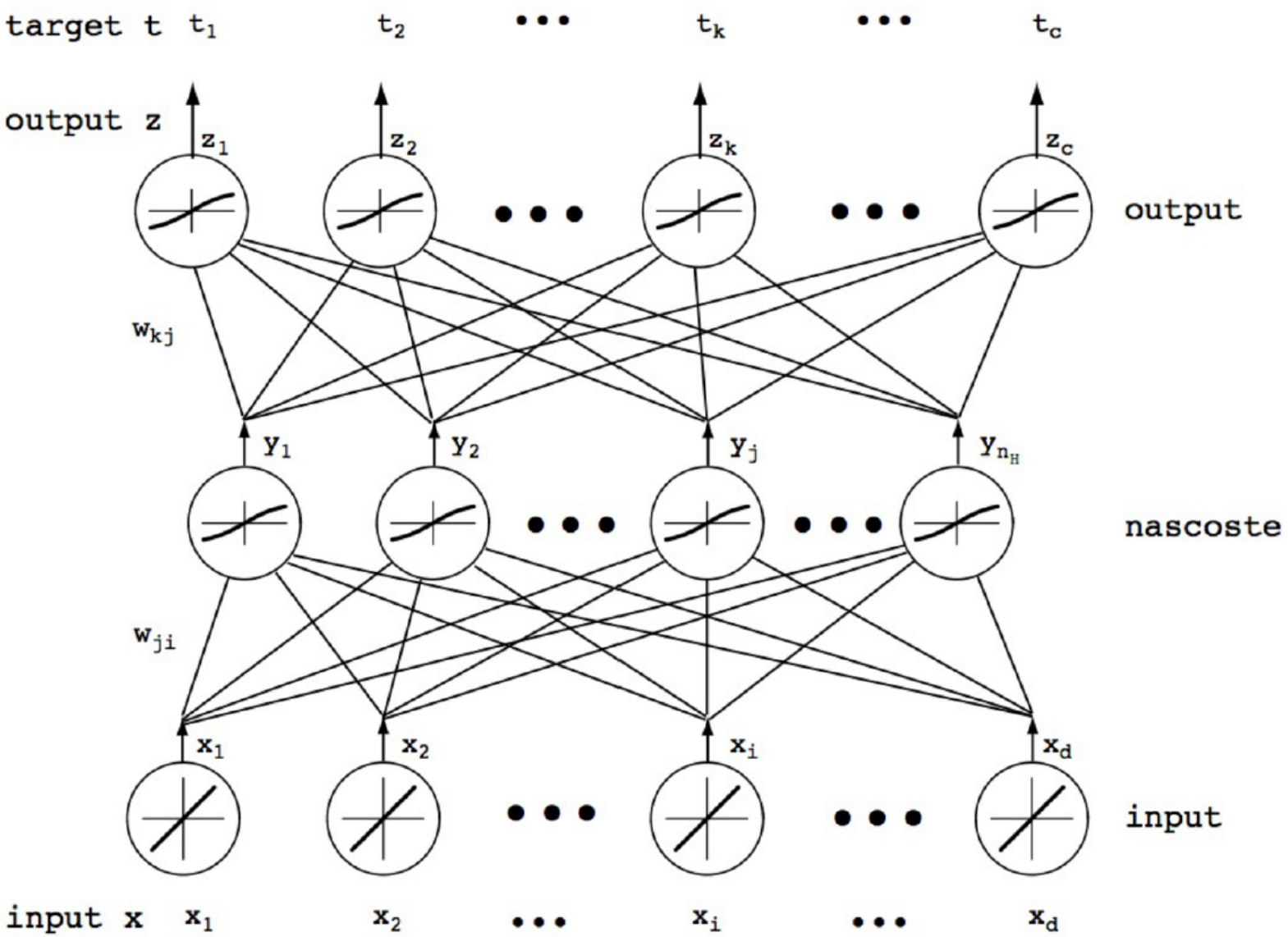
Reti Neurali (Parte III)

