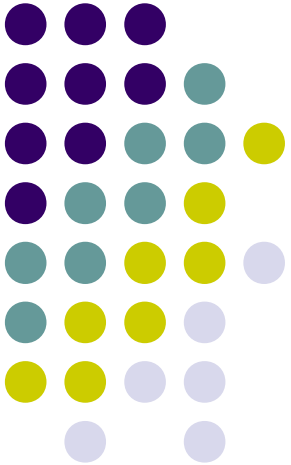
Modelli di classificatori

Apprendimento statistico

Classificatori a supporto vettoriale

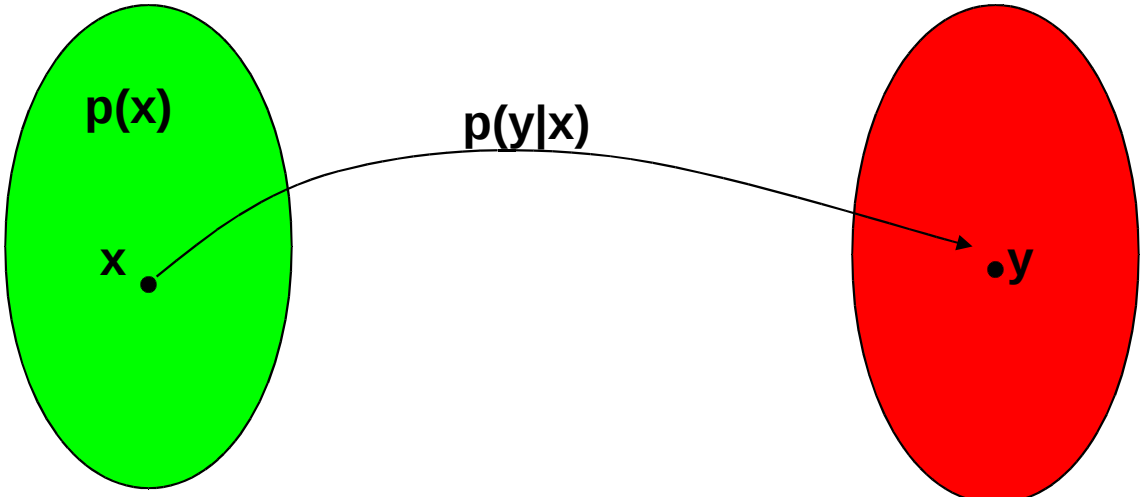
AdaBoost

Applicazioni



Il problema dell'apprendimento

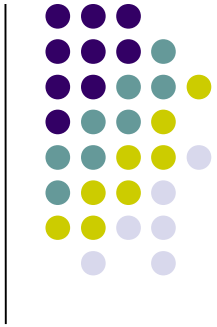
- Inquadriamo da un punto di vista statistico il problema dell'apprendimento di un classificatore
- Un training set $S=\{(x_1,y_1),\ldots,(x_n,y_n)\}$ può essere visto come un insieme di campioni estratti i.i.d. da XxY, dove X e Y sono due insiemi di v.a.



p(x,y)=p(y|x) p(x)

Fissata, ma ignota

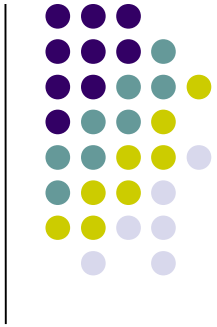
Il learning come approssimazione



Lo scopo principale del learning è quindi quello di utilizzare S per costruire una funzione f_S che, applicata ad un valore di x non visto precedentemente, predice il valore di y corrispondente:

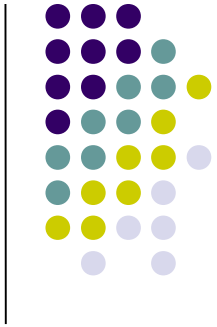
$$y_{pred}=f_S(x_{new})$$

Il learning come approssimazione



- Per valutare la qualità della funzione trovata, si utilizza una *funzione di costo* (*loss function*) $L(f_S(x),y^*)$.
- Esempi:
  - $[f_S(x)-y^*]^2$
  - $|f_S(x)-y^*|$

Valutazione del costo



Scegliendo una certa funzione f, il rischio atteso è dato da:

$$R(f)=\int L(f(x),y)\,dp(x,y)$$

L'obiettivo è quello di minimizzare R(f), ma p(x,y) non è conosciuta.

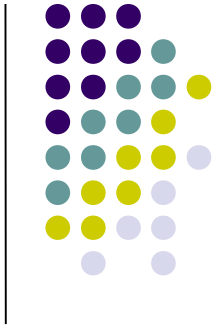
L'unica possibilità è fare una valutazione sui dati a disposizione (training set):

$$R_S(f)=\sum_i L(f(x_i),y_i)$$

detto *rischio empirico.*

Che differenza c'è tra rischio empirico e rischio atteso ?

Cause di errore



- Lo scopo finale dell'algoritmo di apprendimento è quello di individuare la funzione f_0 che minimizza il rischio atteso.
- La scelta, però, sarà fatta all'interno dell'insieme Λ di funzioni f_n che l'algoritmo di apprendimento è capace di implementare.
- Le cause di errore sono quindi due:

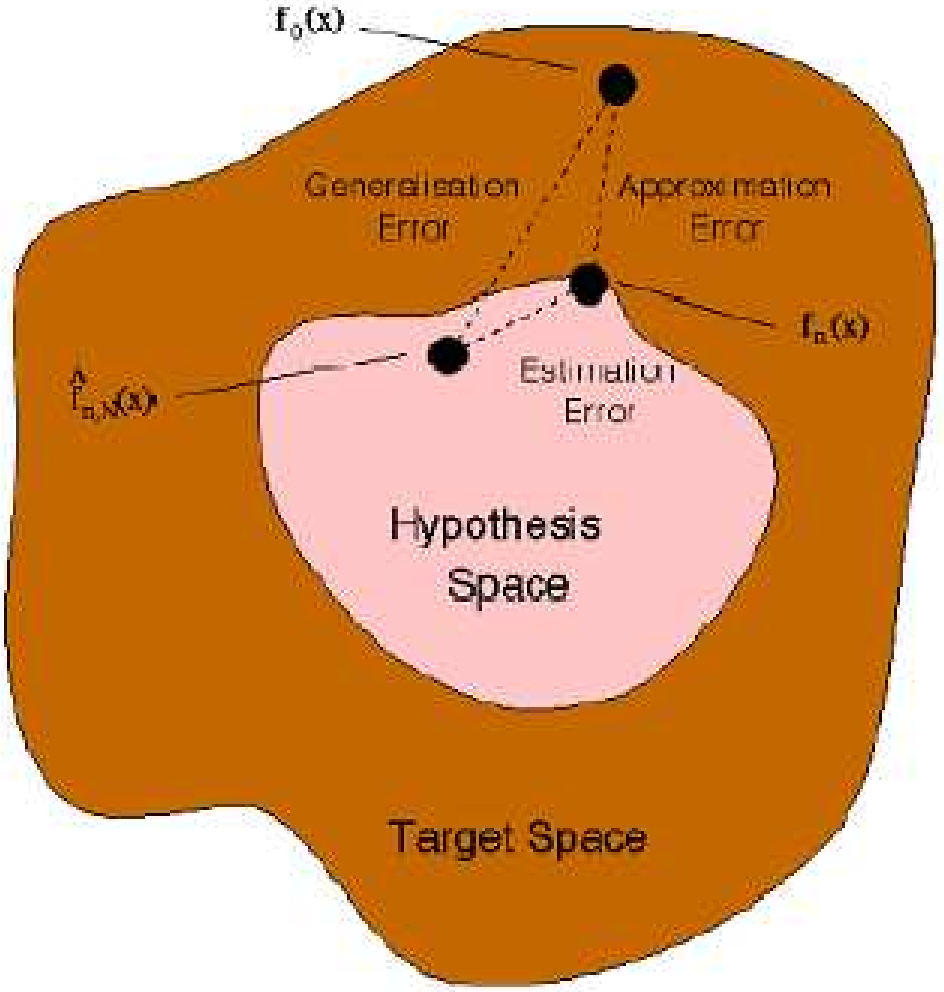
  Errore di Approssimazione

  è la conseguenza della limitatezza dell'insieme Λ di funzioni implementabili dall'algoritmo di apprendimento.

  Errore di Stima

  è l'errore dovuto alla procedura di apprendimento che, scegliendo la funzione $f_{n,N}$ sulla base del rischio empirico, non individua la funzione ottimale all'interno di Λ.
- Insieme, questi due errori formano l'**errore di generalizzazione**.

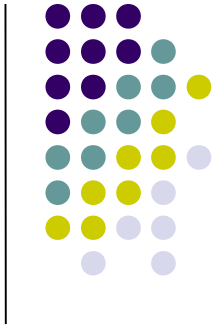
Quanti errori !!!



Errore e generalizzazione

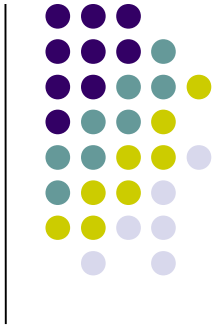
- Le sole informazioni che abbiamo in fase di learning sono i campioni del training set su cui realizzare l'addestramento.
- Come abbiamo visto, la minimizzazione del rischio empirico non ci garantisce che minimizziamo anche il rischio atteso: **problema della generalizzazione**

Errore e generalizzazione



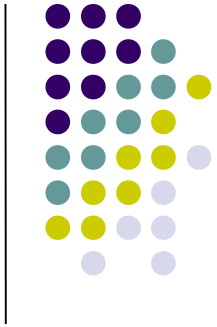
- Da un lato l'apprendimento mira ad ottenere un classificatore ad elevata capacità, in grado cioè di minimizzare il rischio empirico.
- Dall'altro, si vorrebbe un classificatore che garantisca buone dosi di generalizzazione.
- Queste, però, sono esigenze contrastanti.

La dimensione VC



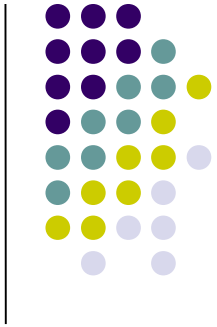
- La dimensione di Vapnik-Chervonenkis (dimensione VC) è una misura della capacità (o della complessità) di una classe di funzioni f(α).
- Coincide con il più grande numero di campioni su cui la famiglia di funzioni f(α) può operare una corretta discriminazione.

La dimensione VC



- La dimensione VC fornisce quindi una valutazione quantitativa della capacità di una classe di funzioni f(α).
- Tramite la dimensione VC è quindi possibile fornire un limite superiore del rischio atteso in funzione del rischio empirico e del numero di campioni disponibili

Una limitazione del rischio atteso



E' stata dimostrato (Vapnik) che sussiste la seguente disuguaglianza :

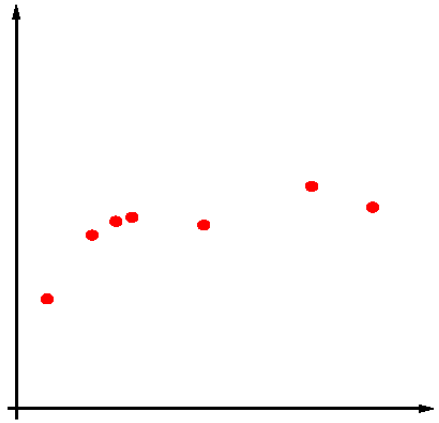
$$R(f) \leq R_S(f) + \sqrt{\frac{h\left(\log\left(2|S|/h\right)+1\right)-\log\left(\eta/4\right)}{|S|}}$$

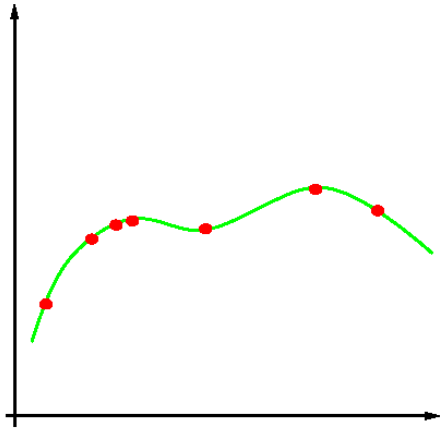
con probabilità 1-η.

