



Università degli Studi di Modena e Reggio Emilia
Facoltà di Scienze della Comunicazione e dell'Economia
Corso di Laurea Specialistica in Economia e Gestione delle Reti e dell'Innovazione

Appunti sui percettroni

Fabio Ruini – matricola nr: 7496

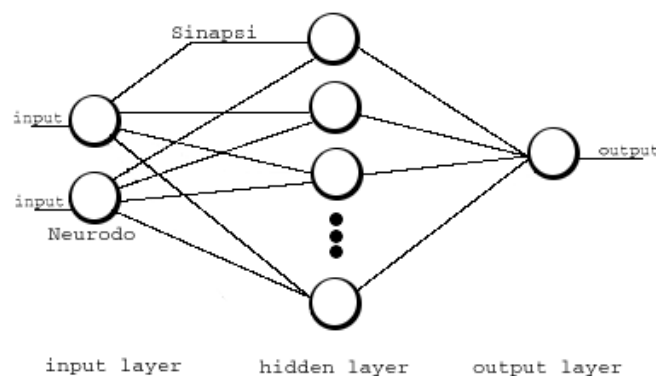
e-mail: fabio.ruini@unimore.it

web: <http://www.fabioruini.eu>

Percettroni

Il [modello di Hopfield](#), per quanto estremamente interessante ed utile in determinate circostanze, presenta diverse limitazioni. Una limitazione intrinseca, che emerge nel momento in cui lo si utilizza allo scopo di operare delle classificazioni, consiste nel fatto che il modello di Hopfield permette di classificare assieme soltanto quei pattern che si somigliano. Ciò significa che la rappresentazione che forniamo al modello deve già essere, in qualche modo, adatta al compito.

Reti di tipo diverso possono tuttavia superare questa limitazione. Dotando una rete neurale di una struttura, che consente di dividere concettualmente lo strato di input da quelli/o intermedi/o e da quello di output, diventa possibile far emergere nella rete la capacità di “astrarre” a partire da rappresentazioni inappropriate.



[Immagine tratta da: <http://edu.os3.it/html/manual/impararec/node584.html>]

La figura riportata qui sopra mostra una delle strutture “tipiche” di una rete neurale: uno strato di input (composto in questo caso da 2 neuroni), uno strato intermedio (spesso chiamato anche “strato nascosto”) ed uno strato di output (caratterizzato dalla presenza di un unico neurone).

Ancora più semplice, tuttavia, è il caso della rete neurale composta da due soli strati: uno di input ed uno di output, quest'ultimo con un solo neurone di tipo booleano. Seguendo la definizione fornita da Floreano e Matteucci (*“Manuale sulle reti neurali”*, 1996 - ed. “Il Mulino” - pag. 68), una rete neurale con un unico strato di connessioni unidirezionali dai nodi di input ai nodi di uscita è detta **percettrone**.

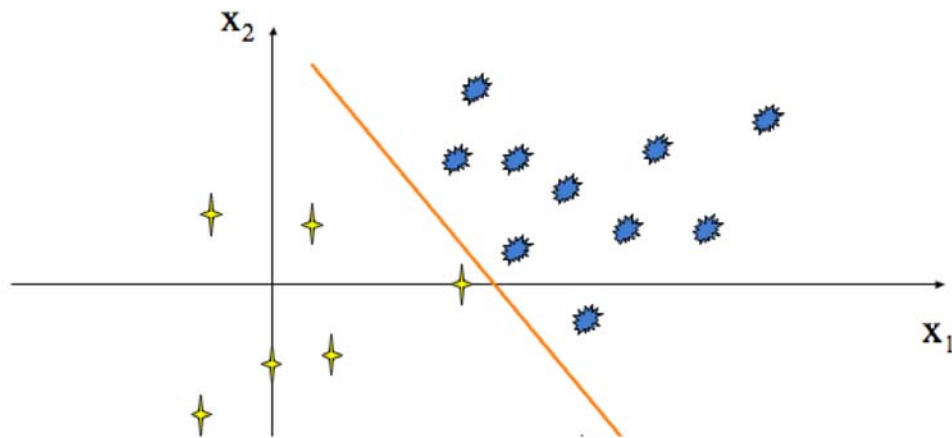
Come si evince dal nome, il neurone booleano è in grado di produrre soltanto due pattern di output, sintetizzabili in “attivo” o “non attivo”. La sua regola di decisione è implementata mediante un comportamento “a soglia”: se la sommatoria dei pattern di attivazione dei singoli neuroni che compongono lo strato di input, ponderata per i rispettivi pesi, oltrepassa una certa soglia, allora il neurone di output adotterà il pattern di output “attivo”. Viceversa, il neurone di output si manterrà nello stato “non attivo”.

