Percettroni

Percettroni

- **Corso di laurea in Informatica**
  (anno accademico 2024/2025)
  - Insegnamento: Apprendimento ed evoluzione in sistemi artificiali
  - Docente: Marco Villani

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA
Dipartimento di Scienze Fisiche, Informatiche e Matematiche

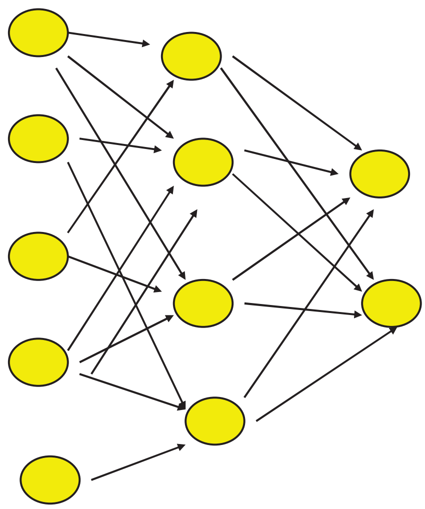
Limitazioni del modello di Hopfield

- Prestazioni inferiori ad altri modelli in termini di numero di pattern che possono essere memorizzati, a parità di numero di neuroni
  - scalano linearmente con N; mentre la capacità di codifica scala con 2^N; l' onere computazionale scala con N^2
- interferenze fra pattern diversi, "confusione"
- mancanza di struttura e di distinzione fra neuroni di input, interni e di output
- elevato numero di connessioni
- possibilità di intrappolamento in minimi locali
- l' "apprendimento" è limitato
  - scrittura diretta nelle connessioni del contributo dovuto a un singolo pattern
  - nessun feedback basato sulle prestazioni della rete

Una limitazione intrinseca

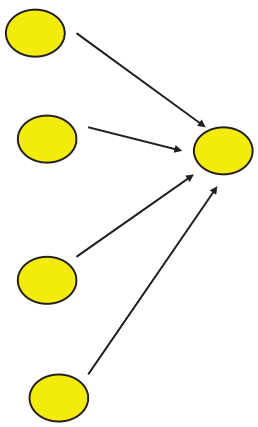
- il modello di Hopfield consente di classificare assieme quei pattern che si somigliano
- ciò significa che la codifica che forniamo deve già essere adatta al compito
  - due pattern che" stanno assieme" devono essere già simili
  - Controesempio: la parità
- reti di tipo diverso possono superare questa limitazione, imparando ad astrarre da rappresentazioni inappropriate
- per fare questo è necessario dotare la rete di una struttura

Reti a strati: la semantica dei neuroni



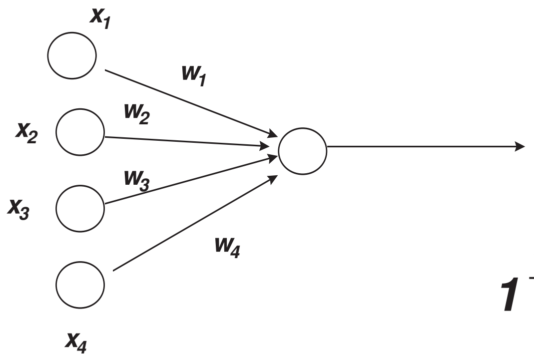
- uno strato di neuroni di ingresso
  - ad esempio, una retina artificiale
  - un insieme di sintomi
- eventualmente, uno o più strati interni
  - dal significato non predefinito
- uno strato di output, classificazione
  - ad esempio, viso maschile / viso femminile
  - diagnosi diverse, p.es. influenza o febbre di Lassa

La struttura più semplice

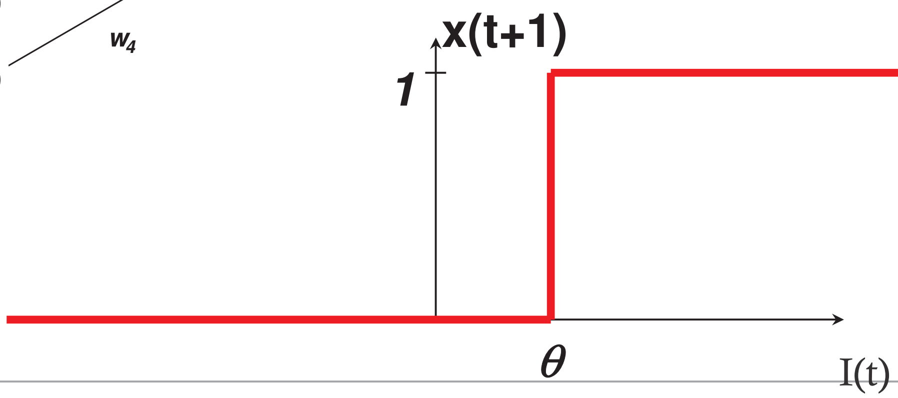


- la rete più semplice è quella che è composta di due soli strati
  - input
  - output (anche un solo neurone)
- il neurone booleano può servire per distinguere gli input in due classi
- il neurone booleano realizza una regola di decisione che corrisponde alla suddivisione dello spazio delle features di input mediante un iperpiano

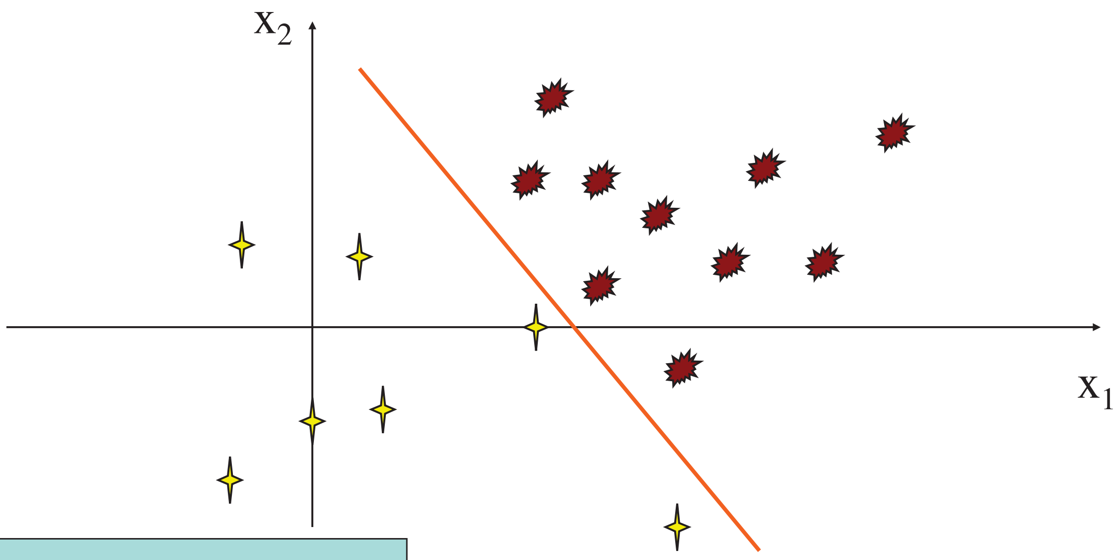
Neuroni booleani



$$I = w_1x_1 + w_2x_2 + w_3x_3 \ldots$$



La regola di decisione corrisponde a un iperpiano



Input continui, output booleano

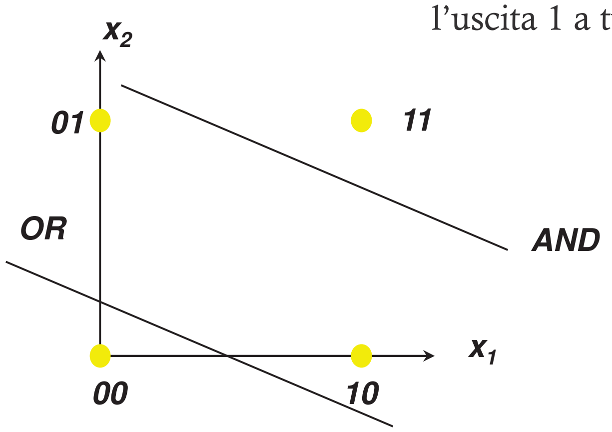
Un percettrone booleano

- Nel caso in cui anche l' input sia binario, eseguire una classificazione equivale a determinare una funzione booleana del vettore di ingresso
  - che vale 1 in corrispondenza dei casi sopra soglia
  - che vale 0 in corrispondenza dei casi sotto soglia
- esempio