Corso di AA, anno 2017/18, Padova

Fabio Aiolli

08 Novembre 2017

Reti Neurali Multistrato

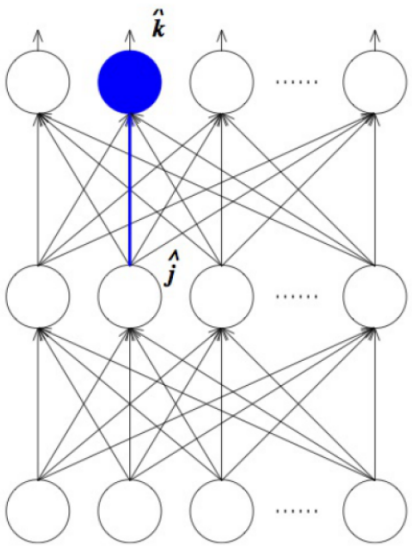


Reti Neurali Multistrato: terminologia

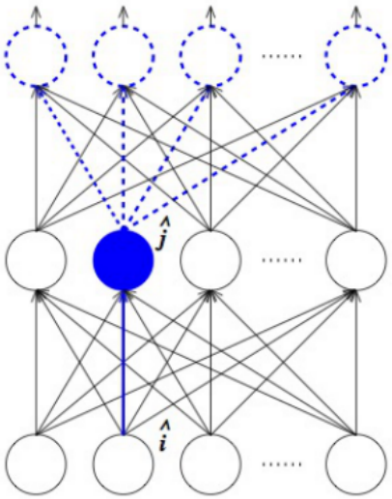
- d unità di ingresso: dimensione dei dati in ingresso $\mathbf{x} \equiv (x_1, \ldots, x_d)$, ($d+1$ se includiamo la soglia nel vettore dei pesi $\mathbf{x}' \equiv (x_0, x_1, \ldots, x_d)$)
- N_H unità nascoste (con output $\mathbf{y} \equiv (y_1, \ldots, y_{N_H})$)
- c unità di output: dimensione dei dati in output $\mathbf{z} = (z_1, \ldots, z_c)$ e dimensione degli output desiderati $\mathbf{t} = (t_1, \ldots, t_c)$
- w_{ji} peso dalla unità di ingresso i alla unità nascosta j ($\mathbf{w}_j$ è il vettore pesi dell'unità nascosta j)
- w_{kj} peso dalla unità nascosta j alla unità di output k ($\mathbf{w}_k$ è il vettore pesi dell'unità di output k)

Calcolo gradiente per i pesi di una unità di output

$$
\begin{aligned}
\frac{\partial E}{\partial w_{\hat{k}\hat{j}}} &= \frac{\partial}{\partial w_{\hat{k}\hat{j}}} \left[\frac{1}{2cN} \sum_{s=1}^{N} \sum_{k=1}^{c} (t_k^{(s)} - z_k^{(s)})^2 \right] \\
&= \frac{1}{2cN} \sum_{s=1}^{N} \frac{\partial}{\partial w_{\hat{k}\hat{j}}} \left[\sum_{k=1}^{c} (t_k^{(s)} - z_k^{(s)})^2 \right] \\
&= \frac{1}{2cN} \sum_{s=1}^{N} 2(t_{\hat{k}}^{(s)} - z_{\hat{k}}^{(s)}) \frac{\partial}{\partial w_{\hat{k}\hat{j}}} \left[(t_{\hat{k}}^{(s)} - z_{\hat{k}}^{(s)}) \right] \\
&= \frac{1}{cN} \sum_{s=1}^{N} (t_{\hat{k}}^{(s)} - z_{\hat{k}}^{(s)}) \frac{\partial}{\partial w_{\hat{k}\hat{j}}} \left[t_{\hat{k}}^{(s)} - \sigma(\mathbf{w}_{\hat{k}} \cdot \mathbf{y}^{(s)}) \right] \\
&= -\frac{1}{cN} \sum_{s=1}^{N} (t_{\hat{k}}^{(s)} - z_{\hat{k}}^{(s)}) \sigma'(\mathbf{w}_{\hat{k}} \cdot \mathbf{y}^{(s)}) y_{\hat{j}}^{(s)} \\
&= -\frac{1}{cN} \sum_{s=1}^{N} (t_{\hat{k}}^{(s)} - z_{\hat{k}}^{(s)}) z_{\hat{k}}(1 - z_{\hat{k}}) y_{\hat{j}}^{(s)}
\end{aligned}
$$



