

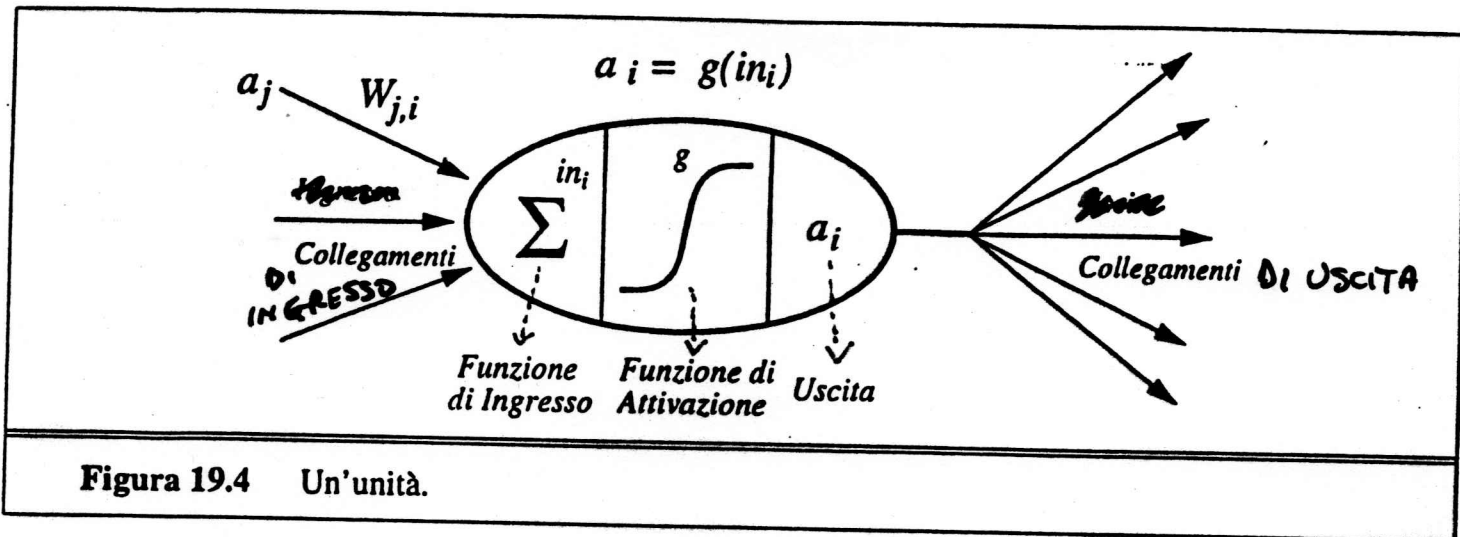


Figura 19.4 Un'unità.

$$in_i = \sum_j W_{ji} \cdot a_j = \bar{W}_i \cdot \bar{a}^{in}$$

$$\bar{a}^{in} = \langle a_1, a_2, \dots, a_m \rangle$$

$$a_i = g(\bar{W}_i \cdot \bar{a}^{in}) \Leftarrow \text{PASSO ELEMENTARE DI COMPUTAZIONE}$$

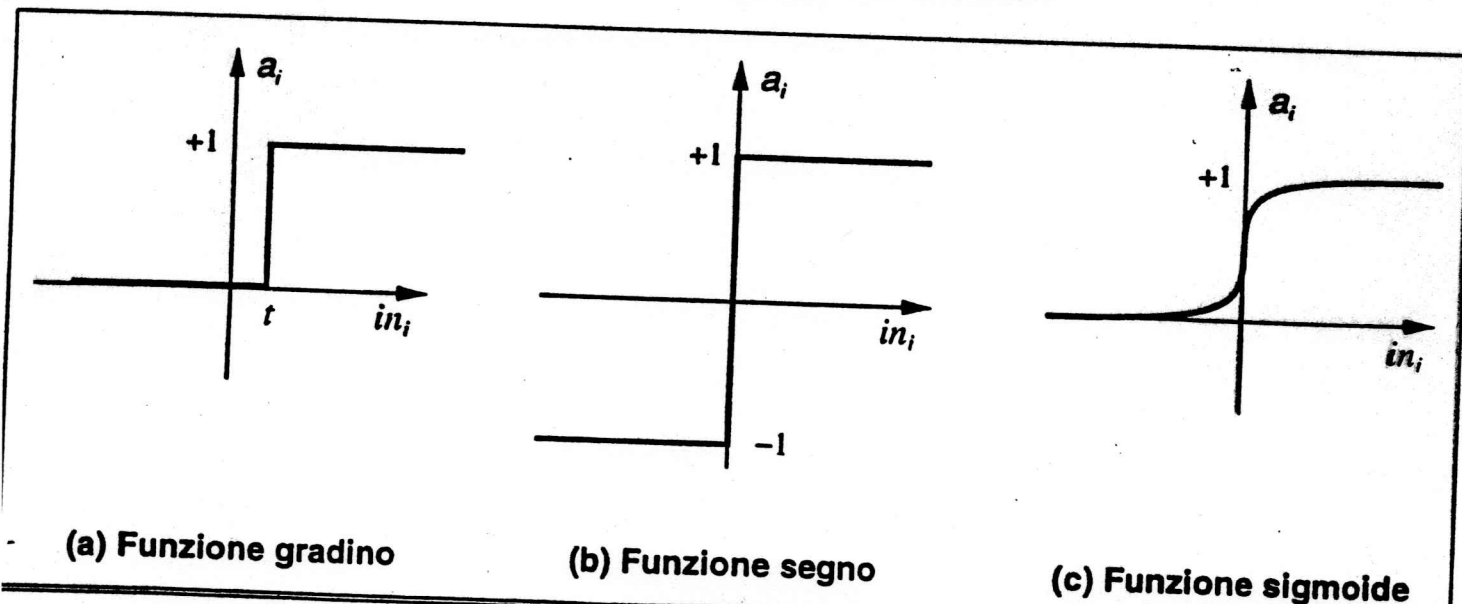


Figura 19.5 Tre diverse funzioni di attivazione per le unità.

$$\text{gradino}_t(x) = \begin{cases} 1, & \text{if } x \geq t \\ 0, & \text{if } x < t \end{cases} \quad \text{segno}(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \quad \text{sigmoide}(x) = \frac{1}{1 + e^{-x}}$$

(STEP)

$t = \text{VALORE SOGLIA} = \left\{ \begin{array}{l} \text{VALORE MINIMO DI} \\ \text{INPUT PESATO CHE È} \\ \text{NECESSARIO PER} \\ \text{GENERARE UN'IMPULSO} \\ \text{DA INVIARE ALL'ESTERNO} \end{array} \right.$

UNA RETE CHE USA $[\text{Step}_t]$ ($t \neq 0$) PUO' ESSERE
CONVERTITA IN UNA RETE CHE USA $[\text{Step}_0]$ ($t=0$)

