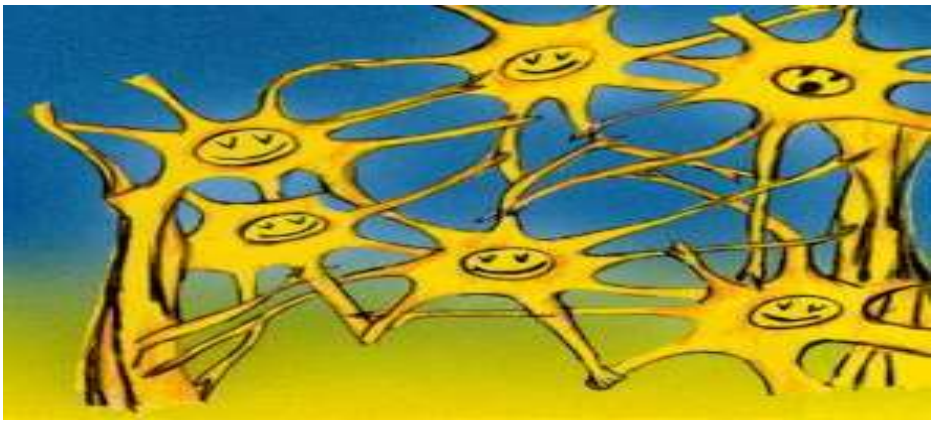
Introduzione alle reti neurali artificiali: modelli, pacchetti software, applicazioni statistiche, agroindustriali e di ricerca

Angelo Fabbri (who's&D/L)

Course Schedule

- Materiali http://angelofabbri.altervista.org
- (h 10:30) Test-in https://esami.polocesena.unibo.it
- Definitions, application examples
- Biological/artificial neuron
- Neuron Network concepts
- (h 13:30) Pause
- (h 14:30) Examples and practice on software
- (h 17+) Test-out



1.Summary

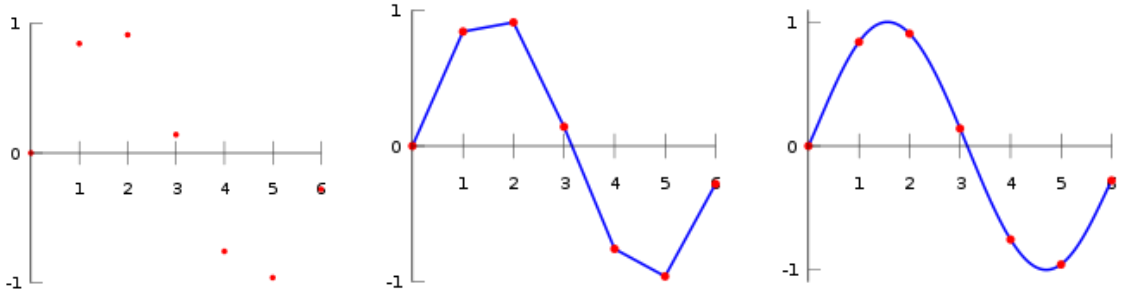
2.Una definizione, una rassegna di applicazioni e qualche riferimento

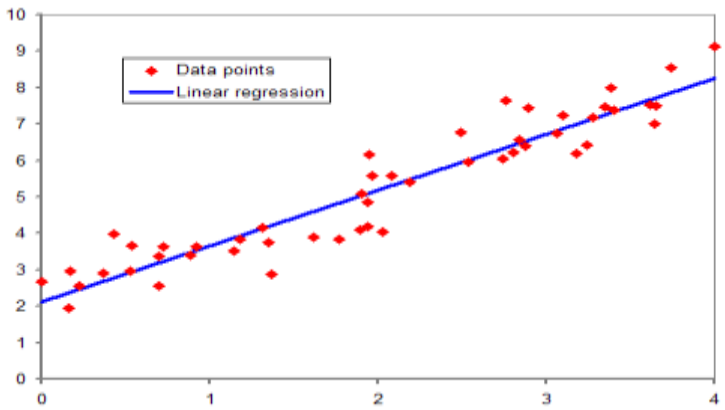
- L'idea di ANN (Artificial Neural Network), rispondendo alla necessità di determinare relazioni tra cause ed effetti, consiste sostanzialmente in una ***tecnica matematica particolarmente efficace nella ricerca di relazioni tra variabili, anche all'interno di set di dati molto complessi*** basata su una imitazione del neurone biologico.

La ricerca di relazioni attraverso le ANN segue un processo di apprendimento simile a quello dell'esperienza umana: stratificazione della conoscenza per analogia. Funziona bene anche con dati scarsi, mancanti o, al contrario, ridondanti. E' dunque molto differente rispetto ad altri metodi statistici, assai più rigidi, come p.e. quelli basati su piani fattoriali o la minimizzazione di una distanza da un modello predeterminato.

2.1.Esempi generali di ricerca di regole, come relazioni tra gruppi di variabili - Soft Computing

- come si può provare a capire quale legge matematica ha generato una serie di dati (lineare o non, interpolante o approssimante)?





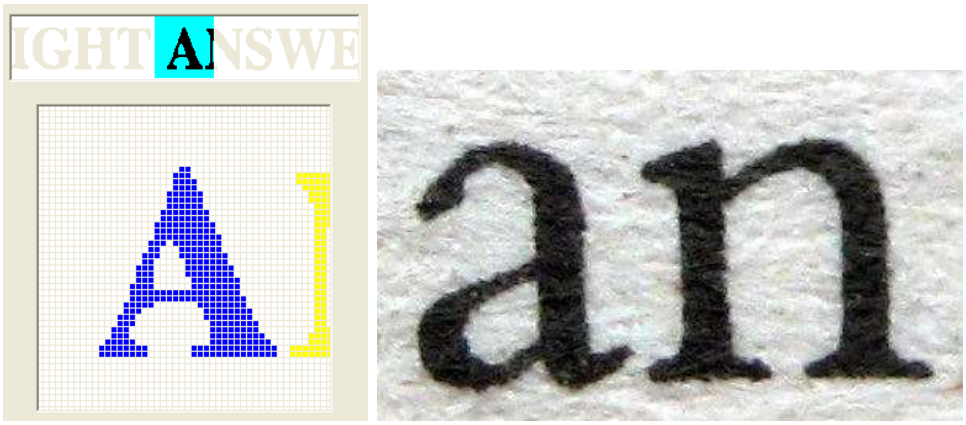
-

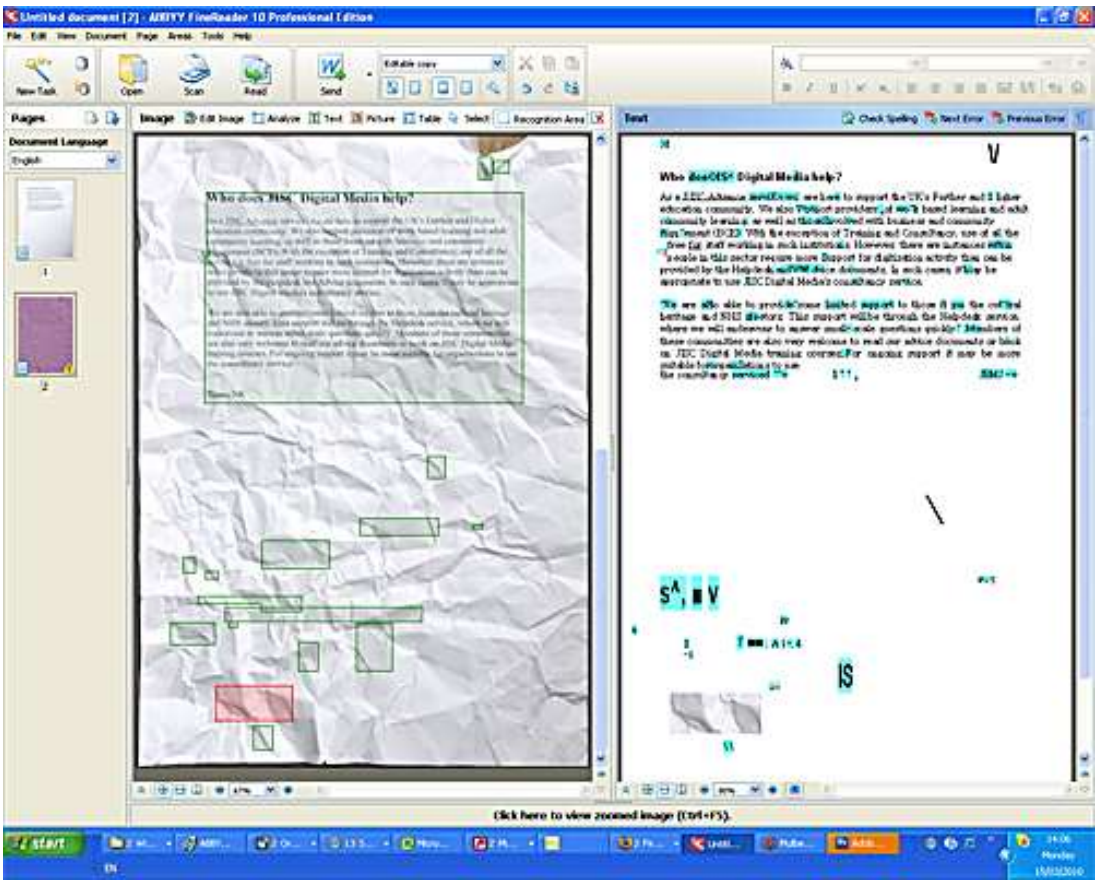
input data => output data

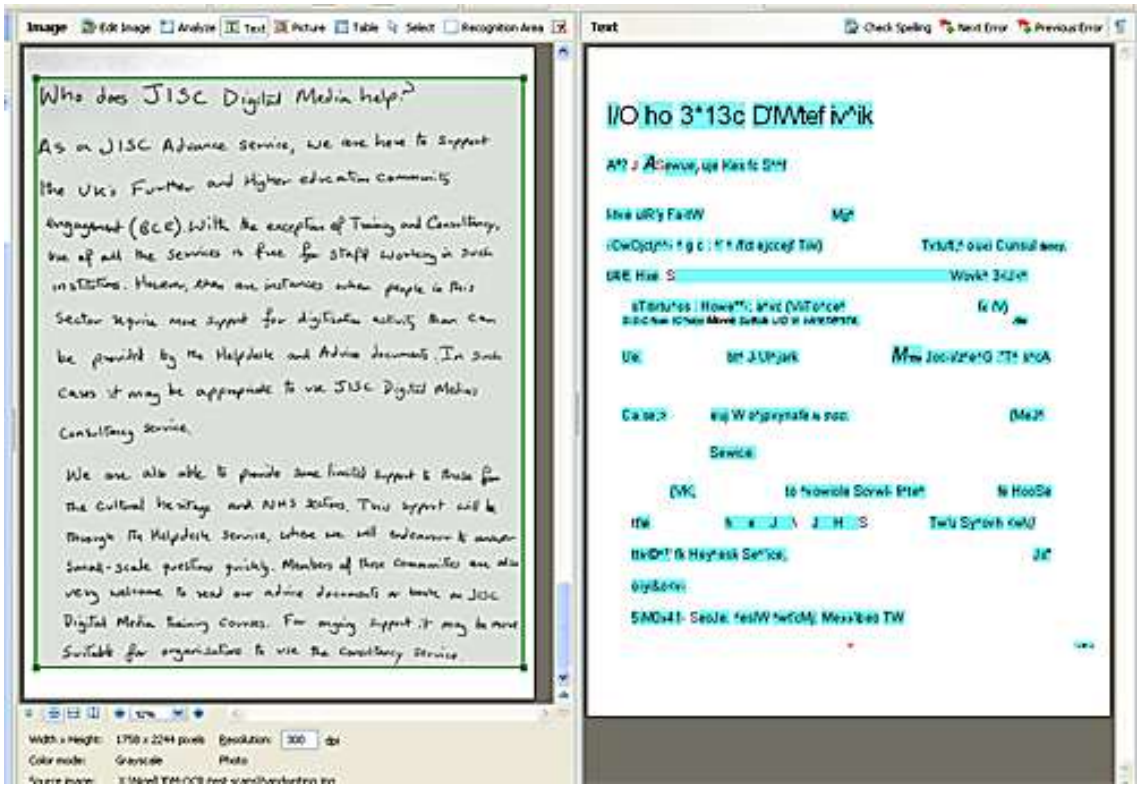
sei	otto	dieci	dodici	quattordici
3	4	5	6	11

