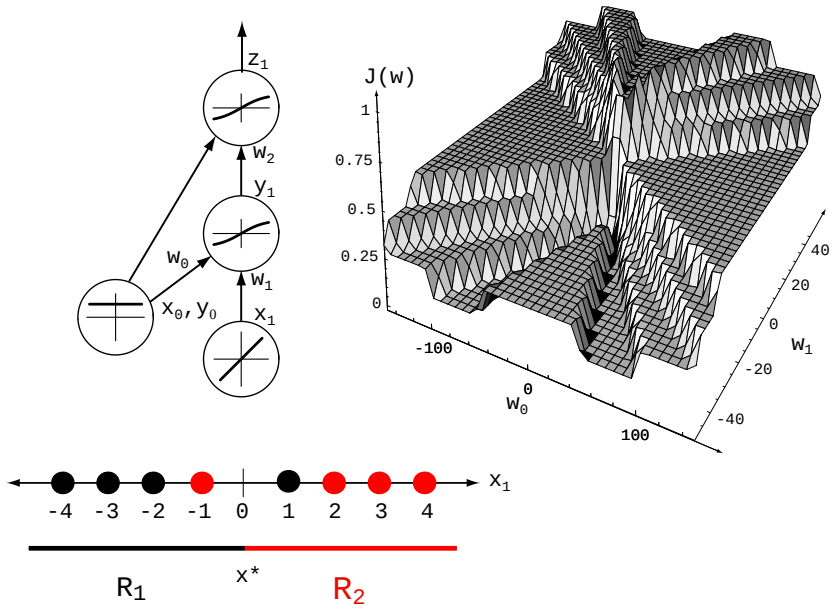
Esempio di Funzione Errore



Discesa di Gradiente Batch e Stocastica

Batch:

Fai finché condizione di terminazione non soddisfatta

1. calcola $\nabla E_{Tr}[\vec{w}]$
2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_{Tr}[\vec{w}]$

Stocastica (Incrementale):

Fai finché condizione di terminazione non soddisfatta

- Per ogni esempio di apprendimento p in Tr
  1. calcola $\nabla E_{p \in Tr}[\vec{w}]$
  2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_{p \in Tr}[\vec{w}]$

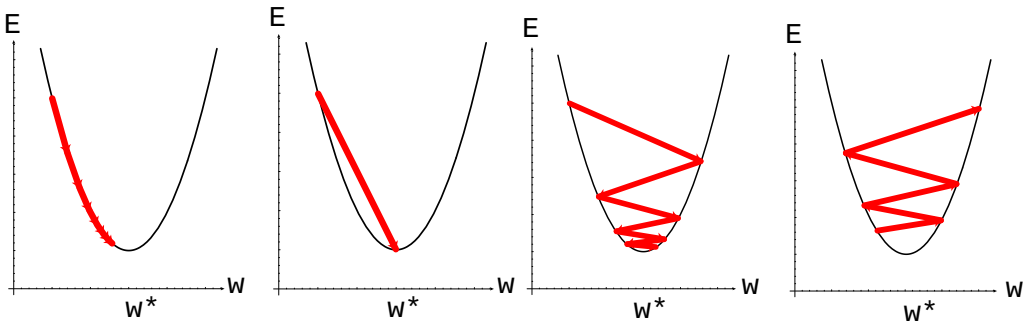
dove

$$E_{Tr}[\vec{w}] \equiv \frac{1}{2cN_{Tr}} \sum_{p \in Tr} \sum_{k=1}^{c} (t_k^{(p)} - z_k^{(p)})^2 \qquad E_{p \in Tr}[\vec{w}] \equiv \frac{1}{2c} \sum_{k=1}^{c} (t_k^{(p)} - z_k^{(p)})^2$$

La discesa di gradiente *Stocastica* (gradiente istantaneo) può approssimare quella *Batch* (gradiente esatto) con precisione arbitraria se η è sufficientemente piccolo

Alcuni Problemi ...