AGGIUNGO UNA UNITA' DI INPUT I_0 CHE INVIA
SEMPRE IL VALORE DI ATTIVAZIONE -1 ALLA UNITA' i

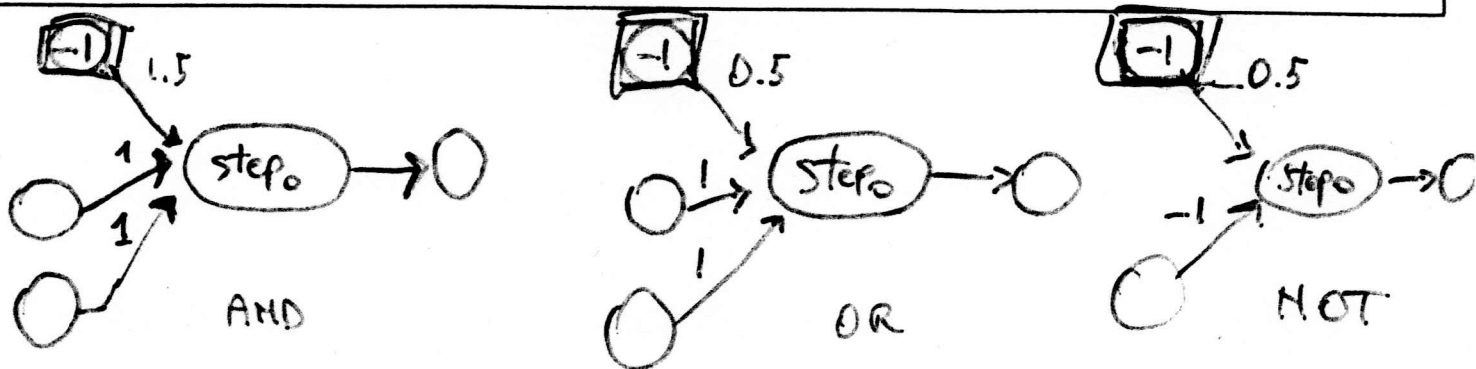
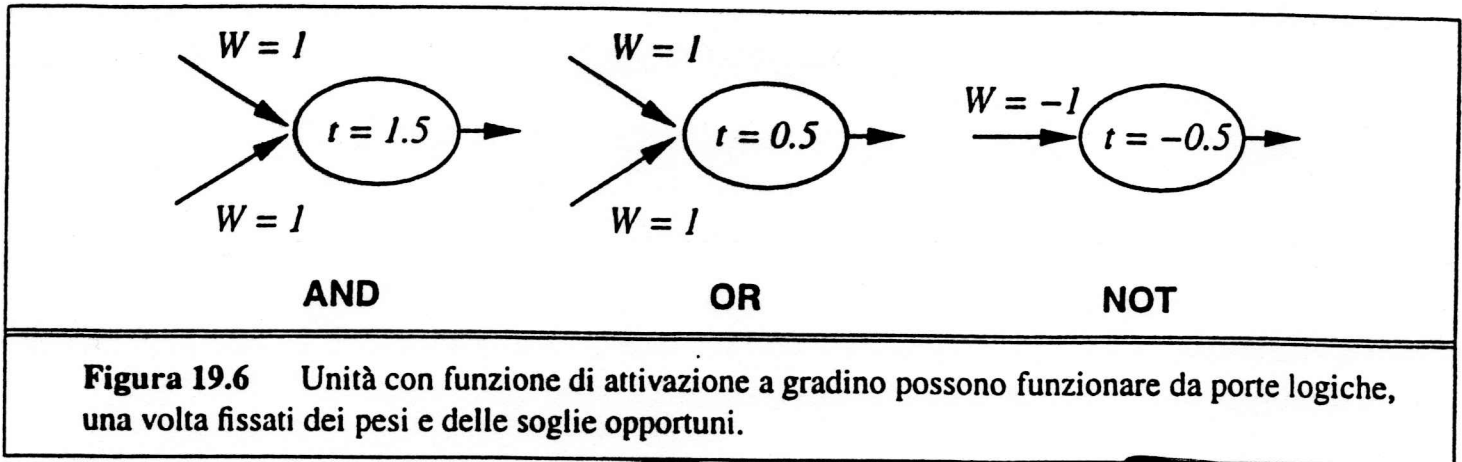
IL PESO W_{0i} E' "t"

$$a_i = \text{step}_t \left(\sum_{j=1}^n w_{ji} a_j \right) = \text{step}_0 \left(\sum_{j=0}^n w_{ji} a_j \right)$$

$$W_{0i} = t \quad a_0 = -1$$

UTILE PERCHE' t PUO' ESSERE "APPRESO" COME GLI ALTRI
PESI DELLA RETE

E' POSSIBILE RAPPRESENTARE UNA QUALSIASI FORMULA PROPOSIZIONALE:



RETI "FEED FORWARD"

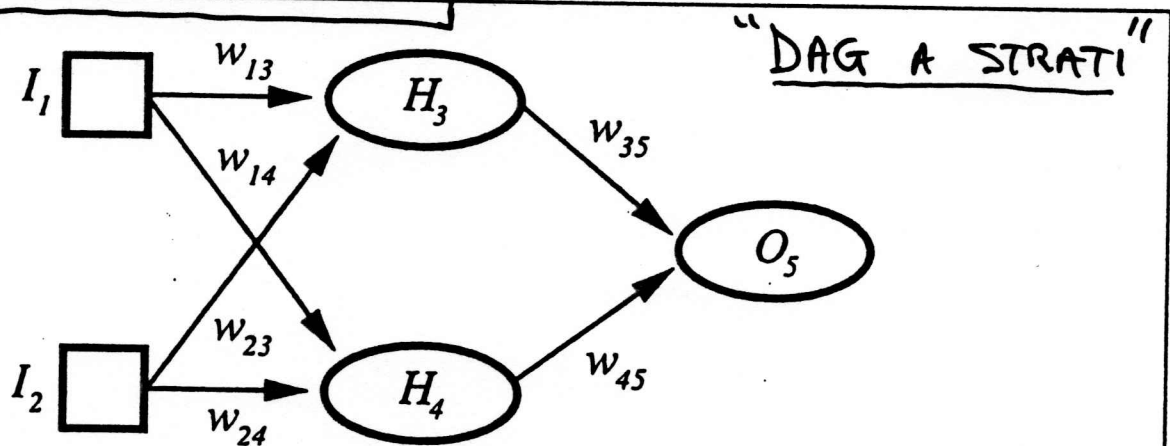


Figura 19.7 Una rete alimentata in avanti a due strati molto semplice, con due ingressi, due nodi nascosti e un nodo di output.

