Metodi supervisionati di classificazione

Giorgio Valentini

e-mail: valentini@dsi.unimi.it

UNIVERSITÀ DEGLI STUDI DI MILANO

DSI - Dipartimento di Scienze dell'Informazione

Classificazione bio-molecolare di tessuti e geni

- Diagnosi a livello bio-molecolare:
  - Identificazione e classificazione di pazienti sani e malati.
  - Classificazione di differenti tipologie patologiche
  - Diagnosi basata sulla integrazione di dati biologici eterogenei
- Predizione di esiti clinici:
  - Predizione delle risposte di pazienti a trattamenti farmacologici
  - Sviluppo di tool prognostici per uso clinico e per la risposta a terapie
- Identificazione di molecole target per lo sviluppo di farmaci
- ...

Classificazione di classi funzionali come problema di apprendimento automatico (1)

I dati generati da bio-tecnologie high-throughput (ad es: DNA microarray) sono rappresentabili come insiemi di coppie (**x**,t):

- $\mathbf{x} \in R^d$, $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ rappresenta i livelli di espressione genica di *d* geni
- t rappresenta un particolare stato funzionale

Es: $t \in C = \{ s, m \}$, s → paziente sano, m→malato

Formato dei dati DNA microarray

X_{jk} rappresenta il livello di espressione del gene *j* nell' array *k*

N è il numero dei geni e $K = K_1 + K_2$ il numero degli array

$$C_1 = \left\{ X_{jk} \middle| 1 \le k \le K_1, 1 \le j \le N \right\} \qquad C_2 = \left\{ X_{jk} \middle| K_1 + 1 \le k \le K_1 + K_2, 1 \le j \le N \right\}$$

	Array1	Array2	...	ArrayK_1	ArrayK_1+1	...	Array$K$
Gene 1	X_{11}	X_{12}	...	X_{1K1}	X_{1K1+1}	...	X_{1K}
Gene 2	X_{21}	X_{22}	...	X_{2K1}	X_{2K1+1}	...	X_{2K}
...	...	...	...	...	...	...	...
Gene n	X_{N1}	X_{N2}	...	X_{NK1}	X_{NK1+1}	...	X_{NK}

Classificazione di classi funzionali come problema di apprendimento automatico (2)

Obiettivo dell' apprendimento automatico:

Apprendere la funzione non nota f :
$f : R^d \rightarrow C$ che mappa i livelli di espressione genica $\mathbf{x} \in R^d$ nella corrispondente classe funzionale $t \in C$ (es: paziente sano o malato)

tramite un algoritmo di apprendimento (learning machine) L che utilizza solo un training set $D = \{(x_i, t_i)\}_{i=1}^{n}$ di campioni distribuiti in accordo alla distribuzione di probabilità congiunta P(**x**,t).

Algoritmi di apprendimento supervisionato e learning machine

- L' algoritmo di apprendimento *L* genera un' approssimazione $g : R^d \rightarrow C$ della funzione non nota *f* utilizzando il training set *D*: $L(D) \rightarrow g$.
- Si desidera che tale funzione sia la più "vicina" possibile ad *f*
- A tal fine si usa una funzione di perdita Loss*(f(***x**),g(**x**)) che misuri quanto *g* differisca da *f*.
- Nei problemi di classificazione si usa la funzione di perdita 0/1:

$$Loss(g(x), f(x)) = \begin{cases} 1 & se \quad g(x) \neq f(x) \\ 0 & se \quad g(x) = f(x) \end{cases}$$

Ma f non è nota (se lo fosse avremmo risolto il problema) ...

Addestramento delle learning machine

- Nella realtà si dispone spesso solo di un insieme relativamente limitato di dati (ad es: un insieme *D* di dati di espressione genica) e la learning machine viene addestrata ad approssimare *f* utilizzando tali dati come una serie di esempi da apprendere:

($\mathbf{x}_1$,t_1), ($\mathbf{x}_2$,t_2), ... ($\mathbf{x}_n$,t_n).