0 0 --> 0	0 0 --> 0
0 1 --> 0	0 1 --> 1
1 0 --> 0	1 0 --> 1
1 1 --> 1	1 1 --> 0
AND	XOR

Un percettrone booleano

- interpretazione geometrica: $\sum w_k x_k - T = 0$ è l' equazione di un iperpiano di dimensioni N-1 in uno spazio N dimensionale
  - la regola del neurone booleano a soglia associa l'uscita 1 a tutti i punti del semipiano superiore

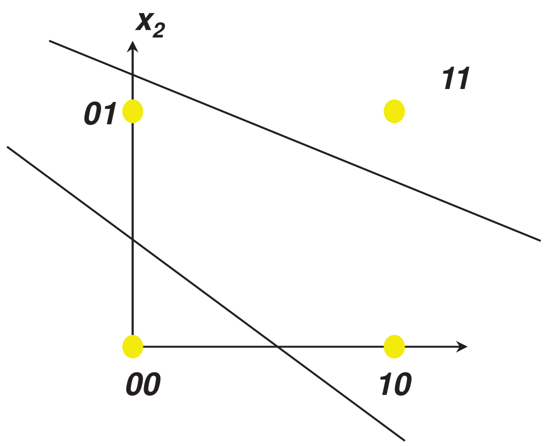


Semipiano inferiore

- anche le funzioni in cui il valore "1" è associato al semipiano inferiore (p.es. NAND, NOR) sono realizzabili con funzioni lineari a soglia
- si parte da un iperpiano che realizza la funzione opposta (p.es. AND, OR); siano $w_{k,old}$, k=1, ... N e T_{old} i valori dei parametri
- si scelgono $w_k = -w_{k,old}$, $T = -T_{old}$ come nuovi parametri, quindi
- $\sum w_k x_k - T = -(\sum w_{k,old} x_k - T_{old})$
- e pertanto il valore y=1 è associato al semispazio inferiore

Funzioni non separabili linearmente

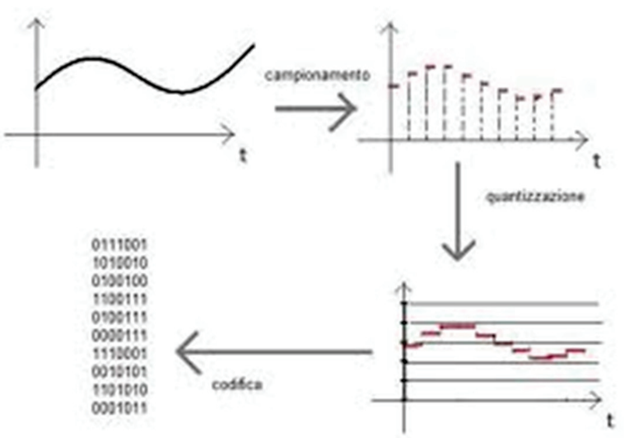
- le funzioni XOR e identità non sono **linearmente separabili**; occorrono due rette per isolarle, quindi una struttura di rete più complessa

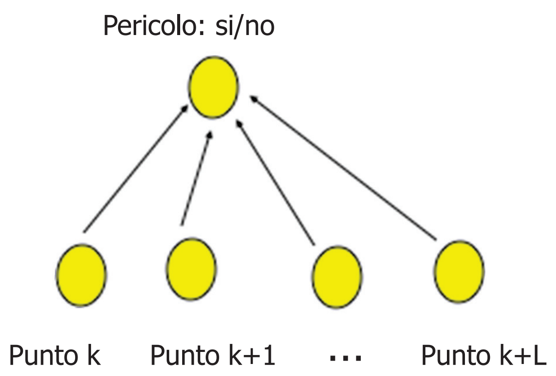


Funzioni non separabili linearmente

- Nelle applicazioni accade quasi sempre che i neuroni dello strato di input abbiano valori continui
- il neurone di output è spesso booleano
  - realizza la consueta funzione H(input-soglia)
- esempi:
  - classificazione di segnali
  - Diagnosi
  - Riconoscimento di immagini
  - …

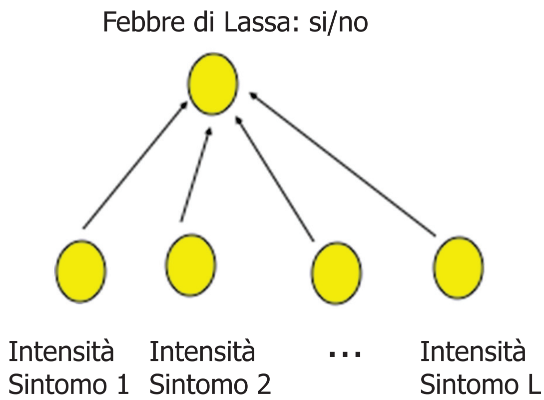
Analisi di segnali





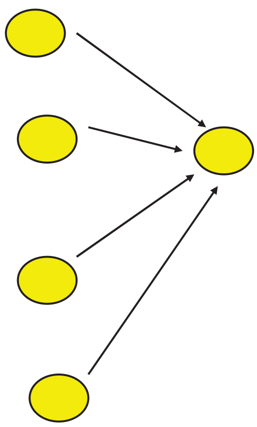
Una struttura con più neuroni di output in parallelo consente di valutare simultaneamente più ipotesi

Diagnosi



Una struttura con più neuroni di output in parallelo consente di valutare simultaneamente più ipotesi

Apprendimento (con output booleano)



- all' inizio i pesi w assumono valori casuali
- abbiamo a disposizione per l' addestramento una serie di esempi, con la corretta classificazione
  - alla rete vengono quindi presentati i diversi casi da classificare
- in corrispondenza di un certo "caso" in ingresso, la rete fornisce una classificazione (sopra soglia / sotto soglia)

Apprendimento (con output booleano)

- se la classificazione è corretta, non si fa nulla
- se la classificazione è sbagliata, si modificano i pesi in modo da migliorare le prestazioni
  - l'apprendimento equivale ad una modifica dei pesi
- come in Hopfield, ma qui è
  - graduale
  - dipendente dalle risposte della rete
- in un certo senso, la rete vive in un universo creato dall'insegnante e cerca di adattarsi alle sue caratteristiche

Apprendimento (con output booleano)

