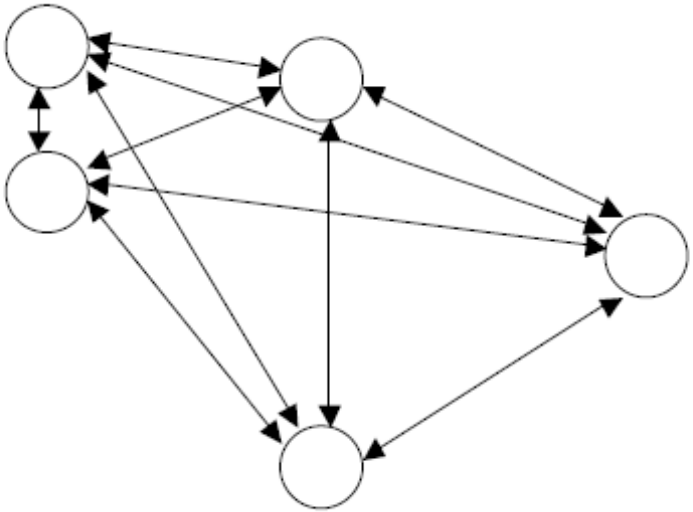
Per completare l'introduzione alle ANN che stiamo conducendo, ne presenteremo due, ma potete facilmente approfondire l'argomento in Rete.

**Reti completamente connesse (senza strati)**

In una rete completamente connessa ogni neurone è connesso bidirezionalmente con ognuno degli altri.

Le connessioni sono rappresentate da una matrice quadrata W, di dimensione pari al numero di neuroni.

Il generico elemento *wij* della matrice rappresenta il peso della connessione tra il neurone *i* ed il neurone *j*.

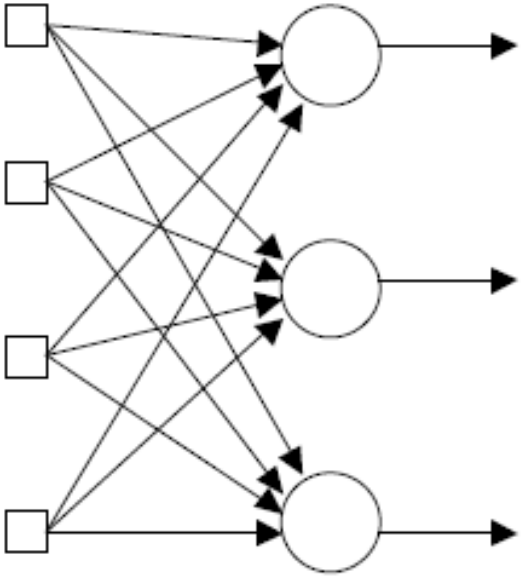


## Reti stratificate

Le reti stratificate sono caratterizzate dalla presenza di neuroni ciascuno connesso con tutti quelli dello strato successivo, senza che esistano connessioni tra neuroni del medesimo strato o tra strati non adiacenti.

Nello strato di input non avviene alcuna computazione (i neuroni di ingresso passano semplicemente i segnali ricevuti allo strato successivo), per cui una rete come quella presentata in figura A viene considerata come una rete con un solo strato.