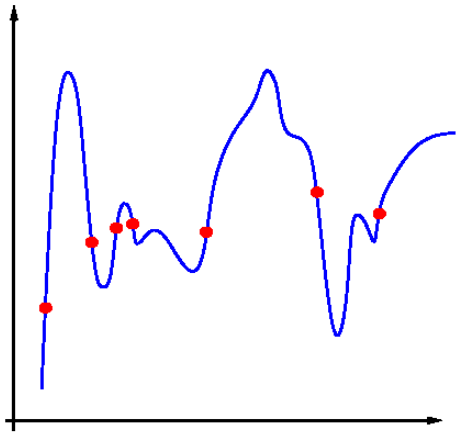
h è la *dimensione di Vapnik-Chervonenkis* (*VC dimension*) della funzione f mentre |S| è la cardinalità dell'insieme di campioni.

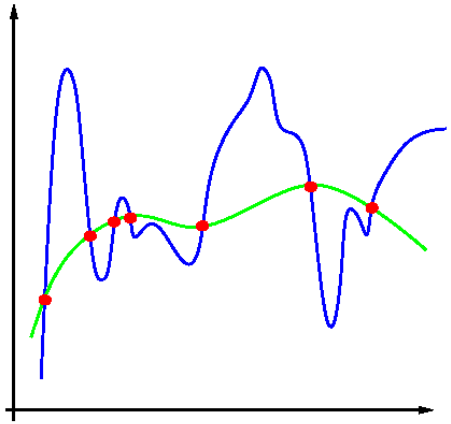
All'aumentare di |S|/h il secondo addendo decresce.

Effetti della capacità









Minimizzazione del Rischio strutturale

Per controllare il rischio atteso occorre allora trovare una f che minimizzi il termine

$$R_S(f) + \sqrt{\frac{h\left(\log\left(2|S|/\mathrm{h}\right)+1\right)-\log\left(\eta/4\right)}{|S|}}$$

dove il primo addendo è il rischio empirico e il secondo addendo è detto *termine di confidenza*.

Si noti che l'intera espressione è indipendente dalla p(x,y) ignota.

Minimizzazione del Rischio Strutturale

- Per un dato insieme di campioni di training è quindi possibile controllare il rischio atteso agendo contemporaneamente sul rischio empirico $R_s(f)$ e sulla dimensione VC h.
- Si noti che il termine di confidenza dipende dalla classe di funzioni scelta, mentre sia il rischio atteso che il rischio empirico dipendono dalla particolare forma della funzione scelta dall'algoritmo di apprendimento (p.es. i valori dei pesi di un MLP).

Classificatori a supporto vettoriale

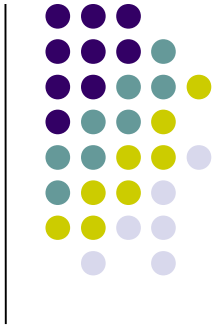
- Consideriamo un problema di classificazione a due classi per cui abbiamo un training set $S=\{x_i,y_i\}$. Le classi sono etichettate con ± 1.
- Supponiamo che i campioni siano linearmente separabili. Ciò significa che esiste un iperpiano $w \cdot x+b=0$ tale che:

$w \cdot x_i+b > 0$ se $y_i=+1$

$$\Longrightarrow \quad (w \cdot x_i+b)\, y_i > 0$$

$w \cdot x_i+b < 0$ se $y_i= -1$

Classificatori a supporto vettoriale



- Consideriamo il punto x_+ (x_-) che ha etichetta +1 (-1) e si trova a minima distanza dall'iperpiano; chiamiamo d_+ (d_-) tale distanza. Definiamo *margine* dell'iperpiano la somma $d_+ + d_-$.
- Se consideriamo uno scaling di w e b per cui, in corrispondenza di tali punti, si abbia $w \cdot x_+ + b = +1$ e $w \cdot x_- + b = -1$, allora $d_+ = d_- = 1/||w||$. Per cui il margine diventa $2/||w||$.

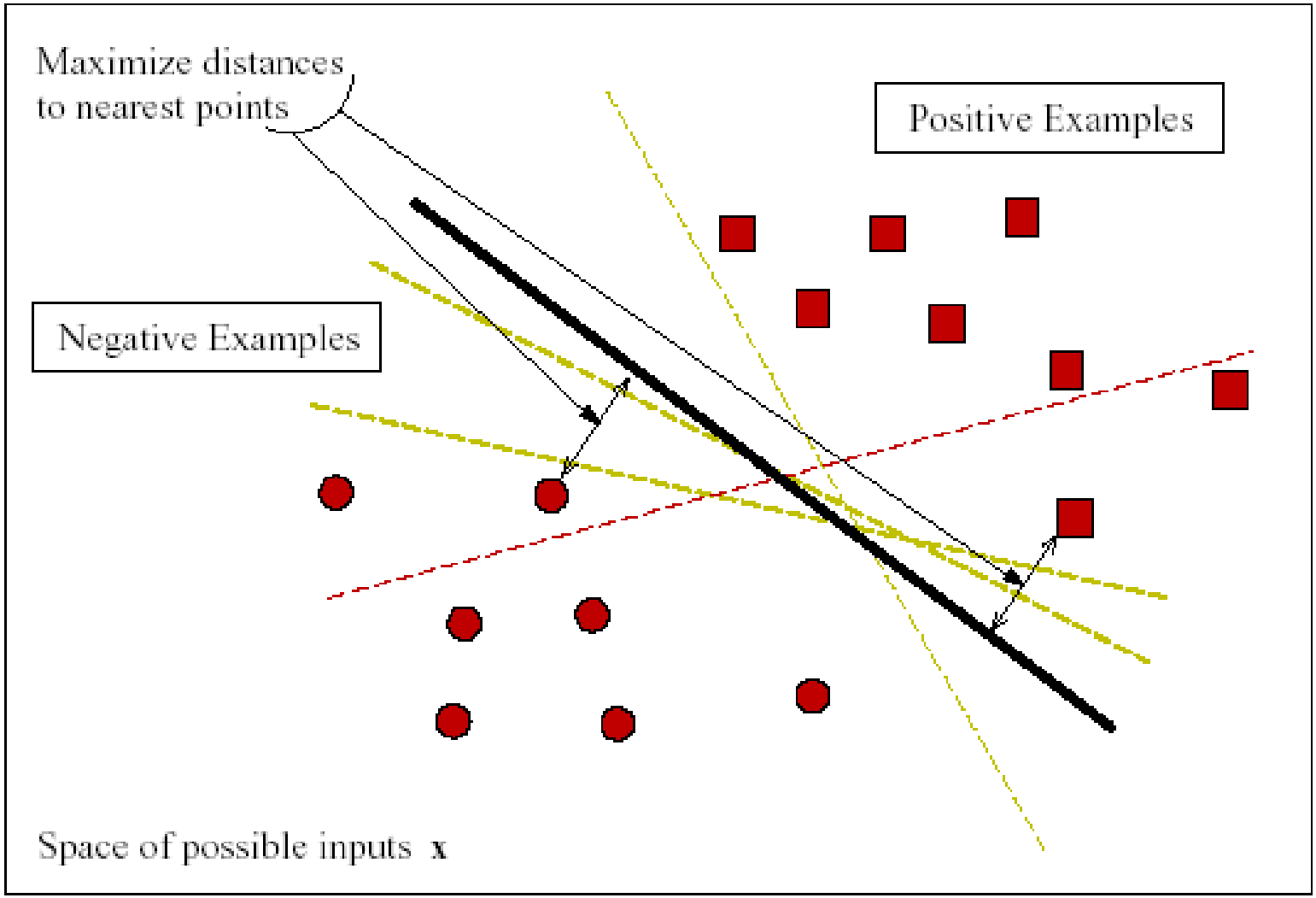
Classificatori a supporto vettoriale e dimensione VC

- Il classificatore così realizzato si definisce *classificatore a supporto vettoriale* o *Support Vector Machine* (*SVM*).
- Si è dimostrato che la capacità di generalizzazione di questo classificatore cresce al crescere del margine in quanto la capacità h soddisfa la seguente proprietà:

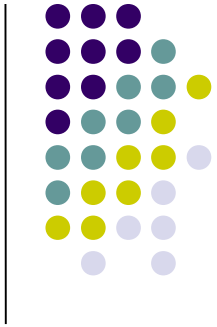
$$h \leq \min\left(ceil\left(\frac{R^2}{m^2} \right), D \right) + 1$$

  dove R è il raggio della più piccola sfera che contiene i campioni.
- Come si vede, la capacità h diminuisce al crescere del margine e quindi la migliore generalizzazione è data dall'iperpiano a margine massimo, detto *optimal separating hyperplane (OSH)*.

L’iperpiano a margine massimo

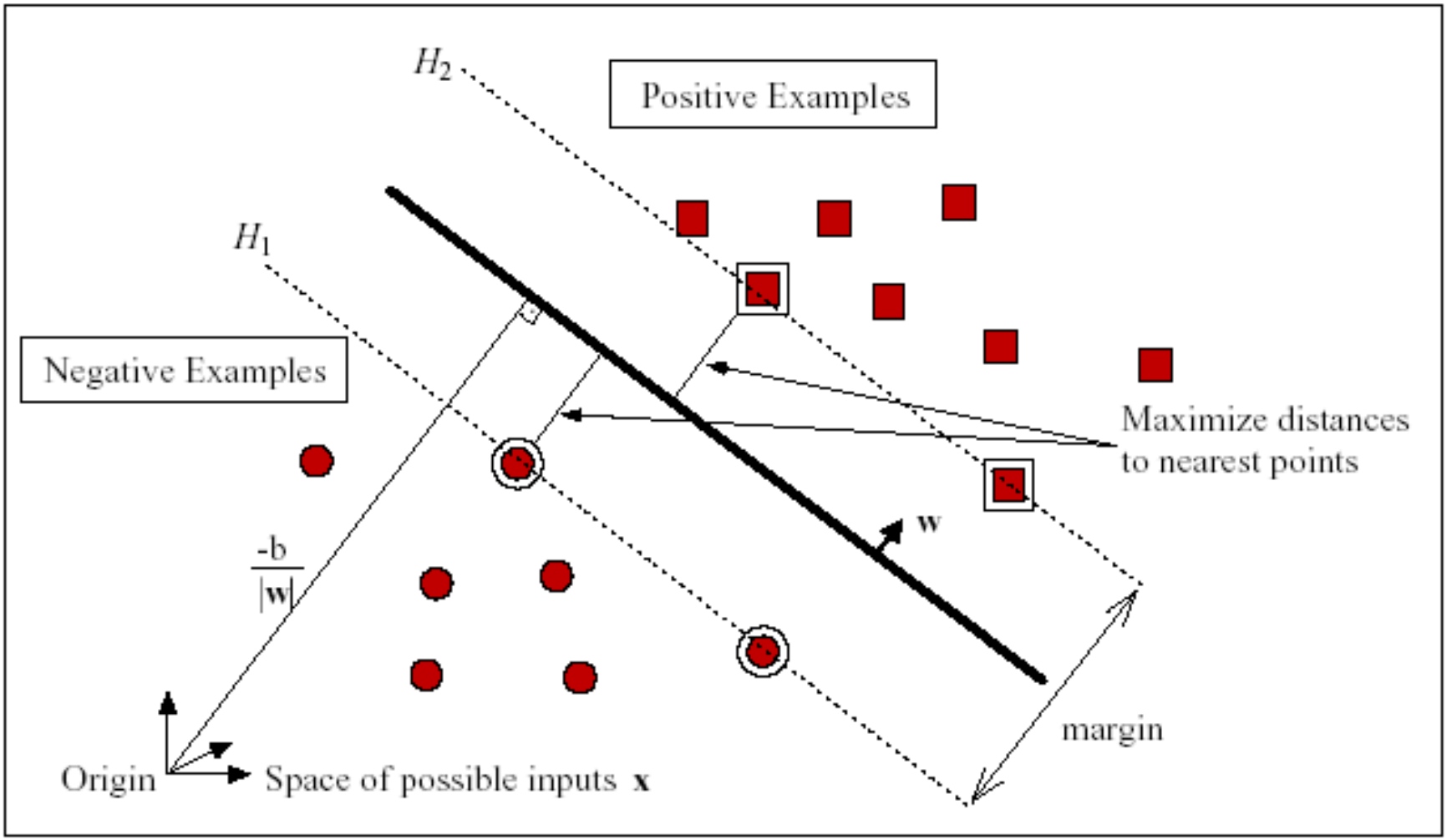


L'iperpiano a margine massimo

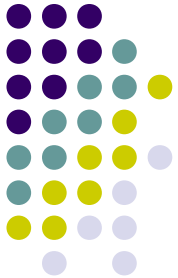


- Grazie alle condizioni imposte su w e b, l'OSH sarà tale che $(w \cdot x_i + b)\, y_i - 1 > 0$.
- I punti più vicini soddisferanno l'equazione $(w \cdot x_i + b)\, y_i - 1 = 0$ che individua due iperpiani H_1 e H_2 paralleli all'OSH, tra i quali non cadrà alcun punto di S.
- I punti che giacciono su H_1 e H_2 sono detti *vettori di supporto* (*support vectors*). Cambiando i vettori di supporto, cambia anche l'OSH.