Calcolo gradiente per i pesi di unità nascoste



$$
\begin{aligned}
\frac{\partial E}{\partial w_{\hat{j}\hat{i}}} &= \frac{\partial}{\partial w_{\hat{j}\hat{i}}} \left[\frac{1}{2cN} \sum_{s=1}^{N} \sum_{k=1}^{c} (t_k^{(s)} - z_k^{(s)})^2 \right] \\
&= \frac{1}{2cN} \sum_{s=1}^{N} \sum_{k=1}^{c} \frac{\partial}{\partial w_{\hat{j}\hat{i}}} \left[(t_k^{(s)} - z_k^{(s)})^2 \right] \\
&= \frac{1}{2cN} \sum_{s=1}^{N} \sum_{k=1}^{c} 2(t_k^{(s)} - z_k^{(s)}) \frac{\partial}{\partial w_{\hat{j}\hat{i}}} \left[-z_k^{(s)} \right] \\
&= -\frac{1}{cN} \sum_{s=1}^{N} \sum_{k=1}^{c} (t_k^{(s)} - z_k^{(s)}) \sigma'(\mathbf{w}_k \cdot \mathbf{y}^{(s)}) \frac{\partial}{\partial w_{\hat{j}\hat{i}}} \left[\sum_{j=1}^{N_H} w_{kj} y_j^{(s)} \right] \\
&= -\frac{1}{cN} \sum_{s=1}^{N} \sum_{k=1}^{c} (t_k^{(s)} - z_k^{(s)}) \sigma'(\mathbf{w}_k \cdot \mathbf{y}^{(s)}) w_{k\hat{j}} \frac{\partial}{\partial w_{\hat{j}\hat{i}}} \left[y_{\hat{j}}^{(s)} \right]
\end{aligned}
$$

Calcolo gradiente per i pesi di unità nascoste

$$
\begin{aligned}
&= -\frac{1}{cN}\sum_{s=1}^{N}\sum_{k=1}^{c}(t_k^{(s)} - z_k^{(s)})\sigma'(\mathbf{w}_k \cdot \mathbf{y}^{(s)})w_{k\hat{j}}\frac{\partial}{\partial w_{\hat{j}\hat{i}}}\left[y_{\hat{j}}^{(s)}\right] \\
&= -\frac{1}{cN}\sum_{s=1}^{N}\sum_{k=1}^{c}(t_k^{(s)} - z_k^{(s)})\sigma'(\mathbf{w}_k \cdot \mathbf{y}^{(s)})w_{k\hat{j}}\sigma'(\mathbf{w}_{\hat{j}} \cdot \mathbf{x}^{(s)})\frac{\partial}{\partial w_{\hat{j}\hat{i}}}\left[\mathbf{w}_{\hat{j}} \cdot \mathbf{x}^{(s)}\right] \\
&= -\frac{1}{cN}\sum_{s=1}^{N}\sum_{k=1}^{c}(t_k^{(s)} - z_k^{(s)})\sigma'(\mathbf{w}_k \cdot \mathbf{y}^{(s)})w_{k\hat{j}}\sigma'(\mathbf{w}_{\hat{j}} \cdot \mathbf{x}^{(s)})x_{\hat{i}}^{(s)} \\
&= -\frac{1}{cN}\sum_{s=1}^{N}\sigma'(\mathbf{w}_{\hat{j}} \cdot \mathbf{x}^{(s)})x_{\hat{i}}^{(s)}\sum_{k=1}^{c}(t_k^{(s)} - z_k^{(s)})w_{k\hat{j}}\sigma'(\mathbf{w}_k \cdot \mathbf{y}^{(s)}) \\
&= -\frac{1}{cN}\sum_{s=1}^{N}y_{\hat{j}}^{(s)}(1 - y_{\hat{j}}^{(s)})x_{\hat{i}}^{(s)}\sum_{k=1}^{c}(t_k^{(s)} - z_k^{(s)})z_k^{(s)}(1 - z_k^{(s)})w_{k\hat{j}}
\end{aligned}
$$