$I_{1,2}$ = UNITÀ DI INPUT

H_3, H_4 = UNITÀ "HIDDEN" (NASCOSTE)

O_5 = UNITÀ DI OUTPUT

FUNZIONI RAPPRESENTATE DA RETI DI TIPO FEED FORWARD

UNA RETE CON STRUTTURA E g FISSE RAPPRESENTA FUNZIONI CON UNA SPECIFICA STRUTTURA PARAMETRIZZATA DAI PESI DELLE CONNESSIONI

ESEMPIO:

$$\begin{aligned} a_5 &= g(w_{3,5} \cdot a_3 + w_{4,5} \cdot a_4) = \\ &= g(w_{3,5} \cdot \underbrace{g(w_{1,3} \cdot a_1 + w_{2,3} \cdot a_2)}_{a_3} + w_{4,5} \cdot \underbrace{g(w_{1,4} \cdot a_1 + w_{2,4} \cdot a_2)}_{a_4}) \end{aligned}$$

g è non lineare $\Rightarrow$ l'intera rete rappresenta una funzione non lineare

PROCESSO DI APPRENDIMENTO $\approx$ PROCESSO DI "TUNING" (AGGIUSTAMENTO) DEI PARAMETRI (PESI) DELLA FUNZIONE RAPPRESENTATA, IN MODO CHE LA RETE ABBLA UN COMPORTAMENTO CONSISTENTE CON IL TRAINING SET

$\uparrow$
PROCESSO DI "REGRESSIONE LINEARE"

1 STRATO SUFFICIENTEMENTE "LARGO" CONSENTE DI RAPPRESENTARE UNA QUALSIASI FUNZIONE CONTINUA

LE RETI FF NON HANNO UNO STATO INTERNO (I PESI DELLA RETE SONO FISSI DURANTE LA COMPUTAZIONE).

UNA RETE FF. COMPIE UNA FUNZIONE

$$F: D_i^n \rightarrow D_o^m$$

n : # UNITÀ INPUT

m : # UNITÀ OUTPUT

D_i : dominio unità input

D_o : dominio unità output

Tipicamente $|D_i| = |D_o| = 2$

PERCETTRONE (FINE ANNI '50)

RETE FF CON 1 STRATO

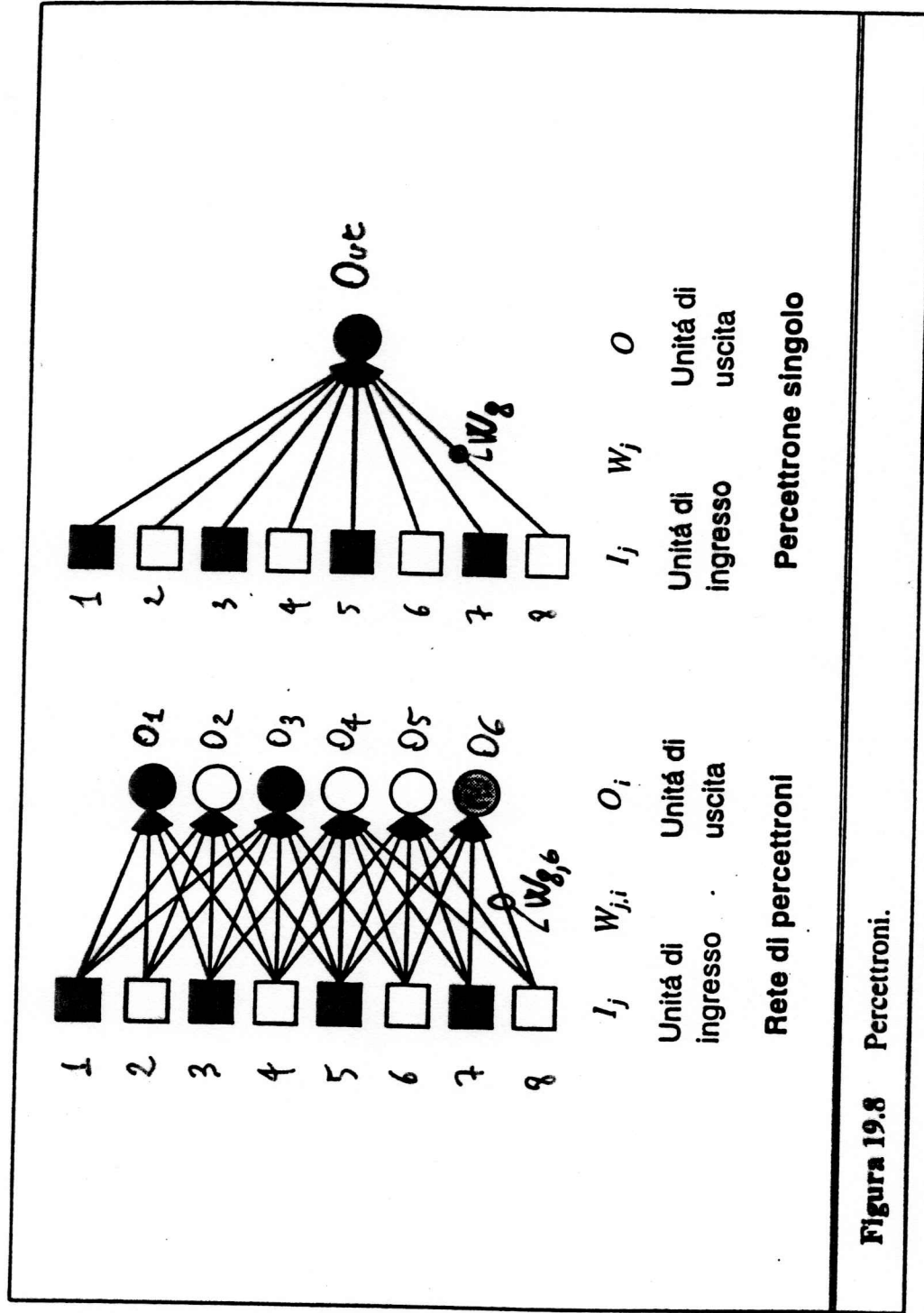


Figura 19.8 Percettroni.

$$O = \text{Step}_0 \left(\sum_j w_j I_j \right) = \text{Step}_0 (\bar{W} \cdot \bar{I})$$

$O = \text{attivazione } O_{out} = \text{VALORE DI USCITA}$

NOTA:

$$O = \text{Step}_t \left(\sum_{j=1}^n w_j I_j \right) = \text{Step}_0 \left(\sum_{j=0}^n w_j I_j \right)$$

$$I_n = -1, w_{n+1} = t$$

UN PERCEPTRONE PUO' RAPPRESENTARE FUNZIONI SEPARABILI LINEARMENTE

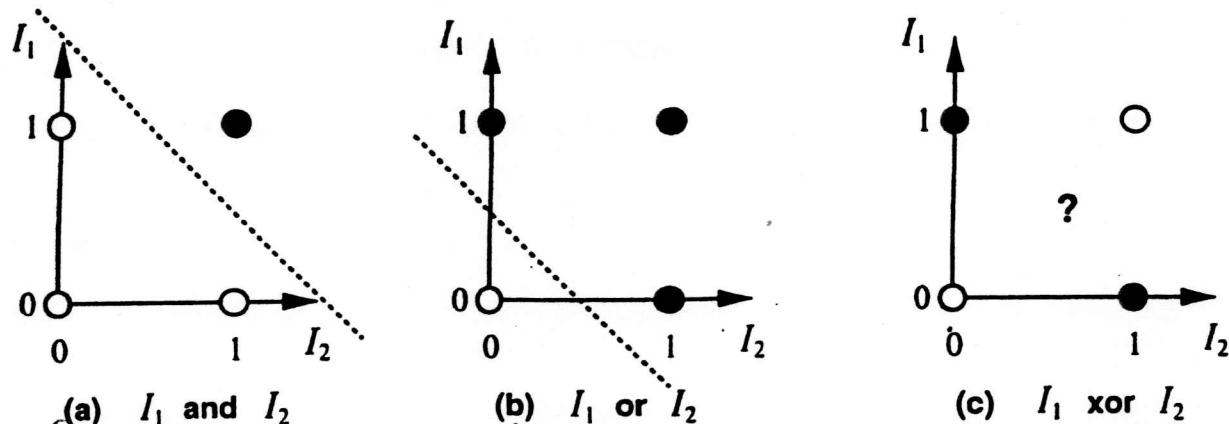


Figura 19.9 Separabilità lineare nei perceptroni.