- come trovare automaticamente un valore dei pesi che consenta di realizzare la funzione desiderata (ammesso che essa sia realizzabile) ?
- caso del percettrone a N ingressi continui, una uscita booleana, senza strati intermedi

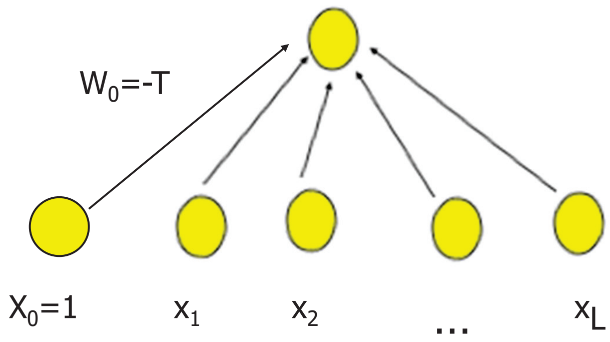
  $y = H(\Sigma w_k x_k - T))$

Apprendimento (con output booleano)

- formalmente, si introduce un neurone in più in ingresso (x_0) e il corrispondente peso w_0; con $x_0 = 1$ e $w_0 = -T$
- quindi la regola può essere scritta in forma compatta

  $y = H(\Sigma w_k x_k)$

  - si noti che, nel percettrone considerato, il vettore dei pesi W e quello dei valori di ingresso X hanno tutti la stessa dimensione (N+1)



Regola del percettrone (locale)

- Sia y l'output del percettrone, y_d quello desiderato e sia $\varepsilon > 0$
- scegli un vettore iniziale di pesi W(0)
- finché W non resta costante per un certo numero di volte consecutive, ripeti
- begin
  - per ogni pattern di input x
    - applica il percettrone ($y = H(\sum w_m x_m)$)
    - se $y = y_d$, $\mathbf{W}(k+1) = \mathbf{W}(k)$
    - se $y < y_d$, $\mathbf{W}(k+1) = \mathbf{W}(k) + \varepsilon\mathbf{x}$
    - se $y > y_d$, $\mathbf{W}(k+1) = \mathbf{W}(k) - \varepsilon\mathbf{x}$
  - incrementa k
- end

Intuitivamente

- se l' output ottenuto y è inferiore a quello voluto y_d, la modifica fa sì che, alla prossima presentazione del pattern x

$$W(k+1) \bullet X = W(k) \bullet X + \varepsilon |X|^2 > W(k) \bullet X$$

- per cui l' input al neurone è maggiore, ed è possibile che superi la soglia e fornisca y=1, come desiderato
  - DEF: $W \bullet X = \sum_k W_k X_k$
- analogamente, nel caso in cui $y>y_d$

$$W(k+1) \bullet X = W(k) \bullet X - \varepsilon |X|^2 < W(k) \bullet X$$

- tende a portare l' input sotto soglia

Teorema del percettrone

- se la funzione booleana è una funzione lineare a soglia (linearmente separabile) allora la regola locale di apprendimento del percettrone è in grado di trovare un insieme di pesi che la realizza in un numero finito di passi (con ε costante o con $\varepsilon \approx 1/k$)

- il teorema del percettrone si applica anche alla regola globale (modifica W non in corrispondenza di un singolo vettore di ingresso, ma in funzione del comportamento sull'insieme dei vettori di ingresso)

Apprendimento globale

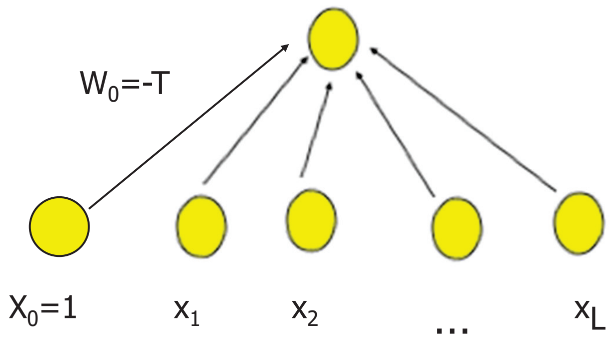
- Scegli un vettore iniziale di pesi W(0)
- finché W(k+1) # W(k), ripeti
- begin
  - per ogni pattern di input x_i
    - applica il percettrone ($y_i = H(\Sigma w_m(k) x_{i,m})$)
    - se $y_i = y_{d,i}$, $\boldsymbol{\Delta_i} = \mathbf{0}$
    - se $y_i < y_{d,i}$, $\boldsymbol{\Delta_i} = \mathbf{x}$
    - se $y_i > y_{d,i}$, $\boldsymbol{\Delta_i} = \mathbf{-x}$
  - $W(k+1) = W(k) + \varepsilon \Sigma_i \Delta_i$
  - incrementa k
- end

Apprendimento globale

- Fase di addestramento: si aggiornano i pesi sinaptici
  - Training set: insieme di esempi usati per adattare i pesi della rete
  - Test set: insieme su cui viene valutata la qualità dell'apprendimento
- In modalità di riconoscimento/classificazione il sistema funziona a pesi costanti

Limiti

- l'algoritmo del percettrone è efficace per reti che hanno solo i due strati di input e output
- esso si basa sul confronto fra il valore dell'output ottenuto e quello desiderato
- quindi non può essere applicato al caso di una rete che abbia anche uno o più strati nascosti
- perché non conosciamo il "valore desiderato" dei neuroni intermedi



Neuroni continui

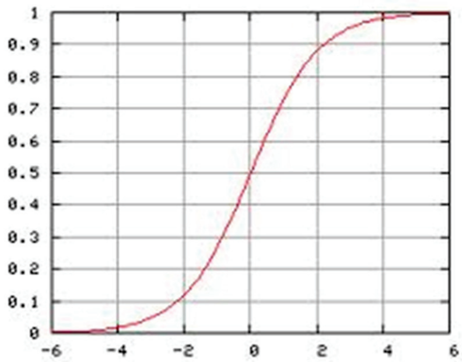
- Esiste un efficace algoritmo di apprendimento anche per reti con più strati
  - retropropagazione del gradiente, *gradient backpropagation*
    - lo discuteremo entro poco
- che si applica a neuroni continui
- i neuroni booleani possono essere approssimati da neuroni continui con una funzione di attivazione sigmoide ripida
- Inoltre, in molti problemi è naturale utilizzare output continui
- è quindi interessante considerare neuroni con funzioni di trasferimento continue del tipo $y = f(\Sigma\, w_k x_k - T)$

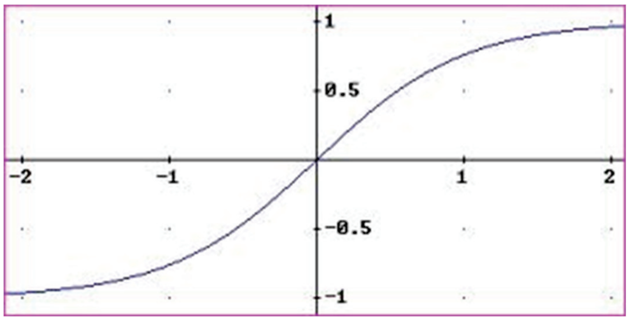
Neuroni con output continui

- squashing functions: funzioni crescenti, limitate sia superiormente che inferiormente