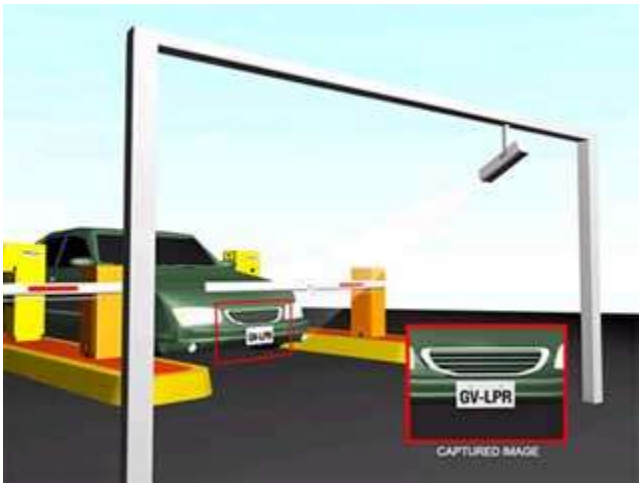
perchè 11?

OCR









- come riesce *GoogleStreetView* ad individuare un volto o una targa automobilistica in milioni di fotografie?

- Tecniche di *pattern matching* utilizzate in software di larghissima diffusione (presenti anche egli smartphone che abbiamo in tasca ad esempio per il riconoscimento facciale):

shazam.com

spotify.com

amazon.com

2.2.Esempi generici di applicazioni nel settore dell'ingegneria agroalimentare

- cosa è in grado di influenzare la crescita di una popolazione di microrganismi? Come si configura una relazione tra tali parametri e come la si determina a partire dalle osservazioni sperimentali?
  Es. la crescita di una colonia batterica è influenzata da diversi fattori: pH, Aw, T, t, ... : *UFC = f(pH, Attività W, Substrato, Temperatura, Tempo)*
- stimare l'umidità di un terreno sulla base dell'indice di cono;
- stimare lo stato di maturazione di un frutto sulla base di misure semplici e non distruttive (es. spettro di assorbimento VIS, NIR, EM);
  Le reti neurali (come altre tecniche di statistica multivariata) diventano fondamentali quando dobbiamo relazionare dati complessi come quelli spettrali (es: relazione tra spettro NIR e indici di maturità)
- stimare le caratteristiche di un prodotto sulla base dei parametri di processo;
- analisi ottica su cernitrici VIS/NIR (partic. individuazione difetti);
- verifica dell'autenticità di un prodotto;
- correlare la produzione di cereale sulla base di differenti regimi irrigui e di fertilizzazione
- dati congiunturali di mercato e dati di vendita
- **http://neuroph.sourceforge.net/tutorials/wines1/WineClassificationUsingNeuralNetworks.html**
- **https://www.sciencedirect.com/search?qs=neural%20networks&pub=Journal%20of%20Food%20Engineering&show=25&sortBy=relevance&origin=jrnl_home&zone=search&publicationTitles=271146**

2.3.Esempi specifici di applicazioni nel settore dell'ingegneria agroalimentare sviluppati in ambito DISTAL

- ***Es**: stima dell'età dell'uovo con metodi dielettrici*
- ***Es**: classificazione Cv albicocche*
  http://www.sciencedirect.com/science/article/pii/S1537511007000724
- ***Es**: olio d'oliva e provenienza pistacchi (con NIR)*
- ***Es**: determinazione aderenza ai disciplinari di produzione prod. caseari*
  http://www.sciencedirect.com/science/article/pii/S0308814611008296
- ***Es**: diagnostica su produzione coni;*

2.4.Storia minima delle reti neurali artificiali

L'idea di rete neurale nasce dal desiderio di riprodurre alcuni aspetti del comportamento umano. Ciò è stato fatto cominciando a riprodurre il funzionamento di una cellula del sistema nervoso.

Anni '40: nascono nell'ambito della neurobiologia come strumenti per lo studio dei meccanismi che governano il sistema nervoso biologico.

- 1943 W.S. McCulloch e Walter Pitts *A logical calculus of the ideas immanent in nervous activity* (rispettivamente un neurologo ed un matematico)

http://www.cse.chalmers.se/~coquand/AUTOMATA/mcp.pdf
https://en.wikipedia.org/wiki/Walter_Pitts
https://en.wikipedia.org/wiki/Warren_Sturgis_McCulloch

- 1949 D. O. Hebb *The organization of behavior* (idea di auto-apprendimento)

https://en.wikipedia.org/wiki/Organization_of_Behavior
http://s-f-walker.org.uk/pubsebooks/pdfs/The_Organization_of_Behavior-Donald_O._Hebb.pdf

Anni '50: con l'avvento dei primi calcolatori, vengono riprese nell'ambito dei primi studi sull'intelligenza artificiale → risultati deludenti per la scarsa potenza di calcolo dell'epoca e limitazioni teoriche dei primi modelli.

- 1958 J. Von Neumann *The computer and the brain*
- 1958 F. Rosenblatt *Phychological review*

AI '70s: http://www.imdb.com/title/tt0062622/quotes
imitazione di funzioni specificamente umane: interpretazione di suoni ed immagini, produzione del parlato; controllo del movimento ed anche aspetti del pensiero legato alla comprensione del linguaggio, organizzazione della conoscenza, istituzione di relazioni.

Anni 80: creazione di modelli più complessi → prime applicazioni di successo in contesti reali; prime interpretazioni delle reti neurali come modelli statistici.

2.5.Approfondimenti bibliografici

- http://it.wikipedia.org/wiki/Rete_neurale
- http://en.wikipedia.org/wiki/Artificial_neural_network
- http://www.obitko.com/tutorials/
- http://www.willamette.edu/~gorr/classes/cs449/intro.html
- http://www.statsoft.com/textbook/neural-networks/
- https://www.taylorfrancis.com/search?key=neural%20network
- https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/
- http://neuralnetworksanddeeplearning.com/chap3.html
- http://natureofcode.com/book/chapter-10-neural-networks/
- https://www.cheshireeng.com/Neuralyst/nnbg.htm
- http://www.saedsayad.com/artificial_neural_network.htm
- https://www.academia.edu/25708860/A_Concise_Introduction_to_Machine_Learning_with_Artificial_Neural_Networks
- https://www.youtube.com/watch?v=q0pm3BrIUFo
- https://deeplearning4j.org/neuralnet-overview.html
- http://www.dkriesel.com/en/science/neural_networks
- http://www.ijcte.org/papers/328-L318.pdf
- **http://dx.doi.org/10.4236/jst.2015.51004**
- **http://dx.doi.org/10.1016/j.aca.2011.06.033**

3.Fenomenologia del cervello e struttura del neurone biologico

Da un punto di vista globale, il ragionamento umano non si basa solo su sillogismi aristotelici ma procede anche per imitazione, giunge a conclusioni non solo deducendo ma anche associando, ricordando oggetti simili. La logica della natura non è bivalente ma sfumata, molto più sintetica che analitica, quindi non è strettamente riproducibile con ragionamenti basati sulla logica classica.
L'allievo musicista ripete i gesti, pur non conoscendo esattamente il modello bio-meccanico che lo porta a certi risultati, sviluppa dopo un lungo allenamento la capacità sia di ripetere le stesse azioni del maestro sia di adattarsi in maniera corretta alle nuove situazioni. Così come nel lancio di un sasso verso un certo obiettivo ci avvicineremo a esso per successive approssimazioni: un lancio teso con una data forza, distendendo il braccio parzialmente, un lancio parabolico e così via finché non otterremo il risultato desiderato.

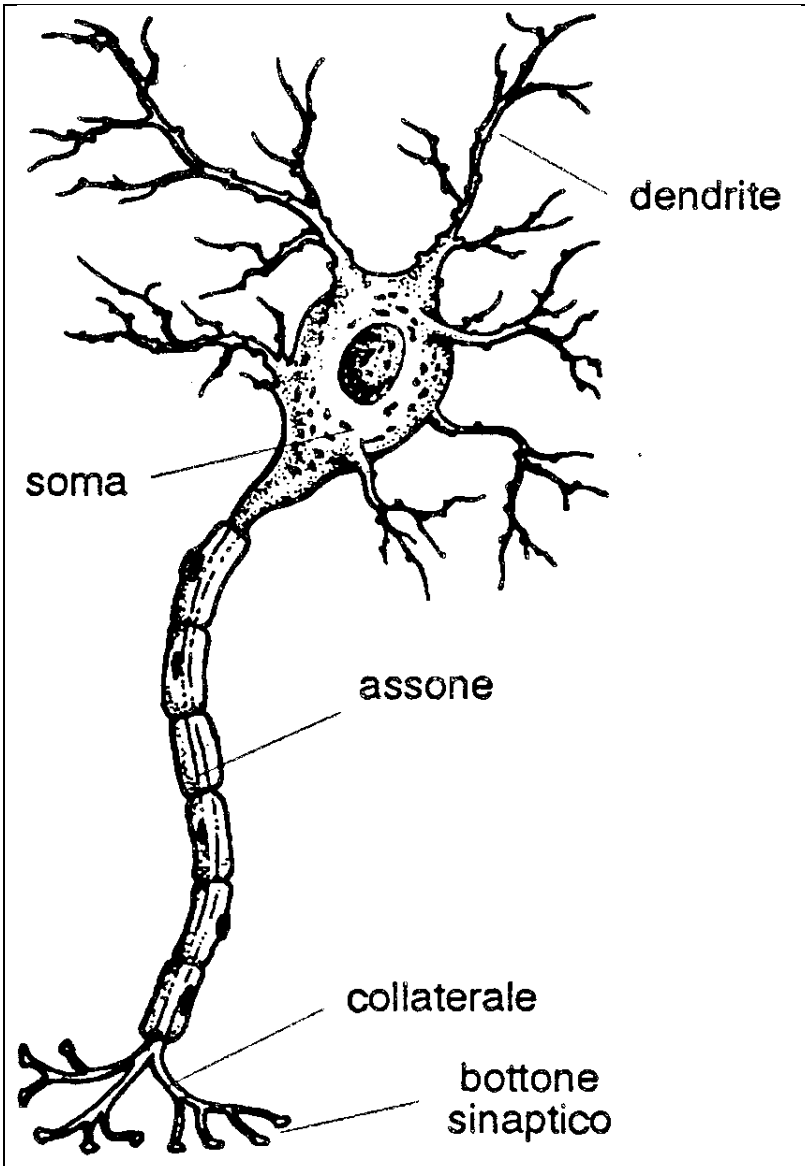
Negli anni '80 sono state sviluppate procedure automatiche per la ricerca di elementi di correlazione tra gruppi di osservazioni, ispirate alla logica ed alla biologia del sistema nervoso, a cui è stato dato il nome di *reti neurali artificiali*. Esse sono dei semplici modelli del sistema nervoso centrale e, come questo, sono composte da unità elementari, i neuroni, connessi tra loro mediante legami funzionali.