$$w_0 = 1.5 \quad w_1 = w_2 = 1$$

$$I_1 = -I_2 + 1.5$$

$$w_0 = 0.5$$

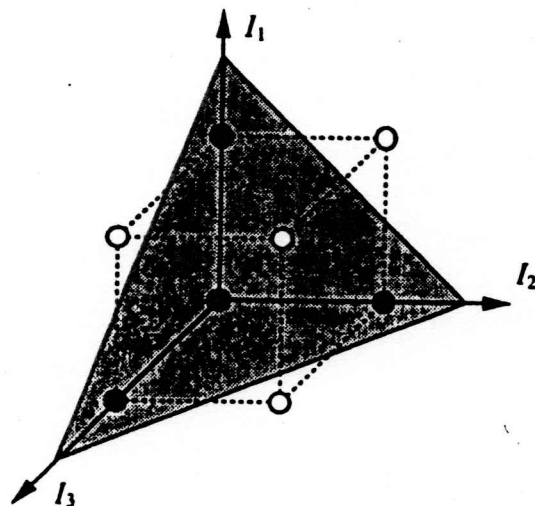
$$w_1 = w_2 = 1$$

$$I_1 = -I_2 + 0.5$$

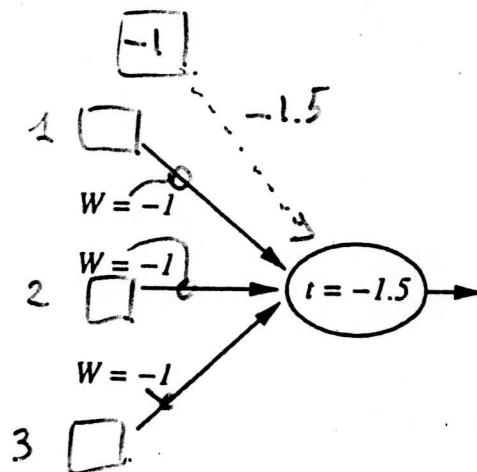
OTTENUTE CON $I_{n_{out}} = 1$

$$I_0 w_0 + I_1 w_1 + I_2 w_2 = 1$$

$$I_1 = \frac{-I_2 w_2 + I_0 w_0}{w_1}$$



(a) Piano di separazione



(b) Pesi e soglia

Figura 19.10 Separabilità lineare in tre dimensioni: la rappresentazione della funzione "minoranza". (UNA MINORANZA DEGLI INGRESSI È VERA (+1))

QUANDO HO PIÙ DI TRE UNITÀ DI INPUT (OLTRE A I_0) LA FRONTIERA DI SEPARAZIONE

TRA ESEMPI IN CUI $\emptyset = 1$ E ESEMPI IN CUI $\emptyset = \emptyset$ È UN IPERPIANO (SUPERFICIE DI DECISIONE)

function APPRENDIMENTO-RETE-NEURALE(*esempi*) returns *rete*

rete ← una rete con pesi assegnati a caso

repeat

for each *e* in *esempi* do

O ← USCITA-RETE-NEURALE(*rete*, *e*)

T ← l'uscita osservata da *e*

si aggiornano i pesi in *rete* basandosi su *e*, *O*, e *T*

end

until tutti gli *esempi* sono predetti correttamente o si

soddisfa un criterio di terminazione

return *rete*

EPOCA

{ es: tempo passato, n° di iterazioni, ecc. }

Figura 19.11 Il generico algoritmo di apprendimento per reti neurali: si correggono i pesi fino a che i valori di uscita previsti *O* e quelli effettivamente osservati coincidono.

O e *T* in generale (reti Feed-forward) sono vettori
nel perceptrone sono singoli valori.

REGOLA DI APPRENDIMENTO (PERCEPTRONE)

$$Err_e = T - O \quad e = \text{esempio}$$

T = valore di output corretto per e

O = valore di output per e

$$Err_e \in \{+1, -1\} \quad O \in \{0, 1\}$$

$$W_j \leftarrow W_j + \alpha I_j Err_e$$

α = incremento (grado/tasso) di apprendimento

I_j = valore attivazione unità input j

LA REGOLA SOMMA O SOTTRA A W_j UNA QUANTITÀ PROPORZIONALE A I_j .
($\alpha > 0$)

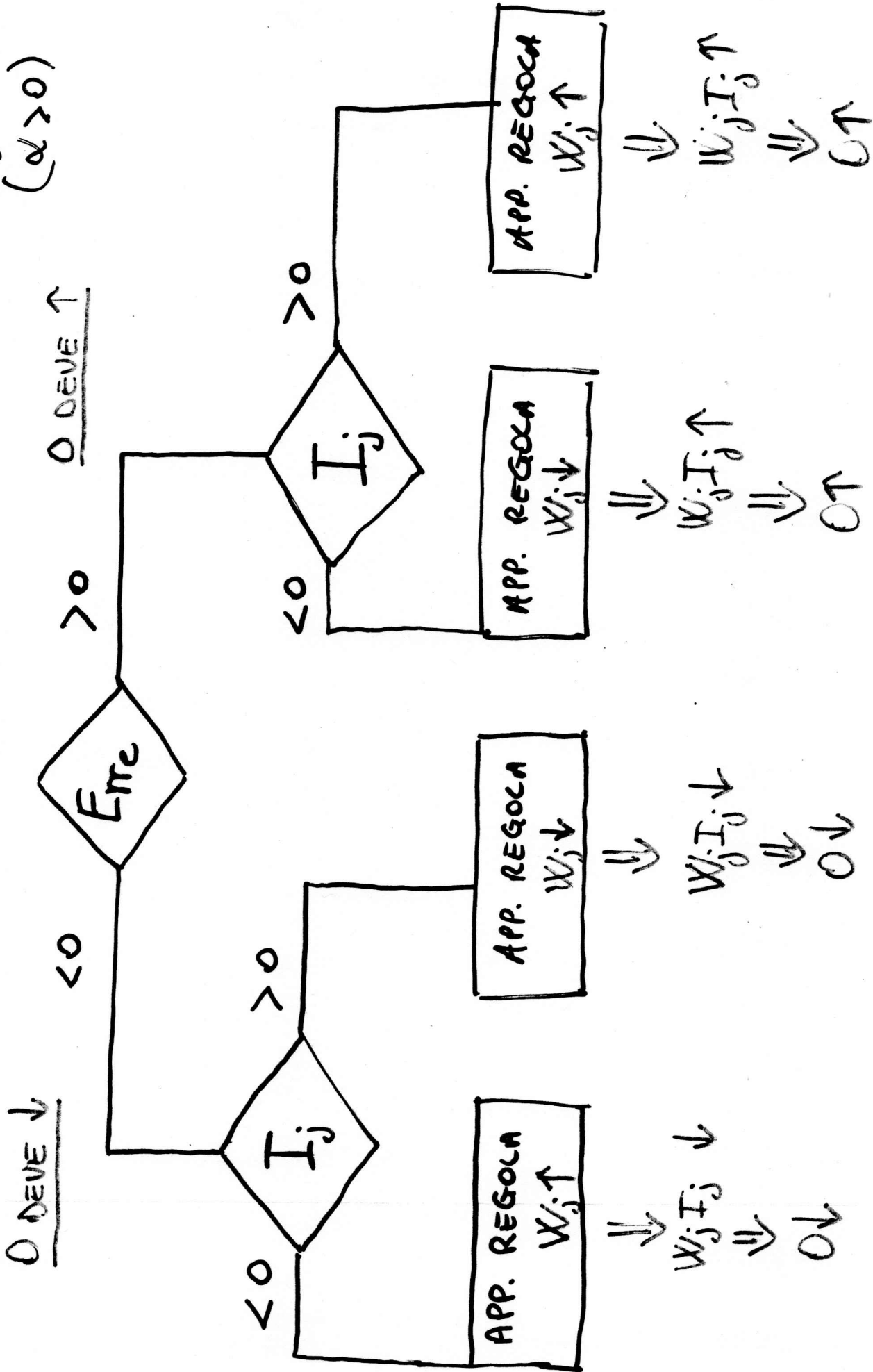
PROCESSO DI APPRENDIMENTO $\Rightarrow$ MODIFICA DELLA "SUPERFICIE DI DECISIONE" (RETTA DI SEPARAZIONE, CON 2+1 UNITÀ DI INPUT)