Se definiamo come $I(t)$ il pattern di attivazione dello strato di input, come $x(t+1)$ il pattern di output del neurone booleano e come σ il valore di soglia, da un punto di vista formale avremo che:

$$I(t) > \sigma \Rightarrow x(t+1) = 1$$

$$I(t) \leq \sigma \Rightarrow x(t+1) = 0$$

Immaginando di avere due soli neuroni di input, con la scelta di una scala opportuna diventa possibile rappresentare su uno spazio bi-dimensionale i possibili patterns di attivazione di questi neuroni.



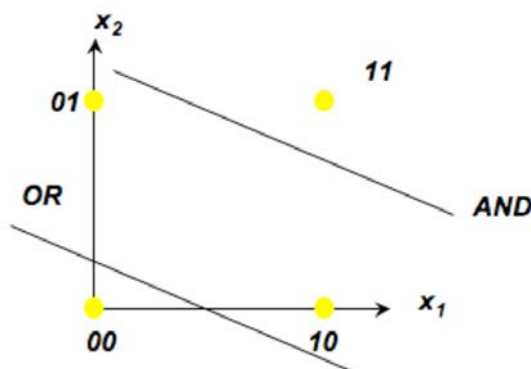
Come si vede nella figura qui sopra, la regola di decisione del neurone booleano corrisponde alla suddivisione dello spazio delle “features” di input, operata mediante un iperpiano. L’equazione che regola la creazione di un iperpiano di dimensioni N-1 in uno spazio N-dimensionale è:

$$\sum w_k x_k - T$$

dove:

- w_k rappresenta il peso sinaptico che collega il neurone di input k-esimo all’unico neurone di output della rete;
- x_k identifica il valore (0 o 1) del neurone di input k-esimo;
- T è una costante.

Se, oltre al neurone di output, anche l’input della rete neurale è di tipo booleano, utilizzare la rete neurale per eseguire una classificazione equivale a determinare una funzione booleana del vettore d’ingresso. Tale funzione assume il valore 1 in corrispondenza dei casi che oltrepassano il valore di soglia, 0 in caso contrario. Con due neuroni booleani di input ed uno di output è ad esempio possibile rappresentare, in maniera estremamente intuitiva, le funzioni AND e OR:



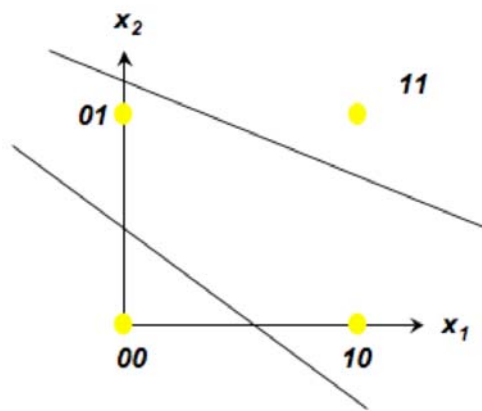
Tutti i punti che stanno al di sopra dell’iperpiano assumono valore 1/VERO, mentre quelli che si trovano al di sotto assumono il valore 0/FALSO.

Con un semplice “trucco” è possibile descrivere allo stesso modo anche quelle funzioni (ad es. NAND e NOR) nelle quali il valore 1 è associato al semipiano inferiore. Partendo da un iperpiano che realizza la funzione opposta rispetto a quella desiderata (ad esempio AND in luogo di NAND), si impongono come nuovi parametri $w_k = -w_{kold}$ e $T = -T_{old}$. A questo punto abbiamo che:

$$\sum w_k x_k - T = -(\sum w_{kold} x_k - T_{old})$$

e, pertanto, il valore 1 viene associato al semispazio inferiore.

Alcune funzioni booleane non sono tuttavia replicabili mediante una struttura di rete come quella vista sino a qui. Le funzioni XOR ed identità, ad esempio, non sono separabili: per isolarle occorrerebbero due rette, implementabili soltanto attraverso il ricorso ad una struttura di rete più complessa.



