Intelligenza Artificiale

Reti Neurali

Roberto Marmo
Laboratorio di Visione Artificiale, Pavia
marmo@vision.unipv.it

Introduzione alle Reti Neurali

Parte 1. Il cervello umano

Parte 2. La rete neurale

Parte 3. L'apprendimento della rete

Parte 4. Creare una rete per classificare

Parte 5. Esempio di applicazione

Parte 6. Seminario su uso reti Kohonen

Testi consigliati

- Bishop, *Neural networks for pattern recognition*, Clarendon Press, Oxford, 1996
- Domeniconi, Jordan, *Discorsi sulle reti neurali e l'apprendimento,* Franco Angeli, 2001
- Cammarata, *Reti neuronali,* Etas
- Floreano, *Manuale sulle reti neurali*, Il Mulino
- *L'apprendimento delle reti artificiali di neuroni,* Le Scienze n.291, novembre 1992
- De Luca, Caianiello, *Introduzione alla cibernetica*, Franco Angeli
- ftp://ftp.sas.com/pub/neural/FAQ.html nozioni e link su Internet

Parte 1

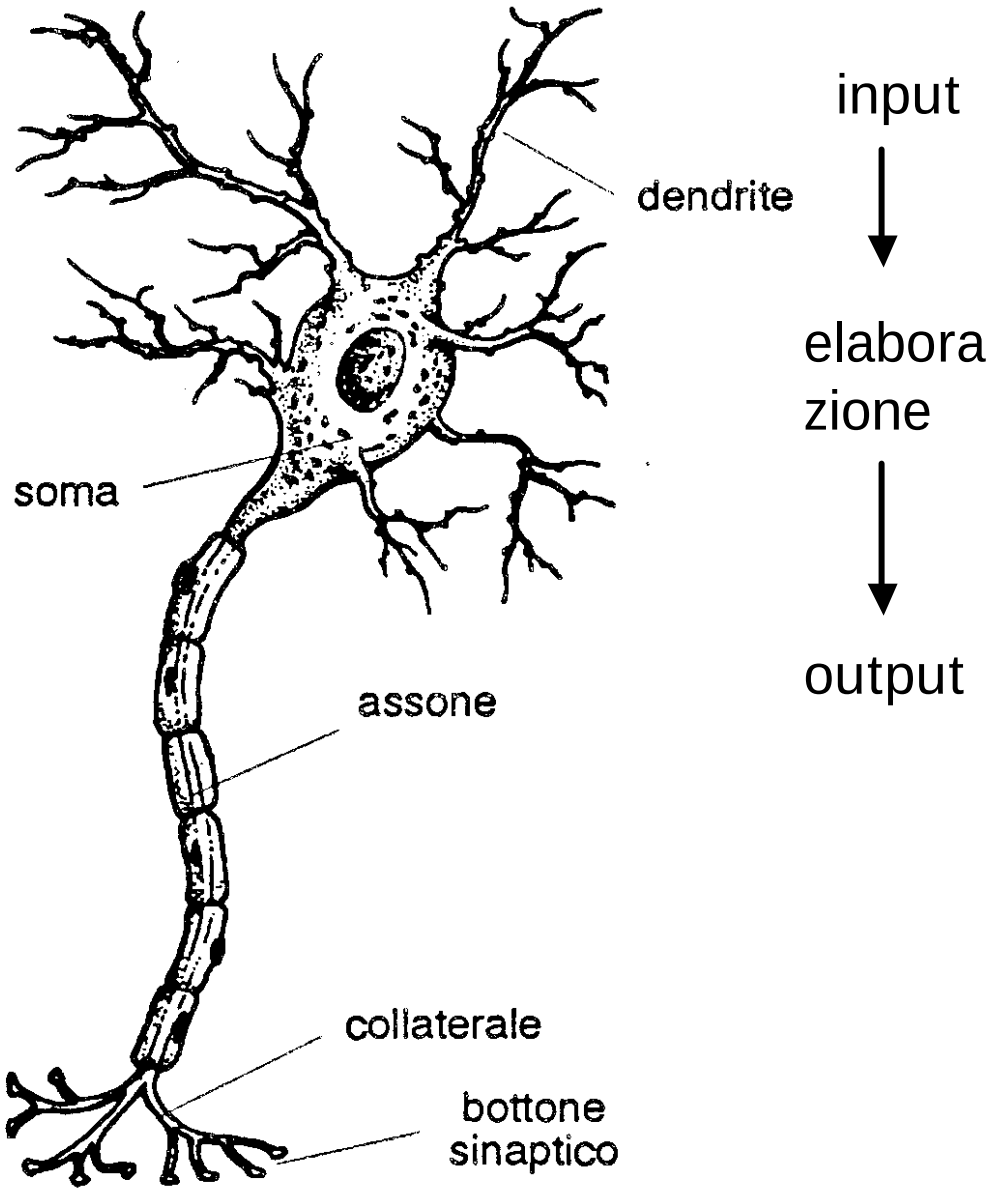
Il cervello umano

John von Neuman: non c'è' alcun modello del cervello più semplice del cervello stesso.

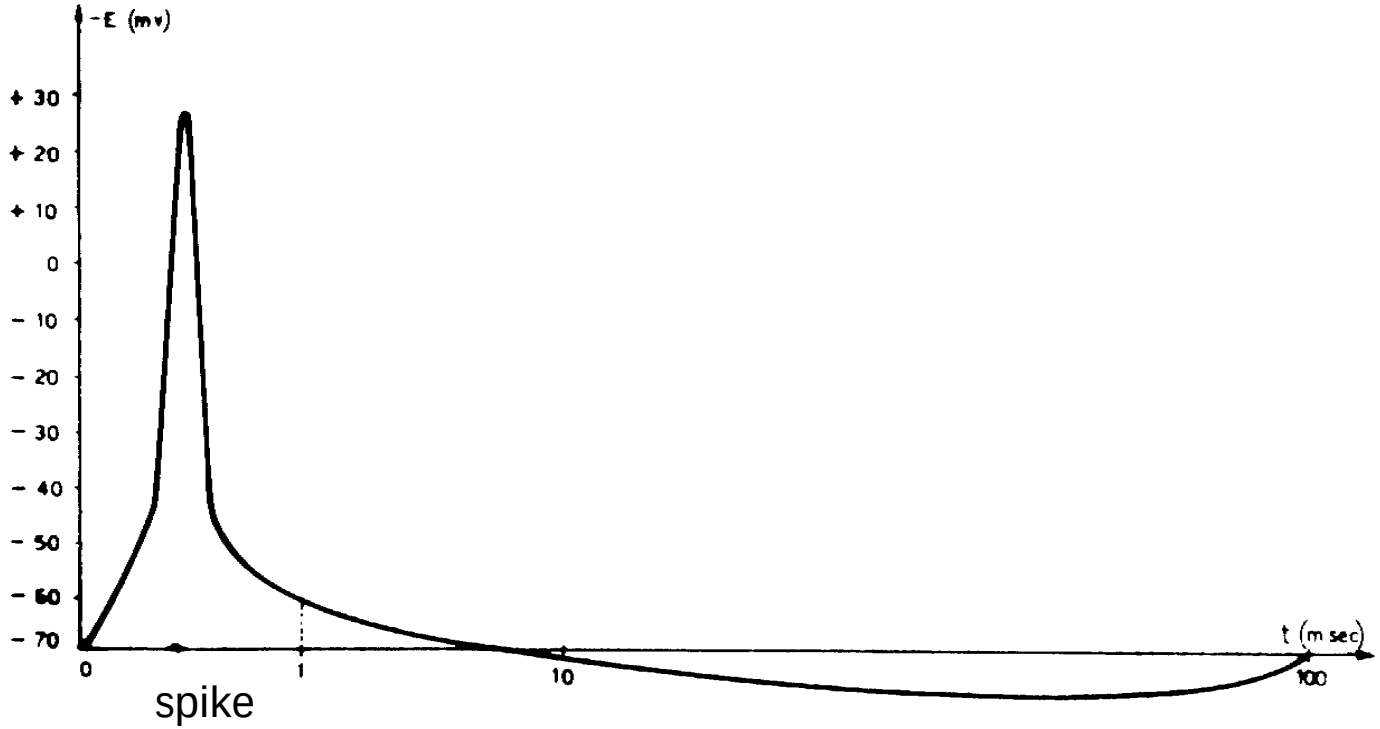
Per simulare alcuni comportamenti della struttura cerebrale degli esseri viventi attraverso un insieme di regole di calcolo, occorre conoscere la struttura del cervello umano e cercare di riprodurlo con un modello matematico.

La cellula neuronale

Unità fondamentale del cervello umano. Ogni neurone riceve come input i segnali elettrici da tutti i dendriti, e se la somma pesata supera il valore di attivazione emette un impulso elettrico in uscita verso l'assone.