- La learning machine verrà addestrata ad apprendere una funzione *g* tale che $g(\mathbf{x}_1)=t_1$, $g(\mathbf{x}_2)=t_2$, ..., $g(\mathbf{x}_n)=t_n$, in modo da minimizzare il rischio empirico R_{emp} rispetto al training set $D=\left\{\left(x_i,t_i\right)\right\}_{i=1}^{n}$:

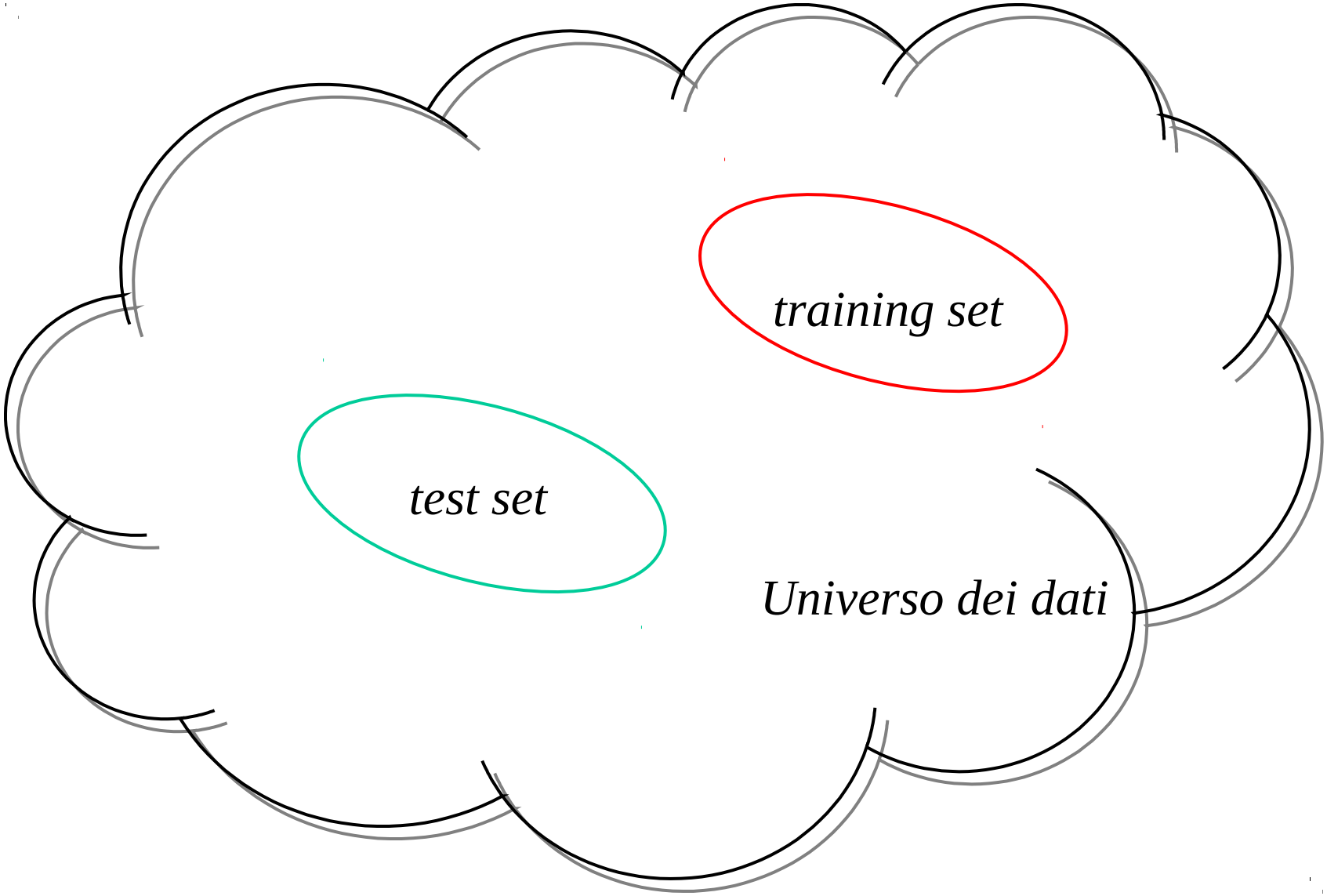
$$R_{emp}=\frac{1}{n}\sum_{i=1}^{n} Loss\left(g\left(x_i\right),t_i\right)$$