Nelle applicazioni reali dei perceptron, accade quasi sempre che i neuroni dello strato di input siano caratterizzati da valori continui, mentre il neurone di output è booleano e realizza la consueta funzione $H(\text{input-soglia})$. Esempio tipico è quello della classificazione di segnali.

Apprendimento nei perceptron

Così come accade per la maggior parte dei modelli di reti neurali, anche nei perceptron è possibile realizzare una funzione di “apprendimento” che si basa sulla modifica dei pesi sinaptici delle connessioni.

All’inizio della fase di training, i pesi w delle connessioni sinaptiche del perceptrone assumono valori completamente casuali. Per l’addestramento abbiamo a disposizione una serie di esempi con la relativa, corretta, classificazione. Alla rete vengono presentati, a turno, i diversi casi da classificare e la rete elabora ogni volta il suo responso (sopra la soglia oppure sotto la soglia). Nel caso in cui la classificazione risulti essere corretta (output della rete uguale a quello previsto), l’algoritmo di training non interviene. Al contrario, se la classificazione risulta essere errata, l’algoritmo provvede a modificare i pesi sinaptici, nel tentativo di migliorare la performance “classificativa” della rete.

Così come nel modello di Hopfield, anche nei perceptron l’apprendimento equivale dunque ad una modifica dei pesi sinaptici. Ma qui si tratta di una modifica graduale e dipendente dalle risposte della rete. In un certo senso è come se il perceptrone vivesse all’interno di un universo creato dall’insegnante e cercasse di adattarsi alle caratteristiche di questo universo.

Se la funzione che si desidera realizzare nel perceptrone è effettivamente realizzabile, ci si pone il problema di come fare a trovare automaticamente un valore dei pesi che consente di implementarla. Nel caso di un perceptrone ad N ingressi continui ed una uscita booleana, senza strati intermedi, l'output è dato da:

$$y = H(\sum w_k x_k - T)$$

Formalmente, quello che si fa è introdurre nel perceptrone un neurone di input in più (che chiameremo x_0) ed il corrispondente peso w_0 . Imponendo che $x_0 = 1$ e $w_0 = -T$, la regola appena abbozzata può essere scritta, in forma più compatta, come:

$$y = H(\sum w_k x_k)$$

Possiamo a questo punto dare una definizione formale della regola di apprendimento ("locale", per motivi che approfondiremo in seguito) del perceptrone. Siano:

- y : l'output del perceptrone;
- y_d : l'output desiderato;
- e : un numero reale maggiore di zero;
- W : il vettore dei pesi sinaptici del perceptrone;
- X : l'insieme dei patterns di input x .

Per ogni pattern x presentato alla rete viene applicata la regola del perceptrone:

$$y = H(\sum w_m x_m)$$

L'output y ottenuto viene quindi confrontato con quello desiderato y_d . I pesi sinaptici vengono modificati di conseguenza, secondo le seguenti regole:

$$y = y_d \Rightarrow W(k+1) = W(k)$$

$$y > y_d \Rightarrow W(k+1) = W(k) + ex$$

$$y < y_d \Rightarrow W(k+1) = W(k) - ex$$

Il procedimento continua fino a quando il vettore dei pesi W non rimane costante per un certo numero di passi.

Così come dovrebbe essere intuitivo, l'algoritmo appena definito fa sì che:

- se l'output y ottenuto è inferiore rispetto a quello desiderato y_d , i pesi del perceptrone vengano modificati in maniera tale che, alla successiva presentazione del pattern x , la sommatoria degli input ricevuti dal neurone di output sia più alta rispetto a quella precedente, con maggiori probabilità che la risposta sia $y=1$ come desiderato. Formalmente:

$$W(k+1) * X = W(k) * X + e|X|^2 > W(k) * X;$$
- se l'output y ottenuto è superiore rispetto a quello desiderato y_d , i pesi del perceptrone vengano modificati in maniera tale che, alla successiva presentazione del pattern x , la sommatoria degli input ricevuti dal neurone di output sia più bassa rispetto a quella

precedente, con maggiori probabilità che la risposta sia $y=0$ come desiderato. Formalmente:

$$W(k+1) * X = W(k) * X + e|X|^2 < W(k) * X .$$

Se la funzione booleana è una funzione lineare a soglia (ovvero se è linearmente separabile), allora la regola locale di apprendimento del perceptrone è in grado di trovare, in un numero finito di passi, un insieme di pesi capace di realizzarla.