(14)

$Err_e = T - 0$

$$W_j \leftarrow W_j + \alpha I_j \cdot Err_e$$

or step₀ ($\sum_j w_j \cdot I_j$)
($\alpha > 0$)



CONVERGENZA (ROSENBLATT ~ 1962)

TEOREMA

IL SISTEMA DI APPRENDIMENTO BASATO
SULLA REGOLA DEL PERCEPTRONE CONVERGE
AD UN INSIEME DI PESI CHE CLASSIFICANO
CORRETTAMENTE GLI ESEMPI (ROSENBLATT ~ 1960)
↓ (RAPPRESENTANO CORRETTAMENTE GLI ESEMPI)

MA:

IL TEOREMA DI CONVERGENZA VALE A PATTO
CHE GLI ESEMPI RAPPRESENTANO UNA FUNZIONE
LINEARMENTE SEPARABILE....

LE FUNZIONI L.S. NON SONO MOLTE (ARTICOLO
MINSKY e PAPERT '69)

PROCESSO DI APPRENDIMENTO $\approx$ RICERCA
LOCALE NELLO SPAZIO DEI PESI DELLA RETE
DI TIPO "DISCESA DEL GRADIENTE"

CONSENTE DI TROVARE IL MINIMO OTTIMO
QUANDO GLI ESEMPI RAPPRESENTANO UNA
FUNZIONE LINEARMENTE SEPARABILE.

$$J(\bar{w}) = \sum_{\bar{x} \in X} |\bar{w} \cdot \bar{x}|$$

$$J(\bar{w}) = 0 \text{ se } X = \emptyset$$

$$J(\bar{w}) > 0 \text{ se } X \neq \emptyset$$

X = sottoinsieme del training set di esempi classificati erroneamente

$\bar{x}$ = vettore di input (attivazione unità input)

$$J(\bar{w}) = \sum_{\bar{x} \in X} \bar{w} \cdot \begin{cases} \bar{x} & \text{se } \bar{w} \cdot \bar{x} \geq 0 \quad (\text{ERRORE} = -1) \\ -\bar{x} & \text{se } \bar{w} \cdot \bar{x} \leq 0 \quad (\text{ERRORE} = 1) \end{cases}$$

$$\nabla J(\bar{w}) = \sum_{\bar{x} \in X} \begin{cases} \bar{x} \\ -\bar{x} \end{cases}$$

VOGLIAMO
MINIMIZZARE $J(\bar{w})$
($\Rightarrow$ ERRORE $\rightarrow 0$)

$$\boxed{\bar{w} \leftarrow \bar{w} - \alpha \nabla J(\bar{w})} = \bar{w} - \alpha \sum_{\bar{x} \in X} \begin{cases} \bar{x} & (\text{err} < 0) \\ -\bar{x} & (\text{err} > 0) \end{cases}$$

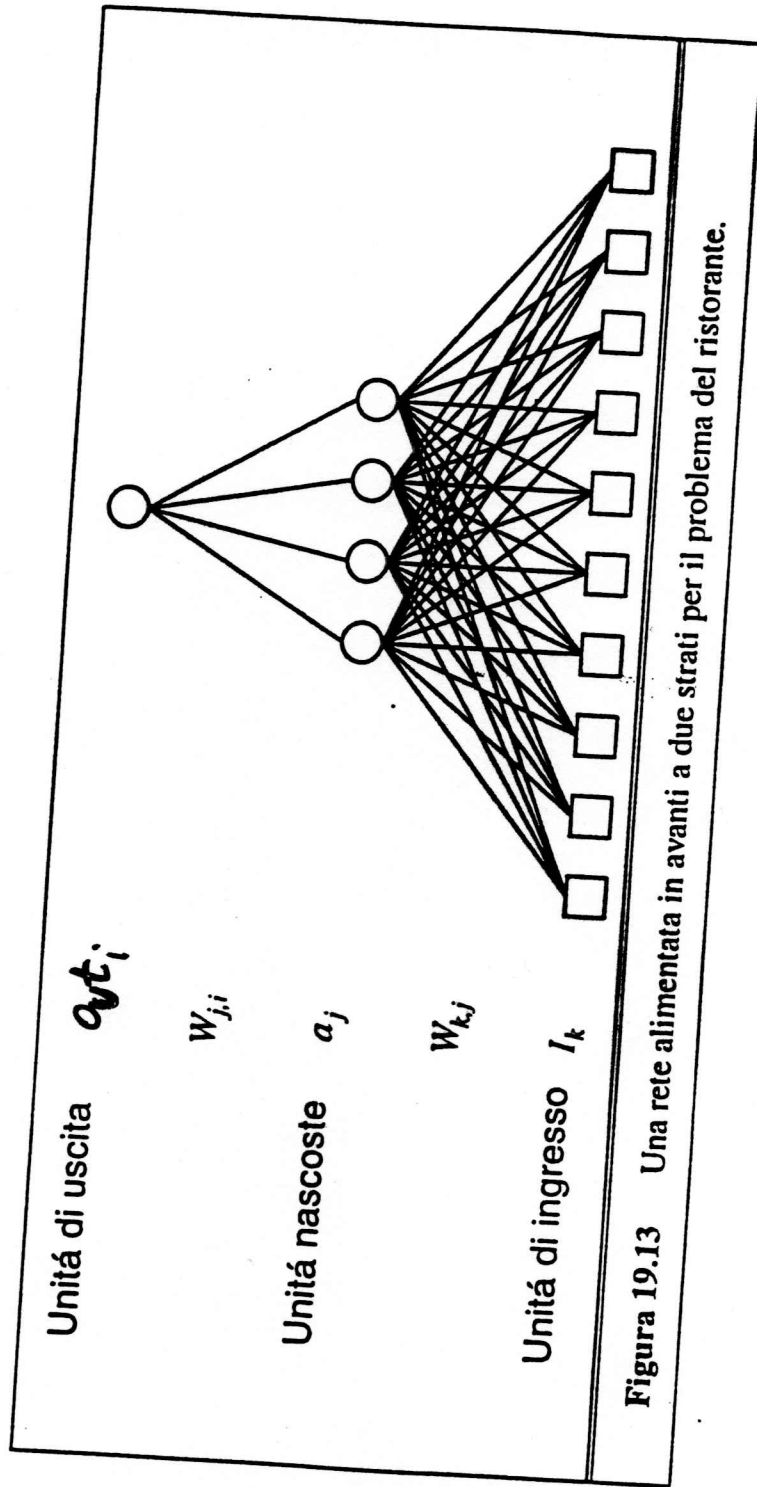
$$\bar{w} \leftarrow \bar{w} + \alpha \sum_{\bar{x}} \begin{cases} -\bar{x} & (\text{err} < 0) \\ +\bar{x} & (\text{err} > 0) \end{cases} \Rightarrow \bar{w} \leftarrow \bar{w} + \alpha \bar{I} \text{Error} \quad (\text{PER UN } \bar{x})$$