Discesa Batch vs. Stocastica

- **Batch**. Finché la condizione di terminazione non è soddisfatta:
  1. Calcola $\nabla E_S[\mathbf{w}]$
  2. $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E_S[\mathbf{w}]$
- **Stocastica**. Finché la condizione di terminazione non è soddisfatta, per ogni esempio di training s:
  1. Calcola $\nabla E_s[\mathbf{w}]$
  2. $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E_s[\mathbf{w}]$
- **Mini-batch**. Finché la condizione di terminazione non è soddisfatta, consideriamo un sottoinsieme di esempi $Q \subseteq S$:
  1. Calcola $\nabla E_Q[\mathbf{w}]$
  2. $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E_Q[\mathbf{w}]$

dove in generale $E_Q[\mathbf{w}] = \frac{1}{2cN} \sum_{s \in Q} \sum_{k=1}^{c} (t_k^{(s)} - z_k^{(s)})^2, Q \subseteq S$.

Algoritmo Backprop con uno strato nascosto

Back-Propagation-1hl-stocastico(S, η):

- Inizializza tutti i pesi a valori random piccoli (es. tra -0.05 e +0.05)
- Finché non è verificata la condizione di terminazione:
  - Per ogni $(\mathbf{x}, \mathbf{t}) \in S$,
    1. Presenta $\mathbf{x}$ alla rete e calcola i vettori $\mathbf{y}$ e $\mathbf{z}$
    2. Per ogni unità di output k,

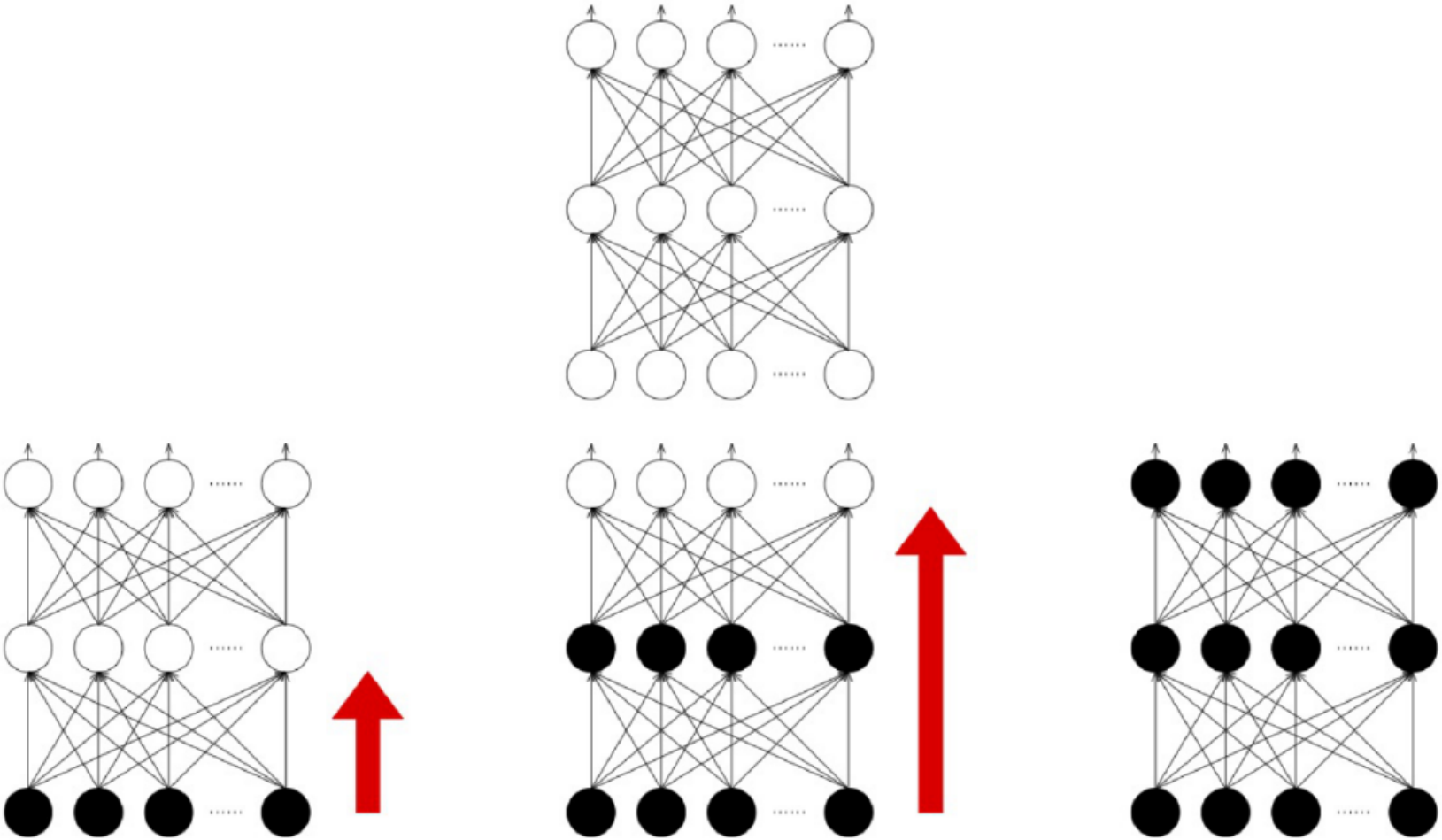
$$\begin{aligned} \delta_k &= z_k(1 - z_k)(t_k - z_k) \\ \Delta w_{kj} &= \delta_k y_j \end{aligned}$$