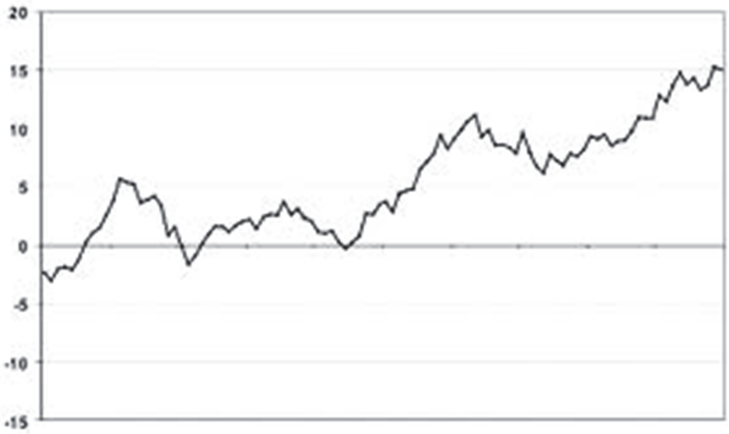
$$f(x) = \frac{A}{1 + be^{-kx}}$$

$$f(x) = A \frac{e^{kx} - e^{-kx}}{e^{kx} + e^{-kx}}$$

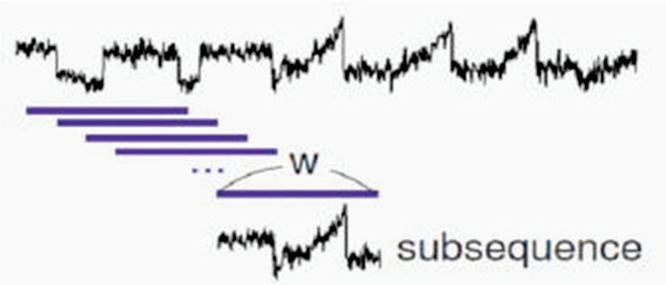




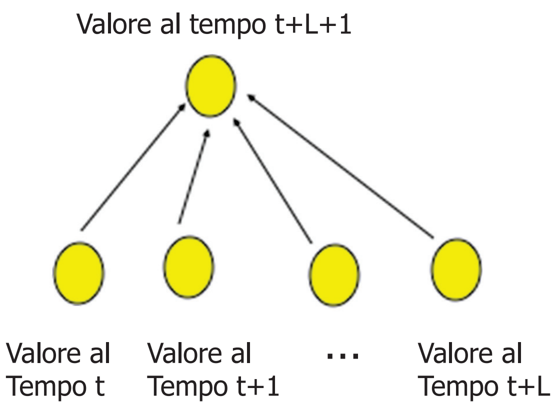
Esempio: previsione di serie temporali



Esempio: previsione di serie temporali



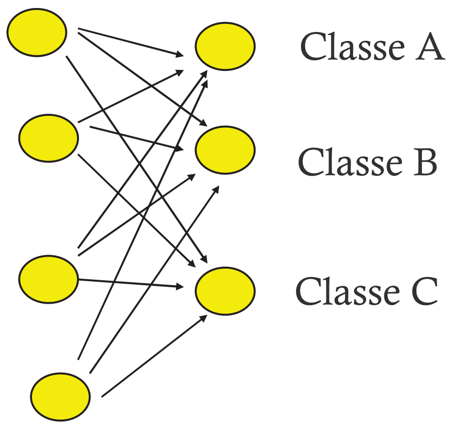
- L'input è composto dai valori precedenti
- L'output è il valore successivo



Alcune applicazioni dei percettroni continui

- approssimazione di funzioni a valori reali
  - $y=f(\mathbf{x})$; $y: R^n \rightarrow R$
  - nella fase di apprendimento si mostrano esempi in cui il vettore di ingresso è **x** e il valore di output desiderato è y
  - in generalizzazione si esamina il comportamento della rete su valori di x non appartenenti al training set (se la funzione f è nota si può valutare la capacità della rete)
- i percettroni continui possono anche essere applicati ai compiti di classificazione (p.es. diagnosi)
- i diversi neuroni dell'output corrispondono alle diverse classi
- il valore di attivazione fornisce una stima dell'evidenza in favore della appartenenza dell'input a quella classe

Alcune applicazioni dei percettroni continui



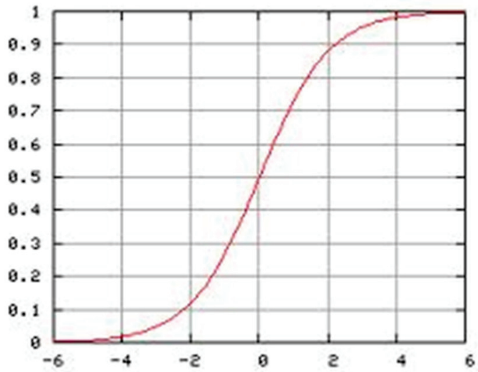
- i percettroni continui possono anche essere applicati ai compiti di classificazione (p.es. diagnosi)
- i diversi neuroni dell'output corrispondono alle diverse classi
- il valore di attivazione fornisce una stima dell'evidenza in favore della appartenenza dell'input a quella classe

Apprendimento

- I percettroni semplici con nodi continui vengono addestrati mediante un procedimento analogo a quello dei percettroni semplici booleani
  - presentazione di un pattern del training set
  - verifica della risposta
  - eventuale correzione dei pesi
  - presentazione del pattern successivo, fino a soddisfare qualche criterio
- studiamo come realizzare un algoritmo di apprendimento per neuroni continui

Una rete elementare con output continuo

- un neurone di ingresso, uno di output
- y = F(I)   --- differenziabile, non decrescente con I
- I = w·x



Training algorithm

- presentiamo uno stato di ingresso x_i, calcoliamo l'output corrispondente e confrontiamolo col valore desiderato
- modifichiamo ogni volta i pesi in modo che l'errore si riduca
- per ogni pattern presentato, l'errore è misurato da $E_i=(y-y_i^d)^2$, se y_i^d è il valore desiderato in corrispondenza di quel pattern in ingresso x_i
- L'errore globale su tutti gli stati del training set è $E = \Sigma_i E_i$
- Il problema di apprendimento diventa un problema di minimizzazione della distanza

Consideriamo un particolare pattern

$$y = F(I) = F(wx)$$

$$E = (y - y^d)^2 = \left[F(I) - y^d\right]^2 = \left[F(wx) - y^d\right]^2$$

- fissata la funzione di attivazione F, E è una funzione del peso w
- Consideriamo un pattern alla volta: lo presentiamo, e modifichiamo w in modo da ridurre E
- Δw deve essere tale da far sì che ΔE<=0
- se E è una funzione crescente di w, w deve diminuire
- se E è una funzione decrescente di w, w deve aumentare

Consideriamo un particolare pattern

- Δw deve avere il segno di –dE/dw
- si noti che in questo modo il sistema si avvicina ad un minimo locale
- ma poiché Δw è finito può anche "andare oltre"
- una buona regola è quella di fare modifiche limitate, e di entità proporzionale a dE/dw
- $\Delta w = -\varepsilon dE/dw$
  - in questo modo la discesa rallenta avvicinandosi al minimo

La «delta rule»