Generalizzazione

- L viene addestrata su un *training set* $D \subset U$.
- La learning machine L è utile se può fare delle previsioni sull' Universo non noto U dei dati:

  vogliamo cioè che *generalizzi* correttamente su dati che non conosce.
- A questo fine L deve prevedere correttamente non tanto i dati D su cui è stata addestrata, ma i dati $(\mathbf{x},t) \in U$, $(\mathbf{x},t) \notin D$ che non conosce.
- Siccome di solito non si conosce a priori U o equivalentemente la distribuzione di probabilità congiunta $P_U(\mathbf{x},t)$, le capacità di generalizzazione di L vengono valutate rispetto ad un test set T separato da D, cioè tale che $T \subset U$ e $T \cap D = \emptyset$.

Universo dei dati e campioni



Apprendimento supervisionato

- *Apprendimento da dati "etichettati"*: ciascun campione viene etichettato (ad es: normale o malato)
- *Supervisionato* in quanto un "supervisore" assegna le etichette ai campioni da apprendere: cioè la learning machine è addestrata tramite un insieme di dati etichettati (**x**,*t*)
- La learning machine impara ad associare un determinato campione **x** ad una classe *t*
- L' obiettivo della learning machine consiste nell' *assegnare un' etichetta corretta a campioni la cui classe di appartenenza non è nota a priori* (ad es: deve essere in grado di predire sulla base dei dati di espressione genica se un paziente sia sano o malato)

Obiettivo dell’ apprendimento supervisionato

- Consiste nel predire correttamente la classe si appartenenza di campioni non noti (*generalizzazione*).

La generalizzazione dipende da:

- Accuratezza del classificatore sul training set
- Complessità della funzione computata dal classificatore
- Dimensione training set

In caso i dati siano caratterizzati da ridotta cardinalità e/o elevata dimensionalità, può sorgere il problema di *overfitting* (sovraadattamento)

Caratteristiche dei dati di espressione genica

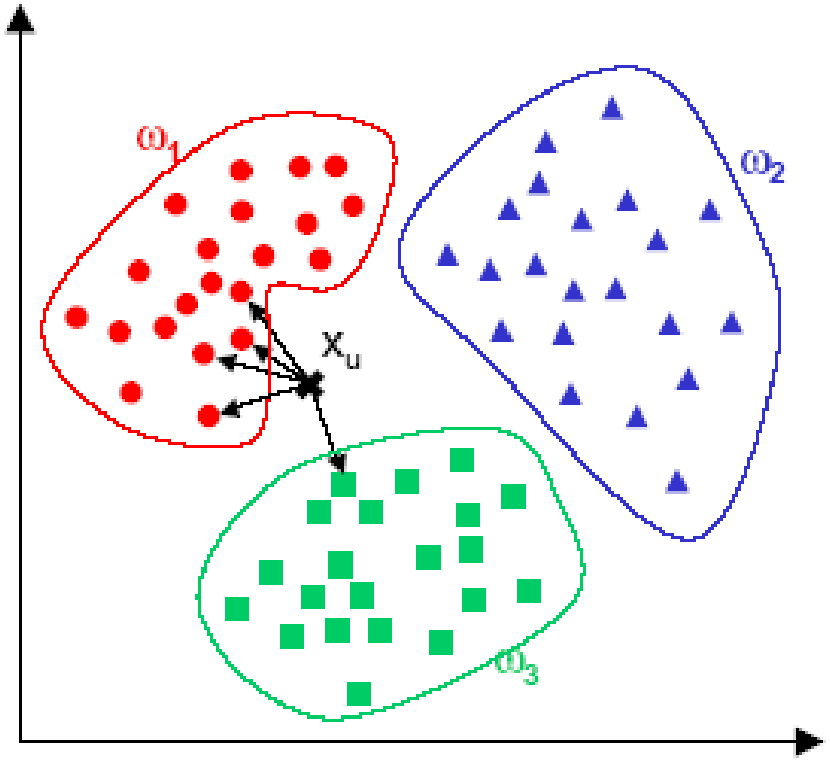
- Campioni di ridotta cardinalità
- Elevata dimensionalità dei dati
- Rumore
- Dati mancanti