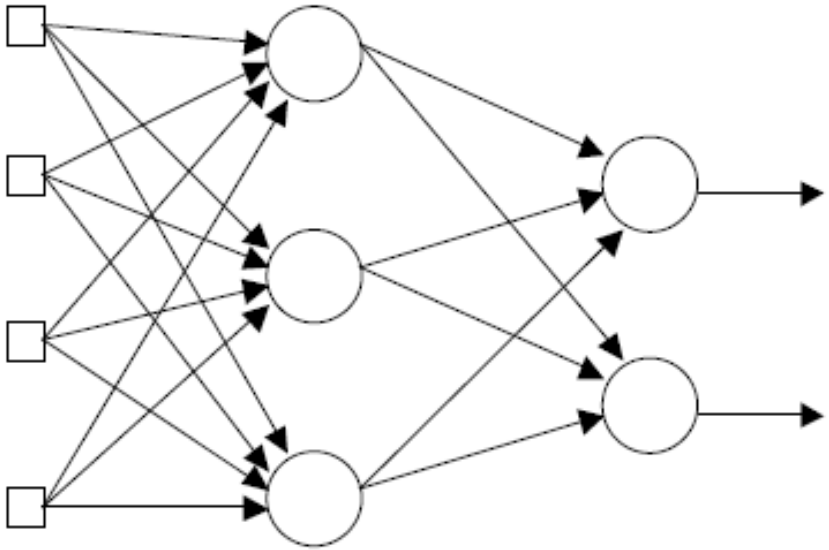
Figura A:



strato di ingresso

strato di uscita

Figura B:



strato di ingresso

strato nascosto

strato di uscita

In questo tipo di rete i segnali viaggiano dallo strato di ingresso verso lo strato di uscita e si dice che la rete è ***feedforward***.

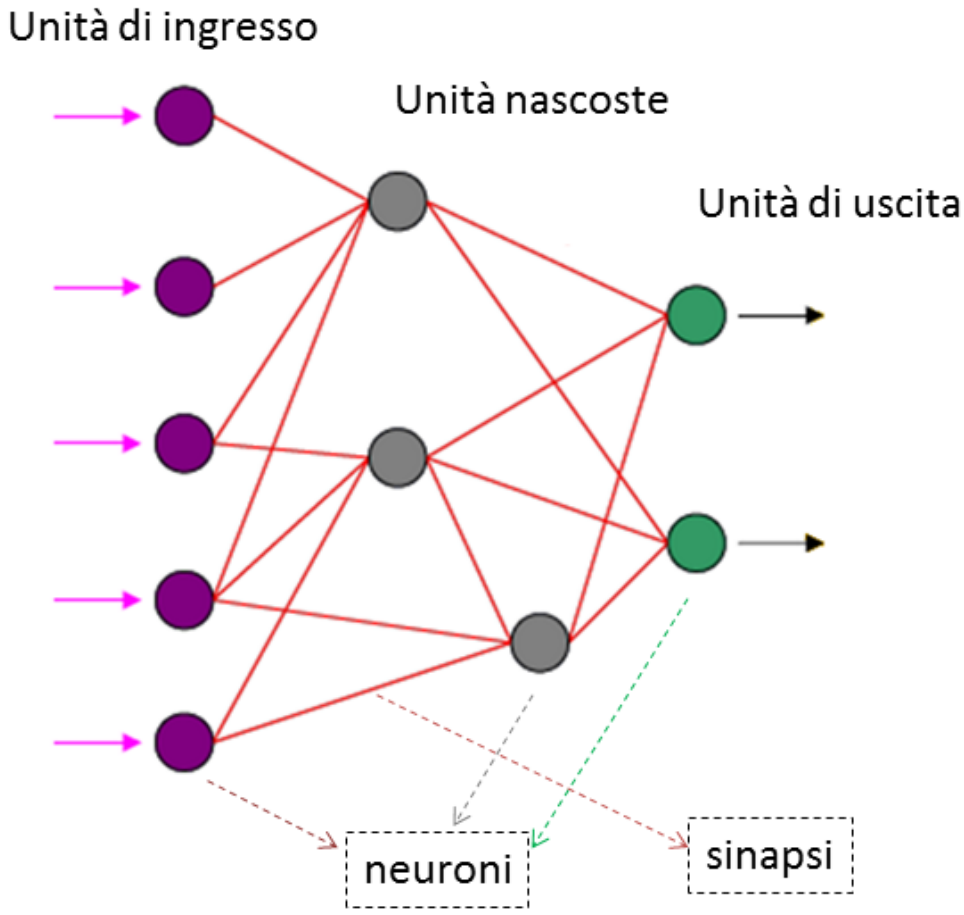
In figura B è invece mostrata una rete stratificata feedforward con uno strato nascosto, ovvero uno strato i cui neuroni non comunicano in modo diretto con l'esterno.
Possono essere presenti uno o più strati nascosti, che permettono alla rete di costruire opportune rappresentazioni interne degli stimoli in ingresso, in modo da facilitare e organizzare il compito della rete.
Le connessioni tra i neuroni di una rete a strati sono rappresentate tramite tante matrici quante le coppie di strati adiacenti.
Ciascuna matrice contiene i pesi delle connessioni tra le coppie di neuroni di strati adiacenti.