- Scelta della topologia della rete $\rightarrow$ determina lo Spazio delle Ipotesi;
- Scelta del passo di discesa (valore di η):

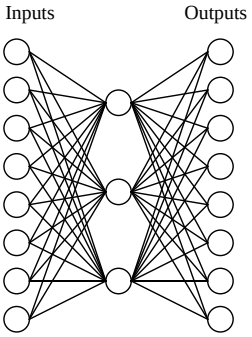


- apprendimento lento..., ma calcolo di output veloce
- MINIMI LOCALI !!

Bias Induttivo: sia nella rappresentazione che nella ricerca

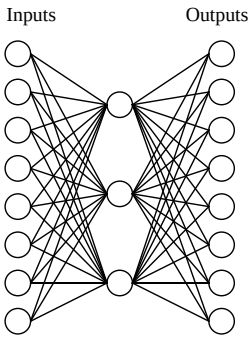
Esempio di Apprendimento per Rete Feed-forward

Compressione di Dati



Input	Output
00000001	00000001
00000010	00000010
00000100	00000100
00001000	00001000
00010000	00010000
00100000	00100000
01000000	01000000
10000000	10000000

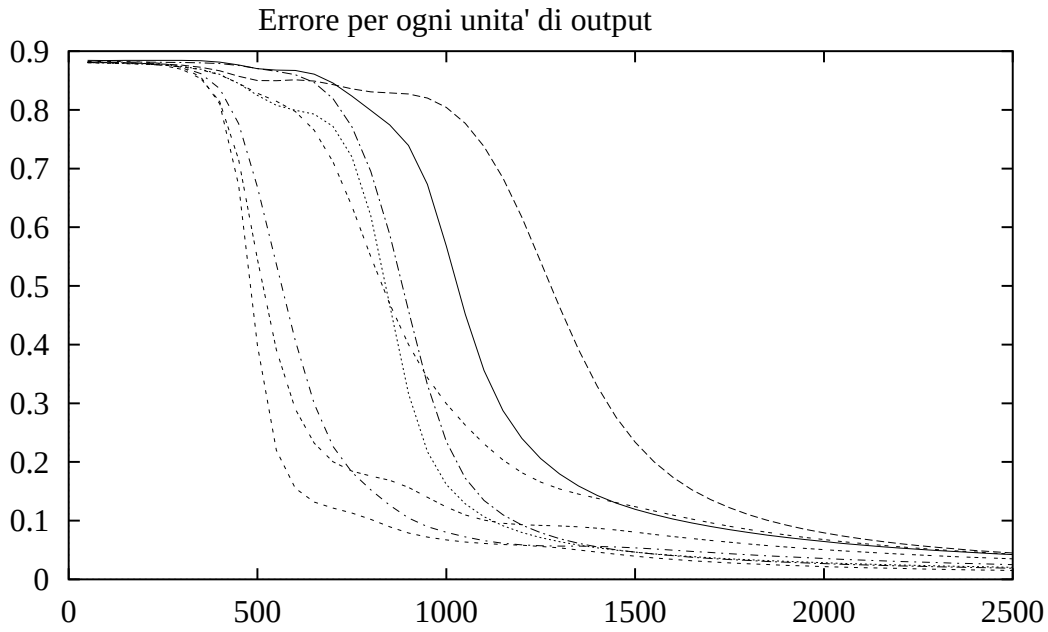
Esempio di Apprendimento per Rete Feed-forward