Un problema complesso di apprendimento automatico

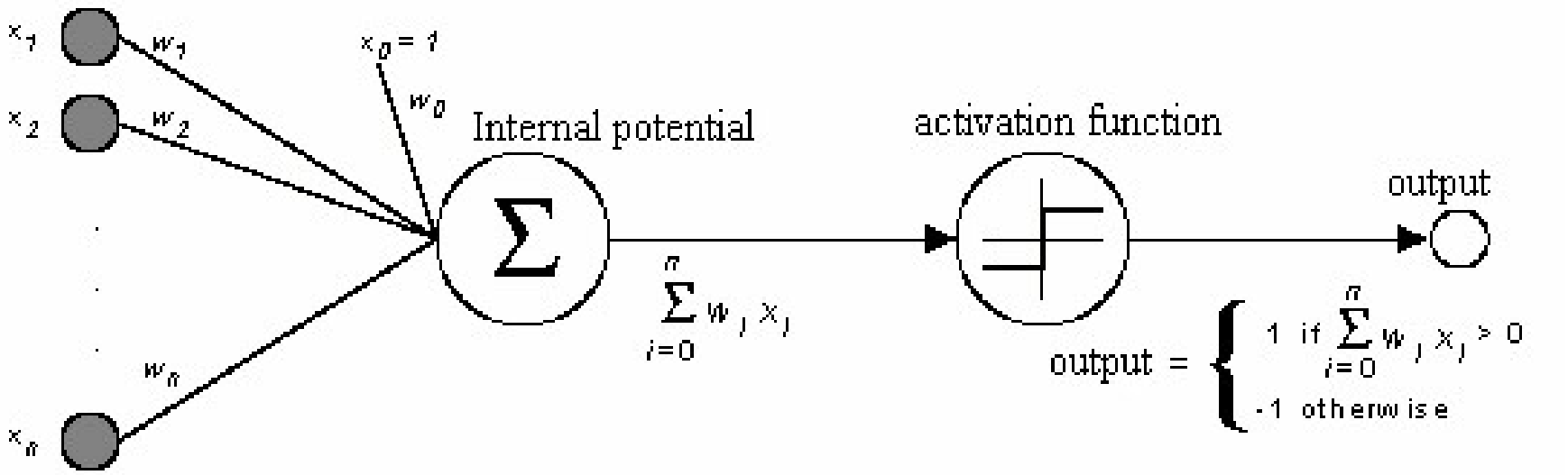
Metodi supervisionati per l’ analisi di dati di espressione genica

- Discriminanti lineari e quadratici (Dudoit et al., 2002)
- K-Nearest Neighbours (Pomeroy et al., 2002)
- Reti neurali (Khan et al. 2001)
- Support Vector Machine (Brown et al. 2000, Furey et al. 2000)
- Alberi di decisione (Dudoit et al. 2000)
- Metodi di ensemble (Dudoit et al., 2002; Diettling and Buhlmann, 2003)

K-nearest-neighbour



Percettrone (modello lineare di neurone)