L’iperpiano a margine massimo



Costruzione dell'iperpiano ottimo



Per costruire l'iperpiano a margine massimo bisogna risolvere il problema:

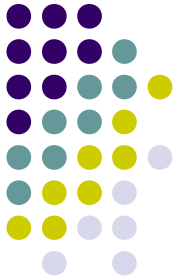
$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

soggetto a $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 \geq 0 \quad \forall\, i$

Ciò porta alla minimizzazione del Lagrangiano:

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i y_i \left(\mathbf{w}\cdot\mathbf{x}_i + b\right) + \sum_{i=1}^{\ell} \alpha_i, \qquad \alpha_i \geq 0.$$

Costruzione dell'iperpiano ottimo

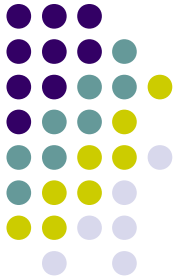


- Il problema risulta essere un problema convesso di programmazione quadratica per il quale la soluzione è:

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$$

- Nell'espressione che fornisce w solo alcuni moltiplicatori di Lagrange α_i saranno non nulli.
- Di conseguenza, solo i corrispondenti punti del training set entreranno come vettori di supporto nella definizione dell'iperpiano.

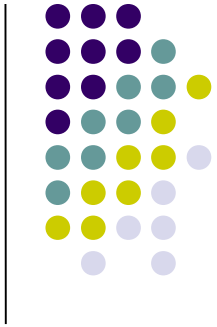
Costruzione dell'iperpiano ottimo



- Risolto il problema, la funzione discriminante sarà data da:

$$
\begin{aligned}
f(\mathbf{x}) &= \mathbf{w} \cdot \mathbf{x} + b \\
\Rightarrow f(\mathbf{x}) &= \left(\sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i \right) \cdot \mathbf{x} + b \\
\Rightarrow f(\mathbf{x}) &= \sum_{i=1}^{\ell} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b.
\end{aligned}
$$

Casi non linearmente separabili



- Per come è stato definito, il classificatore a supporto vettoriale non può gestire casi in cui le classi non siano linearmente separabili.
- Per risolvere questo problema ci sono due approcci:
  - rilassare i vincoli di corretta classificazione, tollerando un certo numero di errori
  - considerare altre superfici oltre l'iperpiano

Generalizzazione dell'iperpiano ottimo

Nella fase di ottimizzazione si rilassa il vincolo di esatta classificazione dei campioni del training set *(soft margins)*, introducendo delle variabili *slack* ξ_i. I vincoli così diventano:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +1 - \xi_i, \quad \text{for } y_i = +1$$

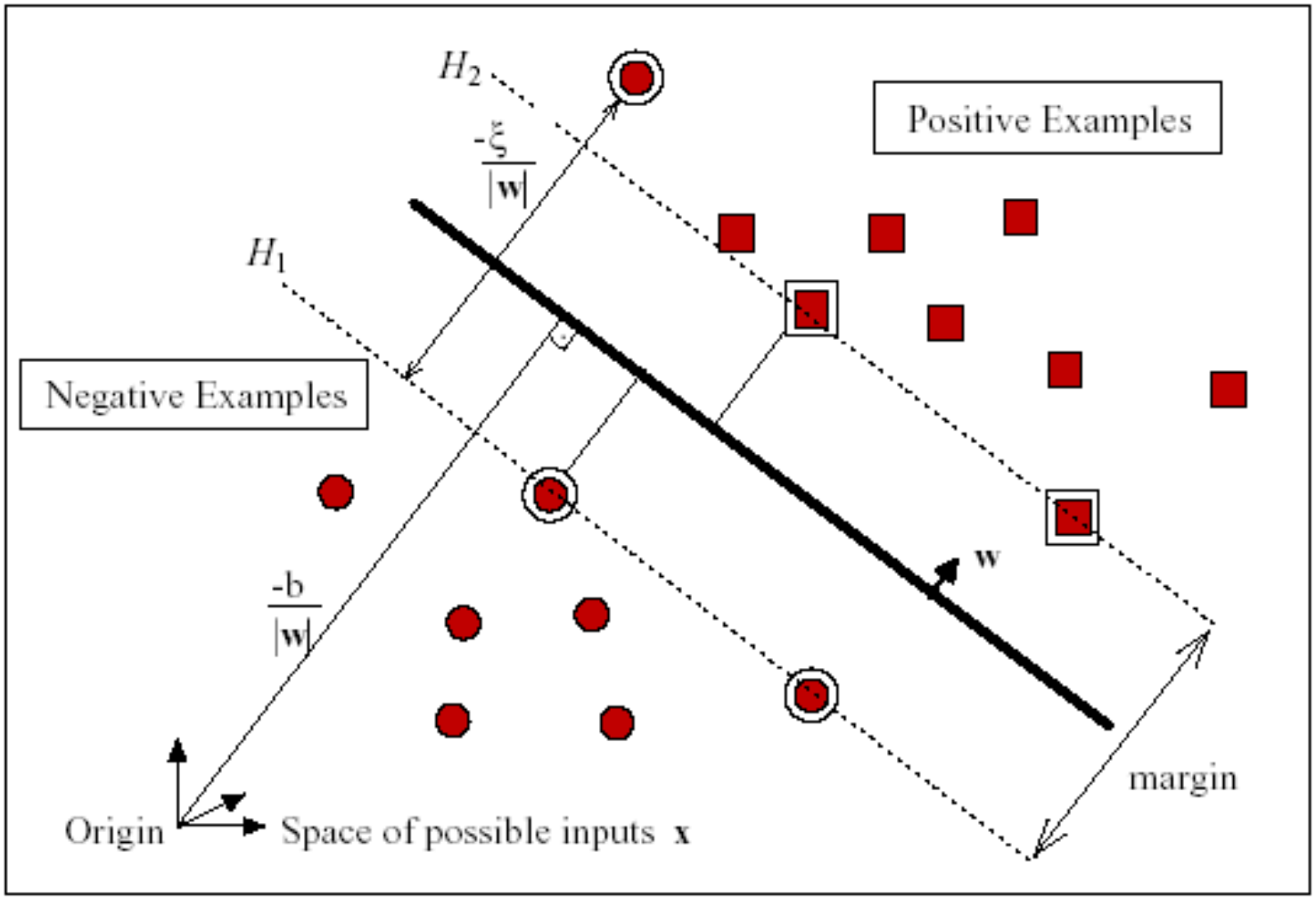
$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 + \xi_i, \quad \text{for } y_i = -1$$

$$\xi_i \geq 0, \qquad \forall\, i.$$

Generalizzazione dell'iperpiano ottimo

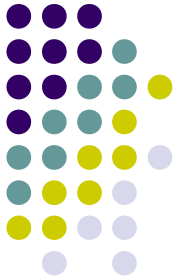
- A seconda del valore assunto dalla corrispondente variabile slack, i punti del training set saranno:
  - disposti al di là degli iperpiani H_1 e H_2 e correttamente classificati ($\xi_i=0$)
  - posti tra gli iperpiani H_1 e H_2 e correttamente classificati ($0<\xi_i<1$)
  - erroneamente classificati ($\xi_i>1$)

Generalizzazione dell'iperpiano ottimo



gli Studi
del L.M.

Generalizzazione dell'iperpiano ottimo

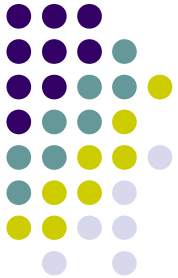


- Con l'introduzione delle variabili slack, il Lagrangiano da minimizzare diventa:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i\} - \sum_i \mu_i\xi_i$$

- Il parametro C permette di gestire il compromesso tra l'ampiezza del margine (C basso) e il numero di errori tollerati in fase di training (per C=∞ si torna al caso di iperpiano perfettamente separabile).

Generalizzazione dell'iperpiano ottimo



- La soluzione ottenuta per l'iperpiano è la stessa del caso originale:

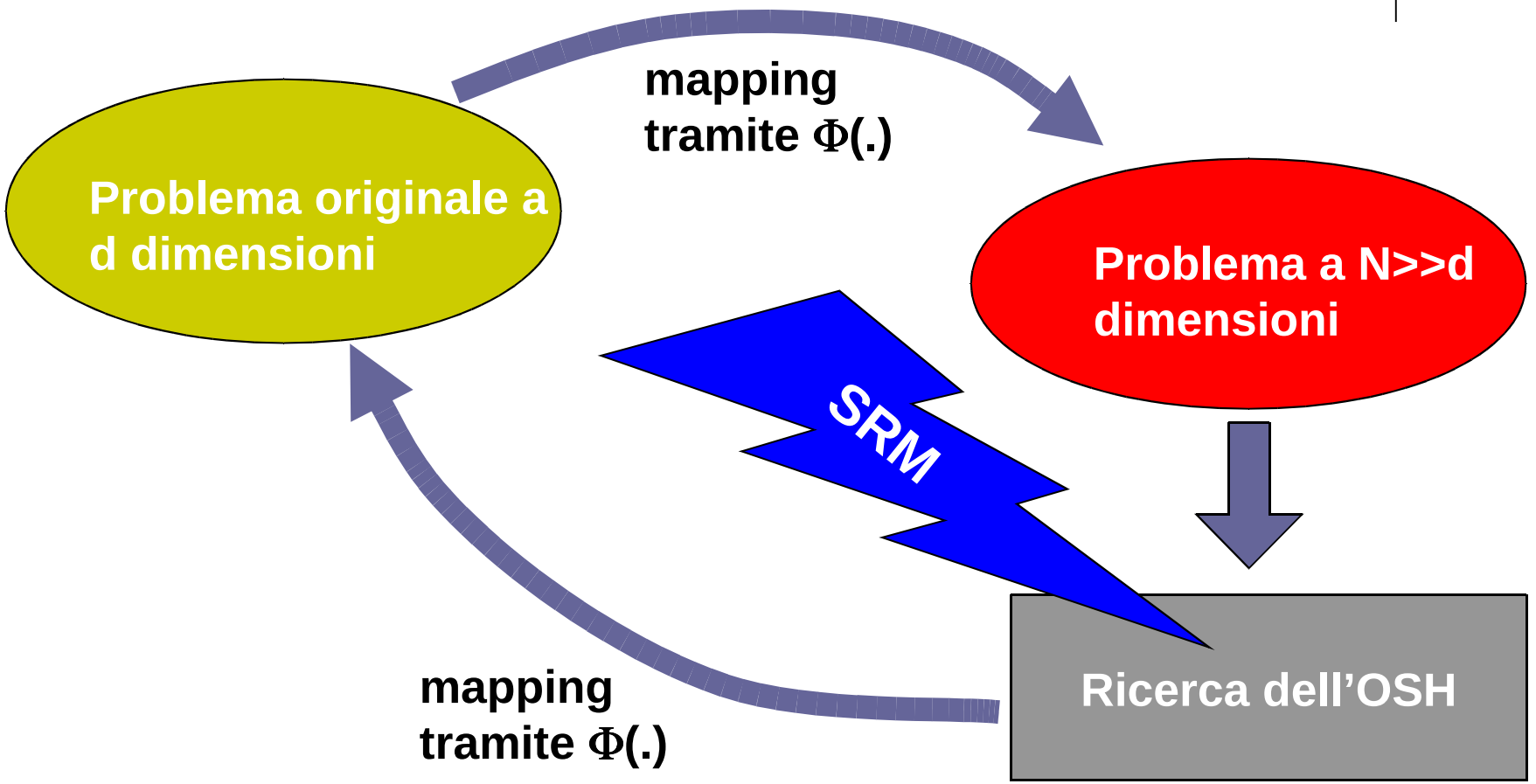
$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

- L'unica differenza è che ora gli α_i rispettano la condizione $0 \leq \alpha_i \leq C$.