Questo teorema, detto “**teorema del perceptrone**” è applicabile anche nel caso della regola “globale”, che modifica il vettore dei pesi sinaptici W , non in corrispondenza di un singolo vettore di ingresso, ma in funzione del comportamento del perceptrone sull’intero insieme dei vettori di input.

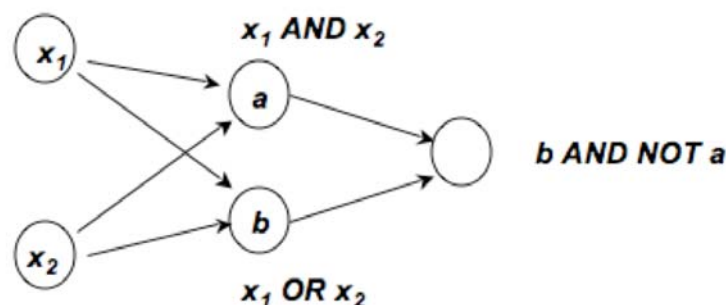
Tuttavia, non tutte le funzioni sono separabili. A dire il vero, come si evince dalla tabella che segue, esse sono decisamente poche e la loro proporzione sul totale delle funzioni realizzabili tende a zero al crescere del numero di bit:

n	$f. \text{ booleane } (f(n))$	$f. \text{ separabili } (c(n))$	$\text{proporzione } c/f$
2	16	14	0.9
3	256	104	0.4
4	65536	1882	0.03
6	$1.8 * 10^{19}$	$1.5 * 10^7$	10^{-12}
8	$1.16 * 10^{77}$	$1.7 * 10^{13}$	10^{-64}

Si potrebbe pensare di mettere in parallelo più perceptron: la struttura risultante potrebbe così apprendere un maggior numero di funzioni, tutte comunque appartenenti al sottoinsieme delle funzioni linearmente separabili.

Per poter realizzare una più ampia classe di funzioni è necessario introdurre nel perceptrone, tra lo strato di input e quello di output, dei neuroni intermedi che permettano di realizzare una sorta di rappresentazione interna dell’input. Il perceptrone risultante viene detto “**multi-layer perceptron**” (**MLP**).

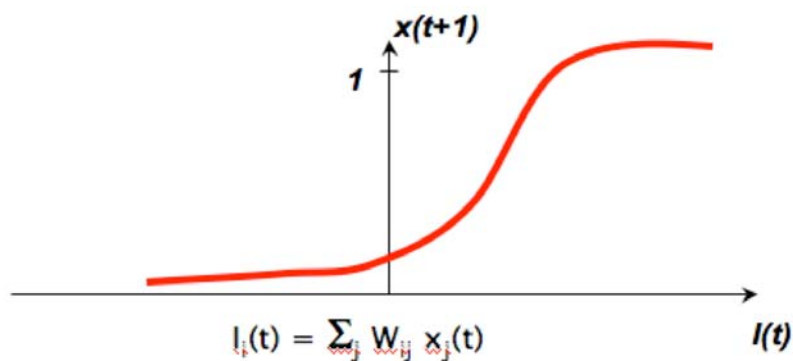
Un perceptrone multi-strato come quello raffigurato qui sotto, ad esempio, è in grado di realizzare la funzione XOR, che abbiamo visto in precedenza non essere realizzabile attraverso un perceptrone semplice.



Una configurazione di questo genere permette ai due neuroni di specializzarsi ciascuno su di una particolare funzione logica. Nel caso dello XOR, ad esempio, i due neuroni possono rispettivamente realizzare le funzioni logiche AND e OR.

I neuroni biologici esibiscono un comportamento “a soglia”, ovvero del tipo “o tutto o niente”, anche se attualmente non è ancora ben chiaro come all’interno del sistema nervoso sia codificata l’informazione. In alcuni casi, ad esempio, si è visto che essa dipende dalla frequenza di firing, ossia dall’intervallo di tempo che passa dallo “sparo” di una corrente elettrica lungo la sinapsi di un neurone al “colpo” successivo. In sistemi artificiali come quelli che stiamo studiando, tuttavia, può avere un senso anche considerare neuroni con funzioni di trasferimento continue.