Funzione di sparo del neurone



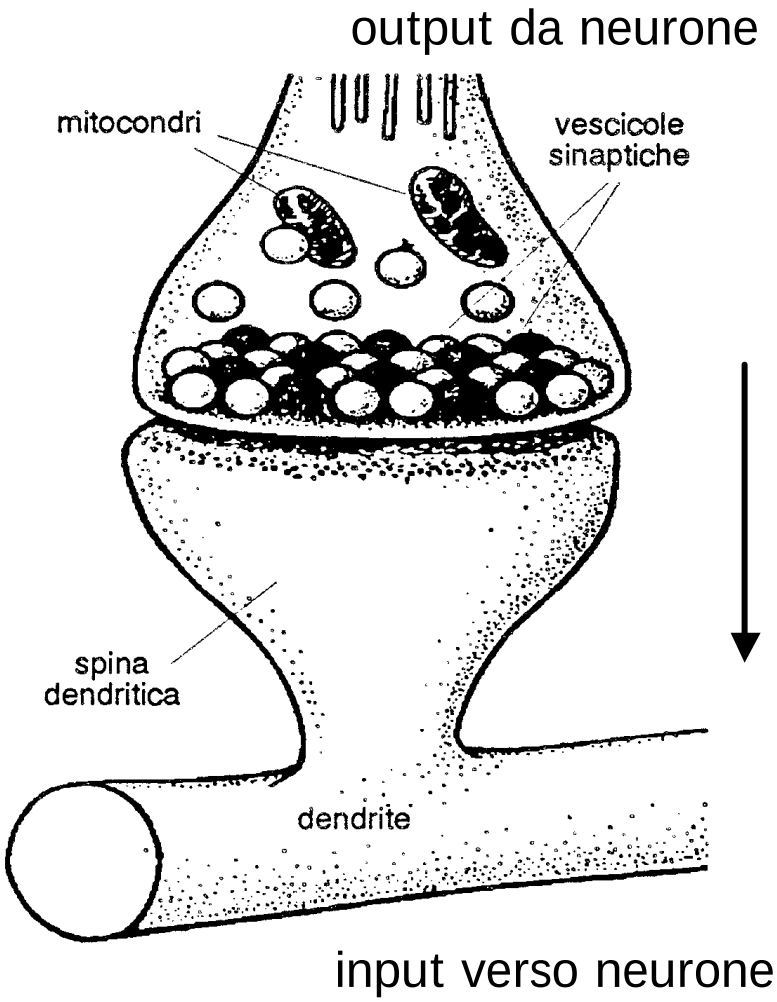
spike

Andamento di –E, differenza di potenziale fra interno ed esterno della membrana cellulare del neurone.

Per uno stimolo con adeguata intensità, il neurone risponde con uno spike o non risponde: non c'è nessun tipo di risposta intermedia.

Contatto sinaptico

Punto di contatto tra due neuroni. Le sinapsi aumentano o diminuiscono nel tempo. La parte superiore trasforma il segnale elettrico in sostanza chimica (neurotrasmettitore) che passa dall'altra parte e viene riconvertito in segnale elettrico. La sinapsi può fare azione eccitatoria o inibitoria del collegamento tra i due neuroni variando i neurotrasmettitori.



Il cervello umano

- nome scientifico: encefalo
- esistono vari tipi di neuroni
- 10 miliardi di neuroni, 80000 neuroni per mm^2
- tessuto gelatinoso con peso di 1300-1500 grammi
- ogni neurone interagisce con 1000-10000 neuroni
- ogni impulso elettrico viaggia alla velocità di 130 metri al secondo
- sinapsi tra neuroni anche molto distanti
- non importa il numero di neuroni ma il numero di sinapsi

Il cervello umano

- l'elaborazione dell'informazione in parallelo e diffusa tra migliaia di neuroni porta all'emergere di processi cognitivi
- l'elaborazione risiede in due emisferi separati da un corpo calloso che sono, a loro volta, suddivisi in regioni ben localizzate negli emisferi
- struttura cerebrale sempre in evoluzione: ogni giorno centinaia di neuroni muoiono, ma non c'è declino mentale perché aumentano le sinapsi per compensare le perdite

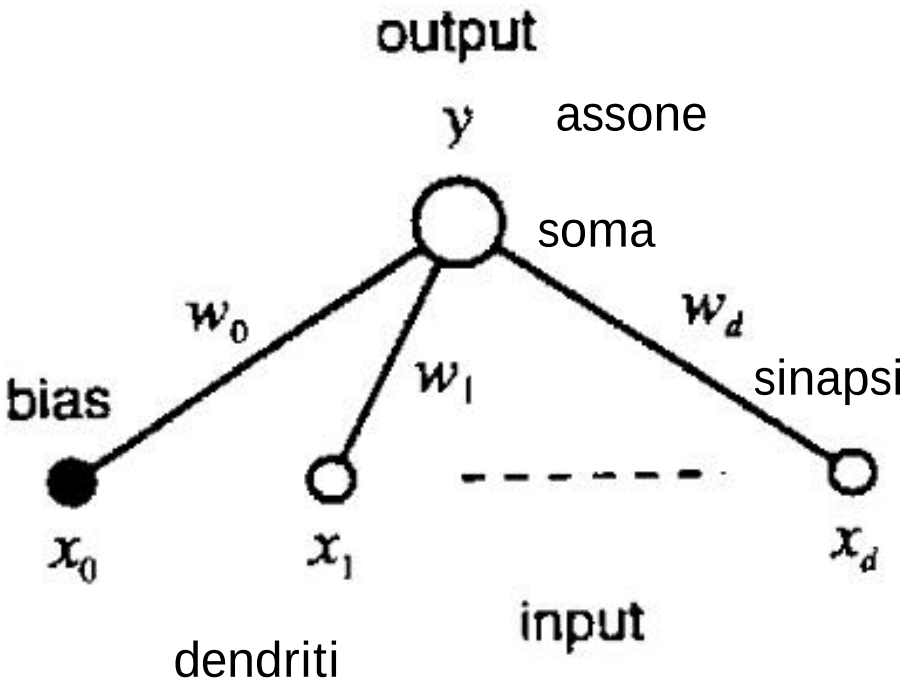
Parte 2

La rete neurale

Il neurone artificiale

Modello matematico molto semplificato del neurone biologico.

Ad ogni input x_i è associato un peso w_i con valore positivo o negativo per eccitare o inibire il neurone. Il bias varia secondo la propensione del neurone ad attivarsi, per variare la soglia di attivazione del neurone.



Algoritmo del neurone

1. caricare i valori degli input x_i e dei pesi relativi w_i
2. calcolare la somma dei valori input pesata con i relativi pesi
3. calcolare il valore della funzione di attivazione g con il risultato della somma pesata
4. l'output del neurone y è il risultato della funzione di attivazione

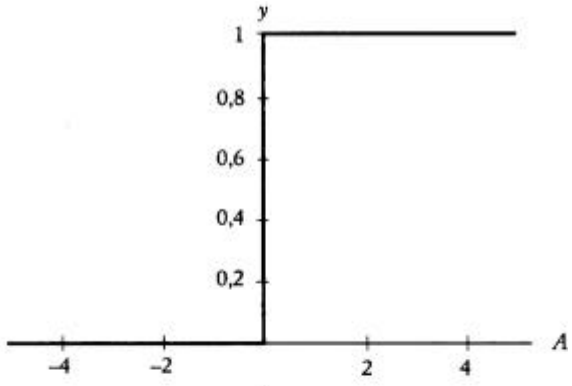
$$y(x) = g\left(\sum_{i=1}^{d} w_i x_i + w_0\right) = \overline{w}^T \overline{x}$$

Funzioni di attivazione

Determina la risposta del neurone.

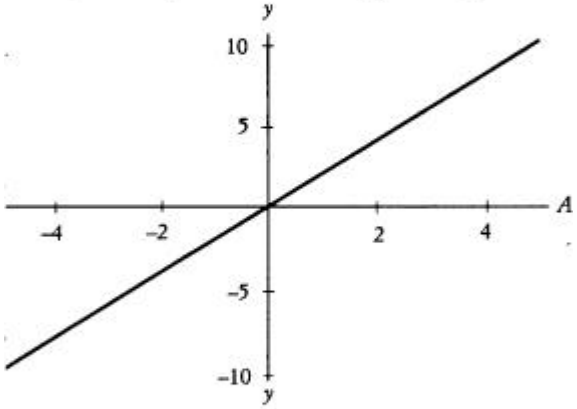
A gradino

$$g(A) = \begin{cases} 1 \ se \ A > 0 \\ 0 \ altrimenti \end{cases}$$



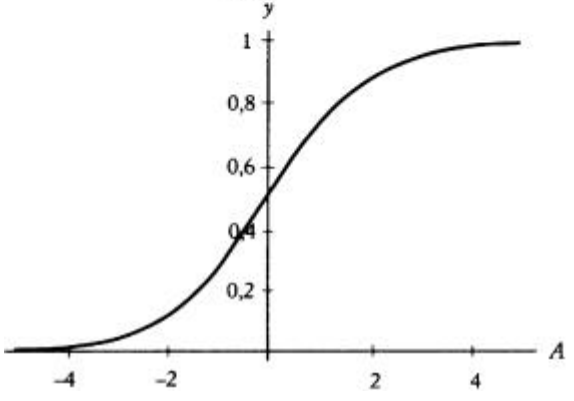
Lineare continua

$$g(A) = kA$$



Sigmoide o logistica $$g(A) = \frac{1}{1+e^{-A}}$$

valori positivi, continua e derivabile



La rete neurale

Sistema dinamico avente la topologia di un grafo orientato con nodi, i neuroni artificiali, ed archi, i pesi sinaptici. Il termine rete è riferito alla topologia dei collegamenti tra i neuroni.