Per i tipi di problemi che più comunemente potrete incontrare nelle vostre attività, e per semplicità di trattazione, noi ci occuperemo principalmente di **reti supervisionate**, pur accennando, per completezza, anche agli altri approcci. In particolare vedremo nel

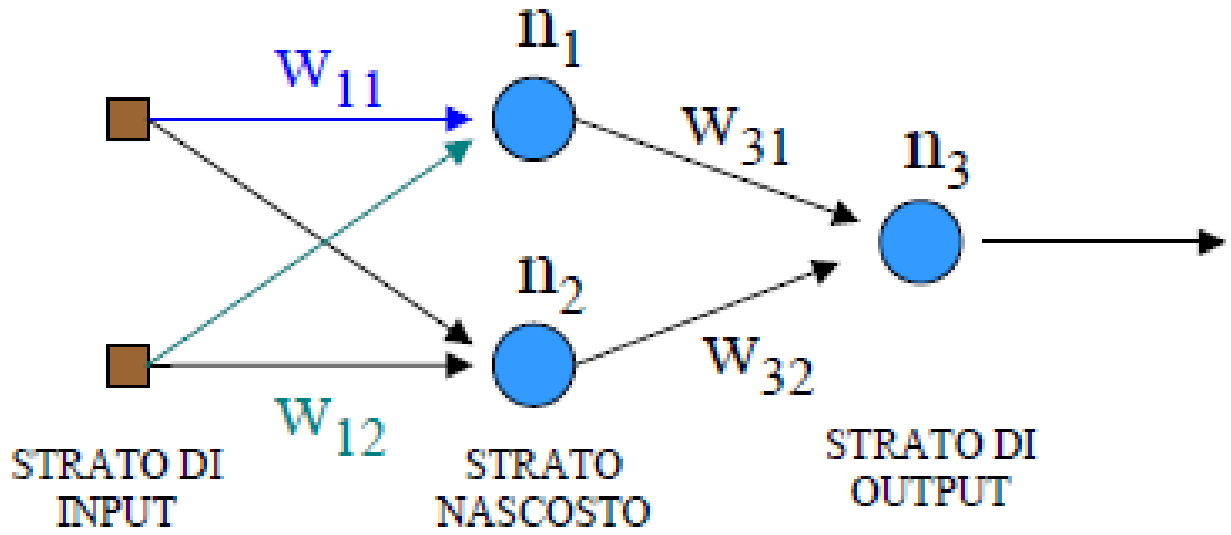
dettaglio come si progetta e sviluppa una ANN multilayer perceptron.

Possiamo dire che l'architettura generale di una rete neurale sia quella mostrata nella figura accanto, costituita da

- unità di ingresso (input unit), che ricevono i segnali dall'ambiente esterno;
- unità nascoste (hidden units), opzionali. Partecipano alle comunicazioni all'interno della rete;
- unità di uscita (output unit), che forniscono la risposta della rete.



Abbiamo già parlato del perceptrone elementare (senza strati nascosti) e del fatto che non può classificare pattern che non siano linearmente separabili .