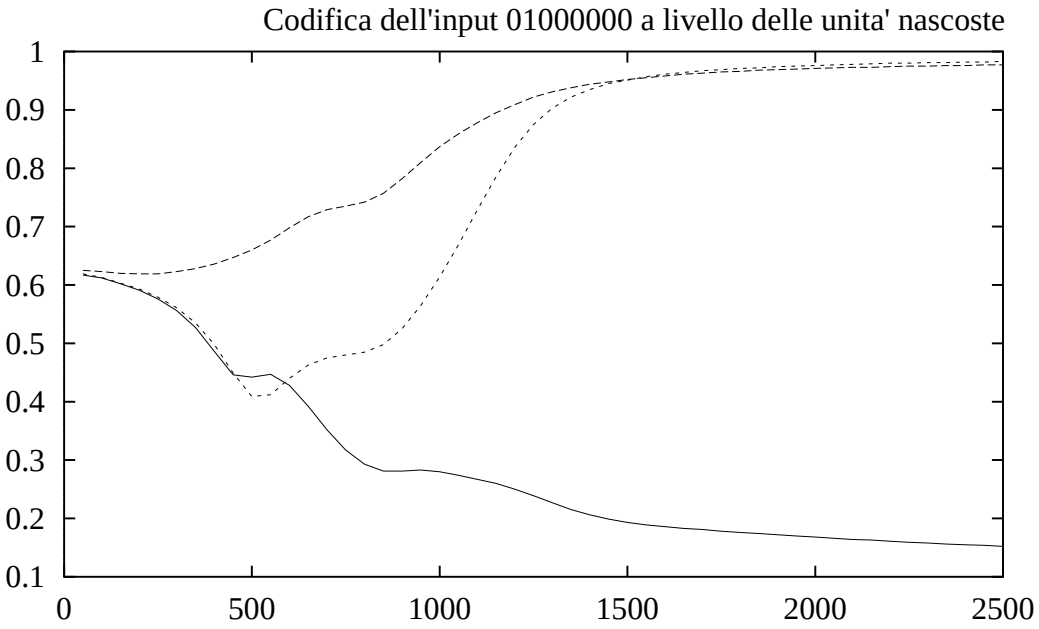
Compressione di Dati

Input		Valori Nascosti		Output
10000000	→	0.89 0.04 0.08	→	10000000
01000000	→	0.01 0.11 0.88	→	01000000
00100000	→	0.01 0.97 0.27	→	00100000
00010000	→	0.99 0.97 0.71	→	00010000
00001000	→	0.03 0.05 0.02	→	00001000
00000100	→	0.22 0.99 0.99	→	00000100
00000010	→	0.80 0.01 0.98	→	00000010
00000001	→	0.60 0.94 0.01	→	00000001