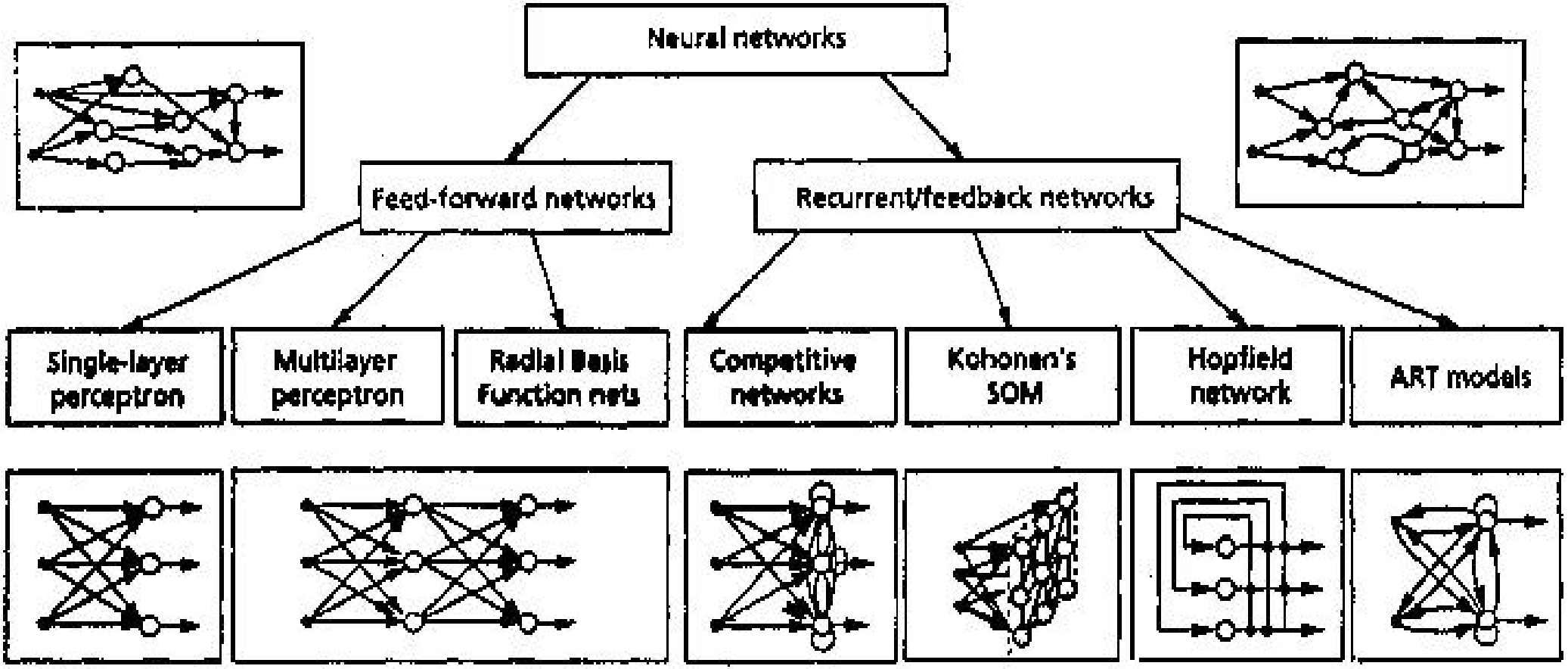
Altre definizioni:

- scatola nera, di cui si può ignorare il funzionamento, che associa un input a un output e le associazioni possono essere create con l'apprendimento
- modello matematico che calcola la funzione output=f(input,pesi) al variare dei pesi e senza specificare la forma della funzione f

La rete neurale è un modello matematico che usa neuroni matematici, la rete neuronale usa neuroni biologici.

In Bishop un approccio statistico e dimostrazioni matematiche.

Esempi di reti neurali



Vantaggi

- adatte per problemi che non chiedono risposte accurate, ma risposte approssimate con un grado di errore o di variazione
- generalizzazione: producono buone risposte anche con input non considerati durante la creazione e l'addestramento
- facili da implementare, basta definire il neurone e poi crearne delle copie e creare i collegamenti tra i neuroni
- funzionamento veloce perché parallelo; ogni neurone usa solo il suo input
- stabilità dell'output rispetto a valori di input: incompleti, con rumore, non ben noti, che accettano un grado di errore o di variazione
- determinano il risultato tenendo conto contemporaneamente di tutti gli input

Svantaggi

- incapacità di rendere conto dell'elaborazione: non si può capire perché ha dato quel risultato specifico
- non si può descrivere e localizzare la conoscenza memorizzata nella rete
- carenza di hardware con cui implementare, si usano su computer seriali
- tecniche di addestramento sofisticate che richiedono molto tempo di calcolo
- non sempre esiste una rete che risolve il problema, perché non sempre esiste un algoritmo di apprendimento che converge dando un output della rete con basso errore
- i valori di output non sono precisi, ma hanno un margine in cui possono variare
- serve una casistica di esempi molto ampia per ottenere un buon apprendimento e un basso errore di output

Campi di applicazione

1. pattern classificazione
2. clustering
3. approssimazione di funzioni
4. predizioni in serie temporali
5. ottimizzazione
6. memorie associative
7. controllo di apparati
8. elaborazione di segnali ed immagini

Rapporti con algoritmi genetici

Gli algoritmi genetici possono essere applicati alle reti neurali per risolvere problemi come:

- scegliere la struttura della rete
- scegliere i valori dei pesi per ridurre l'errore di output

Rapporti con logica fuzzy

Le reti neurali non calcolano con le variabili linguistiche e non usano regole qualitative. Esistono le reti neuro-fuzzy, strumento matematico molto efficiente e complesso che unisce i vantaggi delle due tecniche.

Rapporti con i sistemi esperti

Le reti neurali si differenziano molto dai sistemi esperti perché:

- non usano conoscenze esplicite, ma conoscenze implicite contenute in una casistica molto ampia di esempi formata da vettori di numeri
- usano numeri invece di simboli e regole
- non usano regole tipo IF...THEN..
- tutti i neuroni collaborano insieme
- non vengono programmate, vengono addestrate con l'apprendimento
- accettano dati parziali e con rumore
- il sistema esperto può spiegare perché e come ha ottenuto una conclusione, le reti neurali non può giustificare perché e come ha ottenuto dei valori specifici in uscita

Rapporti con i sistemi esperti

- un sistema esperto può fare ragionamenti come “Socrate è un uomo, gli uomini sono mortali, Socrate è mortale”, la rete neurale no ma può facilmente riconoscere una faccia

- un sistema esperto non tiene conto del supporto materiale che ragiona, ma si dedica solo alla funzione del ragionamento e cerca di imitare la mente umana. Invece il connessionismo è l’approccio allo studio della mente con le reti neurali; considera fondamentale la struttura del supporto materiale che ragiona, per cui cerca di imitare il cervello umano

Parte 3

L’apprendimento della rete

La programmazione

Esempio: per riconoscere un computer bisogna scrivere un programma con struttura IF...THEN che include tutti i casi possibili, quindi bisogna prevederli tutti:

IF ha microprocessore THEN

IF ha memoria THEN

IF ha tastiera THEN

.....

PRINT “è un computer”

L'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 computer. Si creano tanti esempi di computer e si fornisce ogni esempio alla rete neurale dicendogli che è un esempio di computer. La rete neurale impara e si crea una esperienza sul riconoscimento dei computer, così la prossima volta che ha in input una descrizione diversa di un computer lo riconosce subito.

Non programmare ma apprendere

La programmazione serve solo per creare il software che crea la rete e l'algoritmo di apprendimento. Per insegnare alla rete a risolvere un problema, occorre un periodo di apprendimento in cui insegnare alla rete come comportarsi con l'input che riceve, perché all'inizio la rete non ha nessuna forma di conoscenza.

L'operatore crea la struttura della rete e quindi i pesi sono gli unici parametri che possono essere modificati. Infatti la conoscenza è memorizzata sui pesi e la rete apprende usando tecniche di ottimizzazione per variare i valori dei pesi, cercando di minimizzare una funzione di errore. Tipi di apprendimento:

- supervisionato
- non supervisionato
- hebbiano.

Ogni modello di rete neurale ha il suo specifico tipo di apprendimento.

Apprendimento supervisionato

Alla rete viene presentato un training set preparato da un supervisore esterno, e composto da molte coppie significative di valori (input,output atteso);

- la rete riceve l'input e calcola il suo corrispondente output
- per un certo input, l'errore è dato dalla differenza tra l'output della rete e l'output atteso; serve a supervisionare l'apprendimento per far capire alla rete quanto si sbaglia nel calcolare quell'output
- la rete modifica i pesi in base all'errore cercando di minimizzarlo e commetterà sempre meno errori.

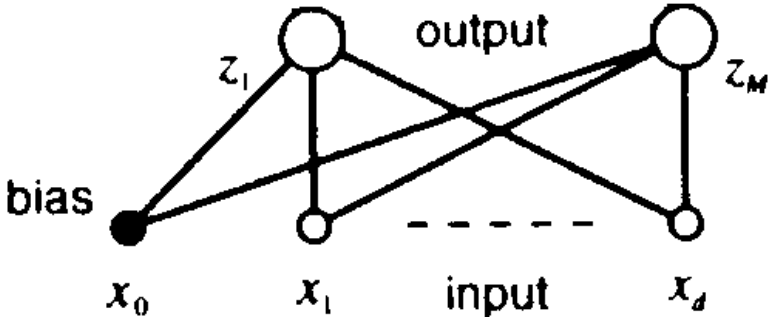
Occorre quindi preparare alcuni esempi di funzionamento studiati appositamente, e la rete impara da questi esempi.

Modelli che usano questo apprendimento:

- perceptron
- multi layer perceptron
- radial basis function

Il perceptron

Modello semplice creato da Rosenblatt nel 1957 per riconoscere immagini simulando la percezione umana, da cui il nome del modello. Ogni neurone ha funzione di attivazione a gradino ed uscita con valori binari: 1 indica la presenza di un oggetto, 0 l'assenza. Questo modello è composto da una fila di neuroni affiancati.