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

- è sempre $dF/dI \geq 0$, quindi dE/dw ha il segno di $(y-y^d)x$
- la regola $\Delta w = -\varepsilon dE/dw$ implica che
  - se $y>y^d$ allora Δw è proporzionale a $-x$
  - se $y<y^d$ allora Δw è proporzionale a x
  - proprio come nel percettrone booleano! in quel caso, se $y>y^d$ sottraevamo dai pesi una quantità proporzionale al vettore di ingresso (e se $y<y^d$ la aggiungevamo)
  - la "delta rule" rappresenta quindi l'analogo continuo della regola booleana del percettrone

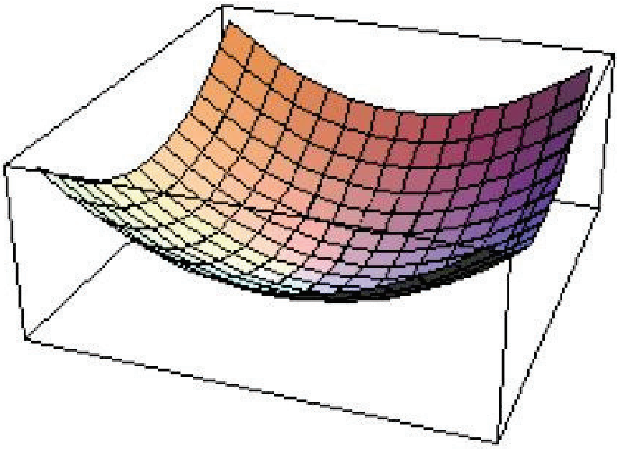
Se ci sono diversi neuroni di input, uno di output

- La funzione di costo E dipende da tutti i pesi della rete
- Per cercare il minimo di una funzione incognita, una tecnica molto diffusa è quella della discesa secondo il 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

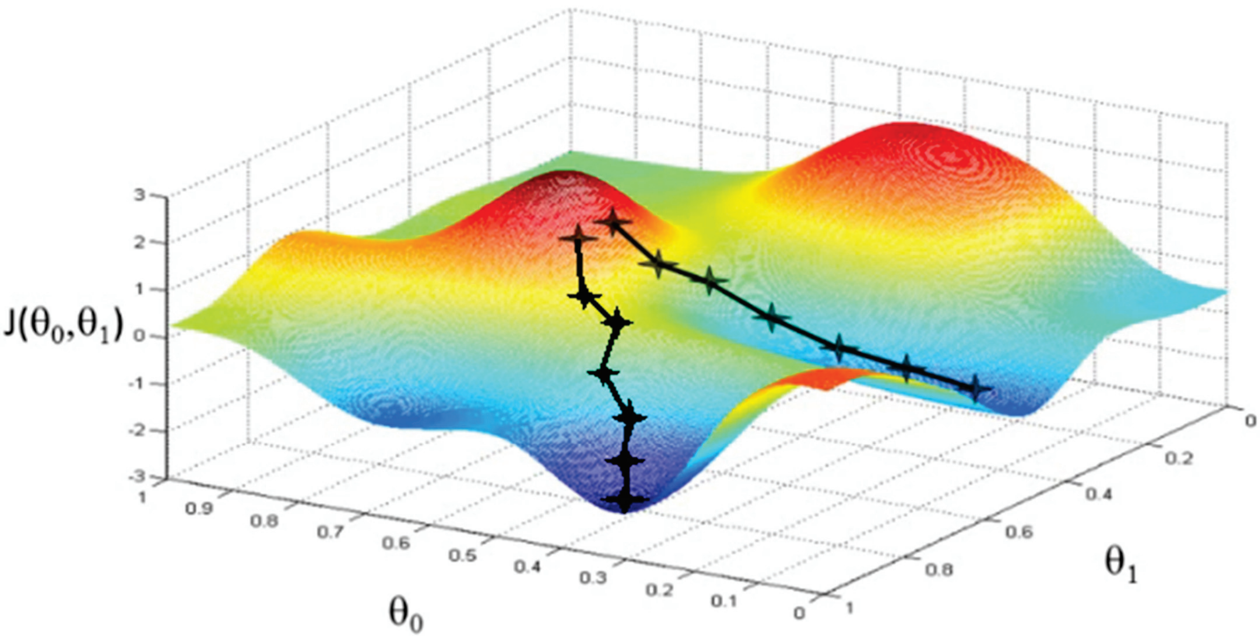
$$\Delta\vec{w} = -\varepsilon\vec{\nabla}E(\vec{w})$$

$$\vec{\nabla} \equiv \left(\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2} \dots \frac{\partial}{\partial w_N}\right)$$

Funzione con un solo minimo



Discesa secondo gradiente con più minimi



Commenti sulla discesa secondo gradiente

- da un dato punto di partenza trova un solo minimo
  - quello nel cui bacino di attrazione ricade lo stato di partenza della ricerca
- è un metodo deterministico
- può restare intrappolato in minimi locali
- rallenta quando ci si avvicina al punto di minimo e la derivata diventa piccola
- si possono introdurre correttivi alla discesa secondo gradiente

Ci possono essere più neuroni di output

- in generale, non c'è un solo neurone di output ma ve ne sono diversi; si definisce una distanza fra il pattern di uscita ottenuto e quello desiderato, e si cerca di minimizzare questa distanza
- si usa spesso il quadrato della distanza euclidea: se vi sono R neuroni nello strato di output, l'errore su uno degli esempi del training set è

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

- E la delta rule è formalmente la stessa

$$\Delta\vec{w} = -\varepsilon\vec{\nabla}E(\vec{w})$$

Limiti del percettrone semplice

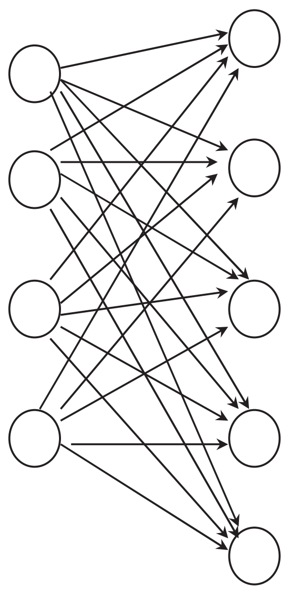
- Come sappiamo, i percettroni senza strati intermedi sono in grado di apprendere efficacemente diverse funzioni, ma non tutte!
- Per esaminare i limiti torniamo al percettrone "tutto booleano" che può apprendere le funzioni linearmente separabili da una serie di esempi
  - La soluzione non è univoca
- Tuttavia molte funzioni non sono linearmente separabili

Nel caso di funzioni booleane

n	f. booleane (f(n))	f. separabili (c(n))	c/f
2	16	14	.9
3	256	104	.4
4	65536	1882	.03
6	$1.8*10^{19}$	$1.5*10^{7}$	10^{-12}
8	$1.16*10^{77}$	$1.7*10^{13}$	10^{-64}