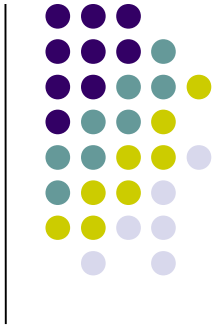
Casi non linearmente separabili

- Nel caso in cui non ci sia soluzione (insiemi non linearmente separabili), si introduce un mapping $\Phi(x)$ ad uno spazio di dimensione molto più grande in cui gli insiemi corrispondenti siano linearmente separabili.
- Quindi, invece di aumentare la complessità del classificatore (che resta un iperpiano) si aumenta la dimensione dello spazio delle features.
- In dipendenza della dimensione dello spazio in cui è formulato il problema originale, il mapping può portare anche a dimensioni molto elevate (~10^6) dello spazio trasformato.

Casi non linearmente separabili



Casi non linearmente separabili



- In questo caso, la funzione da minimizzare diventa:

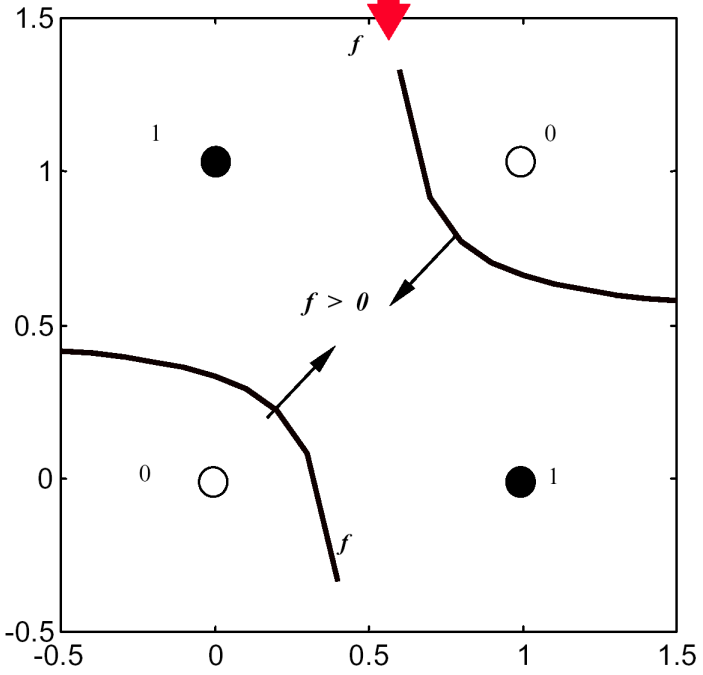
$$L_P = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{\ell} \alpha_i \left(y_i \left(w \cdot \Phi(x_i) + b \right) - 1 \right)$$

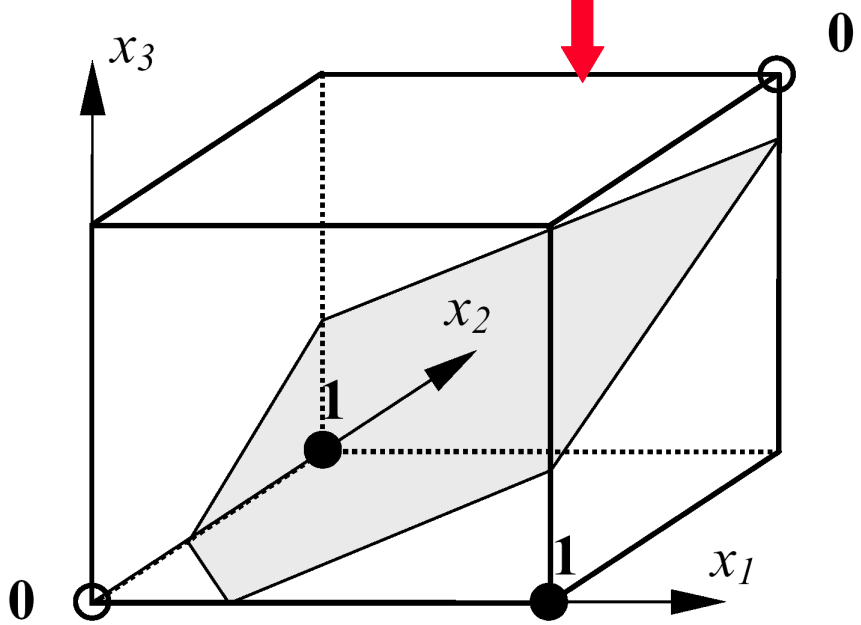
che ha come soluzione: $w = \sum_i \alpha_i y_i \Phi(x_i)$

per cui la funzione discriminante è:

$$f(x) = \sum_i \alpha_i y_i \Phi(x_i) \Phi(x) + b$$

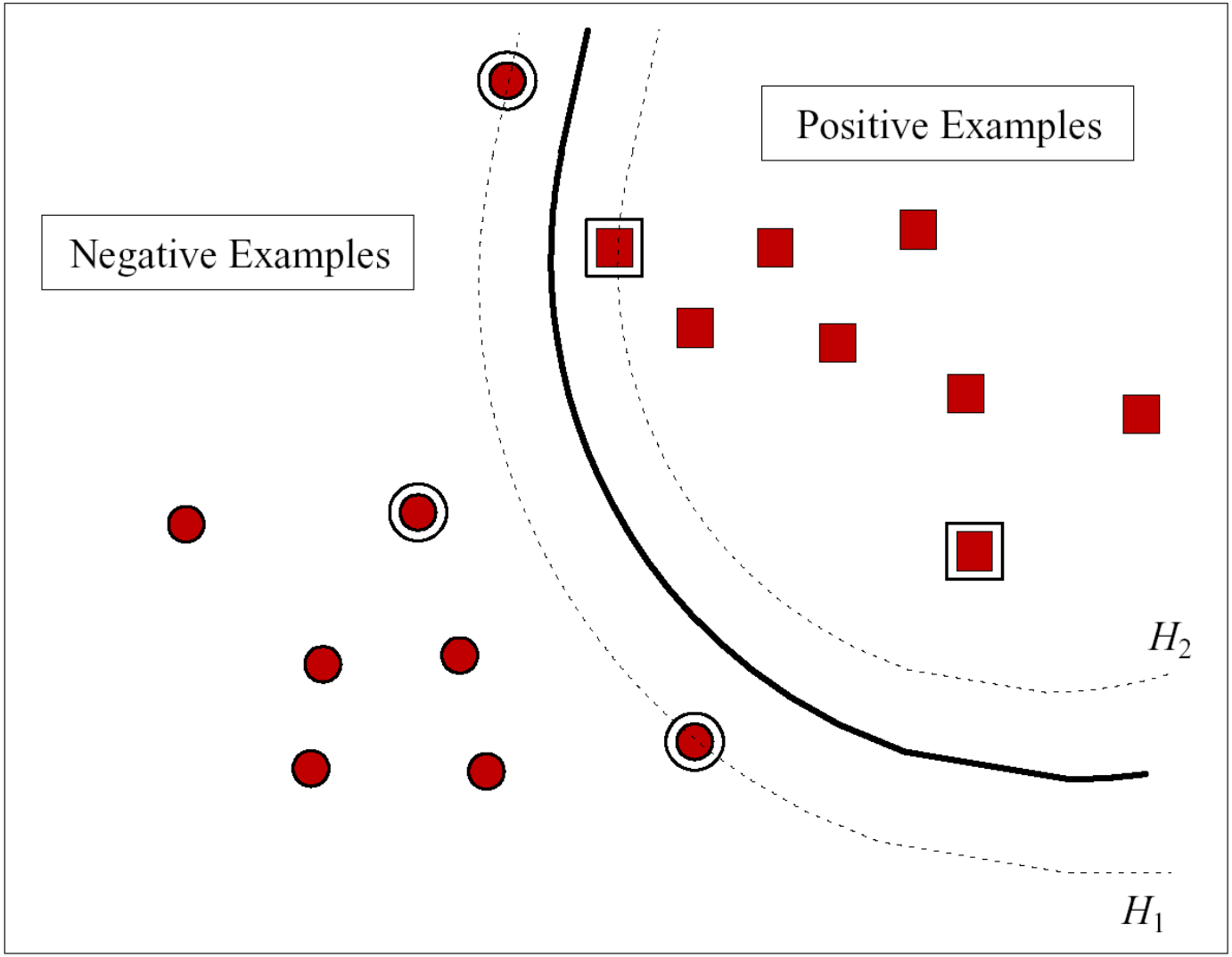
$f(\mathbf{x}) = x_1 + x_2 - 2\,x_1x_2 - 1/3$, $x_3 = x_1x_2$, $f(\mathbf{x}) = x_1 + x_2 - 2x_3 - 1/3$



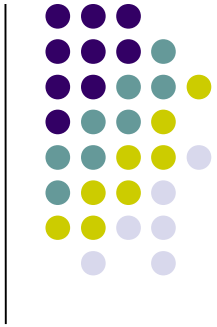


Problema XOR

Casi non linearmente separabili



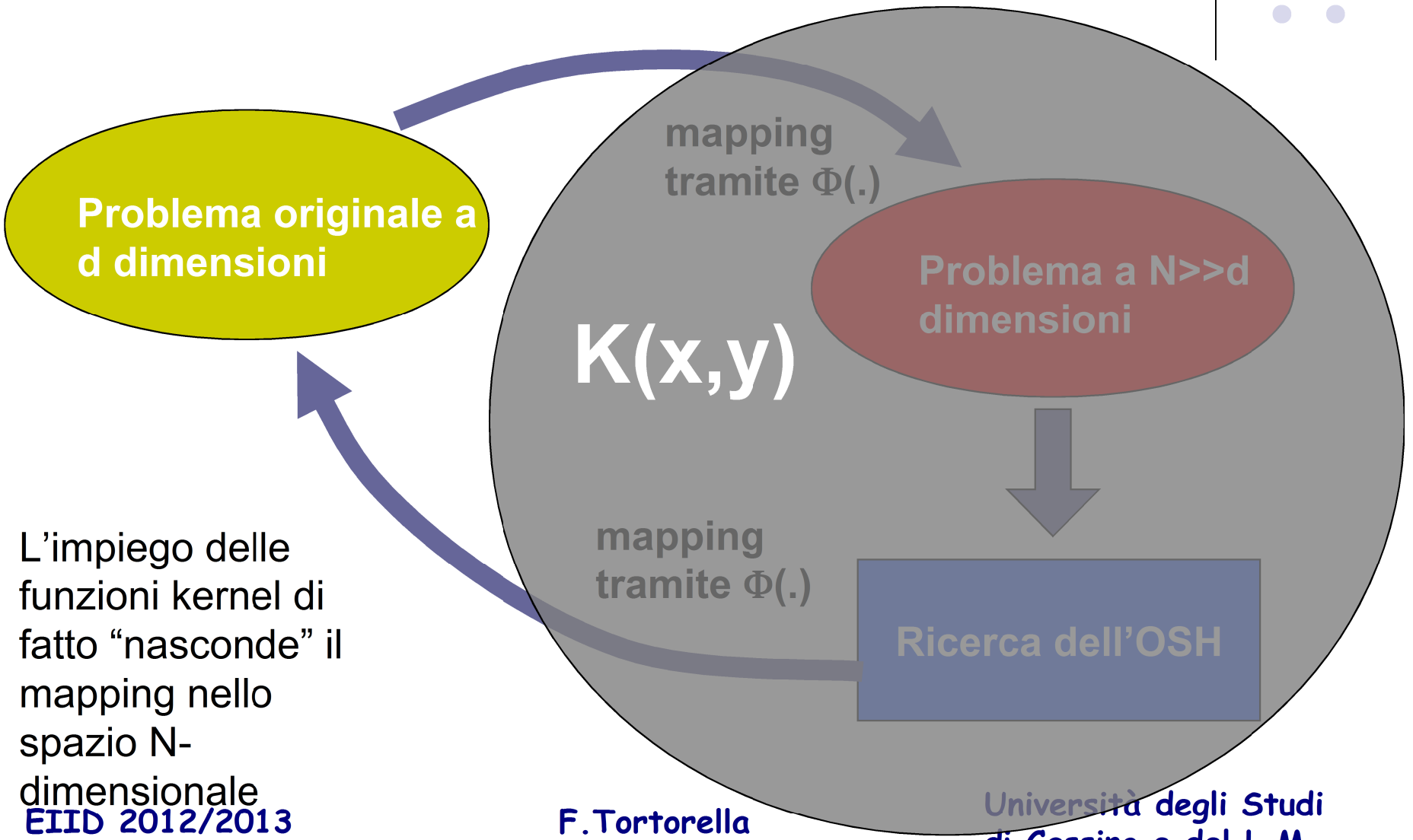
Le funzioni kernel



- Data la dimensione dello spazio trasformato le funzioni di mapping $\Phi(.)$ potrebbero essere molto complesse da valutare.
- In effetti, tutto quello che serve ai fini dell'addestramento e della classificazione è la forma funzionale del prodotto scalare $\Phi(x)\cdot\Phi(y)$.
- Fortunatamente, per il teorema di Mercer, è assicurata l'esistenza di funzioni kernel K(x,y) tali che $K(x,y)=\Phi(x)\cdot\Phi(y)$.
- Di conseguenza, anche la funzione discriminante si modifica in:

$$f(x)=\sum_i \alpha_i y_i K(x_i,x)+b$$

Le funzioni kernel



L'impiego delle funzioni kernel di fatto "nasconde" il mapping nello spazio N-dimensionale