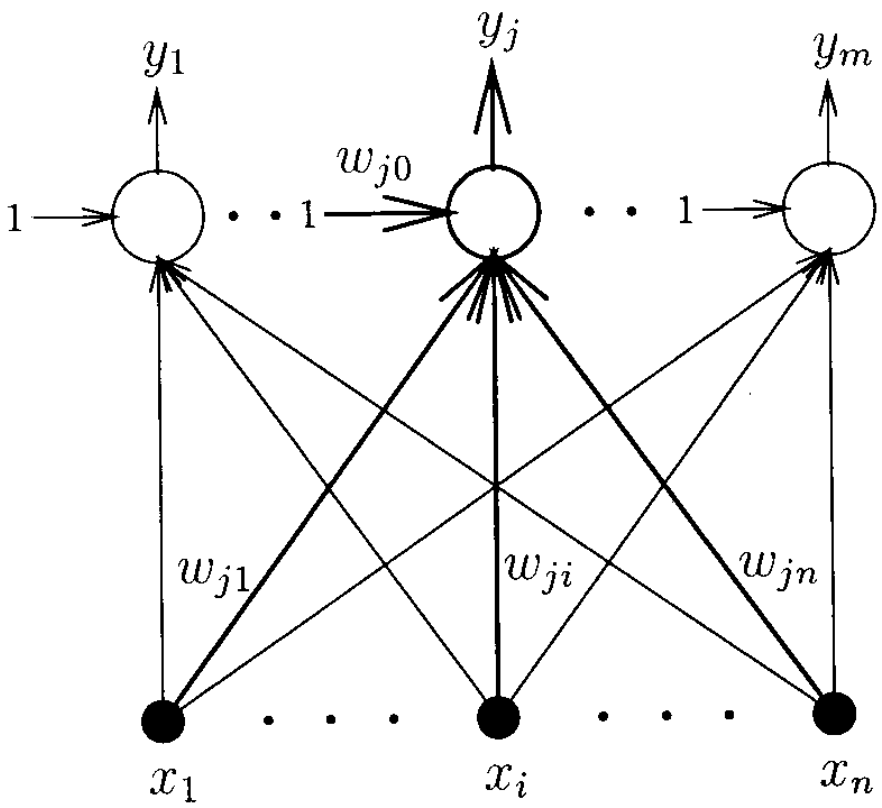


- x_1, ..., x_n - input
- w_1, ..., w_n – pesi sinaptici
- w_0 – fattore costante (bias)

$\forall$ $\sigma(\xi)$ – funzione di attivazione: $\sigma(\xi) = \mathrm{sgn}(\xi)$