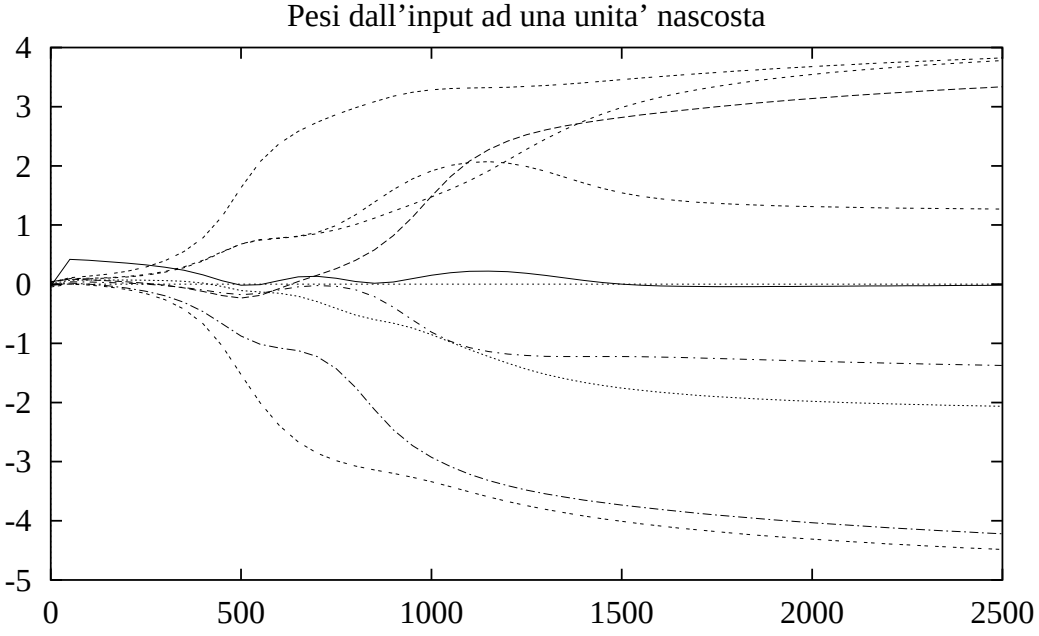
Curve di Apprendimento



Curve di Apprendimento



Curve di Apprendimento



Potere Computazionale Reti Neurali

Il seguente teorema stabilisce l'universalità di reti feed-forward come approssimatori di funzioni continue.

Teorema Sia $\varphi(\cdot)$ una funzione continua monotona crescente, limitata e noncostante. Si indichi con I_n l'ipercubo n-dimensionale $[0,1]^n$ e lo spazio delle funzioni continue su esso definite sia $C(I_n)$. Data una qualunque funzione $f \in C(I_n)$ e $\varepsilon > 0$, allora esiste un intero M e insiemi di costanti reali α_i, θ_i, e w_{ij}, dove $i = 1, \ldots, M$ e $j = 1, \ldots, n$ tale che $f(\cdot)$ possa essere approssimata da

$$F(x_1, \ldots, x_n) = \sum_{i=1}^{M} \alpha_i \varphi(\sum_{j=1}^{n} w_{ij} x_j - \theta_i) \quad (1)$$

in modo tale che

$$|F(x_1, \ldots, x_n) - f(x_1, \ldots, x_n)| < \varepsilon \quad (2)$$

per tutti i punti $[x_1, \ldots, x_n] \in I_n$.

Potere Computazionale Reti Neurali

Notare che qualunque funzione sigmoidale soddisfa le condizioni imposte nel teorema su $\varphi(\cdot)$. Inoltre, l'equazione (1) rappresenta l'output di una rete multistrato descritta come segue

1. la rete ha n nodi di input ed un singolo strato di unità nascoste con M unità; gli input sono denotati da $x_1, \ldots, x_n$.
2. l'i-esima unità ha associati i pesi $w_{i1}, \ldots, w_{in}$ e soglia θ_i.
3. l'output della rete è una combinazione lineare degli output delle unità nascoste, dove i coefficienti della combinazione sono dati da $\alpha_1, \ldots, \alpha_M$.

Quindi, data una tolleranza ε, una rete con un unico strato nascosto può approssimare una qualsiasi funzione in $C(I_n)$.

Si noti che il teorema afferma solo l'esistenza di una rete e non fornisce alcuna formula per il calcolo del numero M di unità nascoste necessarie per approssimare la funzione target con la tolleranza desiderata.

Esempio di Algoritmo Lazy: k-NN

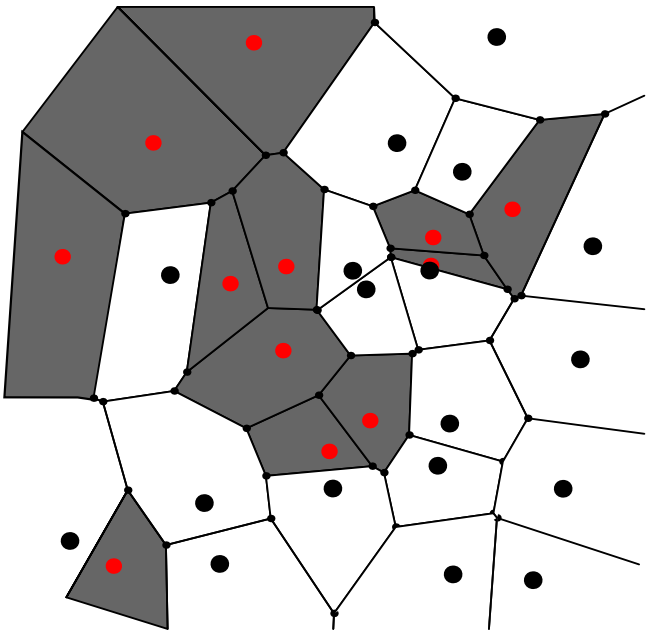
Un esempio di algoritmo Lazy è dato dal k-Nearest Neighbor