Esempi di kernel

- Kernel polinomiali

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

- Kernel gaussiani (RBF)

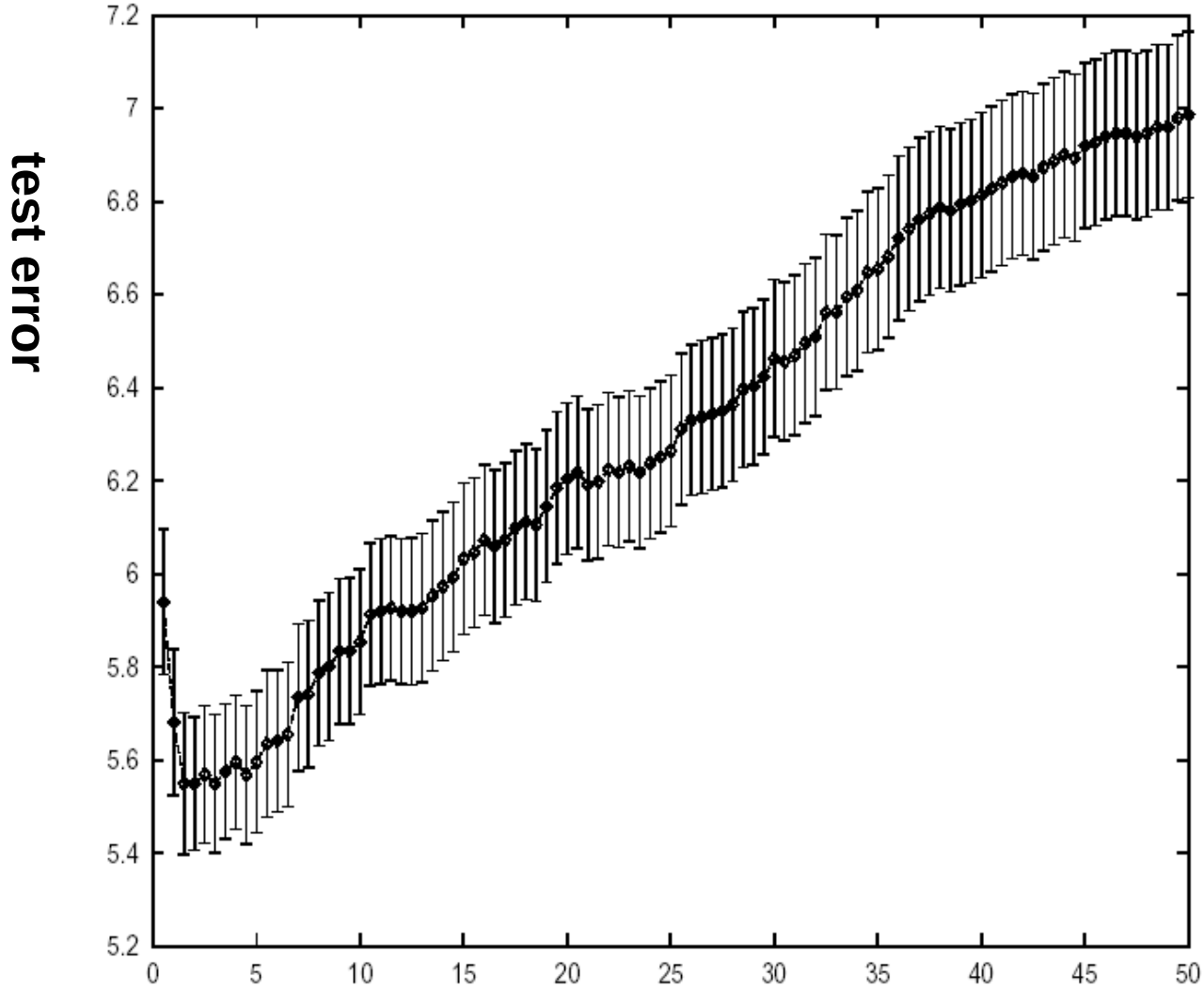
$$K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$$

- Kernel MLP

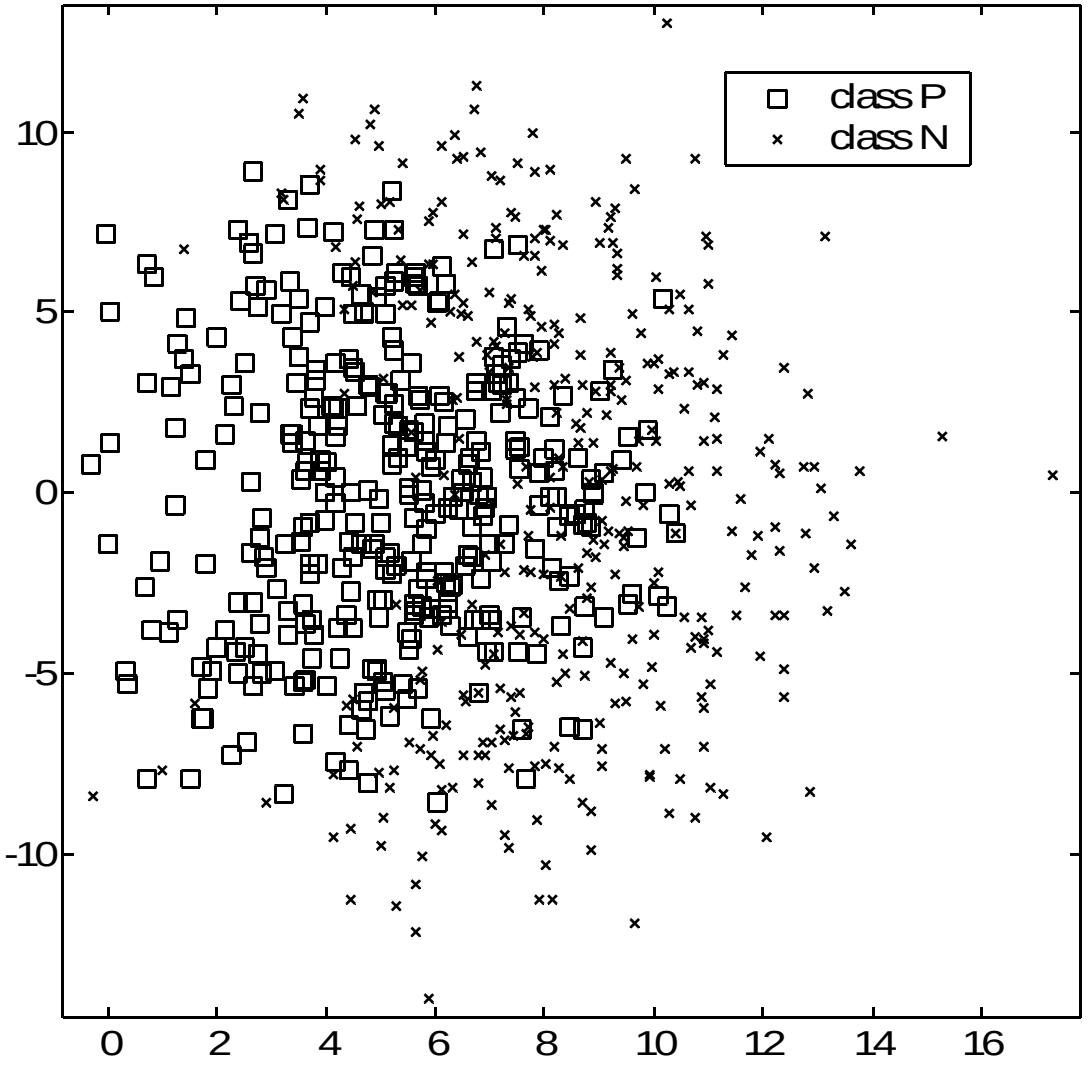
$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x} \cdot \mathbf{y} - \delta)$$

Definizione delle caratteristiche di una SVM

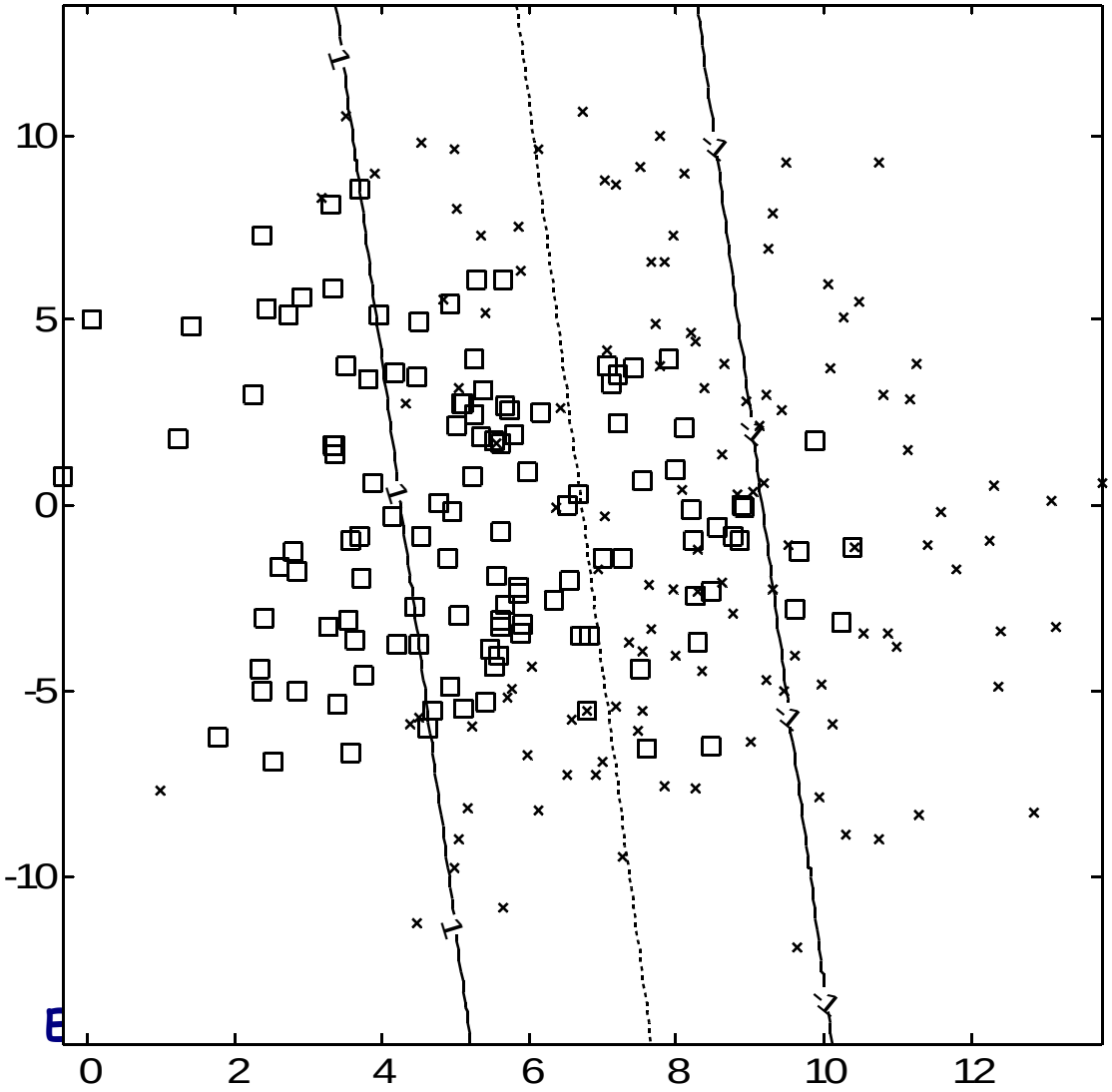
- Per impiegare una SVM è allora necessario definire:
  1. tipo di kernel da impiegare
  2. parametri del particolare kernel
  3. valore di C
- Non esistono criteri teorici per queste scelte; tipicamente va fatta una verifica su un insieme di validazione.