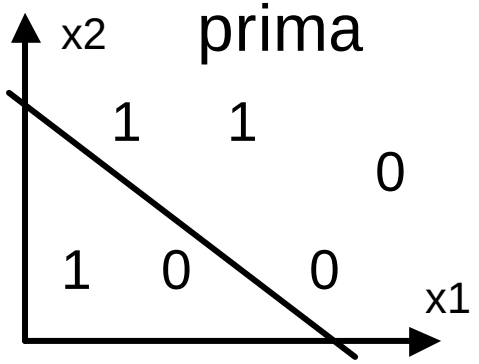


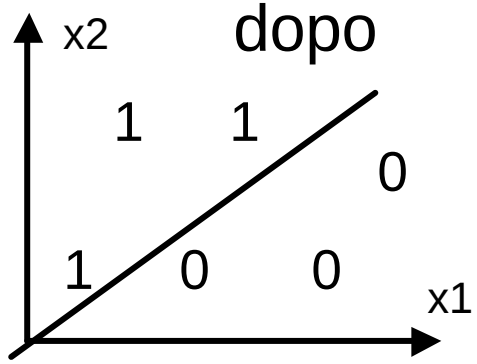
L'apprendimento consiste nel variare i pesi e nel muovere una retta di separazione del piano fino a dividere correttamente i valori input in due gruppi.

Il perceptron

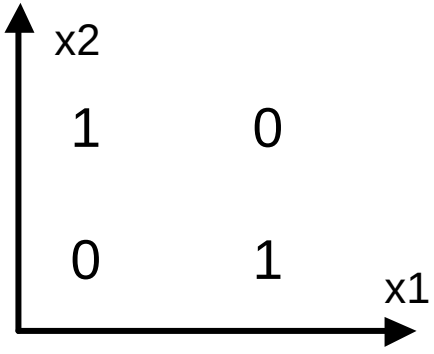
Esempio: un oggetto è caratterizzato dalla sua altezza x1 e larghezza x2, e ci sono due tipi di oggetti. Si riportano sul piano degli input le misure di alcuni oggetti e si rappresentano i due tipi di oggetto con 1 e 0. Si vuole stabilire una semplice regola per decidere a quale classe appartengono.

Spazio di 2 valori input linearmente separabile e Perceptron con apprendimento trova sempre una retta di separazione del piano nei due semipiani con i gruppi di oggetti





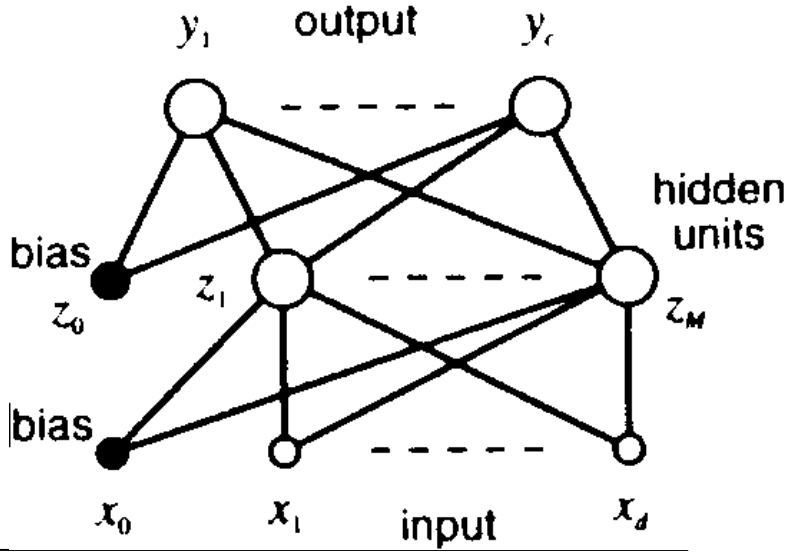
Spazio di 2 valori input non linearmente separabile e perceptron non trova una retta di separazione



Multi Layer Perceptron (Mlp)

- formato dalla sovrapposizione di vari Perceptron
- il livello di input non contiene neuroni, nell'esempio sono 2
- ogni neurone è collegato con tutti i neuroni dello strato precedente e successivo, i neuroni sullo stesso strato non sono collegati, non esistono cicli dallo strato di output verso lo strato di input

•la conoscenza viene elaborata dal livello input verso il livello output, cioè si calcola l'output di tutti i nodi di un livello che diventano poi gli input dei nodi del livello successivo



Struttura di Mlp

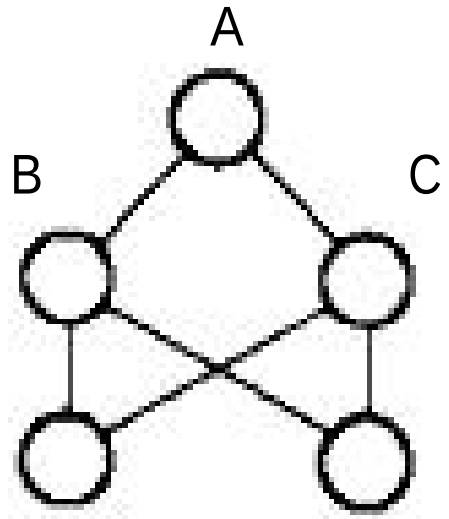
Mlp con d neuroni input, M neuroni hidden, c neuroni output, g funzione di attivazione dei neuroni, w_{KJ} il peso dal neurone k al neurone j

$$y_k(x) = g\left(\sum_{j=0}^{M} w_{kj}^{(2)}\, g\left(\sum_{i=0}^{c} w_{ji}^{(1)} x_i\right)\right)$$

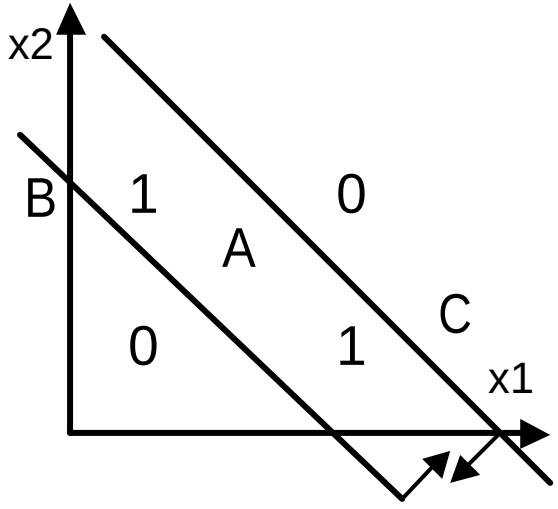
- tutti i neuroni allo stesso livello hanno la stessa funzione di attivazione
- la funzione di attivazione softmax crea ogni output in [0,1] e somma di tutti gli output pari a 1, in modo da interpretare la risposta della rete come stime di probabilità

$$g(a_i) = \frac{e^{a_i}}{\sum_{i=1}^{n} e^{a_i}}$$

Regioni delimitate da Mlp

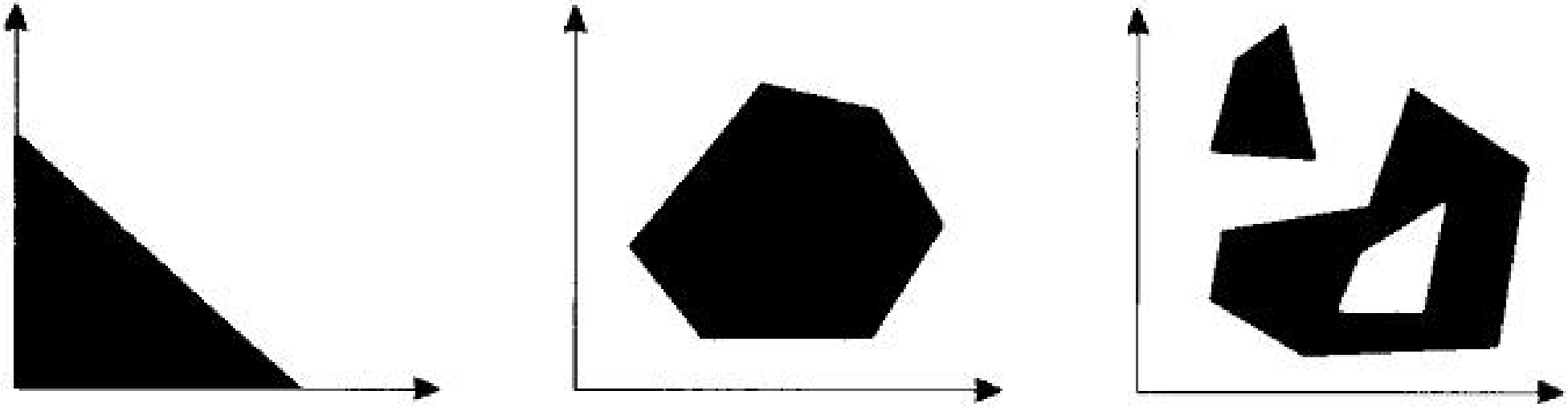


Supera i limiti del Perceptron usando strati di neuroni hidden (interni) che realizzano una rappresentazione interna dell'input più complessa, perché individua regioni arbitrarie intersecando iperpiani nello iperspazio dei valori input.



Nell'esempio i Perceptron B e C creano ciascuno un semipiano individuato dalle rette e il Perceptron A interseca i due semipiani individuando la parte di piano tra le rette.