$$\bar{w} \leftarrow \bar{w} + \alpha \begin{cases} \bar{I} & \text{se } \text{Error} > 0 \\ -\bar{I} & \text{se } \text{Error} < 0 \end{cases} \quad \left(\begin{array}{l} \text{(*)} \\ \leftarrow \text{PER UN } \bar{x} \end{array} \right)$$

(13)

REGOLA PERCEPTRONE: CALCOLO INCREMENTALE DI

BACK PROPAGATION (BRYSON & HO 1969)



$j \in \{1, \dots, 4\}$

$k \in \{1, \dots, 10\}$

QUANTE UNITÀ NASCOSTE? (PROBLEMA APERTO)

APPRENDIMENTO RETE FEED-FORWARD (MULTISTRATO)

COME AGGIORNARE I PESI "ENTRANTI"
NELLE UNITÀ NASCOSTE ?

COME OSSERVARE IL COMPORTAMENTO (VAL DI
"SCORRETTO" DELLE UNITÀ NASCOSTE ? ATTIVAZIONE

BACK PROPAGATION (DELL'ERRORE) : METODO
PER EFFETTUARE UNA CORREZIONE DISTRIBUITA
DEI PESI A FRONTE DI UN ERRORE OSSERVATO
(NELLE UNITÀ DI OUTPUT) CON UN ESEMPIO DEL
TRAINING SET.

IDEA (SIMILE PERCEPTRONE) : MODIFICARE
I PESI IN MODO CHE UN ERRORE OSSERVATO
VENGA DIMINUITO (LA VOLTA SUCCESSIVA IN CUI
SI PRESENTA LO STESSO ESEMPIO ALLA RETE)

ALGORITMO DI BACK-PROPAGATION

✓ esempio:

- (1) CALCOLA Δ_i PER OGNI UNITÀ DI OUTPUT OUT_i , USANDO GLI ERRORI OSSERVATI
- (2) A PARTIRE DAL LIVELLO (STRATO) DI OUTPUT RIPETI I SEGUENTI PASSI ✓ STRATO DELLA RETE, FINCHÉ NON È RAGGIUNTO IL LIVELLO PIÙ BASSO
 - A) PROPAGA I VALORI DI Δ DELLE UNITÀ DEL LIVELLO CORRENTE AL LIVELLO SOTTOSTANTE
(CALCOLO Δ_j A PARTIRE DA Δ_i)
 - B) AGGIORNA I PESI TRA I DUE LIVELLI (USANDO LE REGOLE VISTE)

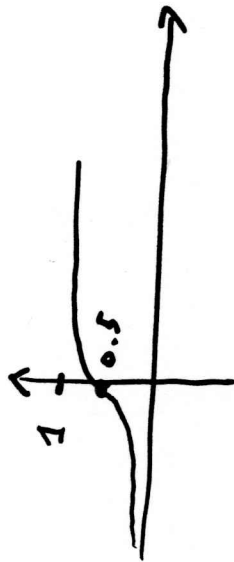
ESEMPIO ...

SOLITAMENTE

$$g = \frac{1}{1 + e^{-x}}$$

(SIGMOIDE)

$$g' = g \cdot (1 - g)$$



function AGGIORNA-PROP-INDIETRO(*rete*, *esempi*, α) returns una rete con pesi modificati

inputs: *rete*, una rete multistrato

esempi, un insieme di coppie ingresso/uscita

α , il tasso di apprendimento

repeat

for each *e* in *esempi* do

/* Calcola l'uscita per questo esempio */

$O \leftarrow \text{LANCIA-RETE}(\text{rete}, I^e)$

/* Calcola l'errore e il Δ per le unità dello strato di uscita */

$\text{Err}^e \leftarrow T^e - O$

/* Si aggiornano i pesi che portano allo strato di uscita */

$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \text{Err}_i^e \times g'(in_i)$

for each strato successivo in *rete* do

/* Si calcoli l'errore in corrispondenza di ciascun nodo */

$\Delta_j \leftarrow g'(in_j) \sum_i W_{j,i} \Delta_i$

/* Si aggiornano i pesi che conducono allo strato */

$W_{k,j} \leftarrow W_{k,j} + \alpha \times I_k \times \Delta_j$

end

end

until *rete* converge

return *rete*

DEVE ESSERE
IL PESO "JECOMO"

Figura 19.14 L'algoritmo di propagazione all'indietro per l'aggiornamento dei pesi in una rete multistrato.