Unico aspetto comune tra il cervello e i computer è la presenza di un flusso informativo orientato tra input ed output: a grandi linee gli input del cervello sono le terminazioni nervose poste negli organi sensoriali mentre gli output sono gli organi motori.

A differenza del procedimento informatico classico in cui il programmatore dispone di un'accurata conoscenza del fenomeno oggetto di studio e costruisce un algoritmo per ogni singolo problema, nelle reti neurali non vi è la progettazione di un classico algoritmo ma l'adattamento del modello generale alle condizioni dell'ambiente in cui si opera.

A differenza dei computer tradizionali in cui la memorizzazione e l'elaborazione dei dati sono separate (RAM e CPU), nelle reti neurali, a somiglianza del cervello umano, tali operazioni vengono fatte contemporaneamente da tutte le unità; i singoli neuroni sono quindi detentori di informazioni e al contempo operano la loro elaborazione. In altri termini mentre il modo di operare dei computer è sequenziale, quello delle reti è intrinsecamente parallelo (e.g. visione).

La ricerca nel campo delle reti neurali è stata molto influenzata dagli studi sulla struttura biologica del sistema nervoso centrale. Pur rimanendo ancora lontani da una conoscenza completa, negli ultimi anni sono stati appresi molti fondamentali aspetti del funzionamento del cervello.



La cellula neuronale

Unità fondamentale del cervello (qualche μm÷1m). Ogni neurone riceve come input i segnali elettrici da tutti i dendriti e, se la loro composizione supera una soglia di attivazione, viene emesso un impulso elettrico in uscita verso l'assone.

Input
↓
elaborazione
↓
output

Il numero di neuroni che compongono il cervello umano è dell'ordine di 10^{10} (sostanzialmente concentrati nella corteccia), corrispondenti ad una capacità di memorizzazione pari a circa 10^{16} bit (≈un milione di GB), pesa appena un chilo e trova comodamente posto nello spazio di 1 dm^3, con un consumo energetico di circa 10W. Ogni neurone riceve stimoli elettrochimici e reagisce generando a sua volta impulsi elettrici che trasmette ad altri neuroni. Ogni neurone può avere fino ad alcune migliaia di dendriti. Un cervello può contenere 10^{14} (0.1 milioni di miliardi) connessioni sinaptiche.

Il corpo cellulare è rivestito da una membrana che ha la capacità di mantenere una polarizzazione elettrica. Il neurone riceve attraverso i vari dendriti delle cariche elettriche che polarizzano la membrana. Se l'effetto cumulativo di questi input supera un certo livello di soglia si verificherà l'eccitamento del corpo cellulare che scarica, mediante l'assone connesso con un dendrite di un altro neurone, un segnale elettrico che, a secondo dei casi, sarà eccitatorio o inibitorio (la composizione dei segnali è assimilabile ad una somma algebrica).

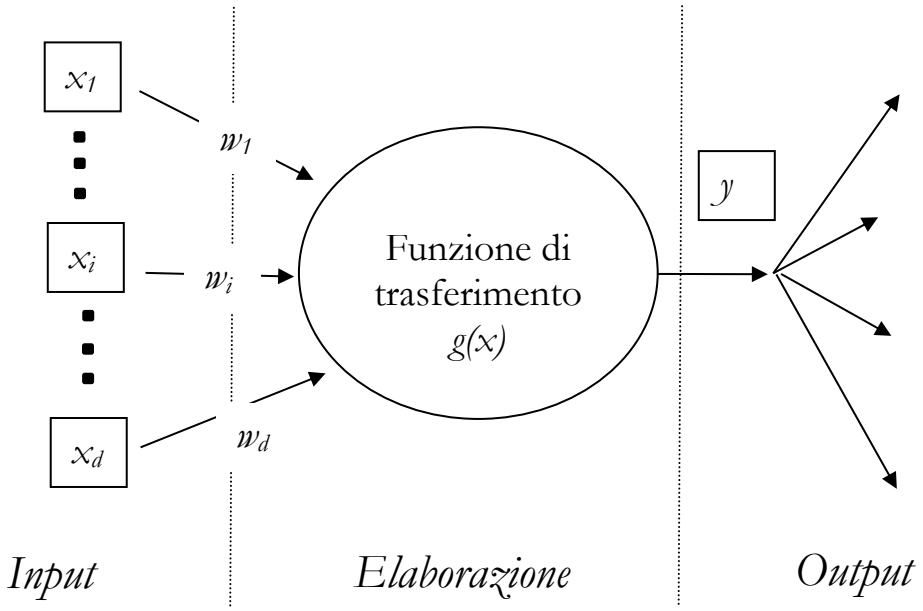
Un neurone può attivarsi cento volte al secondo, quindi il cervello umano può eseguire $100*10^{14}=10^{16}$ attività sinaptiche al secondo (approssimando ad una macchina seriale corrisponderebbe ad un clock di 10^7 GHz).

4.Comportamento del neurone artificiale (processing unit, perceptron, trigger, gain)

Una rete neurale artificiale è formata da vari neuroni connessi tra loro (con riferimento a numeri assai più piccoli di quelli che abbiamo appena visto). Ogni neurone, detto *artificiale* per distinguerlo da quello biologico, è costituito da unità di input e di output, che hanno le stesse funzioni dei dendriti e dall'assone del neurone biologico.

- McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115 - 133.

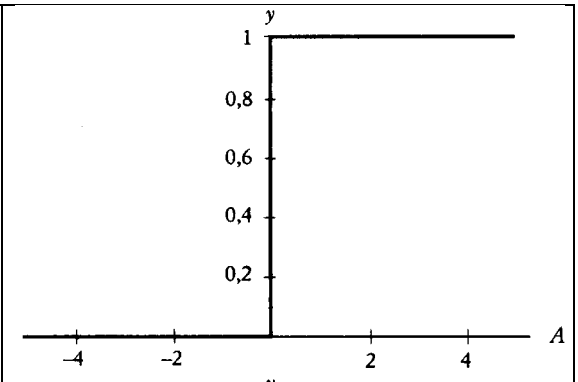
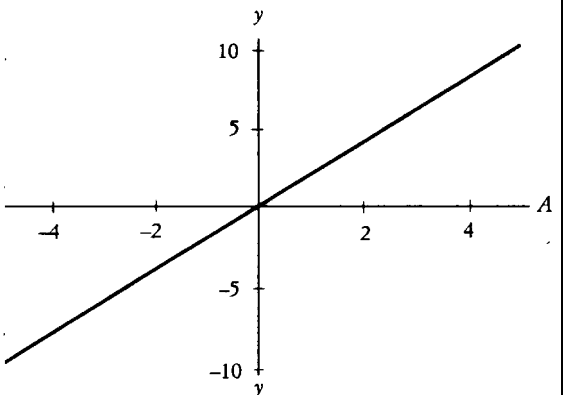
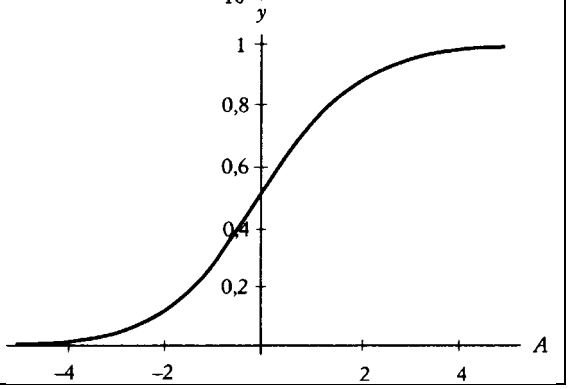
Un generico neurone riceverà (spesso denominato *percettrone*) da ciascuno dei *d* neuroni con cui è connesso un segnale di input x_i (espresso da un valore numerico). Ognuno di questi segnali verrà trasmesso, moltiplicato dalle sinapsi con opportuni pesi *Wi* (dove *Wi* è un numero reale (positivo/negativo => eccitazione/inibizione; maggiore/minore di 1 => amplificazione/attenuazione)):

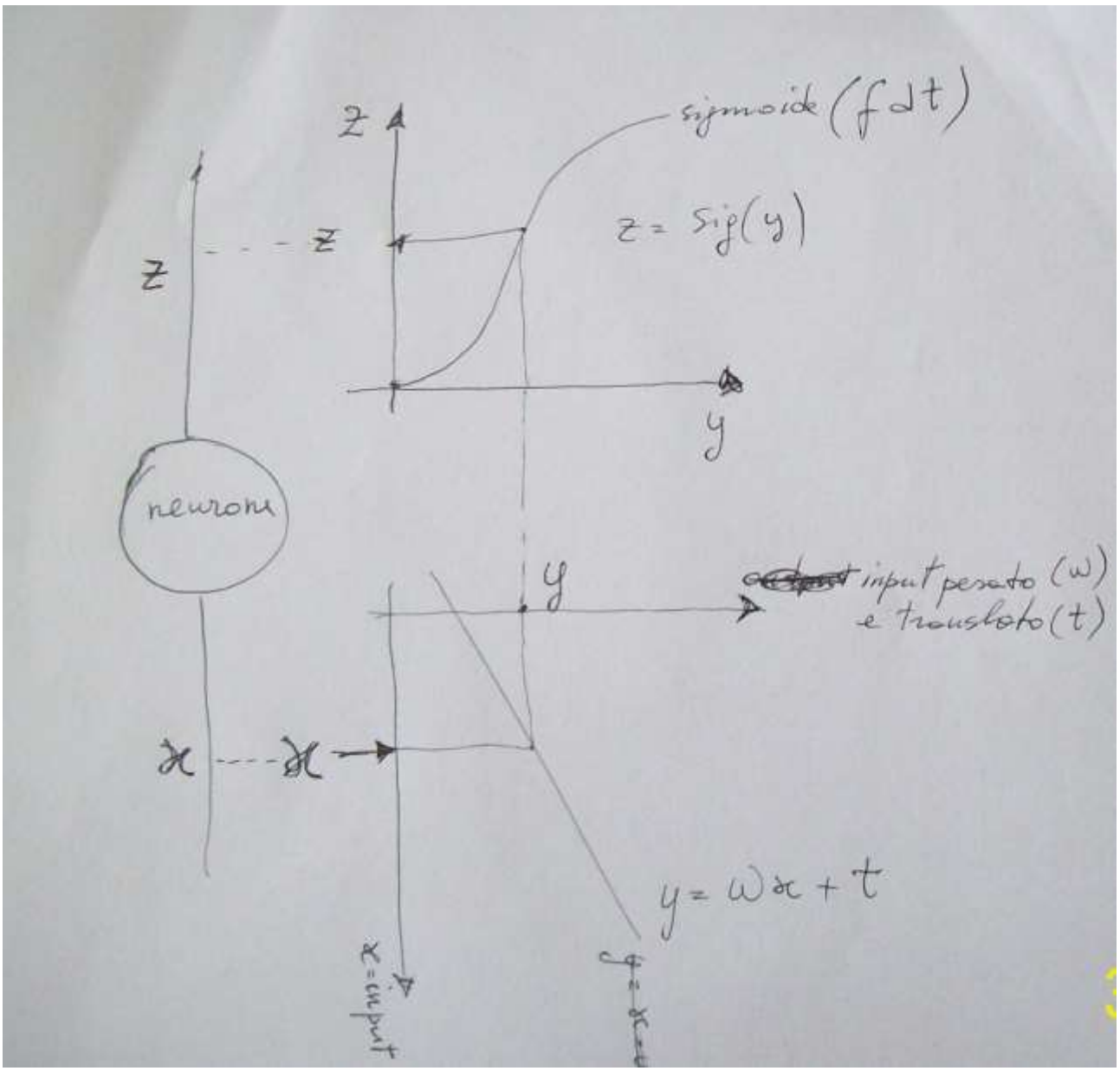


Il neurone produrrà un valore di output *y* elaborando, attraverso una propria *funzione di trasferimento g*, la somma pesata dei segnali di input:

$$y(x) = g\left(\sum_{i=1}^{d} w_i x_i + w_0\right) = \overline{w}^T \overline{x}$$

Come è fatta la funzione di trasferimento?