Regioni delimitate da Mlp

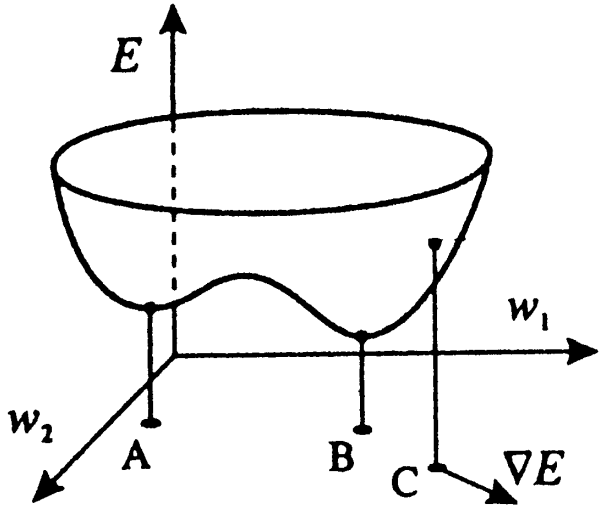


Usando la funzione di attivazione a gradino, a sinistra la regione delimitata da un Perceptron, al centro la regione delimitata da Mlp a 1 livello hidden, a destra la regione delimitata da Mlp con 2 livelli hidden che può delimitare aree arbitrarie.

Funzione di errore

Esprime la differenza fra l'output della rete y e l'output desiderato y' nell'apprendimento. $E(w)$ perché l'output è ottenuto in base al valore dei pesi; quindi E è derivabile nei pesi w e occorre trovare il vettore w che rende minimo l'errore. Esistono molte formulazioni di E.

$$E(w) = \sum_{i=1}^{c} \left(\left(y_i - y_i^{'} \right)^2 \right)$$



E è non lineare: gli algoritmi cercano un minimo nella sua superficie con modifiche di w in base al gradiente di E. A sinistra c'è una superficie di E quadratica nei pesi w_1 e w_2; A e B sono minimi e C punto di calcolo del gradiente locale della superficie, si va nella direzione opposta al gradiente. Può non convergere verso il minimo assoluto B ma verso un minimo locale A; in alcuni punti il gradiente è nullo.

Formule del back-propagation

Ogni neurone computa $a_j = \sum_i w_{ji} z_i$ con z_i l'attivazione di unità che manda connessione a unità j e w_{ji} il peso tra unità j e i, la sommatoria è su tutte le unità connesse all'unità j. Usando funzione attivazione g si ha l'attivazione z_j dell'unità j

$z_j = g(a_j)$ Se le z_j indicano unità input allora sono $z_j = x_j$, se le unità j sono di output l'attivazione è denotata $y_k = z_j$.

$E = \sum_n E^n$ è l'errore per n pattern input.

Formule del back-propagation

La variazione dell'errore dipende dalla $\frac{JE^n}{Jw_{ji}} = \frac{JE^n}{Ja_j}\frac{Ja_j}{Jw_{ji}}$

variazione dei valori dei pesi, in cui

il primo fattore riflette la variazione dell'errore in funzione della variazione dell'input all'unità e l'altro riflette la variazione di un certo peso sull'input

posto $d_j = \frac{JE^n}{Ja_j}$ e $\frac{Ja_j}{Jw_{ji}} = z_i$ si ha $\frac{JE^n}{Jw_{ji}} = d_j z_i$ serve d_j

per ogni unità

Formule del back-propagation

Calcolo d_k per unità output

$y_k = g(a_k)$ $\quad \frac{Jy_k}{Ja_k} = g'(a_k)$ e $\frac{1}{Ja_k} = \frac{g'(a_k)}{Jy_k}$ si ha $d_k = \frac{JE^n}{Ja_k} = g'(a_k)\frac{JE^n}{Jy_k}$

con z_k denotato da y_k, occorre g' e la derivata che si calcola subito perché al livello output si ha output desiderato e output da rete, quindi si calcola subito l'errore e il gradiente

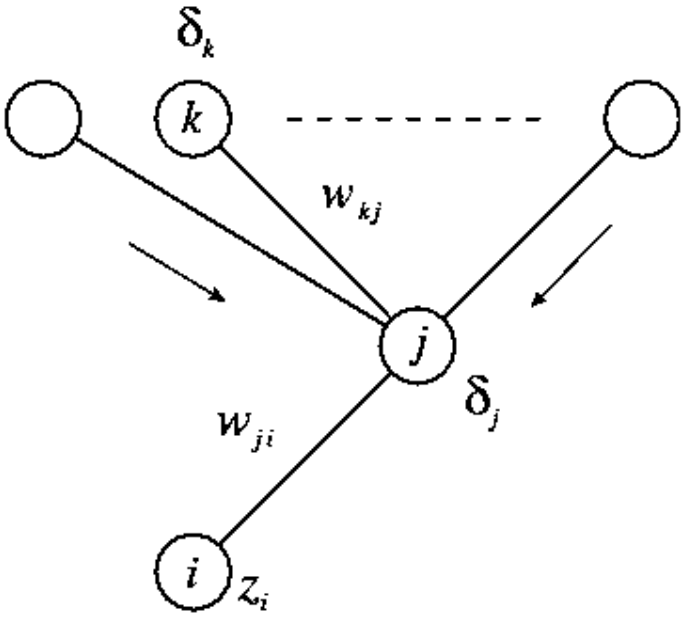
Calcolo d_j per unità input

serve errore da unità di livello superiore per calcolare la funzione di errore locale

$d_j = \frac{JE^n}{Ja_j} = \sum_k \frac{JE^n}{Ja_k}\frac{Ja_k}{Ja_j} = \sum_k d_k \frac{Ja_k}{Ja_j}$ con somma su tutte unità k di livello superiore cui unità j manda connessioni

Formule del back-propagation

cioè la sommatoria dell'errore retropropagato per la quantità di errore dato dall'influenza dell'unità j su unità k



d_j dipende solo da valori input e da pesi di connessioni di j cioè dai neuroni vicini per cui il calcolo è locale, quindi:

- se errore è piccolo e peso ha valore grande allora connessione non porta molto errore

- se errore è grande e peso ha valore grande allora connessione porta molto errore e il peso va cambiato

Formule del back-propagation

Sostituendo la definizione di d e $a_k = \sum_j w_{kj} z_j = \sum_j w_{kj} g(a_j)$

$\frac{Ja_k}{Ja_j} = w_{kj} g'(a_j)$ sostituendo $d_j = g'(a_j) \sum_k w_{kj} d_k$

per cui i valori di un d di una unità hidden dipendono dai d delle unità a livello più alto; poiché i d delle unità output si calcolano subito, gli altri si ottengono ricorsivamente.

Variazione del peso $w_{ji}^{t+1} = w_{ji}^t - h \frac{JE^n}{Jw_{ji}} = w_{ji}^t - h d_j z_i$ in cui si addiziona ad ogni peso un incremento, positivo o negativo determinato dalla sua influenza nella formazione dell'errore, e h è il learning rate (coefficiente di apprendimento) tra 0 e 1.

Algoritmo back propagation

Si esegue un'epoca di apprendimento, cioè si attuano per tutte le coppie (input, output) significative del training set i passi:

1. prende una coppia e calcola la risposta della rete per quell'input; il calcolo procede dal livello input verso il livello output calcolando l'attivazione di tutte le unità, quindi propaga in avanti l'errore
2. calcola l'errore E tra l'output della rete e output della coppia e calcola i d_k delle unità output
3. propaga all'indietro l'errore verso il livello di input, calcolando i d_i per ogni unità hidden
4. variazione dei pesi
5. ripete dal passo 1 fino a terminare le coppie
6. calcola l'errore globale e se è ancora alto si ripete l'epoca di apprendimento.

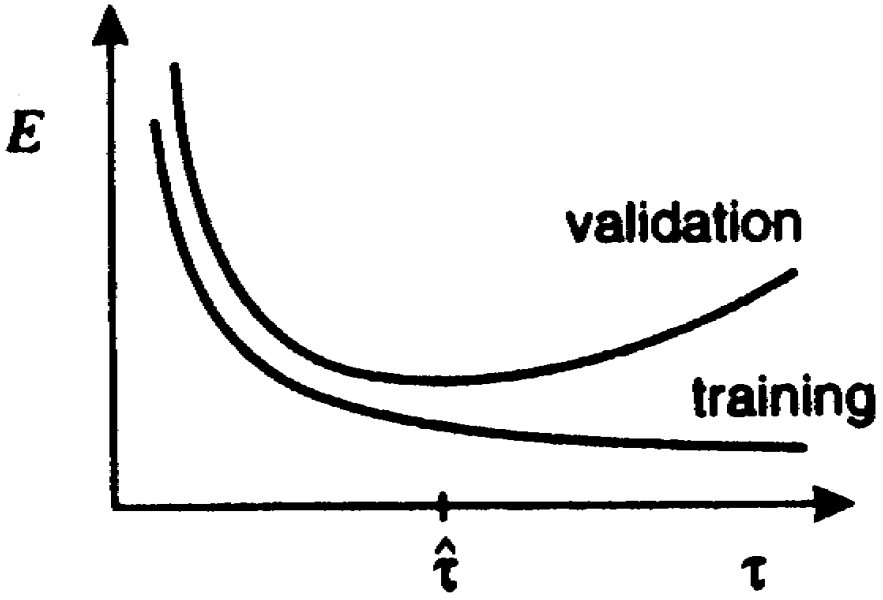
Problemi del back propagation

- algoritmo lento che può finire intrappolato in un minimo di E credendo di aver trovato il valore ottimo dei pesi che rende minima la E
- dipendenza dal valore iniziale dei pesi w; può capitare di cominciare da un punto della superficie di errore già molto vicino a un minimo locale
- scelta critica di h: troppo piccolo crea apprendimento lento, troppo grande crea oscillazioni