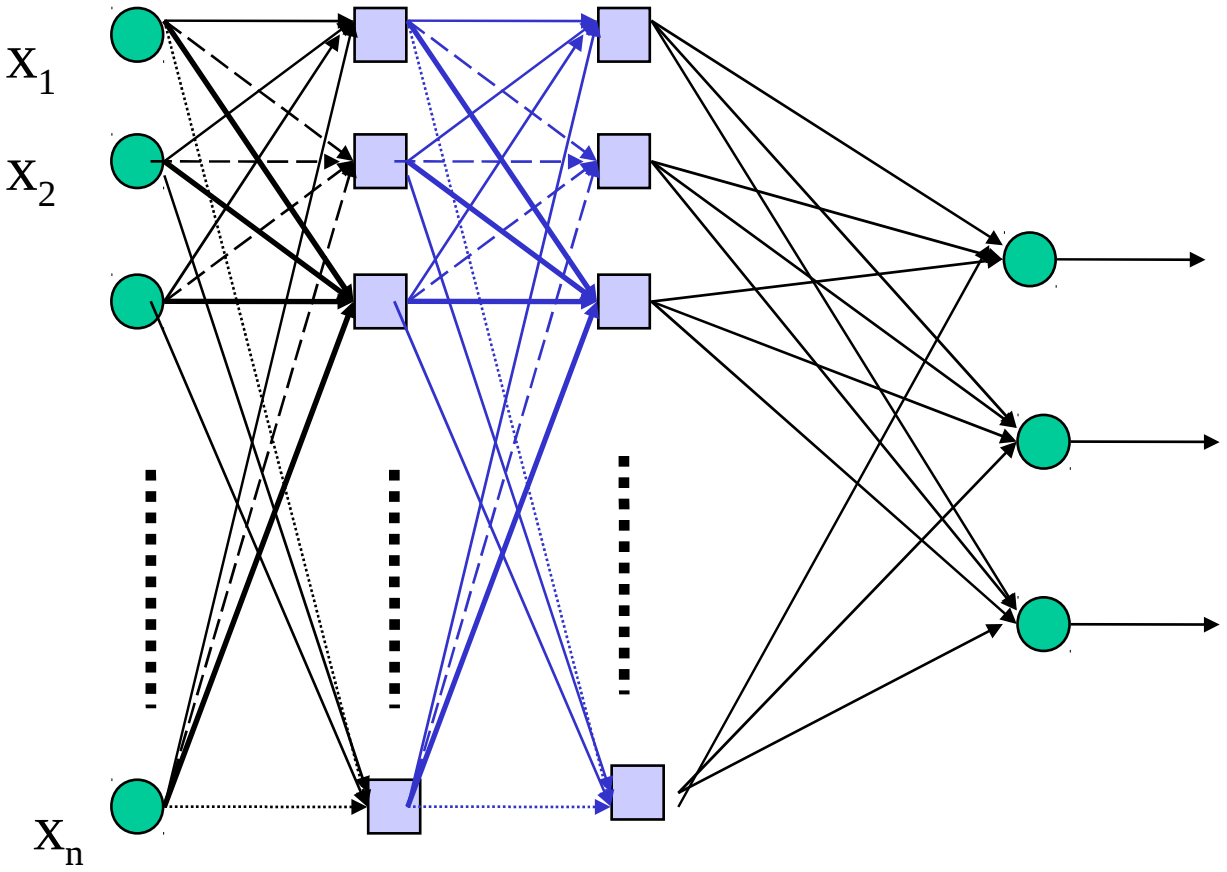
- output: $y = \sigma(\xi)$

Percettrone a singolo strato per classificazione a più classi



- Un neurone y_j per ogni classe j, $1 \leq j \leq m$ completamente connesso all' ingresso x.
- La classe di uscita viene computata tramite tecnica *WTA* (Winner Takes All)

Struttura di un percettrone multistrato (MLP)



Strato di ingresso

Strati nascosti

Strato di uscita

Addestramento dei MLP

Un algoritmo di discesa a gradiente non è direttamente applicabile:

- Ogni strato ha i suoi pesi che devono essere aggiornati
- Solo l' errore rispetto all' uscita è noto

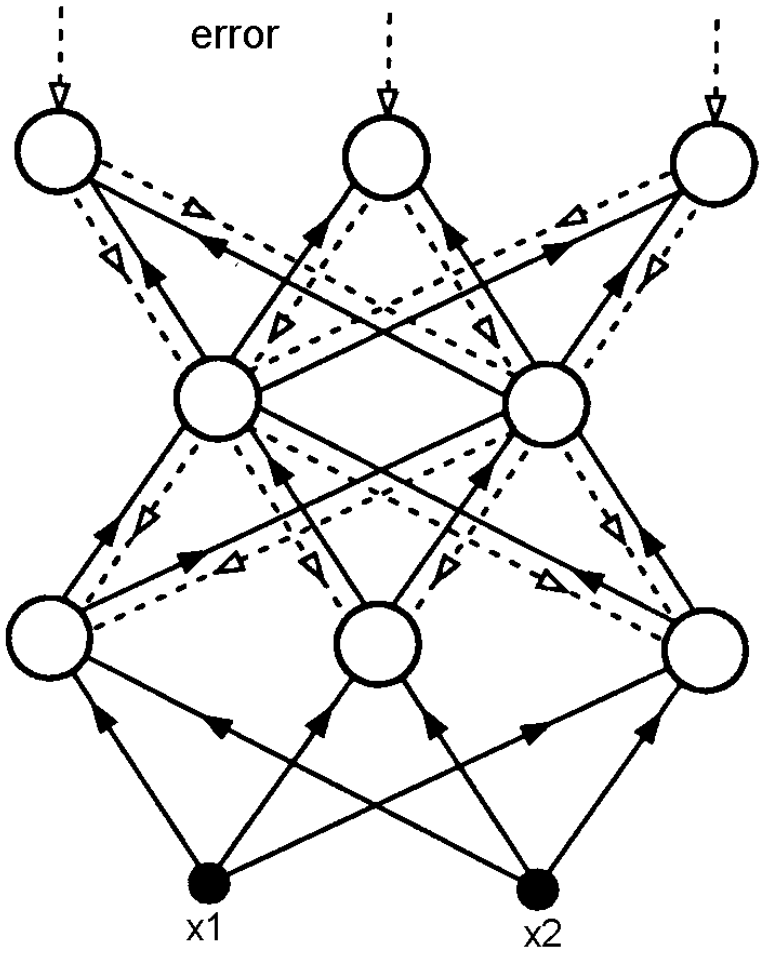
Algortimo di backpropagation (retropropagazione)
(*Rumelhart* et al. 1986)