A gradino	$g(A)=\begin{cases}1 \ se \ A>0\\0 \ altrimenti\end{cases}$	
Lineare continua	$g(A)=kA$	
Sigmoide o logistica	$g(A)=\frac{1}{1+e^{-A}}$ o hTangent	



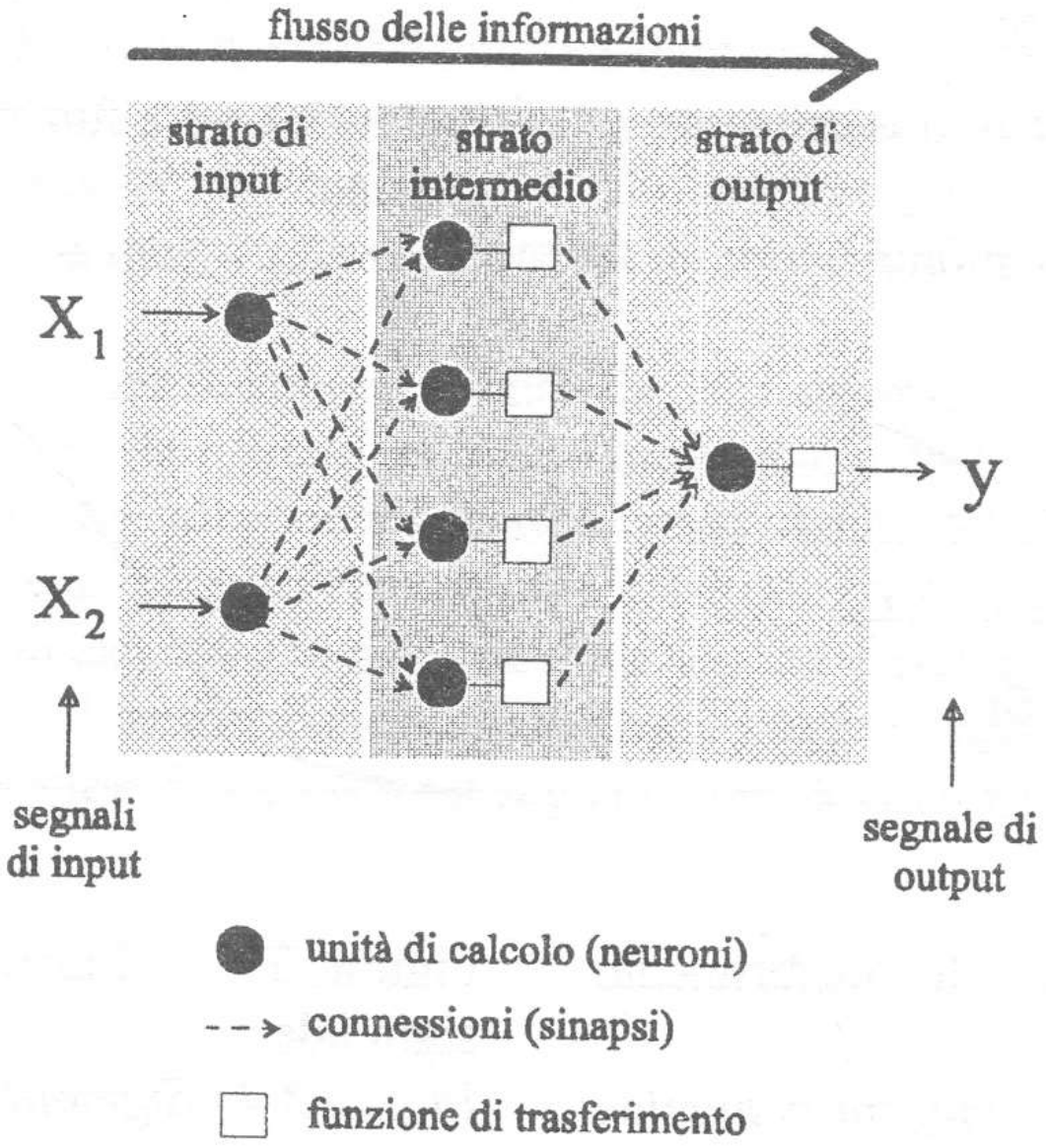
(disegno originale di P.Liberati)

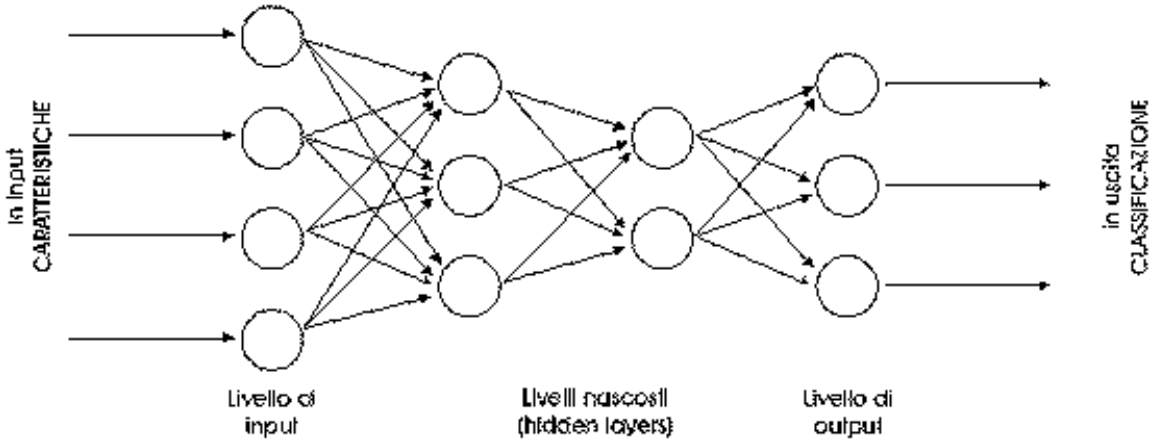
$$z=sigmoid\_function(w\cdot x+t)$$

5.Struttura delle reti neurali

Ora possiamo definire un po' meglio le ANN, come sistemi di elaborazione costituiti da molte unità di calcolo (neuroni), ciascuna dotata di una piccola quantità di memoria, connesse tra loro da canali di comunicazione (sinapsi) che trasportano dati numerici.

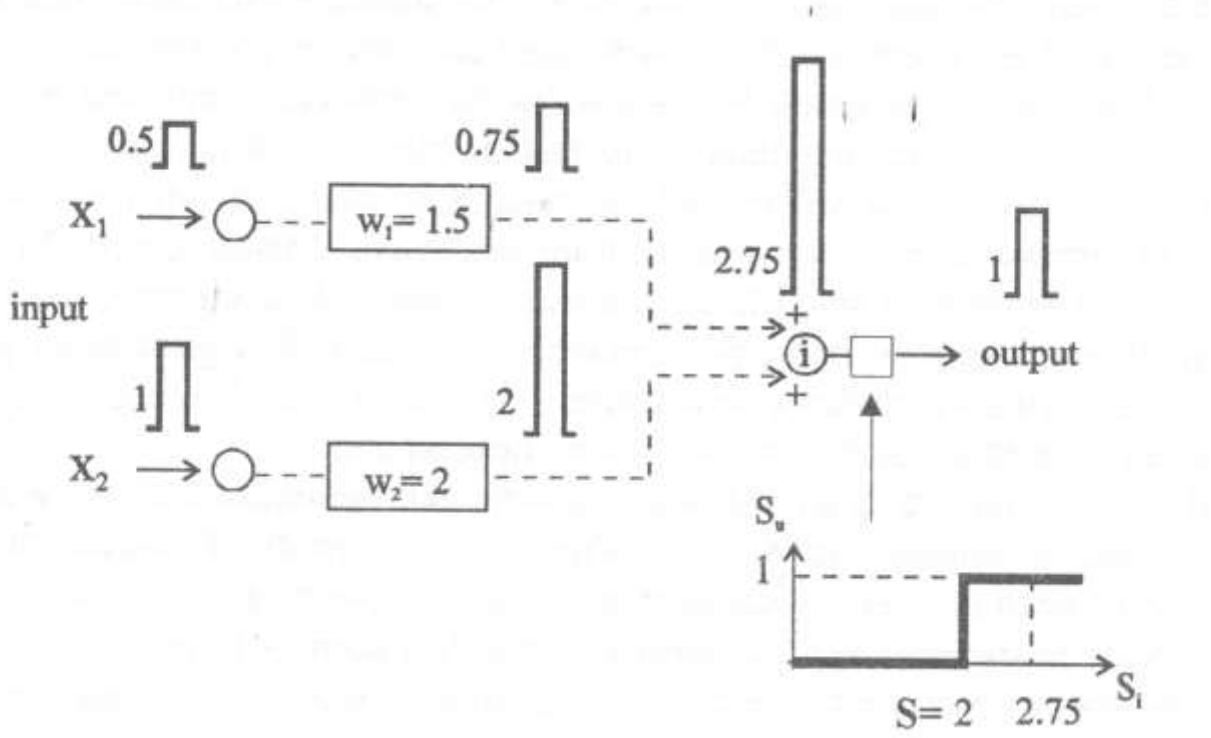
I neuroni sono generalmente organizzati in strati, dei quali alcuni sono preposti a ricevere segnali dall'esterno (neuroni dello strato di input), altri hanno esclusivamente compiti di elaborazione (neuroni degli strati intermedi o nascosti), altri ancora hanno il compito di fornire all'esterno (neuroni dello strato di output) i valori di una data grandezza funzione delle grandezze in ingresso.





Le reti neurali hanno una struttura altamente distribuita e, secondo uno schema assai consueto, ogni neurone di uno strato è collegato a tutti i neuroni dello strato successivo. Con una tale struttura l'informazione viene spezzettata e memorizzata in tante unità separate e, contrariamente ai sistemi tradizionali in cui l'elaborazione è sequenziale, nelle reti neurali le informazioni possono essere elaborate in parallelo.

Come si è detto nel precedente paragrafo le connessioni fra i neuroni hanno una propria forza, chiamata peso. L'esempio numerico riportato nella figura che segue può servire a spiegare quanto detto fin qui. Si abbia una rete neurale a due unità di input. Pervengono dall'esterno due segnali che hanno valore 0,5 e 1. I pesi delle connessioni siano rispettivamente 1,5 e 2, per cui il segnale in ingresso al neurone sarà costituito dalla somma dei due segnali pesati (0,5*1,5 + 1*2 = 2,75). Poiché al neurone arriva un segnale superiore a 2, soglia stabilita dalla specifica funzione di trasferimento, esso si attiva e invia all'uscita un segnale unitario, cioè della massima intensità.