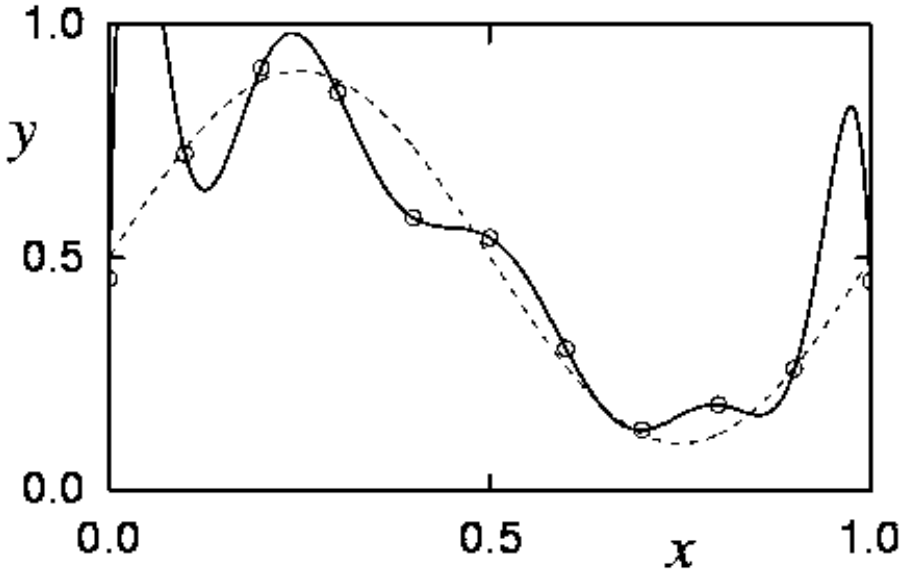
Esistono tecniche che aumentano la velocità di convergenza verso il minimo assoluto di E e che smorzano le oscillazioni aggiungendo alla formula di variazione del peso un altro parametro detto momentum.

Overfitting

La rete neurale deve avere capacità di comprensione del modello statistico dei dati, non memorizzare i soli dati del training set. Solo così può generalizzare, cioè rispondere esattamente a input non in training set. La tecnica migliore per evitare ciò è l'early stopping. Nell'apprendimento l'errore E tende a 0 perché la rete sta imparando meglio. Usando un validation set di coppie non usate nel training set, si misura la E e si crea la seconda curva che tende a un minimo e poi ricresce: da quel punto la rete sta imparando il training set e non il suo modello statistico. Si ferma lo apprendimento al minimo $\hat{t}$ di E rispetto al validation set.



Overfitting



I cerchi indicano i punti che formano il training set, la curva in tratteggio è la funzione da apprendere, la curva continua è la funzione appresa dalla rete neurale con errore nullo di training senza uso di early stopping. Evidente la differenza tra le curve nei punti diversi dai cerchi, perché la rete neurale ha imparato esattamente solo i cerchi e fa grossi errori negli altri punti.

Valutare prestazioni

Per misurare le prestazioni di una rete neurale dopo l'apprendimento, si crea il test set formato da coppie non usate per i training e validation set. In genere il test set è un terzo del training set ed è composto da input critici su cui la risposta della rete deve essere buona, altrimenti si butta via la rete.

Per ogni coppia del test set:

- calcolare la risposta della rete all'input
- calcolare l'errore dato dalla differenza tra output rete e output di coppia

L'errore totale è dato dalla somma degli errori avuti per ogni coppia.

Secondo il tipo di problema, si usano varie tecniche statistiche per decidere se usare o meno la rete creata; in genere si accetta una rete se sul test set ha mostrato un errore inferiore al 20-25% delle coppie.

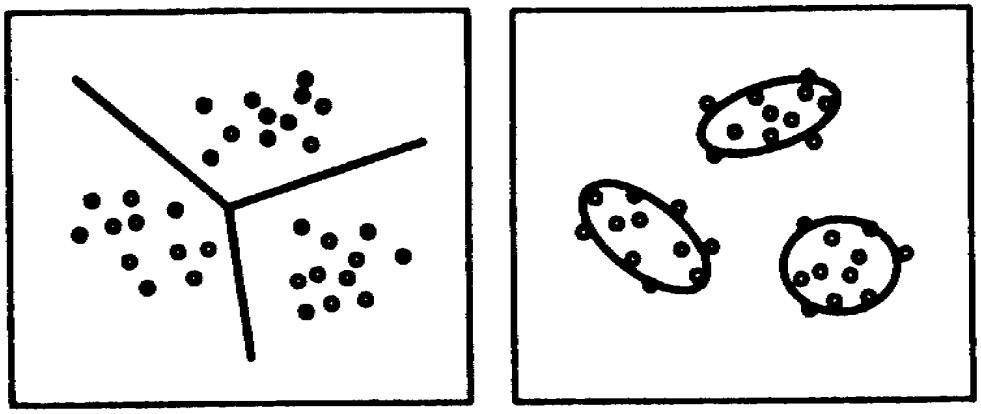
Radial basis function (Rbf)

Un solo livello hidden di M neuroni che realizza funzioni di base $f_j(x)$ circolari centrate sui punti dello spazio di input; esistono varie forme delle funzioni, in genere si usano gaussiane. Ideale per interpolazioni di funzioni e di predizione in serie temporali, in cui stimare il valore successivo in una serie di numeri.

$$y_k(x) = \sum_{i=1}^{M} w_{kj} f_j(x) + w_{k0}$$

Equazione del neurone output k su input x, w_{kj} è il peso tra neuroni k e j

Divisione dello iperspazio dei valori input: a sinistra Mlp con semipiani, a destra Rbf con cluster



Apprendimento non supervisionato

Alla rete vengono presentati solo i valori di input e la rete li divide autonomamente in gruppi usando misure di similarità, senza usare confronti con output noti, e cercando di mettere input simili nello stesso gruppo. E' un apprendimento autonomo e non c'è controllo esterno sull'errore. Adatto per ottimizzare risorse e se non si conoscono a priori i gruppi in cui dividere gli input.

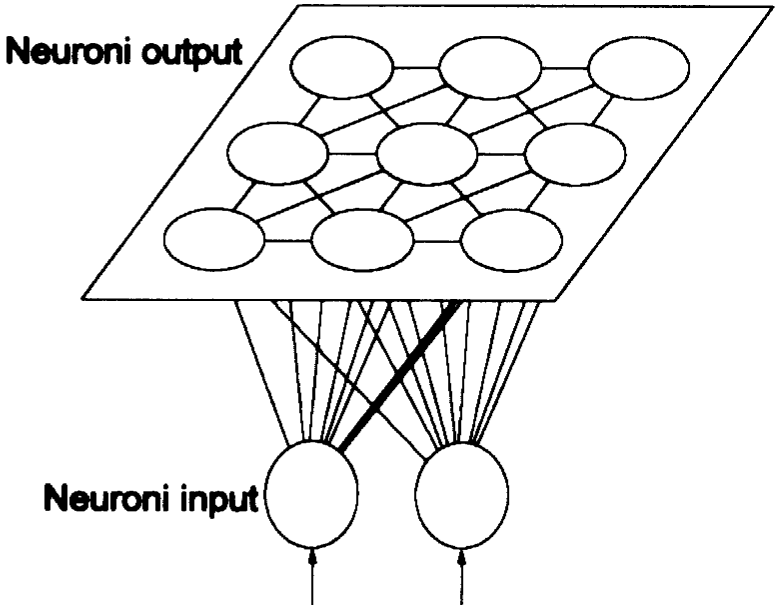
Modelli che usano questo apprendimento:

-Kohonen

-Hopfield

Self Organizing Maps SOM

Ideate da Tuevo Kohonen nel 1982 ispirandosi alla topologia della corteccia del cervello. Tengono conto delle connessioni tra neuroni e dell'influenza che può avere un neurone sui suoi vicini: i neuroni vicini a neuroni attivi rinforzano i legami, mentre a quelli che si trovano ad una data distanza vengono indeboliti i legami.

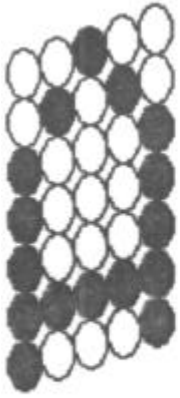


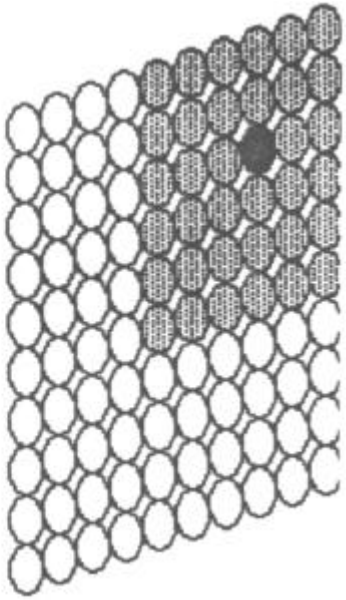
Una rete SOM è composta da un livello di input ed un livello di competizione che sono localizzati su una griglia ad una o due dimensioni e con diverse forme. Ciascun neurone di input è connesso a tutti i neuroni della griglia; ogni neurone di output ha il vettore dei pesi con le stesse dimensioni del vettore di input.