    3. Per ogni unità nascosta j,

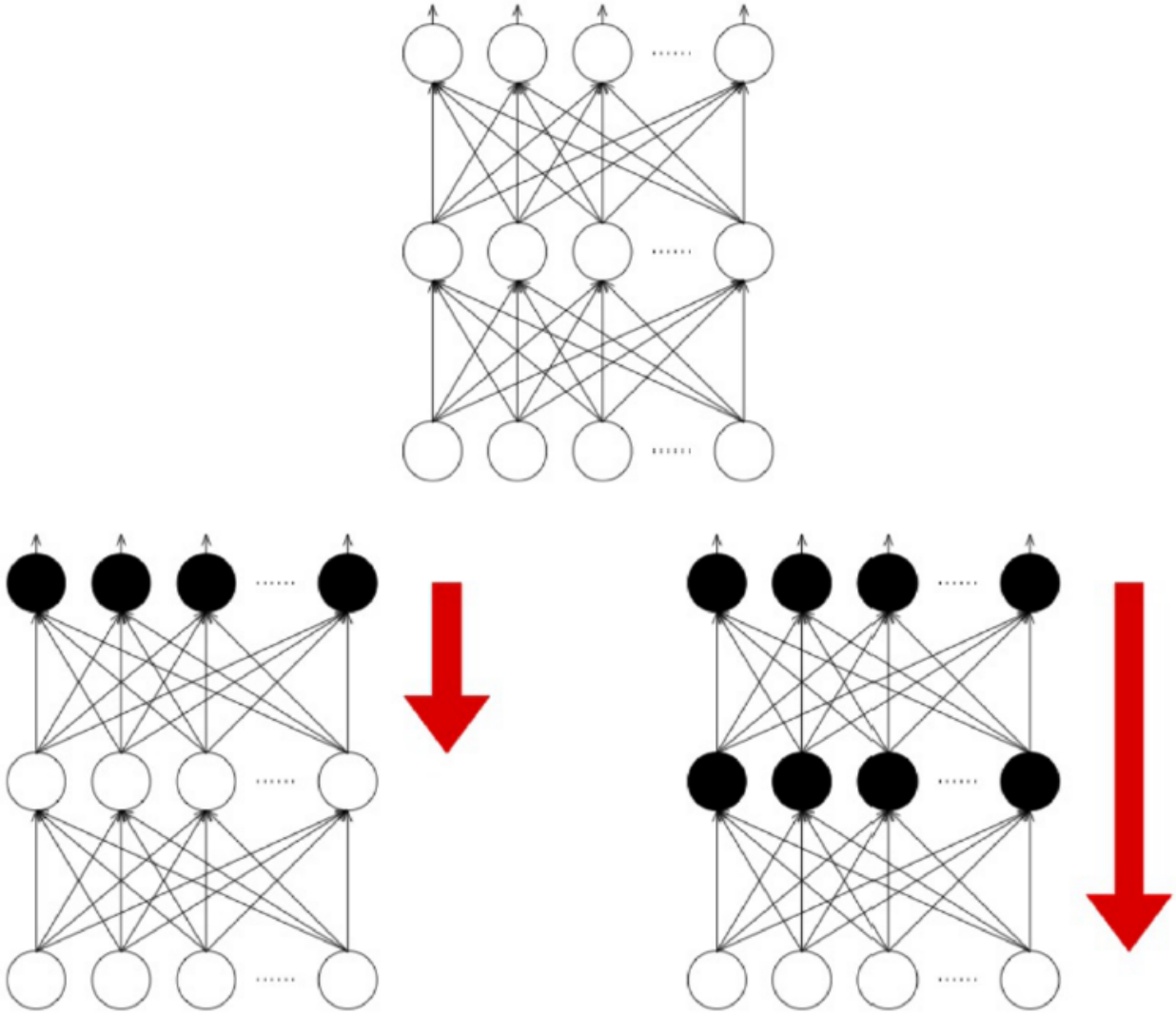
$$\begin{aligned} \delta_j &= y_j(1 - y_j) \sum_{k=1}^{c} w_{kj}\delta_k \\ \Delta w_{ji} &= \delta_j x_i \end{aligned}$$