Nel caso della rete a due neuroni esempio, oltre che dagli input e dai pesi delle connessioni, l'output è determinato evidentemente anche da una funzione di trasferimento, g():

$$y = g(b_0 + b_1 x_1 + b_2 x_2)$$

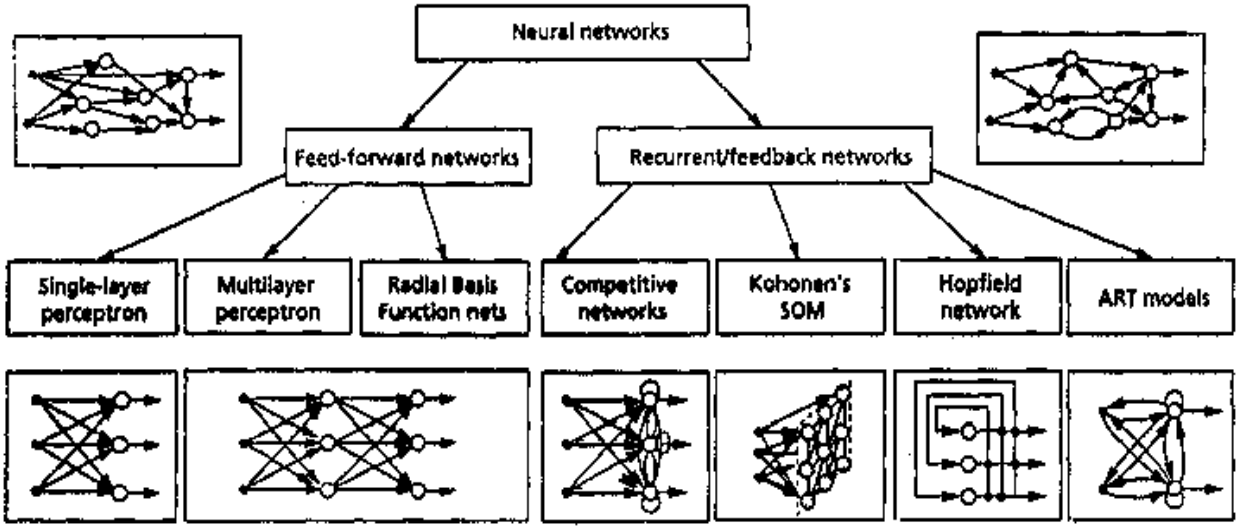
Più in generale, l'output di una rete neurale con *n* unità di input, un solo strato intermedio composto da *h* neuroni ed una singola unità di uscita, anch'essa dotata di una funzione di trasferimento, sarà determinato dalla relazione:

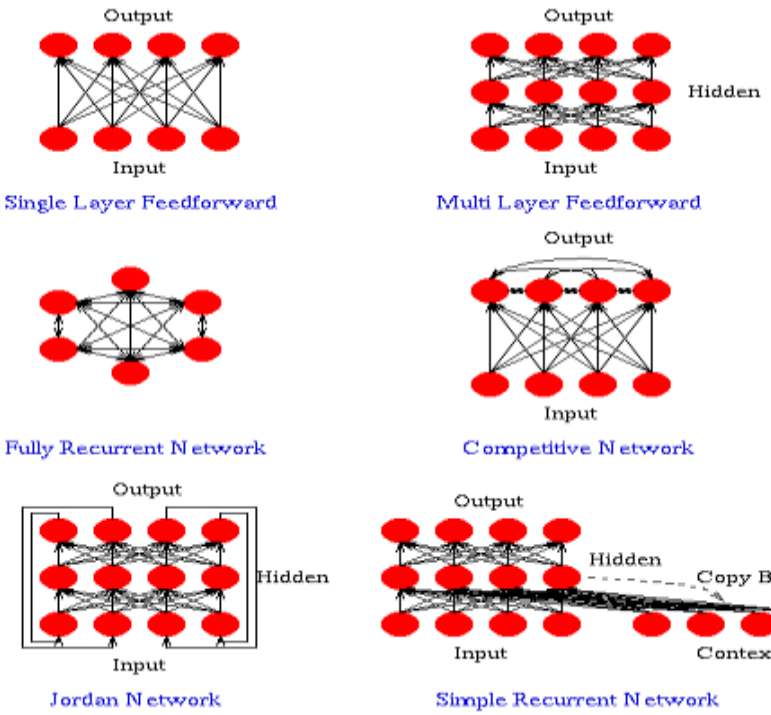
$$y = g\left(\sum_{j=0}^{h} \beta_j \cdot f\left(\sum_{i=0}^{n} \gamma_{ji} x_i\right)\right)$$

in cui:
g, funzione di trasferimento del neurone dello strato di output;
h, numero di neuroni nello strato intermedio;

β_j, pesi delle connessioni tra le unità dello strato intermedio e quelle dello strato di output;
f, funzione di trasferimento dei neuroni dello strato intermedio;
n, numero di unità di input; γ_{ji}, pesi delle connessioni tra le unità dello strato di input e quelle dello strato intermedio;
xj, valori assunti dagli input della rete.

Esistono molte tipologie di reti neurali, ma in particolare si concentra qui l'attenzione sul modello appena visto, detto MLP (multilayer perceptron):





6.Training di una rete (come si determinano i coefficienti w_i?)

6.1.Paradigmi a confronto

Programmazione:
Esempio: per riconoscere una certa varietà frutticola a partire da una serie di immagini bisogna scrivere un programma con struttura IF…THEN che include tutti i casi possibili, quindi bisogna in anticipo prevederli tutti:

IF ha ***colore*** THEN
  IF ha ***forma*** THEN
    IF ha ***dimensioni*** THEN
    ……
  PRINT "è una mela ***golden delicious***"

Apprendimento:
Apprendere significa migliorare la capacità di esecuzione di un certo compito attraverso l'esperienza.

Esempio: all'inizio la rete neurale non conosce il concetto di *golden delicious*. Si presentano alla rete tanti esempi di mela, discriminando il tipo che ci interessa. La rete neurale impara e si crea un'esperienza sul riconoscimento.

6.2.Allenamento di una rete (tutta la conoscenza di una rete è concentrata nei suoi fattori di pesatura!)
Per quanto detto precedentemente, fissata l'architettura di rete e le funzioni di trasferimento dei neuroni, il valore dell'output dipende solo dai valori dei pesi, ma ... qual'è il valore dei pesi?

Inizialmente vengono scelti in modo casuale, e successivamente vengono modificati durante una fase di apprendimento (*training*). Questa fase è fondamentale per le reti neurali che, analogamente ai sistemi biologici, devono apprendere le caratteristiche essenziali di un problema prima di formulare una previsione.

Nelle reti neurali l'apprendimento consiste nel modificare il peso di ciascuna connessione fino a individuare l'insieme dei pesi che consente alla rete di ottenere le migliori prestazioni. In molte reti neurali (reti ad apprendimento con supervisione) ciò avviene fornendo alla rete un adeguato numero di esempi che costituiscono l'input ed i corrispondenti valori desiderati di output.

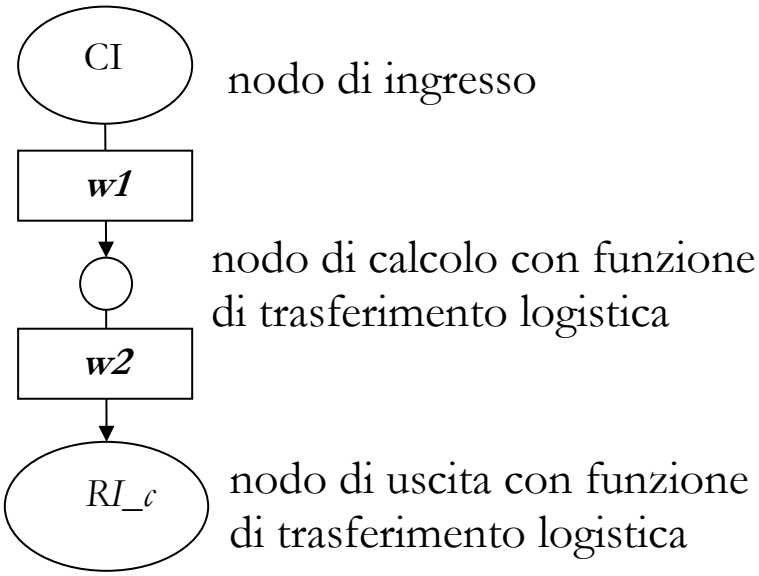
Per ogni esempio di dati la rete propaga l'input al suo interno e, in base ai valori dei pesi, calcola un valore di output; se la differenza fra l'output calcolato e quello fornito dall'esempio è troppo elevata si modificano i pesi in modo da minimizzare la somma degli errori commessi. Esistono vari algoritmi (BackPropagation, Quickprop, ADALINE, Radial Basis Function, ecc.) attraverso i quali, con la reiterata modificazione dei pesi, si riesce a minimizzare l'errore complessivo *E(w)*.

Come **primo esempio** consideriamo un caso semplicissimo (proprio per la semplicità dello schema, una ANN non è pienamente giustificata, ma ignoriamo il fatto in favore della immediata comprensibilità del caso) relativo alla relazione tra grado zuccherino e colore di una certa varietà frutticola.

L'indice di colore *CI* potrebbe essere rappresentato dal rapporto tra la componente rossa e l'intensità totale, mentre il grado zuccherino potrebbe essere espresso dal rapporto *RI* tra il valore attuale (misurato ad esempio in gradi Brix) ed il valore ideale corrispondente al frutto perfettamente maturo.
Si realizzano dunque una serie di misure (distruttive) su frutti a differente grado di maturazione:

ColorIndex	**RipenessIndex_m**
0,96	0,98
0,56	0,75
0,81	0,90
0,09	0,30
0,95	0,97
0,97	0,99
0,38	0,62
...	...

Immaginiamo di voler cercare un legame tra le variabili *CI* ed *RIm* (dove il pedice *m* ricorda che si tratta di valori misurati) attraverso una semplicissima rete neurale. Ad esempio quella di figura, composta da un solo neurone di ingresso, uno di calcolo ed uno di uscita, nella quale sono evidenziati i valori di input/output, i pesi delle connessioni (*w*):



Con riferimento ad una funzione di trasferimento logistica, tale schema concettuale può essere rappresentato analiticamente dall'espressione:

$$RI\_c = \text{Logistica}[\, w2 * \text{Logistica}(w1 * CI + t1)\,]$$

usando il pedice *c* per ricordare che si tratta di valori calcolati.