Per modellizzare il comportamento di un neurone continuo non si può ovviamente utilizzare una funzione a gradino, ma neppure una funzione illimitata. La scelta migliore prevede l’utilizzo di una funzione limitata sia superiormente che inferiormente; tali funzioni vengono dette, in gergo, “squashing functions”.



Molti problemi della vita reale si prestano ad essere descritti mediante input di tipo continuo (o discretizzati con molti valori) oppure misti (binari e continui). I perceptron continui sono spesso utilizzati con successo per effettuare compiti di classificazione (ad esempio quelli relativi alla diagnosi medica) o come approssimatori di funzioni a valori reali. Un famoso teorema ha dimostrato che i perceptron multi-strato continui possono essere considerati degli “approssimatori universali”: una qualunque funzione continua a valori reali può essere approssimata uniformemente da un perceptrone con uno strato intermedio, un neurone di output e funzioni di trasferimento sigmoidali. Ulteriori derivazioni del teorema originale hanno poi dimostrato che esso è valido anche per funzioni continue “a tratti”.

Backpropagation

L’apprendimento dei MLP avviene con un procedimento analogo a quello utilizzato per i perceptron semplici:

- presentazione di un pattern appartenente al training set;
- verifica della risposta;
- eventuale correzione dei pesi sinaptici;
- presentazione del pattern successivo, fino a soddisfare un certo criterio di performance.

La difficoltà consiste nel fornire una regola di apprendimento per gli strati nascosti. La regola del perceptrone semplice tende infatti ad avvicinare la risposta del neurone di uscita a quella desiderata ma, nel caso a più strati, soltanto per reti estremamente semplici possiamo avere un’idea di quale sia il “valore desiderato” applicabile ai neuroni intermedi.

Immaginiamo di trovarci di fronte ad una rete “elementare”, composta da un neurone di ingresso ed uno di output, il cui output è $y = f(I)$, con $I = wx$. Presentando diversi esempi di x , la rete calcola

ogni volta il corrispondente valore di y , il quale viene a sua volta confrontato con quello desiderato y_d . Per ogni pattern presentato, l'errore è misurato da $E = (y - y_d)^2$, che risulta essere una funzione differenziabile.

Una regola semplice per cercare il punto di minimo di una funzione incognita è quella di modificare la variabile indipendente di una quantità proporzionale alla derivata, cambiata di segno. Sappiamo che, dato il pattern x , la misura di errore E dipende dai pesi w :

$$\frac{dE}{dw} = \frac{d(y - y_d)^2}{dw} = 2(y - y_d) \frac{df}{dI} x$$

$$\Delta w = -\varepsilon \frac{dE}{dw} = -2\varepsilon(y - y_d) \frac{df}{dI} x$$

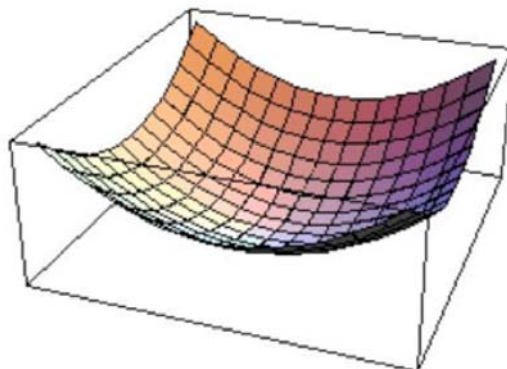
Poiché $\frac{df}{dI} \geq 0$, se $y > d$ allora $\Delta w \approx -x$ e di conseguenza il valore di I diminuisce. In caso contrario, $\Delta w \approx x$ e quindi il valore di I aumenta. Il comportamento ricorda da vicino quello del perceptrone.

Generalmente, non è presente soltanto un neurone di output, ma ve ne sono diversi. Per questo motivo, non potendo calcolare una misura di errore del tipo $E = (y - y_d)^2$, si definisce una distanza tra il pattern di uscita ottenuto e quello desiderato, cercando poi di minimizzare tale distanza.

La distanza tra il pattern di uscita ottenuto e quello desiderato viene spesso misurata con il quadrato della distanza euclidea. Ipotizzando che lo strato di output sia composto da R neuroni, l'errore su uno dei pattern di esempio appartenenti al training set risulta essere:

$$E(W) = \sum_{r=1}^R (y_r - y_r^d)^2$$

Impostata in questo modo, la funzione di costo E dipende da tutti i pesi sinaptici della rete. Per trovare il punto di minimo di una funzione nota di molte variabili è possibile porre a zero tutte le sue derivate parziali rispetto alle diverse variabili. Nel caso di una funzione incognita, invece, una tecnica molto diffusa è quella della discesa secondo gradiente: si calcola lungo quale direzione la funzione diminuisce più rapidamente e si "sposta" il valore delle variabili, di una certa quantità, in quella direzione.