- $f(n) = 2^{2n}$
- $c(n) < 2*2^{n^2}/n!$
- $\lim_{n->} c(n)/f(n) = 0$

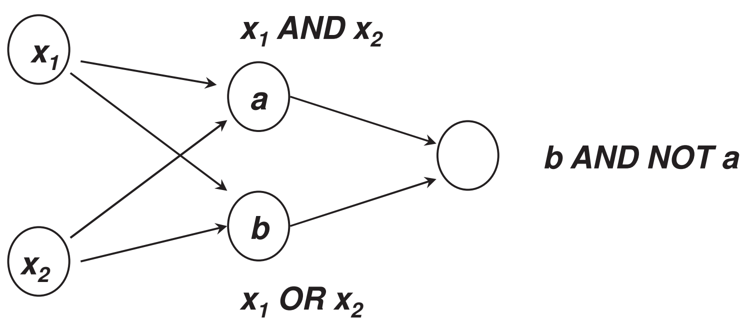
I limiti del percettrone a due strati



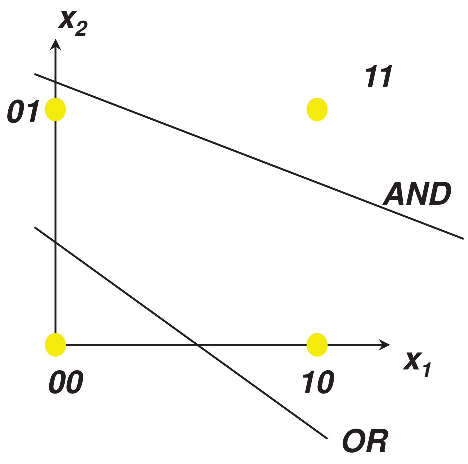
- il percettrone binario a due strati può quindi imparare da esempi solo un numero limitato di casi
- si possono mettere in parallelo più percettroni, che impareranno più funzioni, tutte appartenenti però all'insieme di quelle linearmente separabili
- per poter realizzare una classe più ampia di funzioni è necessario introdurre neuroni intermedi, che realizzino una sorta di rappresentazione interna dell'input

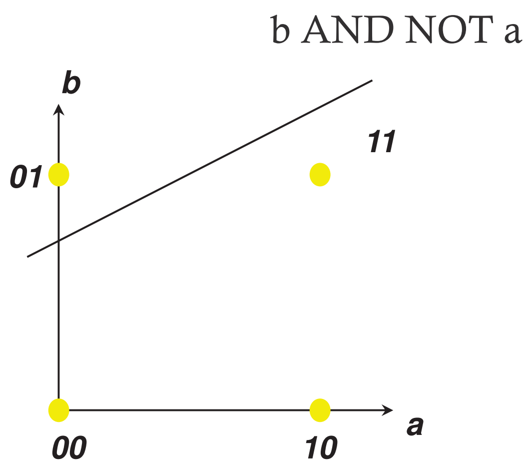
I limiti del percettrone a due strati

- Ad esempio realizzare la funzione non separabile XOR che non può essere realizzata senza strati intermedi

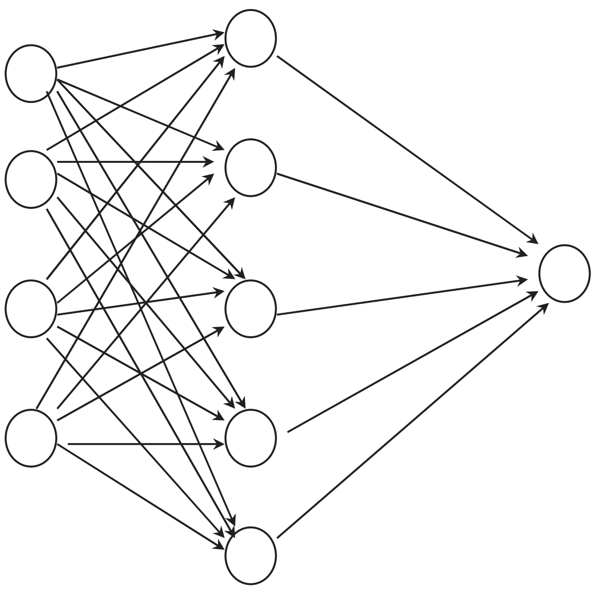


Verifica





MLP: multilayer perceptron



- Consideriamo un percettrone con strati nascosti
- E neuroni interni e di output continui
- Quelli di ingresso sono continui o booleani

MLP: apprendimento

- I percettroni a più strati vengono addestrati mediante un procedimento analogo a quello dei percettroni semplici
  - presentazione di un pattern del training set
  - verifica della risposta
  - eventuale correzione dei pesi
  - presentazione del pattern successivo, fino a soddisfare qualche criterio

- Vediamo come realizzare un algoritmo di apprendimento per percettroni a più strati

Apprendimento

- Come abbiamo visto l'apprendimento diventa un problema di ottimizzazione

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

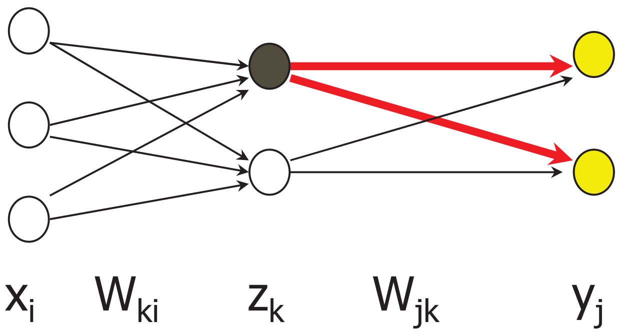
- E è funzione dell'insieme dei parametri W, e il problema può essere affrontato mediante opportuni algoritmi di ottimizzazione

- p.es. col metodo della discesa secondo gradiente
  - efficace per localizzare rapidamente il minimo locale più vicino al punto di partenza della ricerca
  - può non riuscire a trovare un minimo globale

$$\Delta\vec{w} = -\varepsilon\vec{\nabla}E(\vec{w})$$

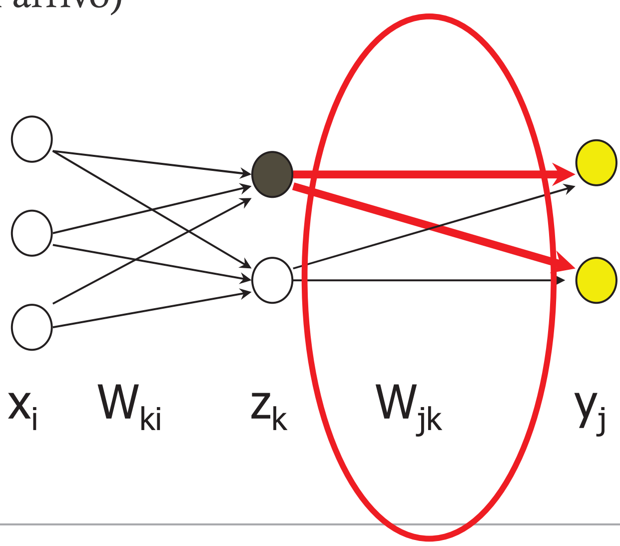
Retropropagazione del gradiente

- Un metodo particolarmente interessante (backpropagation) si basa proprio su questo approccio
  - consideriamo per semplicità un sistema con un solo strato intermedio



Retropropagazione del gradiente

- Le correzioni ai pesi che arrivano allo strato di output si calcolano con la delta rule (è noto il valore “desiderato” dei nodi di arrivo)