Visualizzazione dell' algoritmo di backpropagation in una rete neurale a 3 strati

Output

Strati interme di

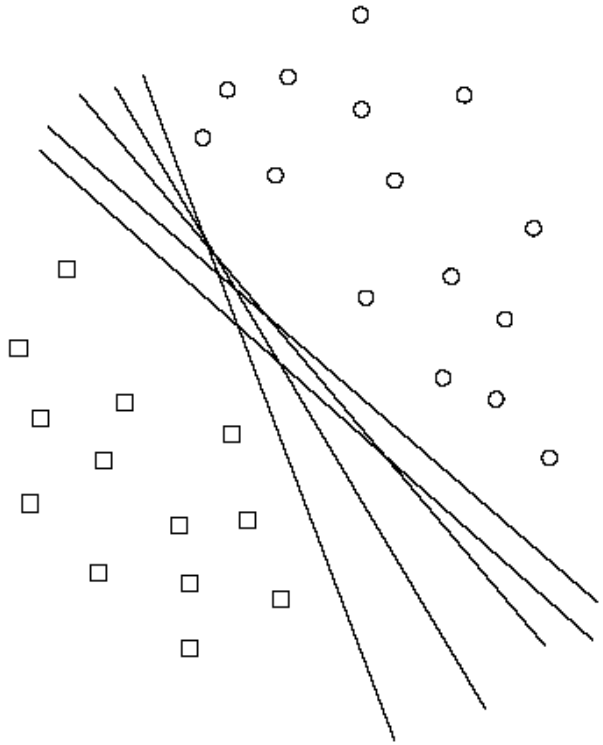
Input



computazione

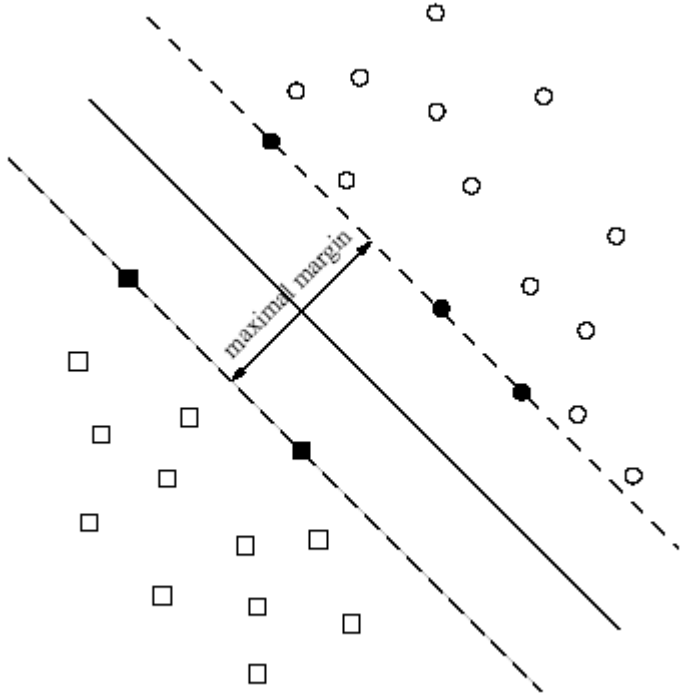
retropropagazione dell' errore

Support Vector Machine (SVM)