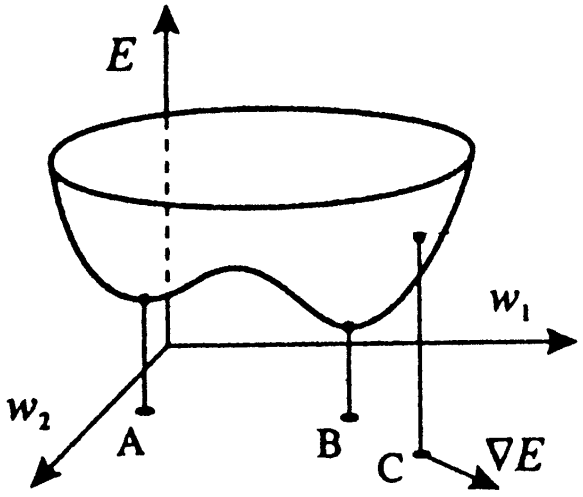
Dunque, definiti (in qualche modo, anche casualmente) i valori dei pesi *w1* e *w2*, è possibile compilare una tabella del tipo:

CI	***RIm***	***RIc***	***SE = (RIc - RIm)²***
0,26	0,51	0,65	0,020
0,58	0,76	0,82	0,003
0,50	0,71	0,62	0,008
0,02	0,12	0,39	0,069
0,37	0,61	0,66	0,003
0,71	0,84	0,82	0,000
0,54	0,74	0,51	0,053
...	...	...	...

La somma di tutti gli errori quadratici ($SSE=\Sigma SE$) costituisce un indice di distanza complessivo tra i dati misurati e quelli stimati attraverso la rete neurale, analogamente al noto criterio dei minimi quadrati.

Al variare dei parametri *w1* e *w2* evidentemente otterremo differenti valori della somma degli errori quadratici *SSE*.
Di conseguenza, assegnati determinati dati misurati, è possibile definire una funzione *SSE(w1, w2)* che, a titolo di esempio, potrebbe avere l'aspetto di figura:

$$E(w) = \sum_{i=1}^{c} \left[x(w)_i^{calc} - x(w)_i^{obs} \right]^2$$



E' dunque immediato capire come vengono calcolati i valori dei coefficienti di pesatura: semplicemente ricercando le condizioni di minimo errore, ovvero di migliore adattabilità della rete ai dati sperimentali.
Per fare ciò si utilizzano molti metodi differenti, come ad esempio quelli basati sul'inseguimento del gradiente (es. metodi di Levenberg-Marquardt, Backtracking).

Un tale criterio per la determinazione dei coefficienti w_i è estendibile a reti di complessità qualsiasi. Semplicemente la ricerca delle condizioni di minimo dovrà essere svolta su uno spazio pluridimensionale, piuttosto che in quello 3D. Ciò non comporta, dal punto di vista algoritmico, alcuna difficoltà supplementare.

Possiamo considerare un **secondo esempio**, ancora semplicissimo ma con qualche dettaglio in più, relativo alla pastorizzazione di un fluido alimentare: la carica batterica residua dipenda da due soli parametri: tempo e temperatura del trattamento.

Si organizza dunque un piano sperimentale di questo tipo:

temperatura, *T*: 60-90°C su 4 livelli (60, 70, 80, 90)
tempo, *t*: 1-5 minuti su 5 livelli (1, 2, 3, 4, 5)

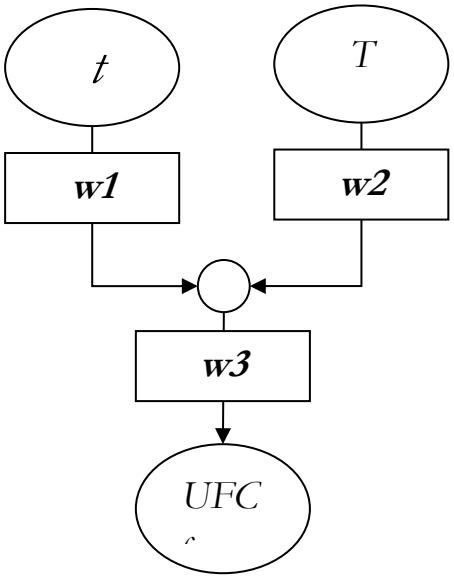
T1&t1, T1&t2, T1&t3, T1&t4, T1&t5
T2&t1, T2&t2, T2&t3, T2&t4, T2&t5
T3&t1, T3&t2, T3&t3, T3&t4, T3&t5
T4&t1, T4&t2, T4&t3, T4&t4, T4&t5

4 x 5 = 20 combinazioni

considerando 10 repliche per ciascuna tesi, abbiamo 10x4x5=200 record, che potremo organizzare in una tabella del tipo (UFCm: centinaia di unità formanti colonia misurato per unità d'area sul prodotto lavorato):

t	***T***	***UFCm***
1	60	UFCm(2,80)
1	70	UFCm(2,70)
1	80	UFCm(2,60)
1	90	UFCm(1,90)
2	60	UFCm(1,80)
2	70	UFCm(1,70)
2	80	UFCm(1,60)
...	...	...

Immaginiamo di voler cercare un legame tra le variabili *t, T* e *UFC* attraverso una semplicissima rete neurale. Ad esempio quella di figura, composta da un solo neurone di ingresso, uno di calcolo ed uno di uscita, nella quale sono evidenziati i valori di input/output, i pesi delle connessioni (*w*), i valori di soglia (*t*), spesso utilizzati in aggiunta ai semplici coefficienti di pesatura, in grado di dare molta adattabilità in più ad una rete:



UFCc = Logistica[*w3**Logistica(*w1***t* + *w2***T* + *t1*) + *t2*]
ricordando che la funzione Logistica(*x*) equivale a *1/(1+exp(-x))*

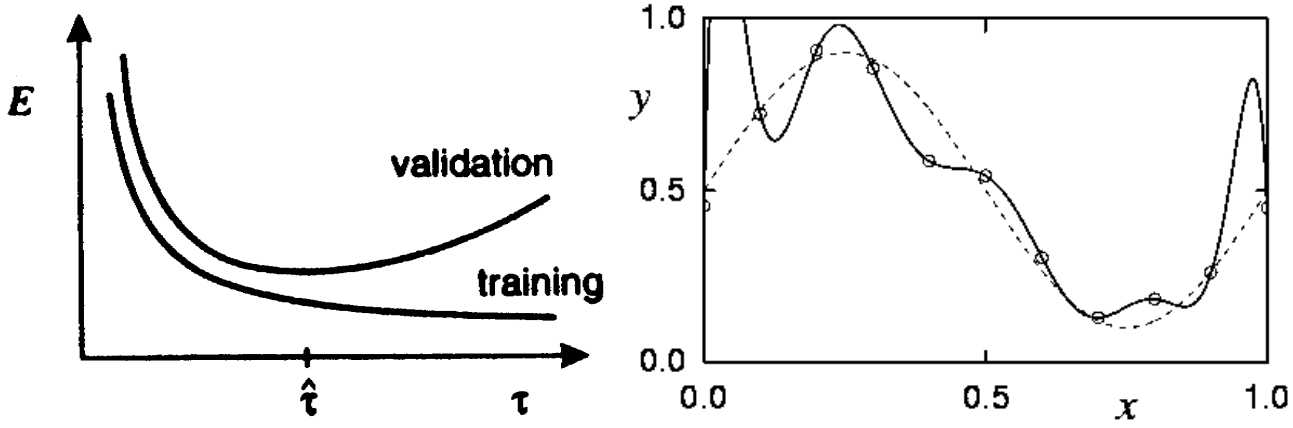
6.3. Overtraining e generalizzazione

Durante la fase di apprendimento la rete calcola i valori degli output e li confronta con quelli forniti nel set di dati (*training-dataset o learning-set*), calcolando il valore di *SSE*, ovvero il principale indicatore delle prestazioni della rete e modificando continuamente i coefficienti di pesatura.

Tuttavia l'SSE viene anche valutato mediante un secondo set di dati (*test-set o validation-set*), differente da quello utilizzato per la fase di apprendimento. Generalmente si utilizza come test set il 20-40% del totale dei dati disponibili. La determinazione dell'errore commesso sul test-set ci consente di valutare la capacità che avrà la rete di prevedere correttamente i valori di output quando gli verranno somministrati valori di ingresso diversi (molto o poco) da quelli utilizzati per l'allenamento.

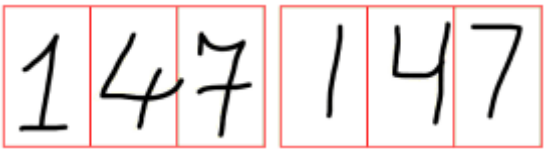
In altre parole possiamo valutare la capacità della rete di adattare l'esperienza acquisita alle nuove condizioni sperimentali; questa proprietà viene chiamata *generalizzazione*.

Una lunga durata della fase di istruzione consente alla rete di stimare bene tutti gli output presenti nel set di dati, ma può indurre una bassa capacità di generalizzazione su nuovi dati: questa patologia si chiama *overtraimng*, ovvero la rete troppo allenata può tendere a ricordare rigidamente gli schemi visti in ingresso. Per questa ragione si considera conclusa la fase di istruzione quando l'errore sul set di test raggiunge il minimo; se si procedesse oltre, l'errore potrebbe riprendere a salire.



training –set: utilizzato per calcolare i valori dei coefficienti di pesatura durante l'apprendimento;
test-set: utilizzato per verificare la bontà della risposta della rete, durante le iterazioni di apprendimento.

Es. una rete ben allenata nel riconoscimento ottico dei caratteri potrebbe utilizzare la propria capacità di generalizzazione per gestire il caso di figura:



viceversa una rete sovra-allenata potrebbe riuscire ad individuare con grande sicurezza le cifre del primo tipo ma non riuscire a capire l'analogia con quelle del secondo.

Similmente una rete in overtraining, allenata nell'esecuzione di somme, potrebbe presentare risultati esatti quando in input si presentano combinazioni numeriche esattamente uguali a quelle utilizzate durante l'allenamento, ma potrebbe fornire risultati completamente sbagliati nel caso di valori di ingresso anche di poco diversi da quelli considerati durante il training.

7.Vantaggi e limiti applicativi delle reti neurali

Sviluppare un modello significa determinare una serie di relazioni matematiche in grado di stimare i rapporti tra le variabili che definiscono lo stato di un sistema fisico. E' spesso difficile ricavare un'espressione matematica che sintetizzi il comportamento del sistema, specie quando esiste una dipendenza non lineare dai parametri considerati. Spesso ci si affida a modelli empirici basati su regressioni che non sempre, però, soddisfano pienamente i dati sperimentali e, peggio ancora, per valori delle grandezze al di fuori dei valori sperimentati, indicano comportamenti della grandezza calcolata che spesso non trovano un riscontro sperimentale.

L'uso delle reti neurali non determina sostanziali vantaggi nei fenomeni lineari, invece quando i fenomeni non sono lineari e coinvolgono un gran numero di input, le reti neurali consentono di evitare sia la scelta a priori di un modello sia la definizione di assunzioni teoriche (p.e. distribuzione delle varianze e degli errori). **Le reti neurali costruiscono autonomamente uno stimatore tendenzialmente ottimo in grado di associare ai valori di un vettore X di n-componenti i corrispondenti valori di un vettore Y di m-componenti.**

Un aspetto delicato nella progettazione di una rete neurale è la scelta del numero di ingressi. Un numero esiguo di parametri di input può risultare insufficiente alla rete per apprendere in maniera corretta, causando quindi una rappresentazione inadeguata del fenomeno in studio.
Sembrerebbe logico utilizzare come ingressi tutte le grandezze che influenzano il fenomeno, ma molti ingressi portano a reti che possano avere dimensioni tali da rendere insufficienti, per un corretto training, gli esempi a disposizione. Infatti, a parità di risoluzione in ingresso, il numero di pattern necessari per mantenere una buona capacità di apprendimento cresce in maniera esponenziale all'aumentare dei parametri di ingresso. Al crescere del numero di ingressi gli esempi disponibili diventano sempre più "sparsi", e quindi insufficienti per un efficace addestramento della rete neurale. Per fissare le idee supponiamo che il numero di ingressi della rete neurale sia 10, che la risoluzione con cui sono forniti i dati di ingresso sia 0.2, e che il range fisico dei valori di ingresso sia 1-30. In tal caso il numero di esempi corrispondenti a tutti i possibili pattern di ingresso risulta essere $(30/0{,}2)^{10}$.
Sebbene una tale stima sia notevolmente pessimistica, in quanto non tutte le combinazioni possibili sono fisicamente plausibili, il problema della dimensionalità resta e induce a ridurre il numero di ingressi, in particolare quando il numero di esempi disponibili non è molto grande.