test error
7.2
7
6.8
6.6
6.4
6.2
6
5.8
5.6
5.4
5.2
0
5
10
15
20
25
30
35
40
45
50
C

Esempio



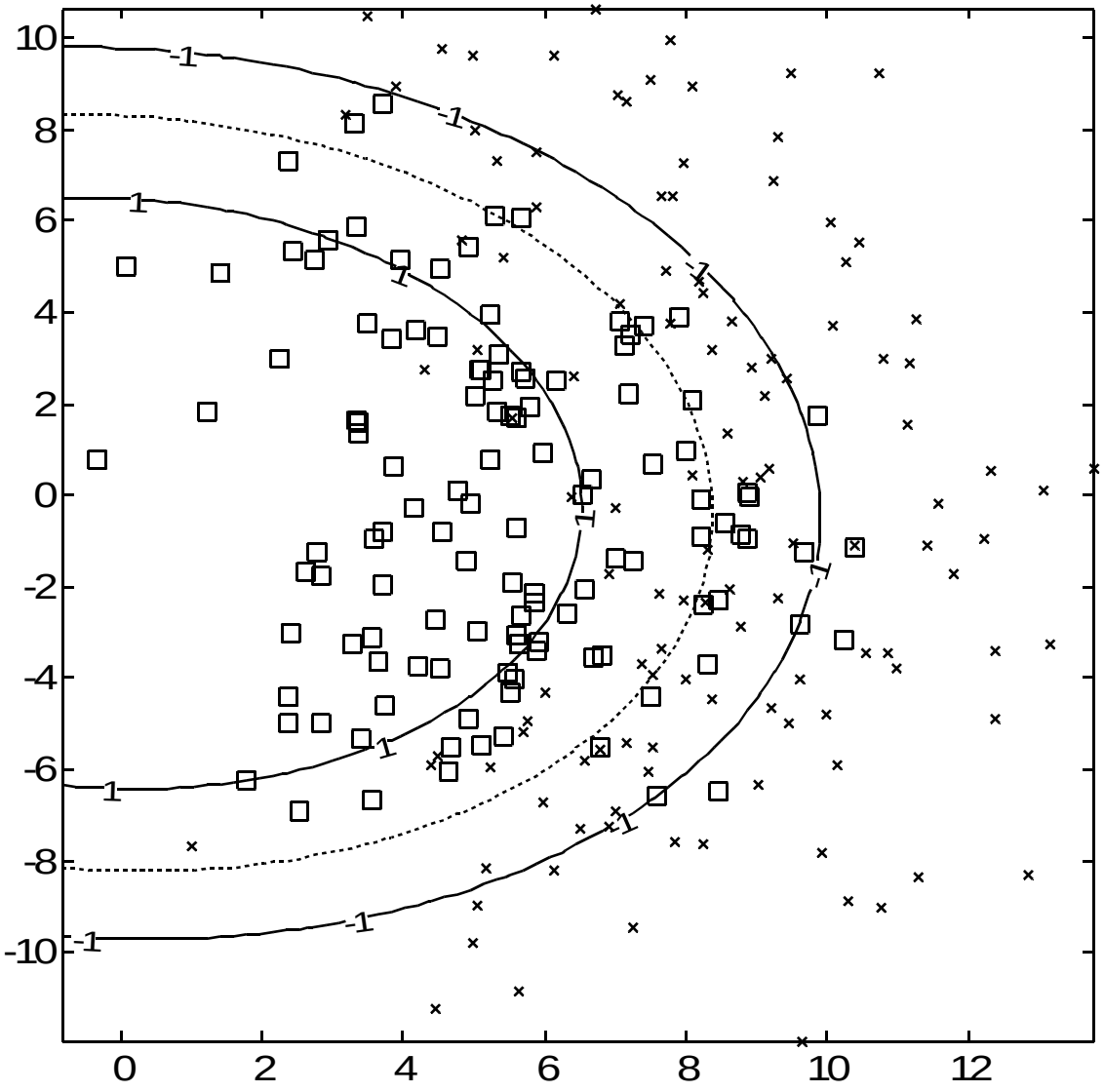
SVM lineare



$K(x,y)=(x\cdot y)$

163 SV su 240 campioni di training

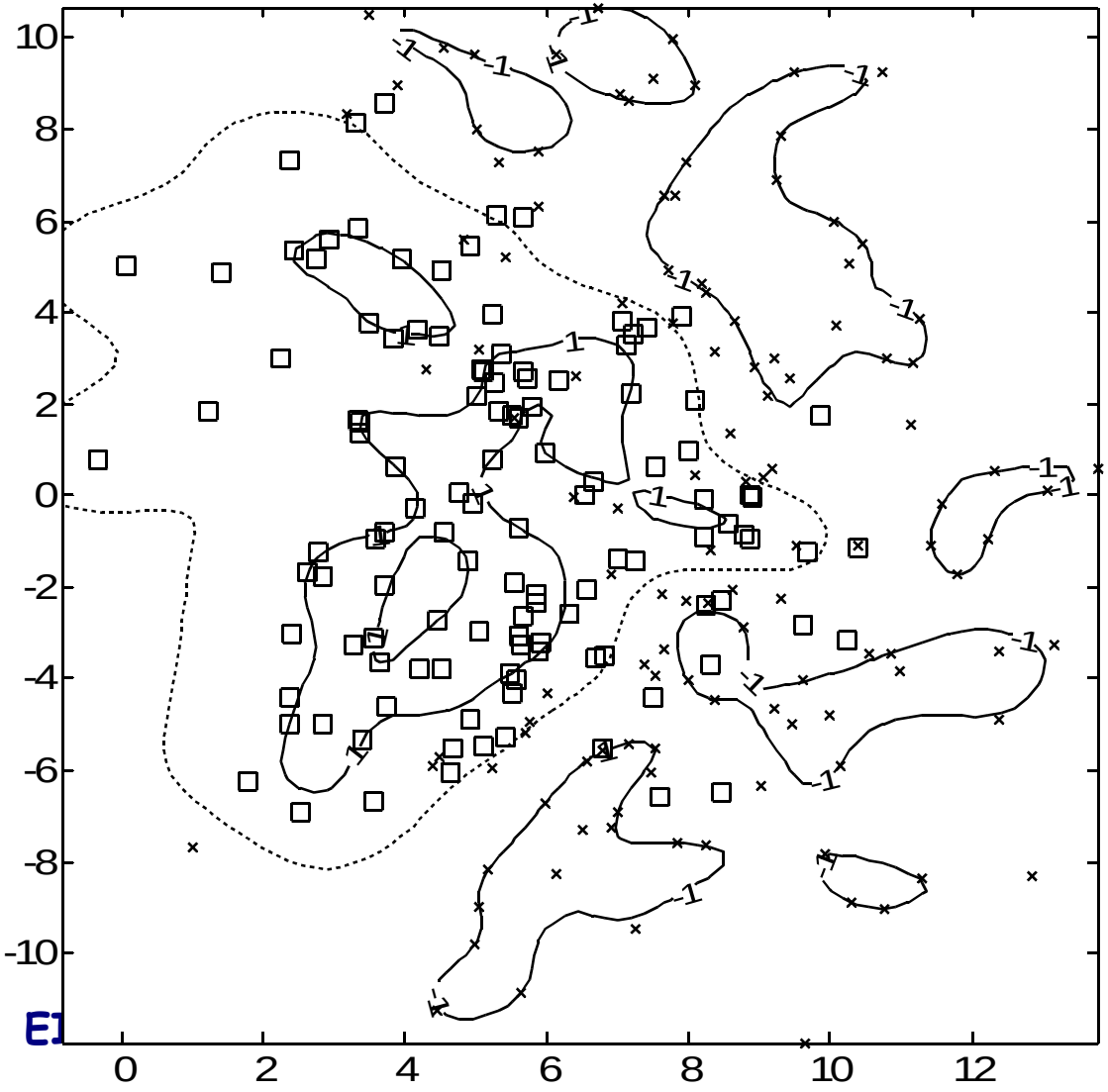
SVM polinomiale



$K(x,y)=(x\cdot y+1)^2$

121 SV su 240 campioni di training

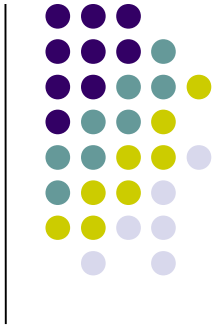
SVM RBF



$K(x,y)= \exp(-0.5 \cdot ||x-y||^2)$

190 SV su 240 campioni di training

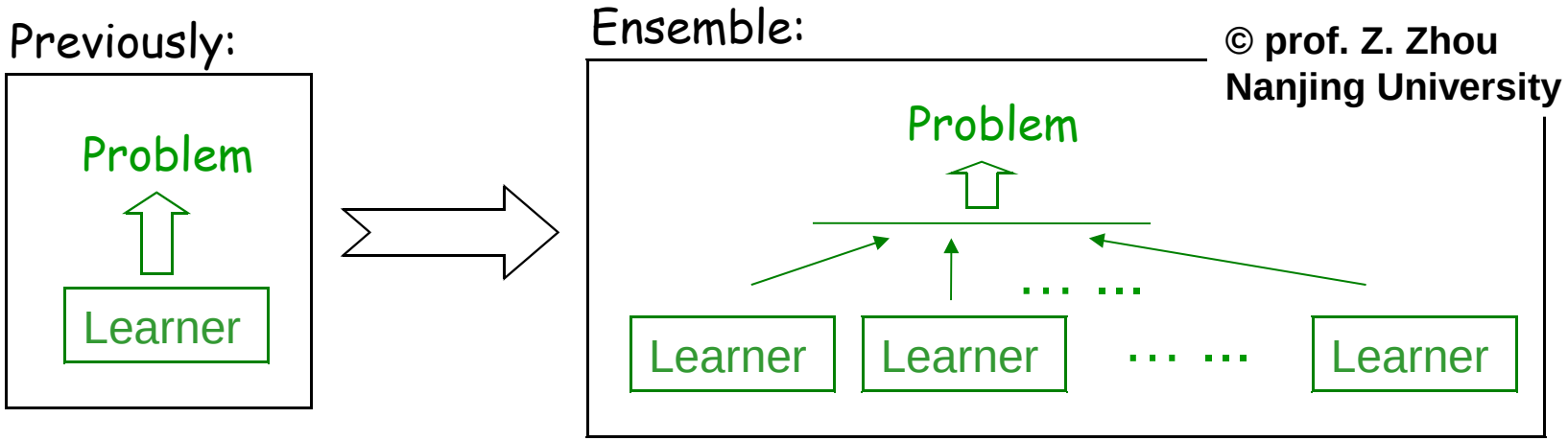
AdaBoost



- Insieme alle SVM è uno dei modelli di classificatore oggi più diffusi.
- Esistono numerose varianti.
- E' basato sulla combinazione di più classificatori componenti.
- Garantisce buone prestazioni
- Può essere usato con diversi classificatori componenti
- Semplice da implementare
-

Ensemble Learning

- Un paradigma dell'apprendimento automatico nel quale si usano più classificatori per risolvere il problema.