Il problema è però risolvibile con l'introduzione di uno strato nascosto. Se nel perceptrone elementare, ad esempio, introduco uno strato nascosto con due neuroni (modello McCulloch-Pitts) posso risolvere il problema dello XOR.

**Ma come si codificano le informazioni in ingresso alla rete?**

Se scegliamo la ***modalità locale***, in cui ogni unità di ingresso corrisponde ad un oggetto, abbiamo la necessità di un numero troppo elevato di neuroni di input, non avremmo flessibilità nella gestione di nuovi oggetti ed inoltre avremmo scarsa resistenza al rumore.

Se invece scegliamo la ***modalità distribuita***, cioè quella in cui la rappresentazione di ogni oggetto utilizza più nodi di ingresso (ognuno relativo ad una caratteristica specifica), superiamo i problemi precedenti.

La scelta di come rappresentare i pattern in ingresso alla rete neurale è determinante.

Inoltre potrebbe essere necessario effettuare una normalizzazione dei dati, ad esempio nel caso in cui provengano da “sorgenti” diverse.

Una delle formule più utilizzate per la normalizzazione consiste nel dividere ogni componente xi per la norma del vettore, secondo la formula:

$$x_i = \frac{x_i}{\sqrt{\sum_{i=1}^{n} x_i^2}}$$

Abbiamo detto che per determinare preventivamente i valori sinaptici delle connessioni fra i noi è necessaria una fase di **addestramento**, che consiste nella modifica graduale dei valori assunti inizialmente dalle sinapsi, sulla base dei pattern scelti (per l'addestramento).

I valori iniziali delle connessioni sinaptiche sono assegnati in modo casuale, per esempio all'interno di un intervallo, oppure sono posti tutti allo stesso valore.

L’addestramento può essere condotto:

- in modo **supervisionato**,
  utilizzando coppie di pattern <vettore di ingresso, risposta desiderata>
  l’aggiornamento dei pesi avviene in conseguenza della misura di errore tra la risposta data dalla rete e quella desiderata
- apprendimento **in auto-organizzazione**
  non viene specificata la risposta attesa dagli esempi forniti per l’addestramento, ma si definiscono regole attraverso le quali la rete possa organizzarsi

La **modifica dei pesi** può avvenire secondo una **modalità on-line**, ovvero in modo che l'aggiornamento avvenga per ogni pattern in ingresso, oppure **per epoche**, cioè effettuata dopo la presentazione di tutti i pattern in ingresso.

Le modifiche calcolate vengono sommate ai valori già presenti.

Il procedimento avviene più volte, in modo che la rete non "dimentichi" quanto ha appreso fino a quel momento.

Un altro aspetto importante è stabilire il **numero di esempi da utilizzare per la fase di training**. Non esiste, tuttavia, una regola ma si possono adottare delle stime. Generalmente per un multilayer perceptron in genere si sceglie un numero di record pari a 2 volte il numero totale delle connessioni.

Per calcolare il **numero delle connessioni** occorre conoscere il numero delle unità nascoste che in questo caso si stimano uguali alla radice quadrata del numero di ingressi moltiplicato il numero di uscite.

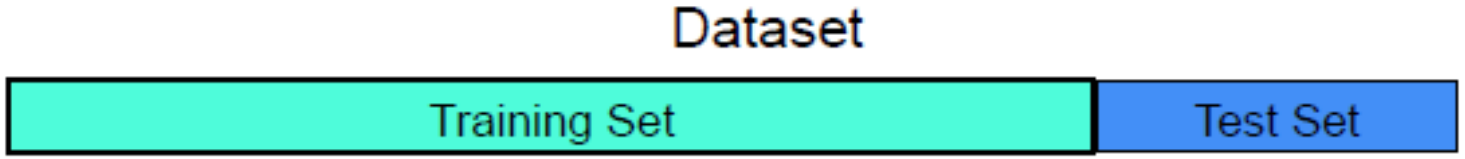
Raggiunto il **valore minimo di errore** della rete, si può fermare l'addestramento.