Interpretazione della mappa

La configurazione finale dei pesi dei singoli neuroni permette di suddividere gli elementi forniti in ingresso in cluster (raggruppamenti di oggetti simili) che di quegli elementi rappresentano una classificazione.

Occorre localizzare sulla mappa i neuroni attivi ed associarli con gli input presentati.





Alla fine dell'addestramento, ogni lettera attiverà una differente combinazione di neuroni contenente il neurone vincitore per quella lettera. Il neurone vincente sarà attivato anche da sottomissione della lettera parziale o con rumore.

Apprendimento in Som

N valori in input creano un punto $\alpha=(\alpha_{1, ..,} \alpha_N)$ in spazio a N dimensioni. Le unità di output O_i sono disposte come in esempio e sono connesse completamente a unità input con pesi w_{ij}. La regola di apprendimento competitivo seleziona come vincitore i* l'unità output avente il vettore dei pesi più vicino all'input α secondo $\left|w_{i^*} - a\right| \leq \left|w_i - a\right| \forall i$ quindi la regola è $\Delta w_{ij} = h\Lambda(i, i^*)(a_j - w_{ij}) \forall i, j$ dove $\Lambda(i, i^*) = 1$ se $i = i^*$ e il valore diminuisce con l'aumento di distanza tra le unità i e i* nel vettore di output. L'unità vincitrice i* e quelle vicine a lei hanno forti modifiche dei pesi; le unità lontane, per cui $\Lambda(i, i^*)$ assume piccolo valore, hanno piccole variazioni dei pesi.

Apprendimento in Som

La Λ è la funzione di vicinato perché contiene le informazioni topologiche dello strato di output della rete.

Trascina verso α il vettore dei pesi dell'unità vincitrice e i vettori dei pesi delle unità vicine, così la rete diventa elastica e si avvicina agli input.

Per avere veloce convergenza la Λ ha all'inizio un ampio raggio di azione e un η elevato, poi vengono gradualmente diminuiti.

Apprendimento in Som

Creare un training set con molti vettori di input significativi; stavolta non ci sono output con cui paragonare la risposta della rete.

Riepilogando per ogni vettore di input:

1. ogni neurone di output riceve le componenti del vettore input e ne calcola la distanza euclidea dal suo vettore di pesi
2. il neurone di output, che ha la minima distanza euclidea dall'input, si attiva e dà una risposta maggiore, modifica poi i suoi pesi e quelli dei neuroni vicini per avvicinarli all'input.

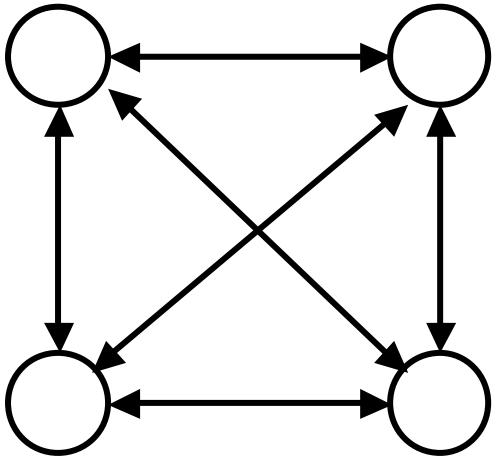
Così input simili attiveranno neuroni vicini.

Occorre stabilire quali sono i neuroni vicini e come modificare i loro pesi.

Reti di Hopfield

Hopfield nel 1982 propone una rete per memorizzare informazioni. E' composta da neuroni completamente connessi con funzione di attivazione. Ogni neurone è nodo di ingresso e di uscita, ha stato attivo o disattivo, ha funzione di attivazione a gradino

$$g(a_i) = \begin{cases} +1 \ se \ a_i > 0 \\ -1 \ altrimenti \end{cases} \qquad a_i = \sum_j w_{ij} x_j$$



Alla rete è associata una funzione energia da minimizzare durante la sua evoluzione con una successione di stati, fino a raggiungere uno stato finale stabile corrispondente al minimo della funzione energia.

Hopfield

Proprietà:

- rappresentazione distribuita su tutti i neuroni
- adatta per realizzare una memoria indirizzabile per contenuti: per recuperare un'informazione basta produrne una versione parziale o qualche indizio
- adatta per compiti di ottimizzazione vincolata, quando il problema può essere espresso in termini di una funzione obiettivo da trasformare in una funzione energia
- recupero efficiente di informazioni anche con danni alla rete

Apprendimento hebbiano

Ideato dallo psicologo Donald Hebb.

L'attivazione simultanea di due neuroni connessi determina la modifica dei pesi delle connessioni che li uniscono, in modo da aumentare la probabilità che uno dei due neuroni si attivi quando l'altro si attiva. Così si rafforza la connessione tra due neuroni molto usati e si indebolisce la connessione tra due neuroni poco attivati contemporaneamente.

Parte 4

Creare una rete
per classificare

La classificazione

Classificare significa dividere un insieme di oggetti in insiemi disgiunti secondo un criterio stabilito a priori; in genere si assegna una etichetta ad ogni insieme creato.

Il pattern recognition (riconoscimento di configurazioni) è la tecnica che consente di creare classificatori numerici e automatici.

Ogni oggetto deve essere rappresentato con un vettore di numeri per essere classificato da una rete neurale, per cui ad ogni oggetto si associa un pattern, un vettore di feature che contraddistingue univocamente l'oggetto.

Il classificatore numerico

Un classificatore numerico si può così definire: date N classi di appartenenza tra cui discriminare, il vettore di input x a L dimensioni delle feature da classificare, il vettore di uscita y che individua la classe formato da N valori, un classificatore riceve in input il vettore x e restituisce in uscita il vettore y dove $y_i=1$ se l'oggetto con input x appartiene alla classe i e $y_j=0$ per $i \neq j$, per i,j=1..N.

E' quindi un mapping, o corrispondenza tra valori di input ed output, che può essere modellato con una funzione non lineare; data la non linearità, conviene impiegare una rete neurale.

Scelta delle feature

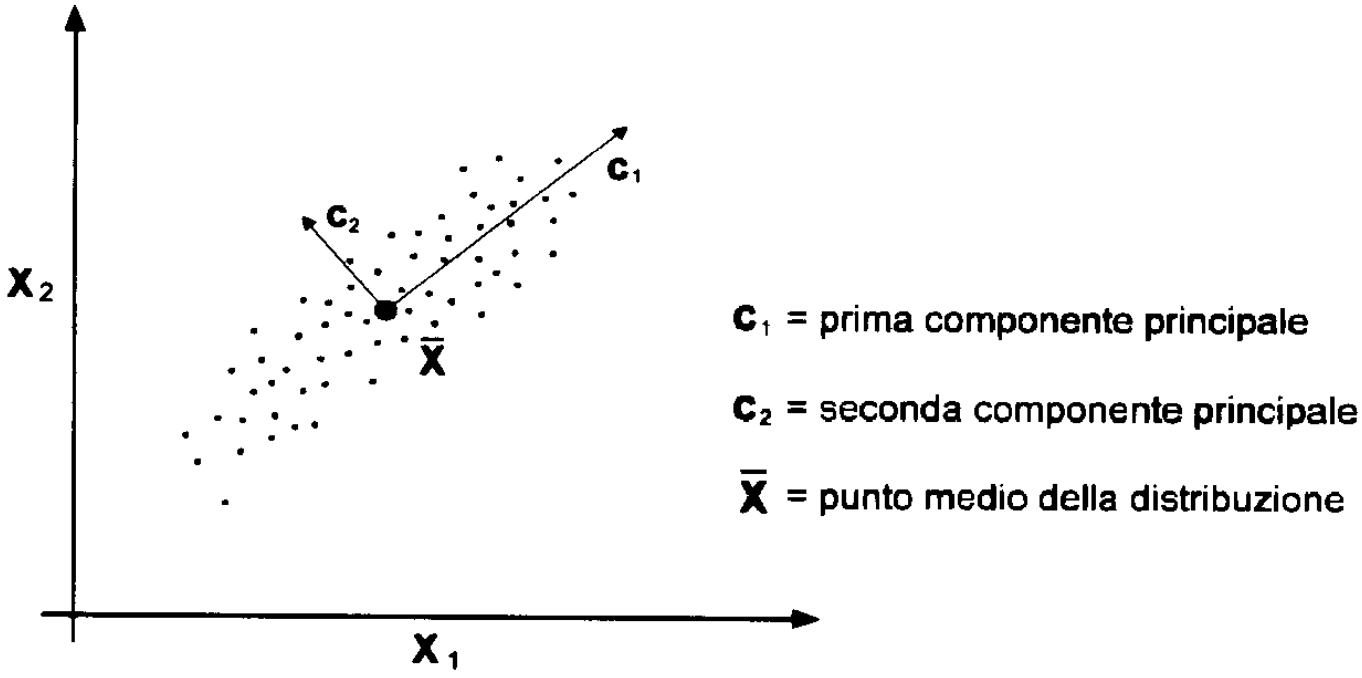
Le feature sono caratteristiche numeriche di un oggetto, ricavate misurando alcune sue proprietà significative. Devono essere scelte con le proprietà:

- discriminanza: i valori delle feature sono simili per oggetti appartenenti alla stessa classe e sono molto diversi per oggetti appartenenti a classi diverse
- indipendenza: i valori delle feature non devono essere correlati tra loro
- minimalità: devono essere il minimo numero possibile di proprietà
- disponibilità: facili e veloci da calcolare.