    4. Aggiorna tutti i pesi: $w_{sq} \leftarrow w_{sq} + \eta \Delta w_{sq}$

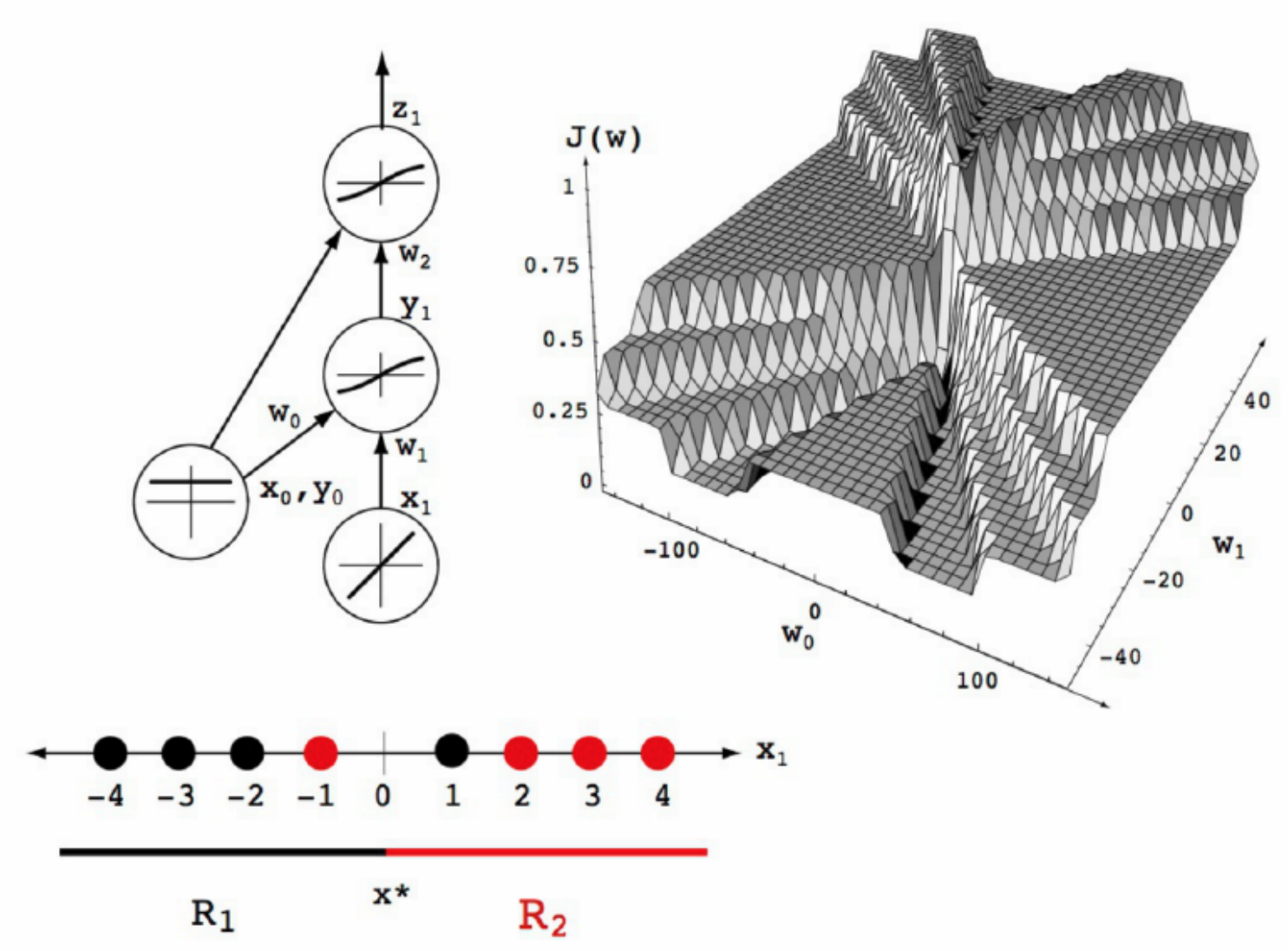
Fase Forward



Fase Backward



Esempio di funzione errore



Universalità delle reti multi-strato

Teorema (Pinkus, 1996) semplificato

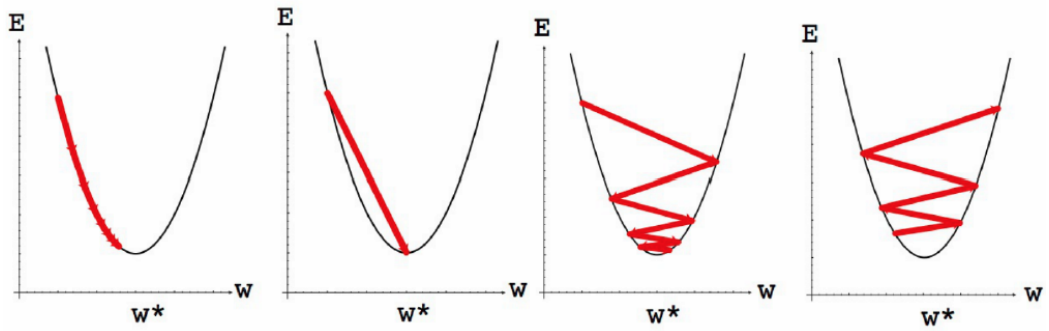
Data una rete feed-forward con un solo livello nascosto, una qualsiasi funzione continua $f : \mathbb{R}^n \to \mathbb{R}$, un qualunque $\epsilon > 0$ piccolo a piacere, per un'ampia classe di funzioni di attivazione, esiste sempre un intero M tale che, la funzione $g : \mathbb{R}^n \to \mathbb{R}$ calcolata dalla rete usando almeno M unità nascoste approssima la funzione f con tolleranza ϵ, ovvero:

$$\max_{\mathbf{x} \in \Omega} |f(\mathbf{x}) - g(\mathbf{x})| < \epsilon$$

Notare che il teorema afferma l'esistenza di una rete con M unità nascoste che approssima la funzione target con la tolleranza desiderata ma non dice niente su come tale numero M possa essere calcolato!