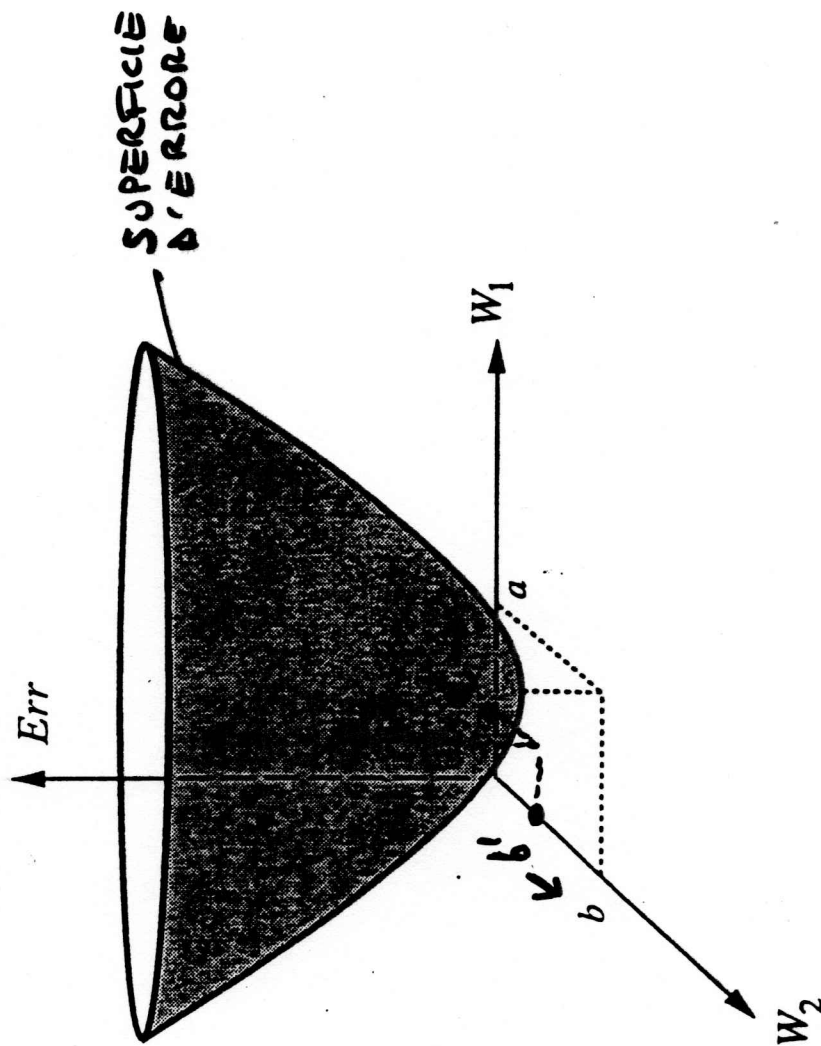


Figura 19.16 Una superficie di errore per una ricerca tramite discesa del gradiente nello spazio dei pesi. Quando $w_1 = a$ e $w_2 = b$ l'errore sull'insieme di addestramento è minimo.

REGOLE AGGIORNAMENTO PESI (RETE A 2 STRATI)



1) CONNESSIONI UNITÀ OUTPUT i

$$w_{ji} \leftarrow w_{ji} + \alpha \cdot a_j \cdot \text{Err}_i \cdot g'(in_i)$$

α = incremento (passo) di apprendimento

a_j = attivazione unità nascosta j

$$\text{Err}_i = T_i - O_i$$

$$in_i = \sum_j w_{ji} \cdot a_j$$

g' = DERIVATA DELLA
FUNZIONE DI
ATTIVAZIONE

$$\Delta_i = \text{Err}_i \times g'(in_i)$$

$$w_{ji} \leftarrow w_{ji} + \alpha a_j \Delta_i$$

↑
ERRORE OSSERVATO PER UNITÀ OUTPUT OUT_i
PESATO CON IL GRADIENTE DI g APPLICATO
ALLA SOMMA TOTALE PESATA DEGLI INPUT DI OUT_i

2) CONNESSIONI AD UNITÀ NASCOSTE (HIDDEN,

$$W_{ji} \leftarrow W_{ji} + \alpha \cdot q_j \cdot \Delta_i \quad \left(\begin{array}{l} \text{REGOLA PER} \\ \text{OUTPUT } \text{OUT}_i, \end{array} \right.$$

NB: Δ_i È USATO PER CAMBIARE TUTTI I PESI
DA UNA UNITÀ j A OUT_i

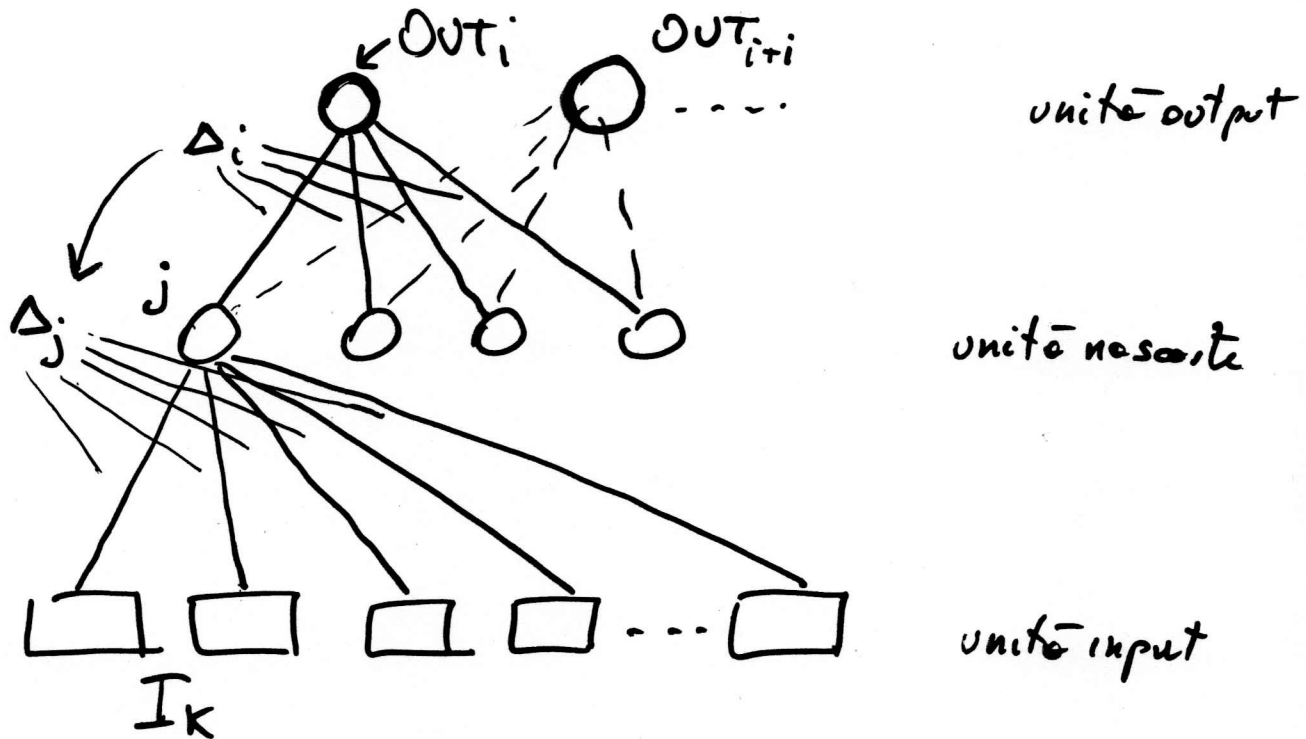
PROBLEMA: IN UNA UNITÀ NASCOSTA NON
C'È IL CONCETTO ESPlicitO DI ERRORE

OSSERVAZIONE: OGNI UNITÀ NASCOSTA j
CONTRIBUISCE ALL'ERRORE Δ_i DI OGNI UNITÀ
OUTPUT i A CUI È CONNESSA PER UNA CERTA
FRAZIONE