Tale valore dipende da vari fattori, quali la precisione che si intende ottenere dalla rete, o il tipo di funzione di attivazione, il numero di uscite della rete, e così via.

Una volta conclusa la fase di addestramento, si può procedere con la fase di test, che permette di osservare il comportamento della rete senza modificare i pesi sinaptici trovati durante il training.

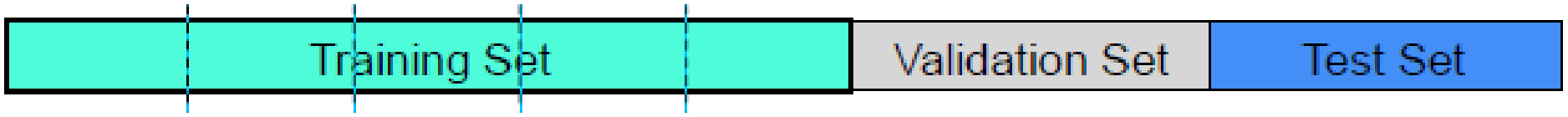
Riassumendo:

- fase di training: alla rete viene fornito un sottoinsieme di coppie input-output. La rete adatta il proprio stato interno per classificarle correttamente.
- Fase di test: viene fornito alla rete un sottoinsieme del dataset diverso dal precedente (di cui si conosce l'output). L'accuratezza della rete è misurata in termini di risposte corrette date dalla rete.



Per suddividere il dataset evitando di introdurre bias, in genere si adotta, in alternativa, uno dei due metodi:

- **Random sampling** (campionamento casuale). E' il più semplice, ma quello che segue fornisce maggiori garanzie.
- **K-fold cross validation**: il dataset è suddiviso in k sottoinsiemi. La rete è addestrata su k-1 sottoinsiemi e testata sul sottoinsieme restante. Il procedimento si itera k volte (con selezioni diverse) e si prende, infine, la media dei risultati. Ovviamente questo metodo è conveniente se si dispone di un numero sufficientemente grande di record nel dataset.



Per evitare l'overfitting (la rete non generalizza), in cui si può incorrere se si adatta troppo il modello ai dati o se si sceglie un modello troppo complesso, è preferibile spesso suddividere ulteriormente il training set, prendendone una parte come **validation set**.
Si userà quest'ultimo sottoinsieme per testare periodicamente l'accuratezza della rete durante l'addestramento: se l'errore aumenta, si arresta il training.

Se poi si deve scegliere tra più modelli o algoritmi di apprendimento, si utilizza il training set suddiviso in k sottoinsiemi di cross-validation, per confrontare le varie alternative.

Si allena il modello scelto sul training set, controllandone periodicamente l'accuratezza sul validation set e si arresta il training quando inizia l'overfitting.

L'accuratezza finale si valuta, in questo caso, sul test set.

Come abbiamo detto, non ci sono regole generali, ma la pratica suggerisce che per le reti neurali l'**approccio migliore** consiste nell'adottare la seguente strategia:

- ovviamente:
  - far corrispondere il numero di nodi in ingresso al numero di feature
  - far corrispondere il numero di nodi in uscita al numero di classi
- Suddividere il dataset in tre sottoinsiemi: training set, validation set, test set.
- Per stabilire il corretto numero di hidden layer e di nodi per livello generalmente si va per tentativi successivi. Adottare la k-fold cross validation sul training set può aiutare a trovare il giusto equilibrio
- Allenare la rete su tutto il training set, limitando l'overfitting con il validation set
  - Valutare l'accuratezza finale sul test set

Vi sono poi un paio di **regole euristiche** che possono aiutare:

- In genere un hidden layer è sufficiente per la maggior parte dei problemi. Inoltre in questo caso l'allenamento è più rapido)
- Conviene sempre cominciare con un numero di nodi interni basso e poi crescere finchè non si noti un miglioramento.