Anche la scelta del **numero di strati e di neuroni** è arbitraria e lasciata a qualche tentativo. Un eccessivo numero di interconnessioni può portare a maggiore tempo

di calcolo, più difficile portabilità della rete e soprattutto bassa generalizzazione. Del resto un basso numero di neuroni può portare a bassi valori di correlazione tra input ed output.
Una buona procedura prevede di partire da una rete monostrato aumentando il numero di neuroni fino a quando non si osservano soddisfacenti valori di correlazione con il validation set.
In questa procedura ci aiuta il teorema di Hornik, Stinchombe e White (1989): qualunque tipo di funzione, relativa ad un qualunque numero di variabili indipendenti o dipendenti, può essere approssimata con qualunque precisione si desideri da un percettrone avente un solo strato nascosto costituito da unità sigmoidali, a patto di disporre del giusto numero di unità e di appropriati valori dei pesi delle connessioni:

- https://www.sciencedirect.com/science/article/pii/0893608089900208
- https://www.researchgate.net/post/How_to_decide_the_number_of_hidden_layers_and_nodes_in_a_hidden_layer
- http://it.mathworks.com/matlabcentral/answers/72654-how-to-decide-size-of-neural-network-like-number-of-neurons-in-a-hidden-layer-number-of-hidden-lay
- https://en.wikipedia.org/wiki/Universal_approximation_theorem

Un'altra proprietà delle reti neurali, assai importante quando si opera con dati di campo, è quella di essere in grado di operare, anche se con minore accuratezza, in presenza di dati incompleti, mancanti o affetti da errore. Questa proprietà, chiamata anche robustezza o resistenza al rumore, può essere spiegata con la struttura distribuita della rete. Quando l'informazione è incompleta o parzialmente errata, se essa risiede in più unità è possibile compensare queste lacune sfruttando la ridondanza dei centri di elaborazione. Per questa ragione la teoria classica della propagazione degli errori non può essere applicata nell'analisi delle prestazioni delle reti neurali. Fenomeni analoghi alla resistenza al rumore possono essere osservati anche nel mondo biologico: se una parte del cervello viene compromessa da un trauma, altre parti del cervello possono far recuperare, almeno parzialmente, le funzionalità dell'area danneggiata.

Le proprietà suddette evidenziano la validità delle tecniche basate sulle reti neurali, tuttavia occorre ricordare anche i limiti di queste tecniche: essi sono riconducibili essenzialmente all'incapacità delle reti neurali di esplicitare la forma del modello interno (relazione input/output).
Attualmente si ottengono informazioni sul legame input-output dall'analisi dell'effetto sull'uscita di variazioni prestabilite di una o più grandezze di input.

Punti di forza

- adatte anche a problemi che non chiedono risposte accurate, ma soluzioni orientative o addirittura solo qualitative;
- generalizzazione: producono buone risposte anche con input non considerati durante la creazione e l'addestramento;
- facili da implementare in un codice di calcolo;
- basso impegno computazionale e scalabilità;
- stabilità dell'output rispetto a valori di input incompleti o affetti da rumore;
- a differenza di un classico metodo interpolatorio, le ANN determinano sia il valore dei parametri che la forma di un modello.

Problemi

- incapacità di rendere conto dell'elaborazione: non è immediato esplicitare le relazioni tra variabili (*black-box,* ha senso utilizzare le ANN quando falliscono i tradizionali metodi di statistica multivariata);
- evidentemente non sempre esiste una rete che risolve il problema;
- funzionano male con pochi casi di esempio.

Un dubbio a questo punto può nascere: si tratta dunque di una *banale* media pesata?

- http://it.wikipedia.org/wiki/Serie_di_Taylor
- http://it.wikipedia.org/wiki/Serie_di_Fourier
- http://www.google.it/search?hl=it&client=firefox-a&rls=org.mozilla%3Aen-US%3Aofficial&hs=S5P&q=taylor+serie+applet&btnG=Cerca&meta=

h13-30 → http://www.cibuscesena.it/menu/

8.Pacchetti software

- Qnet (www.qnetv2k.com
http://www.qnetv2k.com/Qnet2000Manual/contents2000.htm)

- JavaNNS (http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS/welcome_e.html)

- http://neuroph.sourceforge.net/

- OpenStat (?)

- Statistica (http://www.statsoft.com/textbook/stneunet.html)

- Matlab Toolbox (http://www.mathworks.com/products/neuralnet/)
http://www.unibo.it/en/services-and-opportunities/studying-and-beyond/discounts-for-computer-tablet-and-software-1/matlab

-
http://grey.colorado.edu/emergent/index.php/Comparison_of_Neural_Network_Simulators

- http://en.wikipedia.org/wiki/Neural_network_software

- http://statpages.info/

- JustNN http://www.justnn.com/

- https://www.cs.waikato.ac.nz/ml/weka/

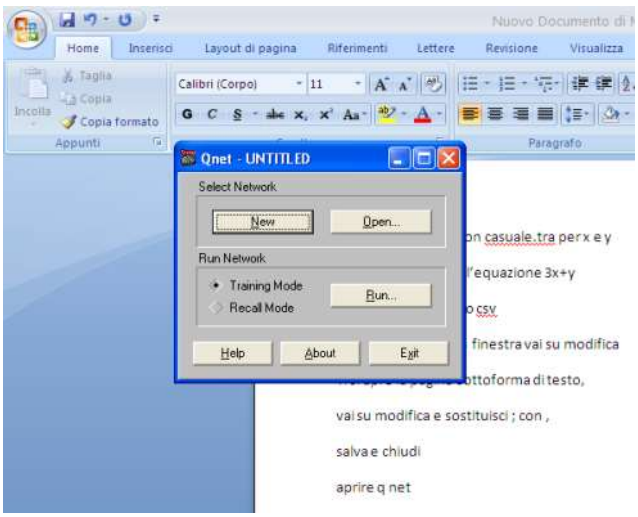
9.Esercitazioni

9.1.Esempio 1 – relazioni algebriche - file algebra.xls

algebra.xls circa 1200 record su quattro colonne (first row #) =CASUALE.TRA(0;99) A) 3*x+1*y B) $3x+y^2$ C) $3x+2y^2+z^3$ D) x+3*x*y+*2*y^z E) 1/(x+3*y) Test 1, 2, 4, 16 PE; even multilayer; show overtraining with high PE number; comma CSV	***Netgraph:*** RMS errorhistory (test+train data); correlation; targets vs. output; output error; input node interrogator; hidden node analyzer. ***Info***: output/targets; node weighting; agreement stats; tolerance checking 20; input node interrogator; hidden node analyzer.

Utilizzando un foglio di Excel preparare circa 1200 record di numeri casuali, organizzati su due colonne: (first row #) =CASUALE.TRA(0;99)
Nella terza fila scrivi l'equazione *3x+y*, essendo *x* ed *y* i contenuti delle prime due colonne. Salva il file in formato *csv*
Riaprire il file csv con un editor di testo. Ad esempio è sufficiente un click sul nome del file con il tasto dx e poi modifica => sostituisci ; con ,
Salva e chiudi.

Settare *qnet* in modalità compatibile con WinXP ed avviarlo



new => network design

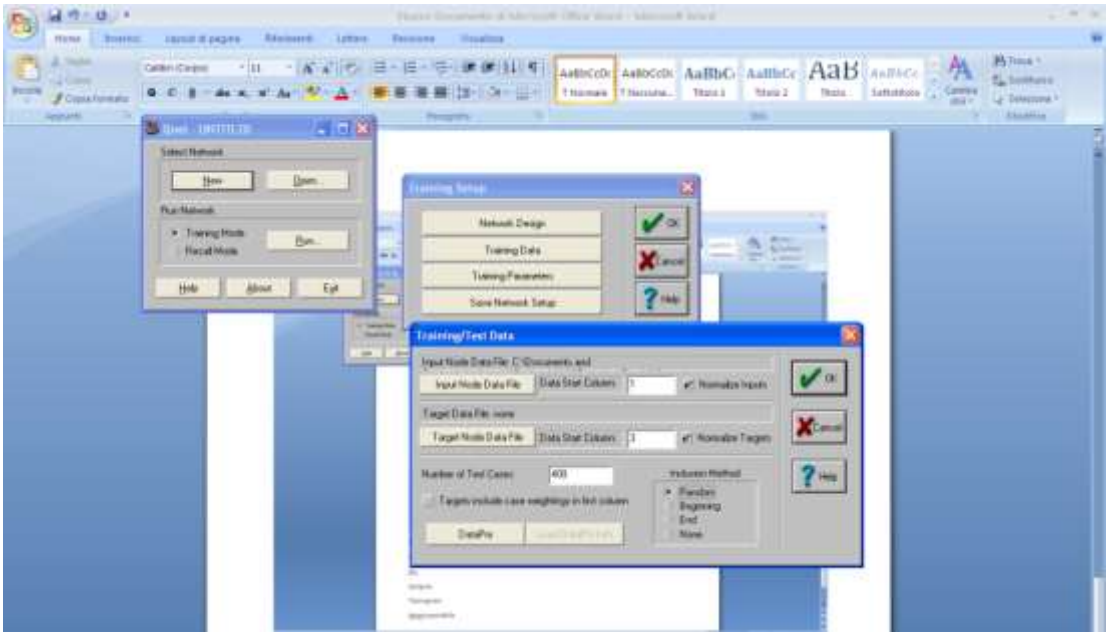
Si apre un'altra pagina sulla quale è possibile definire i parametri della rete

3 layers: input + hidden + output

Training test => Select input Data

Data start column =1 : si considera come input la prima colonna
Output data: iniziano dalla terza colonna del medesimo file. In alternativa è possibile utilizzare file separati per le variabili indipendenti e dipendenti.