L'intervallo dei valori di ogni feature deve essere normalizzato in [0,1] o [-1,1] per non avere ordini di grandezza troppo diversi nell'apprendimento.

Riduzione delle feature

Ridurre il numero delle feature può velocizzare l'apprendimento (crescita di dimensionalità). Una tecnica statistica efficace è la P.C.A., l'analisi delle componenti principali dell'insieme dei valori delle feature. Per descrivere l'insieme nella figura bastano i valori delle prime due componenti.



Progettare la rete

Per un tipo di apprendimento supervisionato:

- individuare le classi in cui dividere l'input secondo il tipo di problema
- scegliere le feature analizzando matematicamente gli oggetti in input
- definire molte coppie (input, output) per i set di training (60%), validation (20%), test (20%)
- definire la codifica numerica: per l'input valori in [-1,1]; per l'output valori binari {0,1}
- scegliere alcuni criteri per valutare la qualità della risposta globale della rete sul test set

Progettare la rete

- scegliere un modello di rete e definirne l'architettura con:
  - funzione di attivazione per ogni neurone
  - numero di livelli hidden e numero di neuroni per ogni livello hidden (non esistono precise regole per determinarli, solo con tentativi e verifiche degli errori commessi)
  - numero di neuroni per lo strato input: tanti quanti i valori delle feature
  - numero di neuroni per lo strato output: tanti quante sono le classi
- scegliere valori piccoli dei pesi per favorire l'apprendimento
- scegliere un algoritmo di apprendimento e i suoi parametri di controllo (es. back propagation)
- scegliere una tecnica per controllare l'apprendimento (es. early stopping)

Usare la rete neurale

Dopo l'addestramento e la prova col test set, si congela la rete e non si modificano più i pesi. Per usarla in un impianto occorre:

- calcolare le feature del nuovo input e di cui non si conosce l'output
- passarle in input alla rete neurale che le elabora calcolando le risposte dai neuroni dal livello input verso il livello output
- la risposta della rete va interpretata per decidere quale classe ha scelto; in genere non si ha un neurone con valore 1 e tutti gli altri 0 per effetto delle funzioni di attivazione e di apprendimento: la classe assegnata è quella con valore di output più alto
- secondo il tipo di problema, se il valore di output è basso (es. 0,7 invece di 1), si può decidere di usarlo o di rifiutarlo facendo classificare ad un altro esperto

Strumenti software

- MatLab, si usano due pacchetti specialistici ben fatti:
- Neural Toolbox, http://www.mathworks.com/products/neuralnet/
- NetLab, http://www.ncrg.aston.ac.uk/netlab/
- Java, esistono vari pacchetti gratuiti su Internet, link su queste pagine:
- http://www.geocities.com/fastiland/NNwww.html
- http://www.mathtools.net/Java/Neural_Networks/
- http://diwww.epfl.ch/w3mantra/tutorial/english/index.html

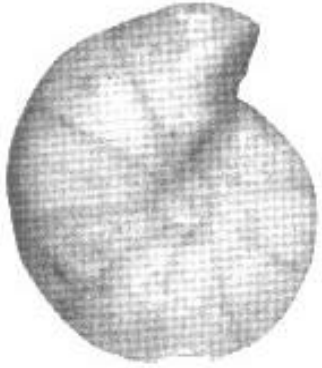
Basta scegliere nei menù le voci corrispondenti alla struttura della rete e alle funzioni desiderate; il toolbox ha algoritmi ottimizzati per l'apprendimento desiderato e fornisce l'output della rete.

Parte 5

Esempio di applicazione

Esempio di classificazione: modi di accrescimento di camere in foraminiferi

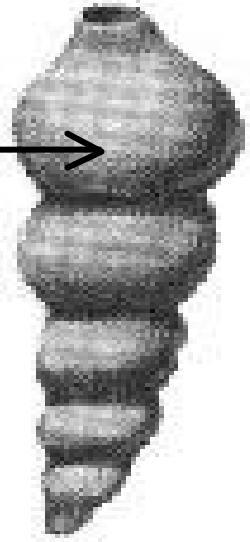
agatostego



spiralato

irregolare

uniseriato



camera

multiseriato

Obiettivo della classificazione

Classificare il modo di accrescimento delle camere di un foraminifero intero setacciato da arenaria e fotografato da microscopio

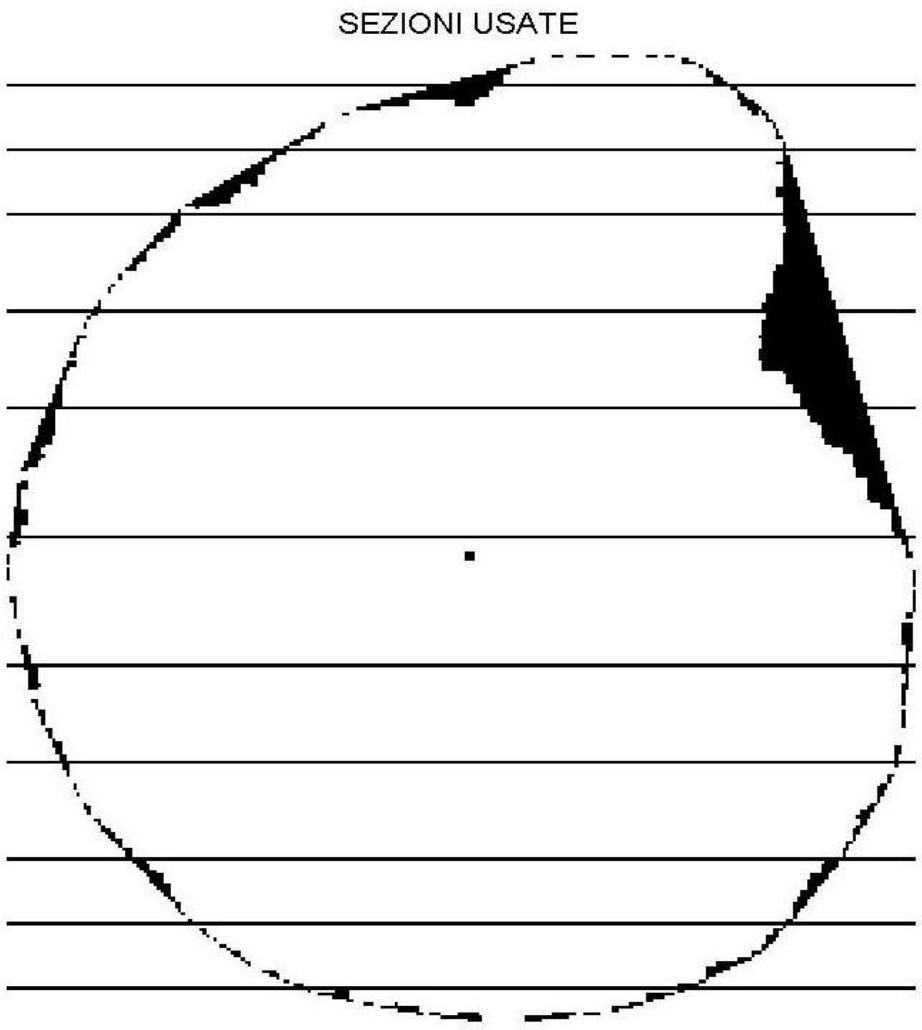
Si sceglie un apprendimento supervisionato e la Mlp perché si conoscono già le classi ed esistono numerosi esempi di classificazione presi da cataloghi di immagini.

I profili degli oggetti sono molto diversi quindi è inutile cercarne una descrizione matematica; non è possibile individuare le singole camere e contarle. Quindi si può usare:

-un campionamento non uniforme e adattivo dello spessore del guscio in varie parti

- feature calcolate su tutto il guscio

Feature scelte



In alto e in basso si misurano più spessori perché in tali zone si differenziano maggiormente gli spessori dei gusci. Altre feature:

- rapporto tra area di cerchio e area del fossile
- eccentricità del fossile (0=retta, 1=cerchio)
- spessore del guscio in pixel in 11 sezioni

totale di 13 feature ridotte a 10

Rete neurale per classificare

- ogni valore di input normalizzato in [-1,1]
- ogni valore di output assume valore in [0,1] e somma 1

- rete Mlp: 10 nodi input , 8 nodi interni, 5 nodi output
- funzione attivazione neurone: softmax
- funzione di errore: cross-entropy
- funzione minimizzazione errore: quasi-newton
- apprendimento con riduzione di errore di generalizzazione tramite early stopping
- training set di 209 immagini
- validation set di 68 immagini

Risultati della classificazione

test set di 70 immagini , 1 non classificato bene percentuale correttezza 98,57%

matrice di confusione per mostrare gli errori di classificazione

	1	2	3	4	5
1	17	0	0	0	0
2	0	17	1	0	0
3	0	0	15	0	0
4	0	0	0	4	0
5	0	0	0	0	16

riga=classe da rete
colonna=classe reale
1=agatostego
2=seriale
3=multiseriale
4=irregolare
5=spiralato

Esempi su sequenze di numeri

- Predizione: una serie di dati può essere data in input ad una rete neurale per fargli decidere quale sarà il prossimo valore, in base al modello statistico che la rete neurale si crea con i dati precedentemente usciti

- Interpolazione: un sensore fornisce un dato ma ogni tanto si guasta e non fornisce il dato; una rete neurale può prendere in input tutti i dati validi forniti dal sensore e ricavare i dati mancanti, in base al modello statistico che la rete neurale si crea con tutti i dati