Come classifica una rete neurale (non regolarizzata)

Soluzioni molteplici (minimi locali multipli)

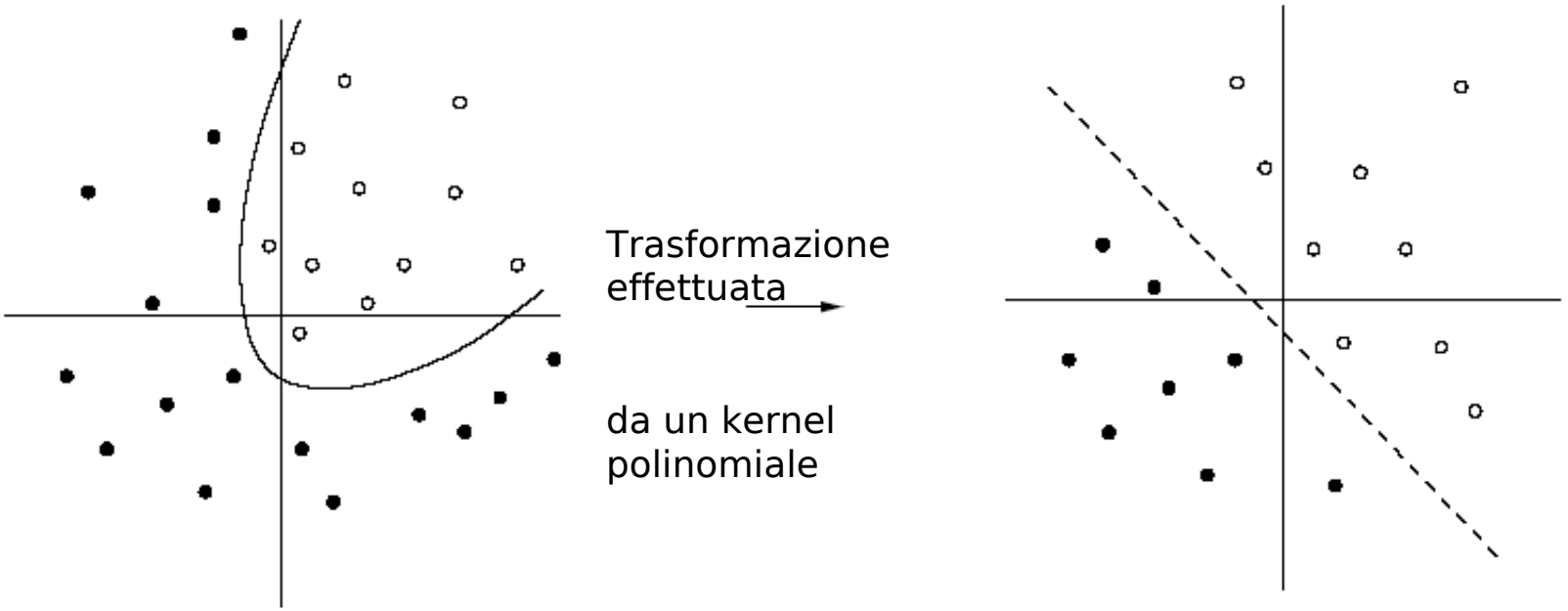


Come classifica una SVM lineare

Soluzione unica (minimo globale)

SVM per problemi non linearmente separabili

- Tramite funzioni kernel (es: funzioni gaussiane e polinomiali) i campioni da classificare sono proiettati in uno spazio iperdimensionale.
- In tale spazio è più probabile che un classificatore lineare riesca a classificare correttamente i campioni (*Teorema di Cover*)



Spazio di ingresso originale: i dati non sono linearmente separabili

La SVM calcola l' iperpiano di separazione ottimale nello spazio trasformato

Package R per la classificazione supervisionata di dati di espressione genica

- Package in R:
  - Nnet, class
  - e1071 (Sviluppata da *Wolski* e *Meyer*, Technische Universitat Wien)
  - Caret, CMA
  - ...
- Diverse librerie in Matlab, C, java, C++, ...