Alcuni problemi..

- La scelta della tipologia della rete determina lo spazio delle ipotesi che si utilizza. Con architetture a 3 livelli (input, hidden, output), il numero delle unità nascoste determina la complessità dello spazio delle ipotesi
- La scelta del passo di discesa (valore di η) può essere determinante per la convergenza:

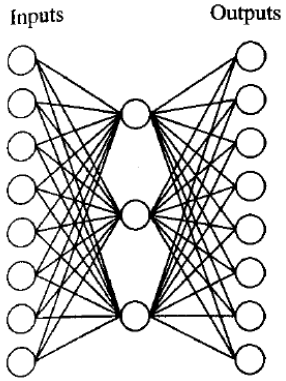


- L'apprendimento è generalmente lento (ma il calcolo dell'output a regime è veloce)
- Minimi locali (anche se efficace in pratica)

Evitare minimi locali

- Aggiunta di un termine (chiamato momento) alla regola di update dei pesi. In pratica viene aggiunto un contributo che deriva dal passo precedente, imponendo una specie di inerzia al sistema
- L'uso di apprendimento stocastico (randomizzando sugli esempi) invece che batch può facilitare l'uscita da minimi locali
- Apprendimento di reti diverse sugli stessi dati con inizializzazioni differenti e selezione della rete più efficace (per esempio valutando su un validation set). Alternativamente, possiamo considerare un "comitato" di reti in cui la predizione è una media (eventualmente pesata) delle predizioni delle singole reti

Rappresentazione dei livelli nascosti

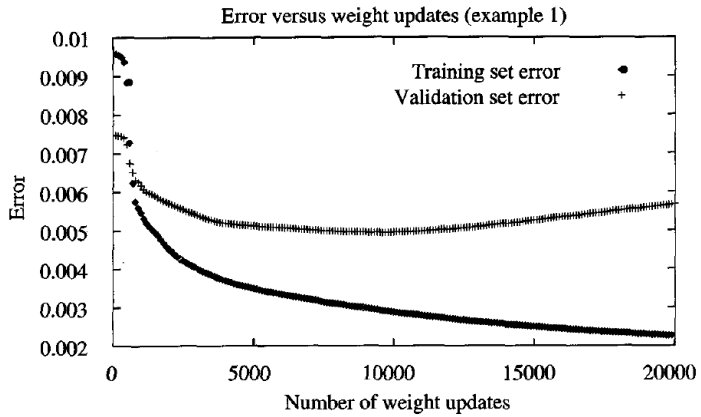


Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

FIGURE 4.7
Learned Hidden Layer Representation. This $8 \times 3 \times 8$ network was trained to learn the identity function, using the eight training examples shown. After 5000 training epochs, the three hidden unit values encode the eight distinct inputs using the encoding shown on the right. Notice if the encoded values are rounded to zero or one, the result is the standard binary encoding for eight distinct values.

Una importante caratteristica delle reti neurali multi-strato è che permettono di scoprire utili rappresentazioni (alternative all'input) dei dati di ingresso. In particolare, l'output delle unità nascoste è una efficace rappresentazione dell'input che permette una più semplice separazione dei dati in output. Nell'esempio si nota come la rappresentazione (encoding) appreso assomigli all'encoding binario su 3 bit ($[100, 011, 010, \dots]$).

Overfitting



- Aumentando il numero di updates sui pesi, l'errore sul validation set prima diminuisce poi incrementa. Perché?
- All'inizio, con valori dei pesi piccoli in modulo (molto simili tra di loro) si riescono a descrivere superfici di decisione più "lisce". All'aumentare del valore assoluto la complessità della superfice di decisione aumenta e con essa anche la possibilità di avere overfitting
- Soluzioni: Monitorare l'errore su un insieme di dati di validazione, oppure utilizzare un termine aggiuntivo nella funzione errore dipendente dalla norma dei pesi (regolarizzazione).