800 records come training data ed restanti 400 come testing data, selezionati in modo casuale



Possiamo salvare la nostra rete con save network setup

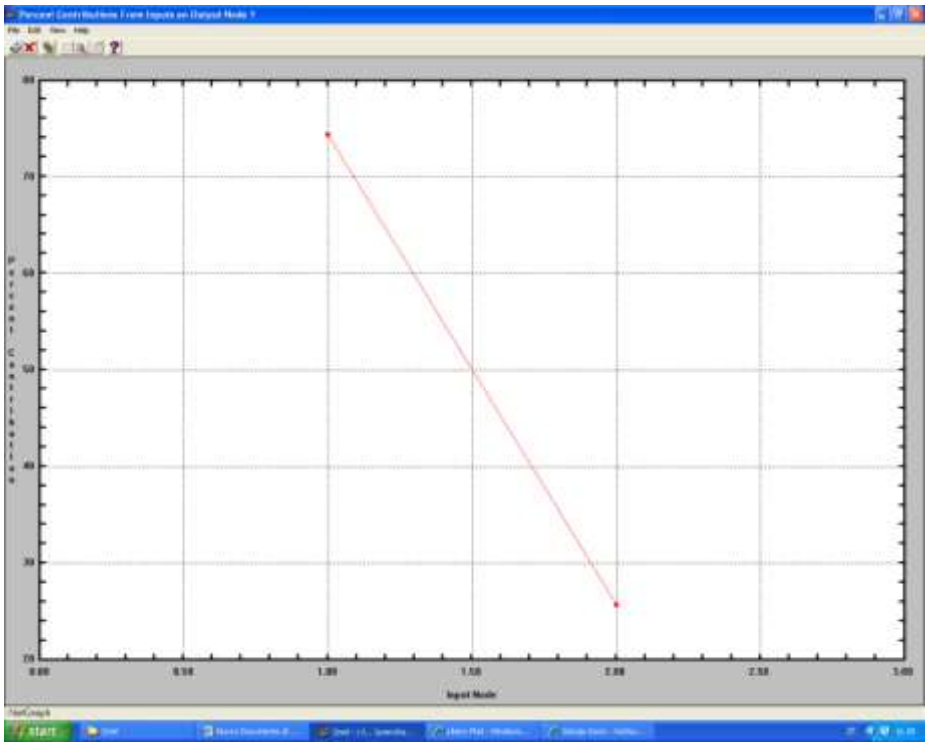
Appena confermiamo con OK, si avvia il calcolo dei coefficienti di pesatura

Analisi dei risultati:

net graph => root mean square error of training set e root mean square error of test set
correlation history => fornisce una misura della qualità della risposta della rete
target vs output => elenco dei valori calcolati dalla rete a confronto con quelli osservati

input node interrogator => rappresentazione dell'importanza delle relazioni tra le variabili di input e variabili dipendenti.

Nell'esempio specifico si vede che la var. dipendente è influenzata principalmente dalla variabile 1 (ricordiamo del resto che abbiamo costruito il dataset in modo che *x* conti il triplo di *y*). Vediamo che la ANN è riuscita a ricostruire molto bene le relazioni tra i dati. x influenza l'output per circa il 75% ed y per circa il 25.
Del resto, ricordando la regola utilizzata per generare i dati z=3x+y, vediamo che la somma dei pesi vale 3+1=4 e che l'influenza relativa delle due variabili è ¾ e ¼.

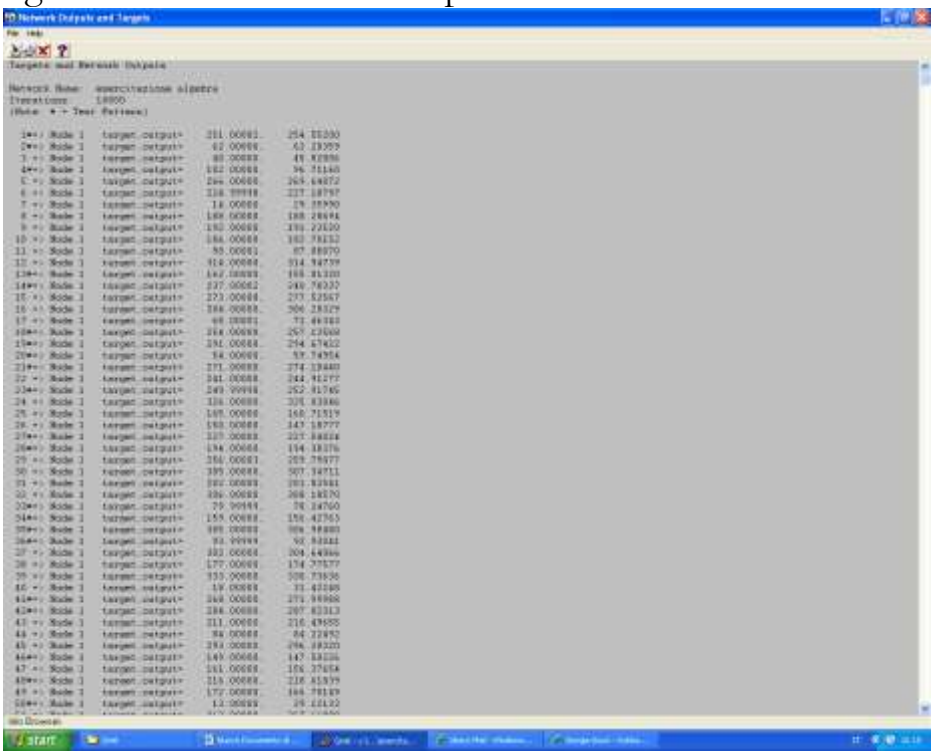


È molto importante questo punto poiché, non solo con le reti vediamo quale parametro è più significativo, ma anche quanto. Tale risultato con un'analisi ANOVA non è possibile.

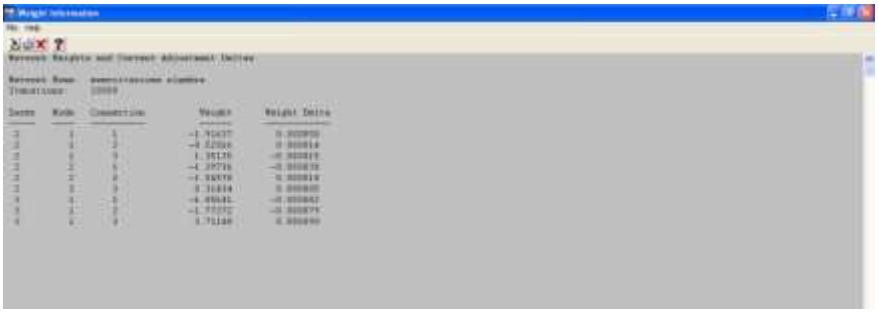
Hidden node analyzer (connection strength) => indica l'importanza dei singoli nodi dello strato nascosto. I nodi poco impegnati possono essere eliminati. Per tentativi possiamo individuare un valore ottimale per quanto riguarda il numero di nodi dello strato nascosto.

Dal menù Info, otteniamo le stesse informazioni in forma tabellare.

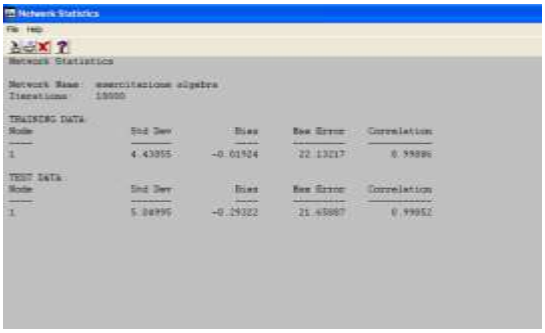
Info => case weighting/output target: valori stimati e i valori reali a confronto. Quelli contrassegnati con * sono stati usati per il test set



info => node weights: è riportato il peso delle connessioni, ovvero il valore dei coefficienti *wi*

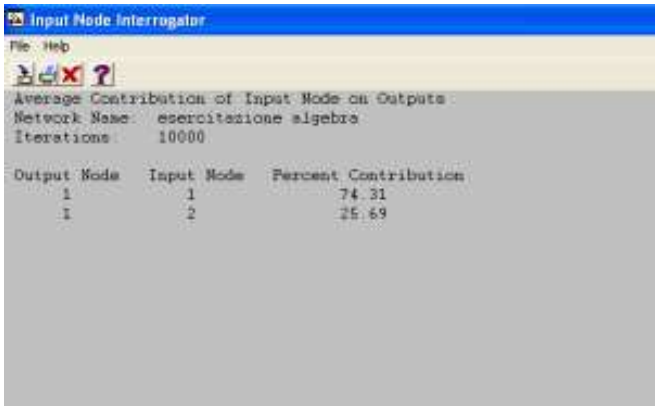


Info => agreement Statistics. Tabella di riepilogo statistico



Info => Network tolerance check: inserito un valore, ad esempio 20, possiamo conoscere quanti dei risultati stimati manifestano un errore superiore a 20

Info => input node interrogator: questa invece è l'importante tabella che riporta i pesi delle connessioni input-output, in sostanza l'espressione delle relazioni tra cause ed effetti



9.2.Nota

Una volta portata a convergenza una rete è possibile ricostruirla in un programma esterno (ad esempio in linguaggio C o Basic) o anche in un foglio di calcolo.

I valori dei coefficienti della rete *wi* e *ti* possono essere recuperati sia attraverso il menù *Info => NodeWeights and Deltas*, sia leggendo direttamente il file .net con un editor di testo. Va comunque ricordato come, per le peculiarità di Qnet2k, tali coefficienti sono calcolati su dati di ingresso normalizzati tra 0.15 e 0.85:

$$x_n = 0.15 + (x - x_{min})\frac{0.85 - 0.15}{x_{max} - x_{min}}$$

In alternativa, il file allegato "Legge e Calc Qnet2k.xls" svolge il servizio di importazione delle rete in un foglio Excel.

9.3.Esempio 2 - funzioni continue - file sin.xls

funzione continua (e.g. sin(x)) 400 record x=[0-10] Modifica=>riempimento=>serie: colonna, da 0 a 9 step 0,1 Test 1, 2, 4, 16 PE show even the test-set choice effect: random/end.	***Netgraph:*** RMS errorhistory (test+train data); correlation; targets vs. output; target output vs. pattern seq.; output error. ***Info***: Case weighting output/targets; node weighting; agreement stats; tolerance checking 0.15; hidden node analyzer.

funzione continua irregolare **e.g. sin(x)*(1+0.1*rnd())** 400 record x=[0-10] Modifica=>riempimento=>serie: colonna, da 0 a 9 step 0,1 Test 1, 2, 4, 16 PE show even the test-set choice effect: random/end.	***Netgraph:*** RMS errorhistory (test+train data); correlation; targets vs. output; target output vs. pattern seq.; output error. ***Info***: Case weighting output/targets; node weighting; agreement stats; tolerance checking 0.15; hidden node analyzer.

9.4.Esempio 3 – ottimizzazione di processo - file forno.xls

biscotti test 2 and 5 nodes hidden layer. 10^5 iterations. tempo cottura: 20-100 min temperatura: 150-200 °C tempo mixing: 100-180 s 200 misure in totale	*variabili di processo* (tempo cottura, temperatura cottura, tempo di miscelazione pastella) ==> *variabili di prodotto* (crispness by a fragility index, browning index). ***Netgraph:*** RMS errorhistory (test+train data); correlation; targets vs. output; output error; input node interrogator; hidden node analyzer. ***Info***: Case weighting output/targets; node weighting; agreement stats; tolerance checking 0.2; input node interrogator; hidden node analyser.

plant parameters optimisation	temperature; concentration; mass flow => shelf life (due to thermal stress?); residual microbial load

10.Qnet samples

Sample_problems (OCR; map generation; sphere drag)
..\..\..\..\..\..\..\..\Program Files\Eng\Stat\Qnet\Qnetv2k.chm

OCR	64 input node ***Netgraph:*** RMS errorhistory + correlation (test+train data); output error. ***Info***: case weighting/output/targets; node weighting; agreement stats; tolerance cecking 0.1; hidden node analyzer.
Map	***Netgraph:*** RMS errorhistory + correlation (test+train data); Target vs. out; target vs. pattern; output error vs. pattern. ***Info***: case weighting/output/targets; node weighting; agreement stats; hidden node analyzer.
Drag add 20 test	***Netgraph:*** RMS errorhistory + correlation (test+train data); Target vs. out; output error vs. pattern. ***Info***: case weighting/output/targets; node weighting; agreement stats; input node analyzer; hidden node analyzer.