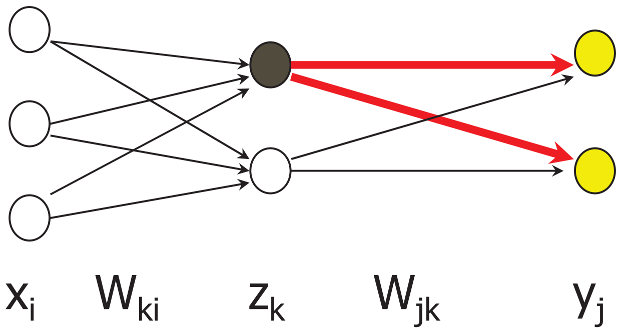


Retropropagazione del gradiente

- Le correzioni ai pesi che arrivano allo strato di output si calcolano con la delta rule (è noto il valore "desiderato" dei nodi di arrivo)
- ma questo non è vero per i pesi che arrivano a neuroni intermedi
- esiste un metodo di "retropropagazione del gradiente" che consente di calcolare le modifiche ai pesi del primo strato, note quelle dell'ultimo strato
- i prossimi luci (facoltativi*) mostrano la derivazione del metodo

Facoltativo: le equazioni

$$y_j = F(A_j)$$
$$A_j \equiv \sum W_{jk} z_k$$
$$z_k = F(B_k)$$
$$B_k \equiv \sum W_{ki} x_i$$
$$E = \sum (y_j - d_j)^2$$



F è una squashing function, limitata e non decrescente

Fac: gradiente di E rispetto i pesi vicini all'output

- E dipende dai pesi dell'ultimo strato come segue

$$E = \sum_j (y_j - d_j)^2$$

$$\frac{\partial E}{\partial W_{jk}} = 2(y_j - d_j)\frac{\partial y_j}{\partial W_{jk}} = 2(y_j - d_j)\frac{\partial F(A_j)}{\partial W_{jk}} = 2(y_j - d_j)F'(A_j)z_k$$

- dove F'(A)= dF/dA

$$y_j = F(A_j)$$

$$A_j \equiv \sum W_{jk} z_k$$

Fac: gradiente di E rispetto i pesi vicini all'output

$$E = \sum_j (y_j - d_j)^2$$

$$\frac{\partial E}{\partial W_{jk}} = 2(y_j - d_j)\frac{\partial y_j}{\partial W_{jk}} = 2(y_j - d_j)\frac{\partial F(A_j)}{\partial W_{jk}} = 2(y_j - d_j)F'(A_j)z_k$$

- si noti che, quando viene presentato un pattern tutti gli elementi del m.a.d. sono noti, quindi il calcolo di queste componenti del gradiente di W è diretto
- e l'informazione può essere usata per aggiornare il valore dei pesi:

$$W_{jk}^s = W_{jk}^{s-1} - \lambda^{(s)} \frac{\partial E^s}{\partial W_{jk}^s}$$

Fac: gradiente di E rispetto i pesi vicini all'input

- E dipende dai pesi del primo strato attraverso la loro azione sui valori degli z_k
  - si noti i) che W_{ki} influenza solo l'elemento k-esimo dello strato intermedio, e che ii) z_k influenza l'errore solo attraverso la sua azione sugli y

$$\boxed{\frac{\partial E}{\partial W_{ki}} = \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial W_{ki}}}$$

Fac: gradiente di E rispetto i pesi vicini all'input

- E dipende dai pesi del primo strato attraverso la loro azione sui valori degli z_k
  - si noti i) che W_{ki} influenza solo l'elemento k-esimo dello strato intermedio, e che ii) z_k influenza l'errore solo attraverso la sua azione sugli y

$$\frac{\partial E}{\partial W_{ki}} = \boxed{\frac{\partial E}{\partial z_k}} \boxed{\frac{\partial z_k}{\partial W_{ki}}}$$

$$\boxed{\frac{\partial z_k}{\partial W_{ki}}} = \frac{\partial}{\partial W_{ki}} F(B_k) = \frac{\partial}{\partial W_{ki}} F\left(\sum W_{ki} x_i\right) = F'(B_k) x_i$$

tutti termini noti al momento del calcolo

$$\boxed{\frac{\partial E}{\partial z_k}} = \sum_j \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_k}$$

Fac: gradiente di E rispetto i pesi vicini all'input

- resta da calcolare solo la derivata di y_j rispetto a z_k

$$\boxed{\frac{\partial E}{\partial z_k} = \sum_j \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_k}}$$

$$\frac{\partial E}{\partial y_j} = 2(y_j - d_j)$$

$$\frac{\partial y_j}{\partial z_k} = \frac{\partial}{\partial z_k} F\left(\sum_k W_{jk} z_k\right) = F'(A_j) W_{jk}$$

- combinando il tutto

$$\boxed{\frac{\partial E}{\partial W_{ki}} = \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial W_{ki}}}$$

$$\frac{\partial E}{\partial W_{ki}} = \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial W_{ki}} = \sum_j \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_k} F'(B_k) x_i$$

$$\frac{\partial E}{\partial W_{ki}} = \sum_j 2(y_j - d_j) F'(A_j) W_{jk} F'(B_k) x_i$$

Fac: perché retropropagazione?

- Finalmente:

$$\frac{\partial E}{\partial W_{ki}} = \sum_j 2(y_j - d_j) F'(A_j) W_{jk} F'(B_k) x_i = \frac{1}{z_k}\frac{\partial E}{\partial W_{jk}} W_{jk} F'(B_k) x_i$$

- si vede che la conoscenza delle componenti del gradiente di E rispetto alle connessioni del secondo strato di pesi consente di calcolare le componenti del gradiente rispetto alle connessioni dei primi pesi

Fac:perché retropropagazione?

- Finalmente:

$$\frac{\partial E}{\partial W_{ki}} = \boxed{\sum_j 2(y_j - d_j)F'(A_j)}\, W_{jk} F'(B_k) x_i = \boxed{\frac{1}{z_k}\frac{\partial E}{\partial W_{jk}}}\, W_{jk} F'(B_k) x_i$$

- si vede che la conoscenza delle componenti del gradiente di E rispetto alle connessioni del secondo strato di pesi consente di calcolare le componenti del gradiente rispetto alle connessioni dei primi pesi

Fac:perché retropropagazione?

- Usando una notazione più sintetica, chiamiamo W_1 e W_2 i vettori dei pesi del primo e del secondo strato, e indichiamo con gli stessi indici i gradienti relativi a quei vettori
- quanto abbiamo visto è che
- si può calcolare direttamente dalle simulazioni il $\mathrm{grad}_2 E$
- poi si vede che il $\mathrm{grad}_1 E$ è esprimibile in funzione del $\mathrm{grad}_2 E$
  - ma sarebbe anche calcolabile direttamente dai dati!

Retropropagazione del gradiente

- il meccanismo di retropropagazione consente quindi di calcolare le modifiche ai pesi verso i neuroni nascosti
  - Il procedimento può essere generalizzato a architetture con più strati intermedi
- dopo aver presentato un pattern, si presenta il successivo, etc.
  - sono necessarie diverse ripresentazioni dell'intero insieme di addestramento
- la validità delle prestazioni della rete viene poi valutata su un insieme di esempi nuovi (generalizzazione) che la rete non ha ancora "visto"

Apprendimento locale o globale?

- l' apprendimento globale è una discesa secondo gradiente
  - utilizza tutte le informazioni prima di modificare i pesi
  - può essere intrappolato in minimi locali
- l' apprendimento locale sovrappone una sorta di rumore alla discesa secondo gradiente
  - ad ogni passo, la forma della funzione energia per un pattern può essere diversa da quella complessiva
  - è quindi possibile per il sistema muoversi contro il gradiente dell'energia complessiva
  - una certa possibilità di fuggire dai minimi locali

Approssimatori universali

- Ogni funzione continua a valori reali può essere approssimata uniformemente da un percettrone con uno strato intermedio, un neurone di output e funzioni di trasferimento sigmoidali
- sono approssimabili anche funzioni continue a tratti (Hornik)
- il teorema si estende al caso di funzioni a molti valori (da R^n a R^m)

Il problema della generalizzazione

- si richiede alla rete di inferire una relazione, o una classificazione, partendo da una sua parziale specificazione, mail problema non ha in generale una soluzione univoca
  - esempio

input	output
1	1
2	1
8	0
3	1
10	0
4	1
11	0
7	0
6	1
5	?