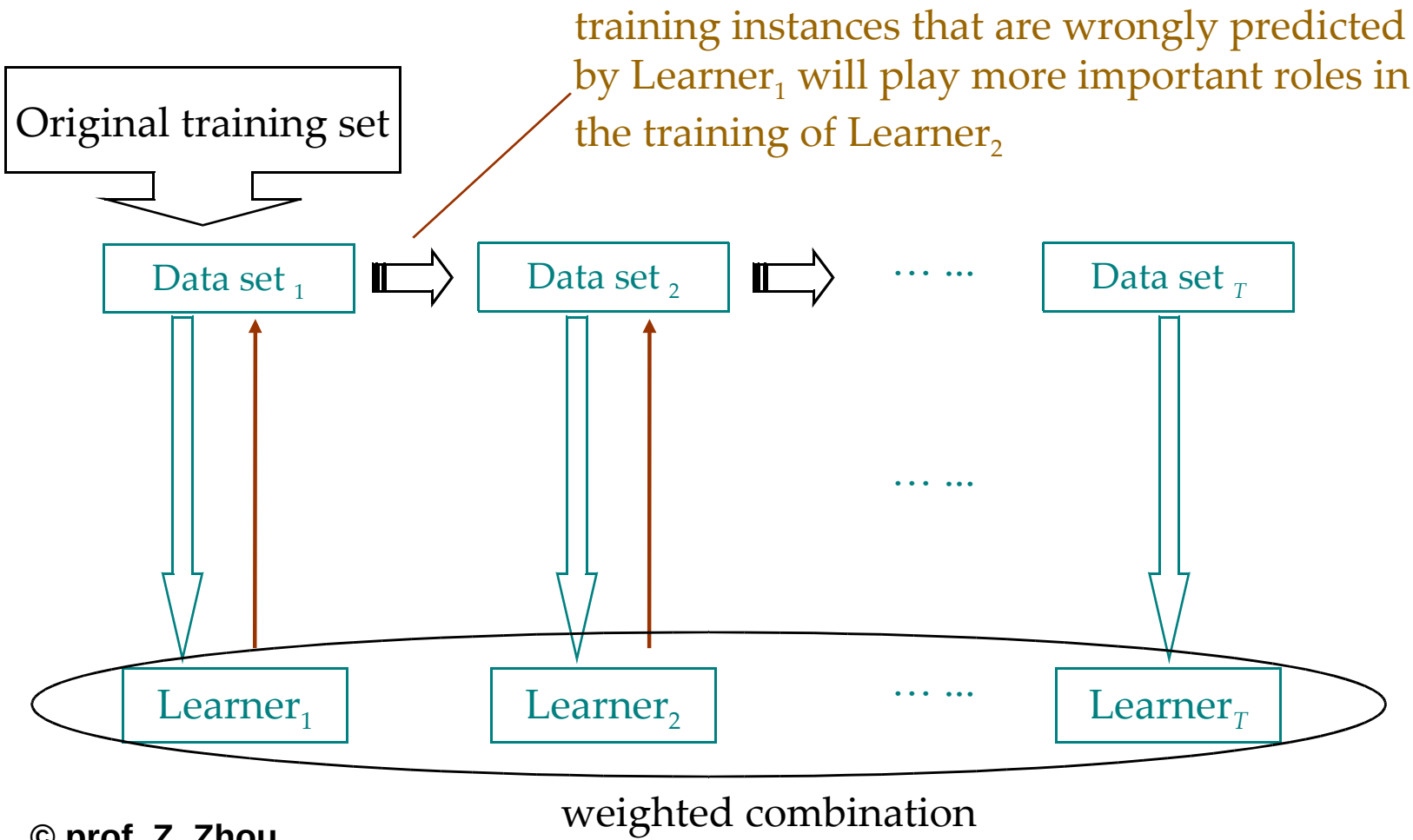


- La capacità di generalizzazione è di solito significativamente migliore rispetto al classificatore unico.
- Il boosting è una delle principali tecniche in questo ambito

AdaBoost: caratteristiche

- Ogni classificatore componente è un “weak learner”, al quale non è richiesta un’elevata accuratezza (basta un recognition rate ≥ 0.5).
- I weak learner vengono determinati sul training set in passi successivi.
- In ogni passo i campioni del TS vengono pesati in maniera diversa, a seconda del fatto se sono stati correttamente classificati nei passi precedenti.

Il processo di apprendimento



Algoritmo

Given: $(x_1, y_1), ..., (x_m, y_m)$; $x_i \in \mathcal{X}, y_i \in \{-1, 1\}$

Initialize weights $D_1(i) = 1/m$ ← All'inizio i campioni sono pesati uniformemente

For $t = 1, ..., T$:

1. (Call *WeakLearn*), which returns the weak classifier $h_t : \mathcal{X} \to \{-1, 1\}$ with minimum error w.r.t. distribution D_t;
2. Choose $\alpha_t \in R$,
3. Update

Incrementa (decrementa) il peso dei campioni erroneamente (correttamente) classificati

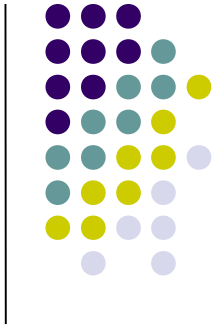
$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor chosen so that D_{t+1} is a distribution

Output the strong classifier:

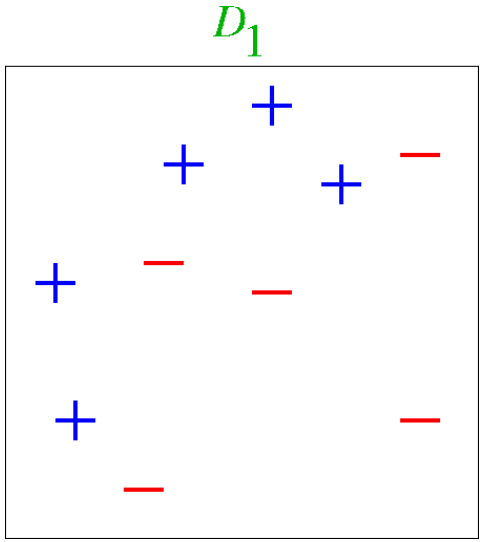
$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Weak Learner



- Il weak learner è un classificatore molto semplice (es. lineare).
- Molto spesso è semplicemente una regola di confronto tra una feature ed una soglia, per cui l'apprendimento consiste nel cercare la soglia migliore su ciascuna feature e scegliere la feature che fornisce l'errore minimo.
- In questo caso, Adaboost realizza una feature selection.

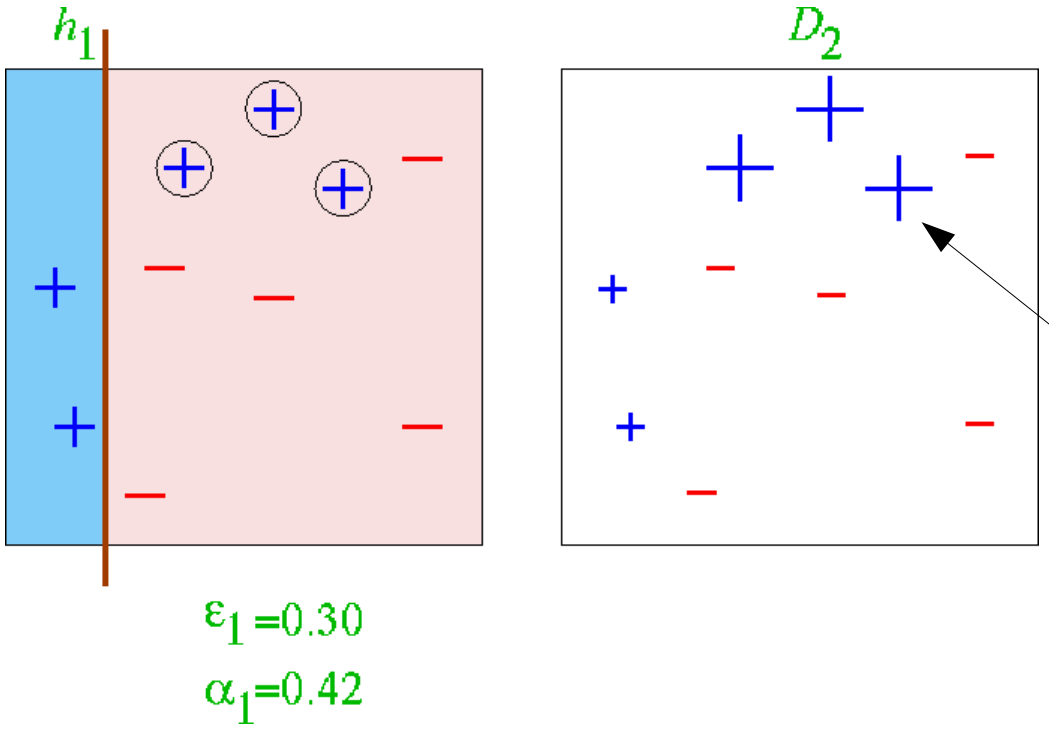
Esempio



All'inizio, tutti i campioni hanno lo stesso peso.

Spazio delle features a due dimensioni (2 features)

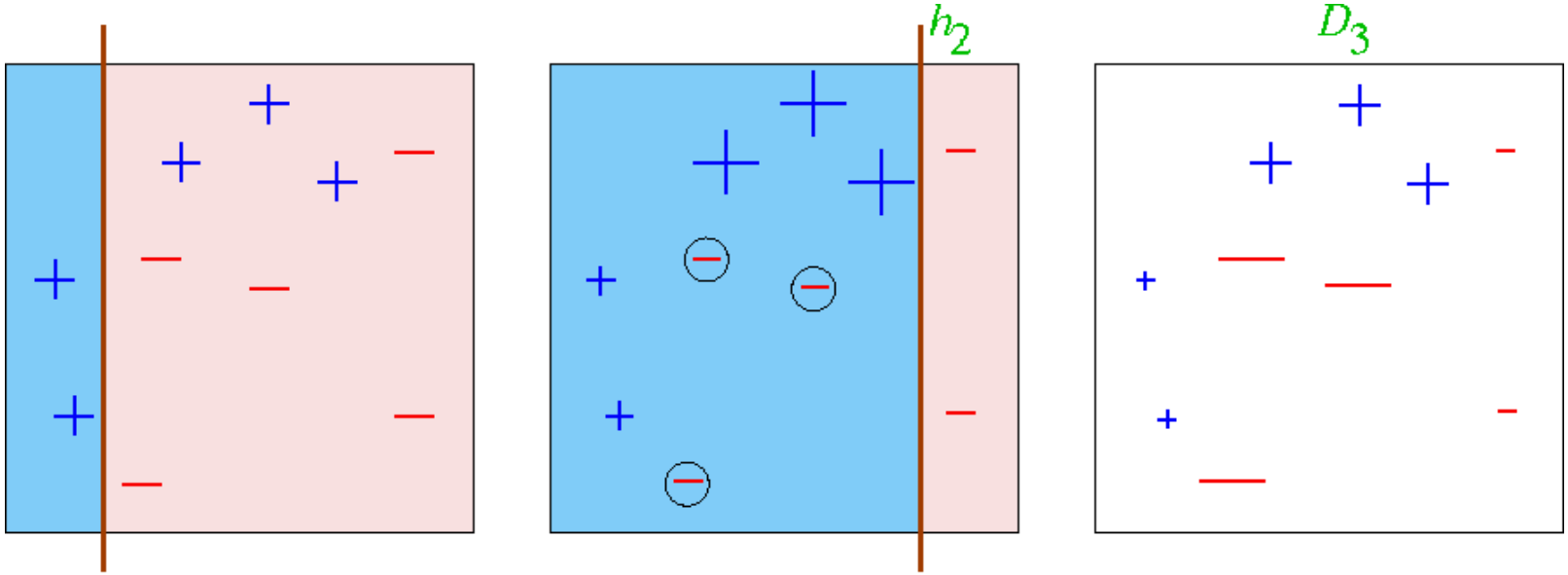
Esempio



Si costruisce il primo weak learner, cui compete un errore del 30%.

Di conseguenza, i campioni errati ricevono un peso maggiore.