Vediamo il caso $k = 1$:

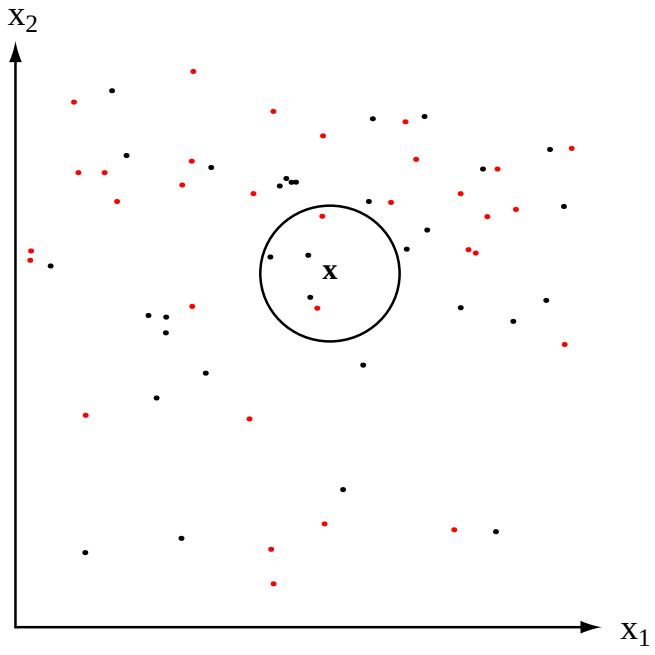
- si memorizzano i dati di apprendimento
- quando si deve effettuare una classificazione di un nuovo ingresso y, si restituisce la classe associata all'esempio più vicino (tramite una metrica opportune: es. Euclidea) memorizzato:

Ovviamente k-NN è lento nel rispondere: il tempo di risposta dipende dal numero di esempi memorizzati

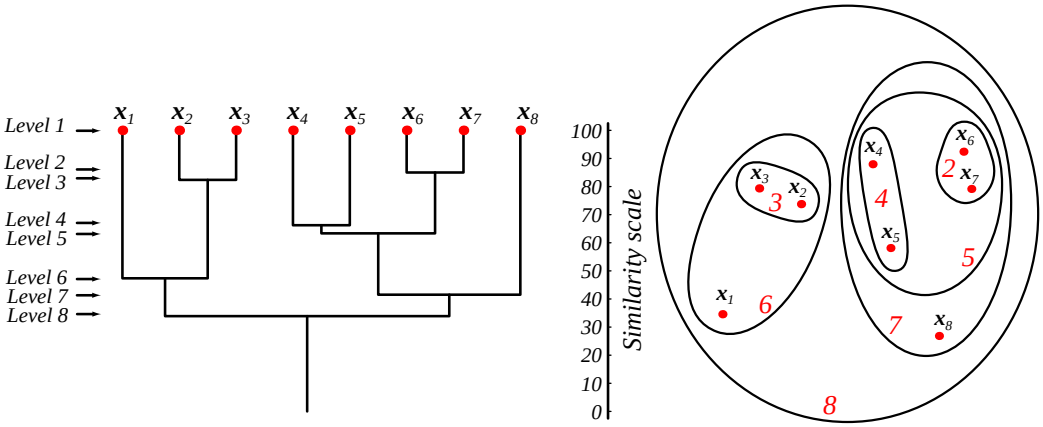
k-NN: diagramma di Voronoi



k-NN: k>1



Un esempio di clustering: Clustering Gerarchico



1. si selezionano i 2 vettori più vicini (es., secondo la distanza euclidea) e si costruisce un sottoalbero con figli dati dai due vettori e padre il centroide (media) dei due vettori;
2. i due vettori vengono sostituiti nell'insieme dei vettori correnti dal centroide, e si torna al passo 1.