CALCOLIAMO Δ_j ($\forall j$) CONSIDERANDO I
 Δ_i DI TUTTE LE UNITÀ OUTPUT i CON CUI j
È CONNESSA PESANDOLI CON W_{ji}

$$\Delta_j = g'(in_j) \cdot \sum_i w_{ji} \cdot \Delta_i$$

$$w_{kj} \leftarrow w_{kj} + \alpha \cdot I_k \cdot \Delta_j$$



Δ_i VIENE PROPAGATA ALL'INDIETRO AD OGNI UNITÀ j COLLEGATA CON i , ED UTILIZZATA PER CORREGGERE I PESI SULLE CONNESSIONI DALLO STRATO INFERIORE DI j

STRUTTURA OTTIMA DELLA RETE

RETE TROPPO PICCOLA $\leadsto$ IL MODELLO PUÒ
ESSERE INCAPACE DI RAPPRESENTARE LA
FUNZIONE DESIDERATA

RETE TROPPO GRANDE $\leadsto$ IL MODELLO PUÒ
ESSERE INCAPACE DI GENERALIZZARE
ADEGUATAMENTE PER INPUT NON VISTI ($\neq$ TRAINING
SET)
(PROBLEMA DELL' "OVERFITTING"
PUÒ SORGERE QUANDO CI SONO TROPPI PARAMETRI - PESI)

$\exists$ VARIE TECNICHE PER DETERMINARE UNA
BUONA STRUTTURA CHE IN PRATICA FUNZIONANO
ABBASTANZA BENE (VEDI RUSSELL & NORVIG)

TEORICAMENTE È UN PROBLEMA APERTO

BACK PROPAGATION COME DISCESA DEL GRADIENTE

LE REGOLE POSSONO ESSERE DERIVATE MATEMATICAMENTE COME REGOLE DI AGGIORNAMENTO DELLO STATO CORRENTE IN UN PROBLEMA DI RICERCA (LOCALE) NELLO SPAZIO DEI PESI DELLA RETE

METODO DISCESA DEL GRADIENTE DELLA FUNZIONE ERRORE CHE DESCRIVE CIASCUN ESEMPIO COME UNA FUNZIONE DI TUTTI I PESI DELLA RETE

$$E = \frac{1}{2} \sum_i (T_i - O_i)^2$$

IL GOAL DEL PROCESSO DI RICERCA È QUELLO DI MODIFICARE I PESI IN MODO CHE E SIA NULLA (O DIA UN VALORE MINIMO) PER OGNI ESEMPIO DEL TRAINING SET

MODIFICA PESI $\leadsto$ SPOSTAMENTO SULLA SUPERFICIE D'ERRORE (IN UNA POSIZIONE CON VALORE INFERIORE)

ESPRESSIVITÀ RETI MULTISTRATO :

$\frac{2^n}{n}$

UNITÀ NASCOSTE PER n INPUT
CONSENTONO DI RAPPRESENTARE
TUTTE LE FUNZIONI BOOLEANE
(IN PRATICA SPESSO MENO, MA DIFFICILE
DIMOSTRARE)

EFFICIENZA :

m esempi e $|\bar{W}|$ pesi

ogni epoca richiede $O(m \cdot |\bar{W}|)$ tempo

NEL CASO PEGGIORE È NECESSARIO UN
NUMERO ESPONENZIALE DI EPOCHE

IN PRATICA, TEMPO CONVERGENZA MOLTO
VARIABILE (INOLTRE POSSIBILI MINIMI
LOCALI)

GENERALIZZAZIONE : EVITARE TRAINING
SET TROPPO GRANDI

TRASPARENZA : UNA R.M. È UNA
"SCATOLA NERA" !

$$E = \frac{1}{2} \sum_i (T_i - O_i)^2 =$$

$$E(\bar{w}) = \frac{1}{2} \sum_i (T_i - g(\sum_j w_{ji} \cdot a_j))^2 =$$

$$E(\bar{w}) = \frac{1}{2} \sum_i (T_i - g(\sum_j \overset{w_{ji}}{w_{ji}} \cdot g(\sum_k w_{kj} I_k)))^2$$

$$E(\bar{w}) = \frac{1}{2} \sum_{i \in \text{OUT}} (T_i - g(\sum_{j \in H} \overset{w_{ji}}{w_{ji}} \cdot g(\sum_{k \in \text{IN}} w_{kj} I_k)))^2$$

OUT : UNITĂ OUTPUT

IN : UNITĂ INPUT

H : UNITĂ NASCOSTE

ESERCIZIO:
FARE PASSAGGI
MATEMATICI

$$\frac{\partial E}{\partial w_{ji}} = -a_j \underbrace{(T_i - O_i)}_{\text{Err}_i} \cdot g'(\underbrace{\sum_j w_{ji} a_j}_{\text{in}_i}) = -a_j \Delta_i$$

$$\Delta_i = \text{Err}_i \cdot g'(\text{in}_i)$$

$$w_{ji} \leftarrow w_{ji} - \alpha \frac{\partial E}{\partial w_{ji}} = w_{ji} + \alpha a_j \Delta_i$$

```

function APPENDIMENTO-PROP-INDIETRO(esempi, rete) returns una rete neurale
  inputs: esempi, un insieme di esempi, ognuno con vettore di input  $x$  e di output
           rete, una rete multistrato con  $L$  strati, pesi  $W_{j,i}$  e funzione di attivazione

  repeat
    for each  $e$  in esempi do
      for each nodo  $j$  nello strato di input do  $a_j \leftarrow x_j[e]$ 
      for  $\ell = 2$  to  $L$  do
         $in_i \leftarrow \sum_j W_{j,i} a_j$ 
         $a_i \leftarrow g(in_i)$ 
        for each nodo  $i$  nello strato di output do
           $\Delta_i \leftarrow g'(in_i) \times (y_i[e] - a_i)$ 
          for  $\ell = L - 1$  to  $1$  do
            for each nodo  $j$  nello strato  $\ell$  do
               $\Delta_j \leftarrow g'(in_j) * \sum_i W_{j,i} \Delta_i$ 
              for each nodo  $i$  nello strato  $\ell + 1$  do
                 $W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$ 
      until non è soddisfatto un qualche criterio di terminazione
  return IPOTESI-RETE-NEURALE(rete)

```

Figura 20.25 L'algoritmo di propagazione all'indietro per l'apprendimento nelle reti multistrato.

Il processo di retropropagazione può essere riassunto come segue.

- ♦ Si calcolano i valori Δ per le unità di output usando l'errore osservato.
- ♦ Cominciando dallo strato di output, si ripete quanto segue per ogni strato della rete fino a raggiungere l'ultimo strato nascosto:
 - si propagano all'indietro i valori Δ verso lo strato precedente
 - si aggiornano i pesi tra i due strati.

L'algoritmo dettagliato è riportato nella Figura 20.25.

Per i lettori con un'inclinazione matematica deriviamo ora le equazioni della retropropagazione partendo da principi fondamentali. Il quadrato dell'errore su un singolo esempio è definito come