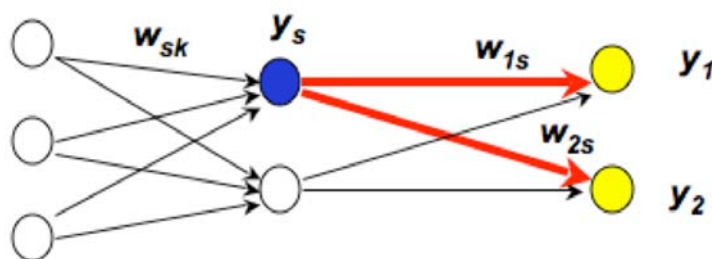
Si tratta di un classico problema di ottimizzazione: individuare i valori estremi di una funzione di molte variabili. Il metodo della discesa secondo gradiente è un metodo molto semplice e che risulta efficace per localizzare rapidamente il più vicino punto di minimo locale. Questa tecnica, tuttavia, presenta l'inconveniente di poter rimanere teoricamente bloccata all'interno di un minimo locale, senza riuscire ad individuare quello globale.

Qualunque sia l'architettura del perceptrone che stiamo analizzando, sappiamo che gli output sono determinati con una funzione del tipo:

$$y_r = f\left(\sum w_{rk}x_k\right)$$

dove w indica i pesi che collegano l'ultimo strato intermedio a quello di uscita. Sappiamo inoltre che, presentando un pattern di addestramento alla rete, sono noti tutti i valori delle attivazioni x dell'ultimo strato intermedio.

E' quindi possibile calcolare direttamente le derivate di E rispetto ai pesi sinaptici w che uniscono l'ultimo strato intermedio a quello di output, allo scopo di aggiustare questi ultimi. Ma si può fare di più. Si consideri un perceptrone MLP come quello mostrato in figura:



L'errore E dipende dai valori di attivazione di y_1 ed y_2 . y_1 ed y_2 sono a loro volta influenzati dai pesi w_{1s} e w_{2s} e dai valori di attivazione di y_s . Ma anche da w_{sk} attraverso l'influenza che esso ha su y_s e, a cascata, su w_{1s} e w_{2s} .

Calcolando le derivate rispetto ai pesi più interni, si possono così determinare le correzioni ai pesi fra lo strato intermedio e l'output; le correzioni ai pesi tra l'ingresso e lo strato intermedio dipendono a loro volta dai pesi verso l'output. Il meccanismo così definito, detto di **backpropagation** (o "retropropagazione del gradiente") consente dunque di calcolare le modifiche ai pesi verso i neuroni nascosti. Essendo un algoritmo più complicato rispetto a quello utilizzato per addestrare un perceptrone semplice si rendono necessarie diverse rappresentazioni dell'intero training set. La validità delle performance della rete viene infine valutata su un insieme di esempi nuovi (generalizzazione), che la rete non ha ancora "visto".

Per la bontà dell'apprendimento è necessario che gli esempi forniti alla rete siano rappresentativi e non contraddittori. Ad esempio, una rete neurale (così come un essere umano) avrebbe enormi difficoltà nell'individuare la regola sottostante questo training set:

<i>esempio in input</i>	<i>output desiderato</i>
1	1
2	1
8	0
3	1
10	0

4	1
11	0
7	0
6	1
5	?

Potrebbero infatti essere valide entrambe le regole “ $y < 7$ ” e “i divisori di 12”.

Si considerino infine i problemi legati all’overfitting ed all’underfitting. Utilizzando una rete con un numero elevato di neuroni è possibile pervenire ad una classificazione molto precisa dell’insieme di addestramento: i confini di separazione tra le varie classi possono essere molto frastagliati e tali da adattarsi perfettamente all’insieme dei casi presenti nel training set. Tuttavia, una rete di questo tipo può mostrare scarse capacità di generalizzazione (overfitting). Problemi analoghi, anche se di senso contrario, si possono avere con una rete il cui numero di neuroni è insufficiente (underfitting).