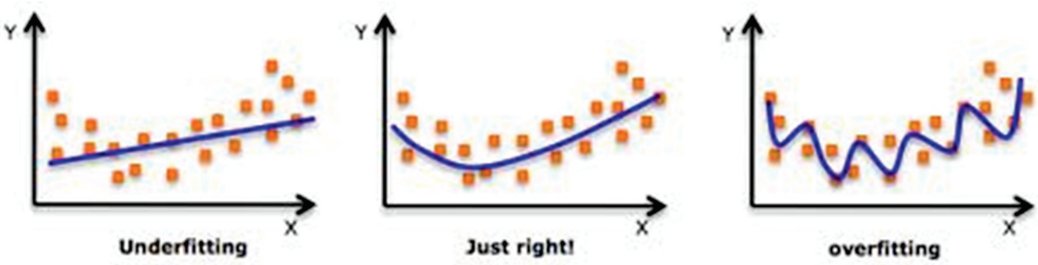
- “y<7” oppure “i divisori di 12?”

Il problema della generalizzazione

- è necessario che gli esempi siano rappresentativi
- esempio di classificazione binaria: se il 99% degli esempi appartiene alla classe A, la rete trova facilmente una soluzione (minimo locale) corrispondente alla risposta “sempre A”
- “y<7” oppure “i divisori di 12?”

Overfitting e underfitting

- Overfitting e underfitting
  - utilizzando un numero elevato di neuroni è possibile pervenire ad una classificazione molto precisa dell’insieme di addestramento; i confini di separazione fra le classi possono essere molto frastagliati e tali da adattarsi all’ insieme dei casi: tuttavia una rete di questo tipo può avere scarse capacità di generalizzazione (con molti parametri si può fittare ogni curva)



Paradigmi di apprendimento

- Vi sono tre grandi paradigmi di apprendimento, ciascuno corrispondente ad un particolare compito astratto di apprendimento.
- Apprendimento supervisionato (supervised learning), qualora si disponga di un insieme di dati per l'addestramento (o training set) comprendente esempi tipici di ingressi con le relative uscite loro corrispondenti: in tal modo la rete può imparare ad inferire la relazione che li lega
  - La rete è addestrata mediante un opportuno algoritmo (tipicamente, la backpropagation), il quale usa tali dati allo scopo di modificare i pesi e altri parametri della rete stessa in modo tale da minimizzare l'errore di previsione relativo all'insieme di addestramento

Paradigmi di apprendimento

- Vi sono tre grandi paradigmi di apprendimento, ciascuno corrispondente ad un particolare compito astratto di apprendimento.
- Apprendimento non supervisionato (unsupervised learning), basato su algoritmi di addestramento che modificano i pesi della rete facendo esclusivamente riferimento ad un insieme di dati che include le sole variabili di ingresso
  - Tali algoritmi tentano di raggruppare i dati di ingresso e di individuare pertanto degli opportuni cluster rappresentativi dei dati stessi
  - Esempio, regola di apprendimento di Hebb

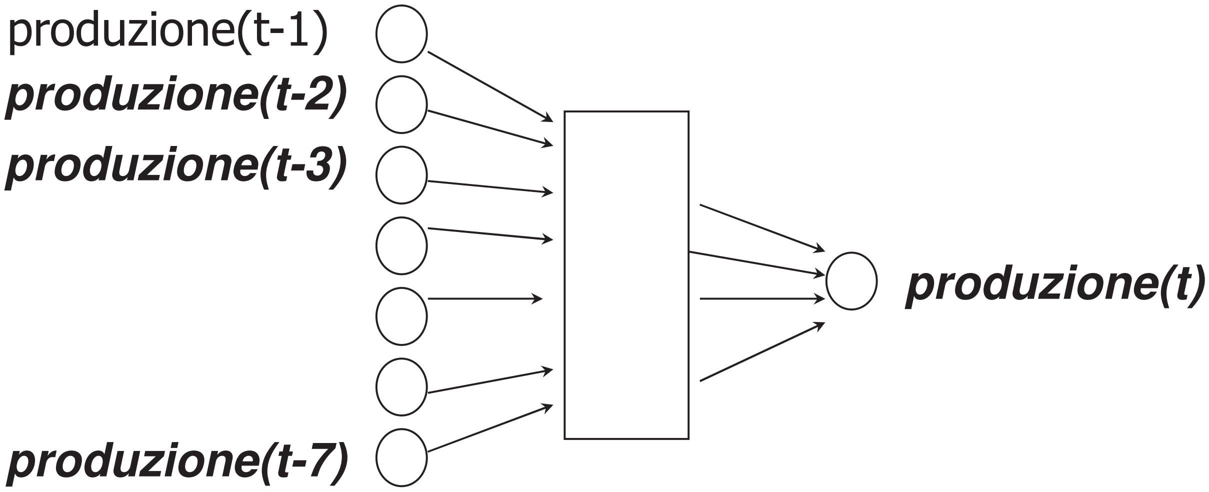
Paradigmi di apprendimento

- Vi sono tre grandi paradigmi di apprendimento, ciascuno corrispondente ad un particolare compito astratto di apprendimento.
- Apprendimento per rinforzo (reinforcement learning), nel quale un opportuno algoritmo si prefigge lo scopo di individuare un certo modus operandi, a partire da un processo di osservazione dell'ambiente esterno
  - Ogni azione ha un impatto sull'ambiente, e l'ambiente produce una retroazione che guida l'algoritmo stesso nel processo di apprendimento. Tale classe di problemi postula che l'entità che apprende sia un agente
  - Esempio, Bucket brigade per il sistema a classificatori
  - Molto usata è la regola chiamata Q-learning

Applicazioni di reti neurali

- diagnosi
- riconoscimento
- analisi di segnali
- forecast
- compressione
- …
- Processi industriali: es. controllo fermentazioni (cefalosporina)

Forecast



Cenno alle applicazioni di reti neurali in economia

- classificazione di agenti economici
  - credit scoring
  - frodi con carte di credito
  - classificazione dei consumatori
- previsione di serie temporali
  - cambi
  - titoli
  - prezzi
- modellazione di agenti economici
  - istituzioni che apprendono
  - modellazione di agenti che interagiscono scambiando merci

Diversi modelli di rete

- esistono molti modelli di reti oltre a quelli esaminati finora
- esempi: reti a strati con feedback, reti con funzione di attivazione gaussiana, reti auto-organizzate di Kohonen (SOM), modelli ART , modelli con neuroni come "oscillatori non lineari", etc.

Deep learning

- Maggiore enfasi negli ultimi anni sui metodi di apprendimento "profondo" (Deep Learning)
- Alcuni miglioramenti tecnici
- Ma soprattutto: tanti strati, tanti nodi per strato
- Ciò richiede moltissimi dati
  - Big Data
- Backprop is still with us!
- Applicazioni interessanti sono in vista…