Esempio

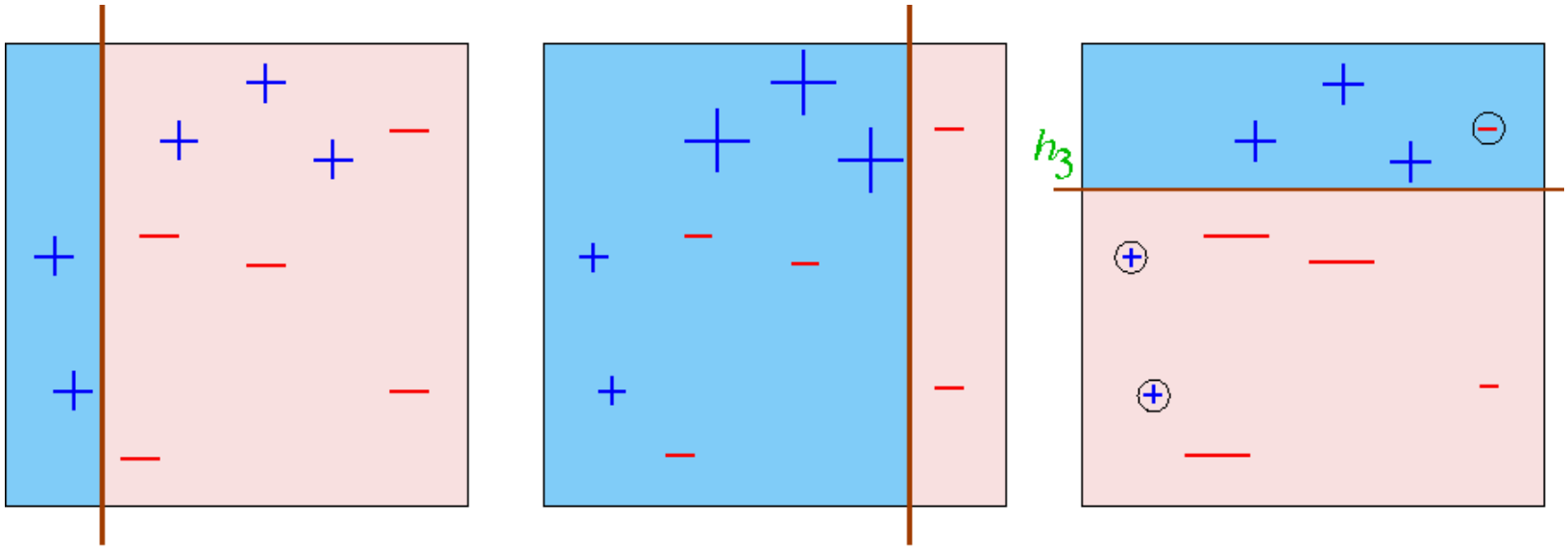


$\varepsilon_2 = 0.21$ ← Errore pesato

$\alpha_2 = 0.65$

Si costruisce il secondo weak learner: sulla base dei suoi risultati, i pesi sui campioni vengono aggiornati.

Esempio

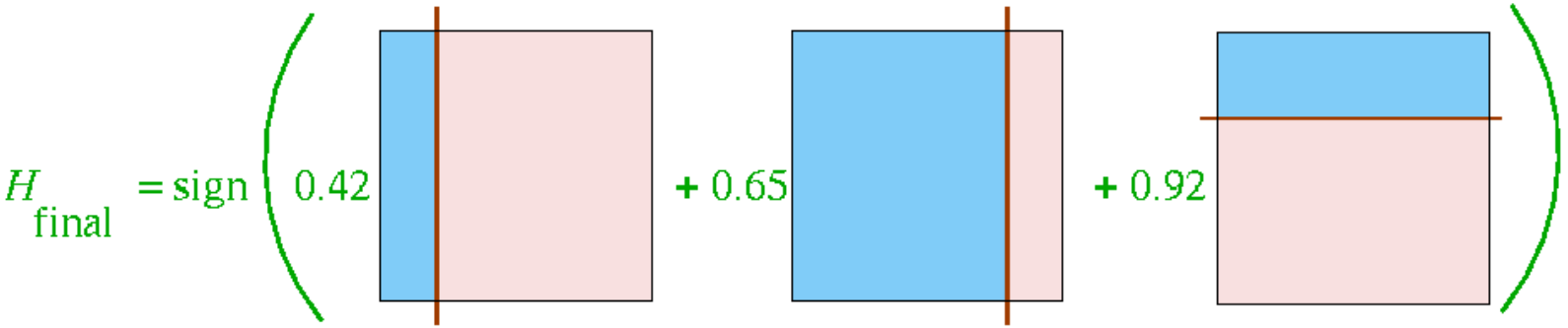


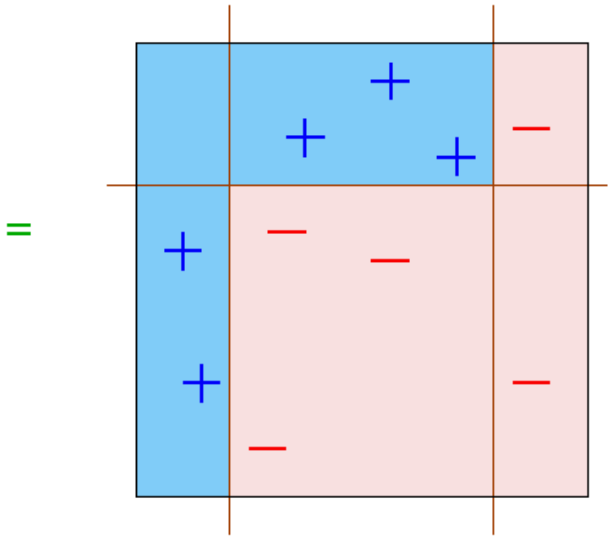
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

Si costruisce il terzo weak learner

Esempio

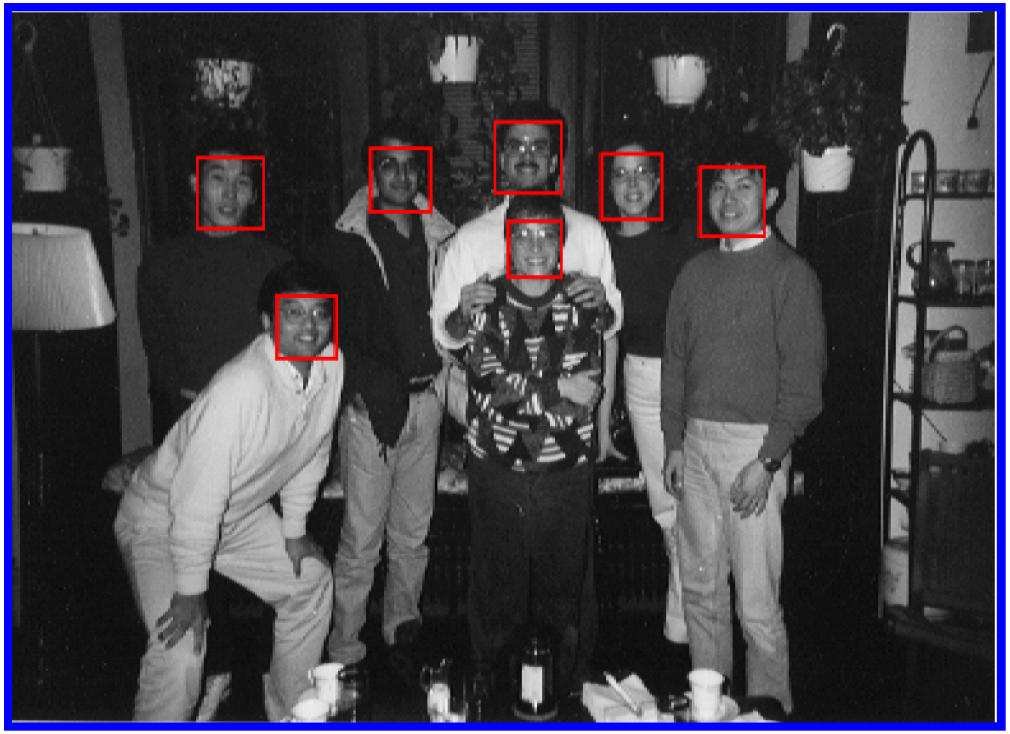




Classificatore finale ottenuto come combinazione lineare dei tre weak learner.

Si noti come si sia individuata la frontiera di decisione corretta.

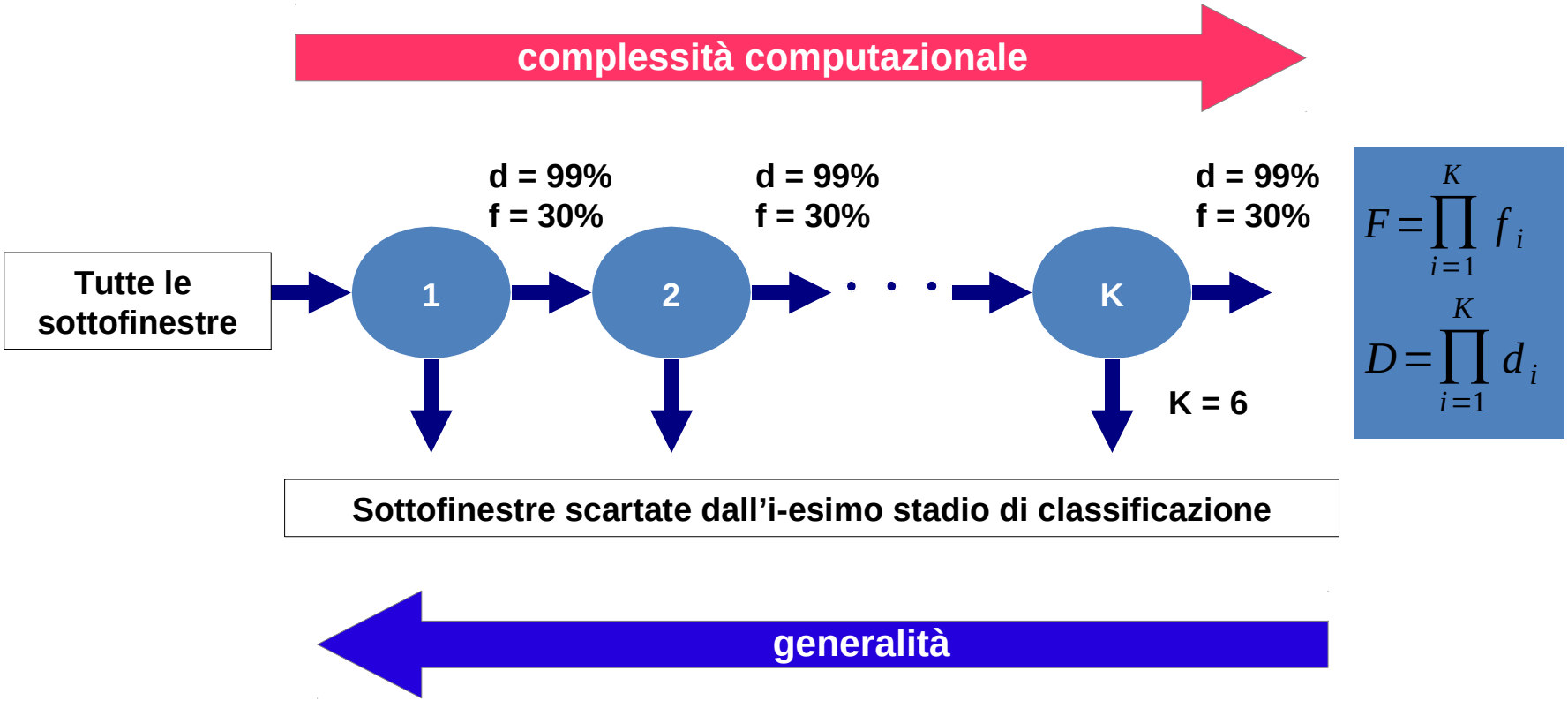
Applicazione: Face Detection



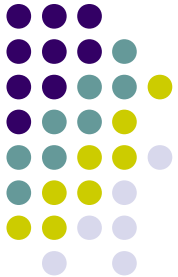
Algoritmo di Viola-Jones

- Tecnica di apprendimento supervisionato
- Raggiunge elevate velocità d'elaborazione ottenendo ottimi detection rate
- Lavora su sottofinestre estratte dall'immagine da analizzare
- Tre caratteristiche fondamentali:
  - Un algoritmo di apprendimento basato su AdaBoost
  - Un set numeroso di semplici feature facilmente calcolabili
  - Un metodo per combinare diversi classificatori in cascata

Schema del detector

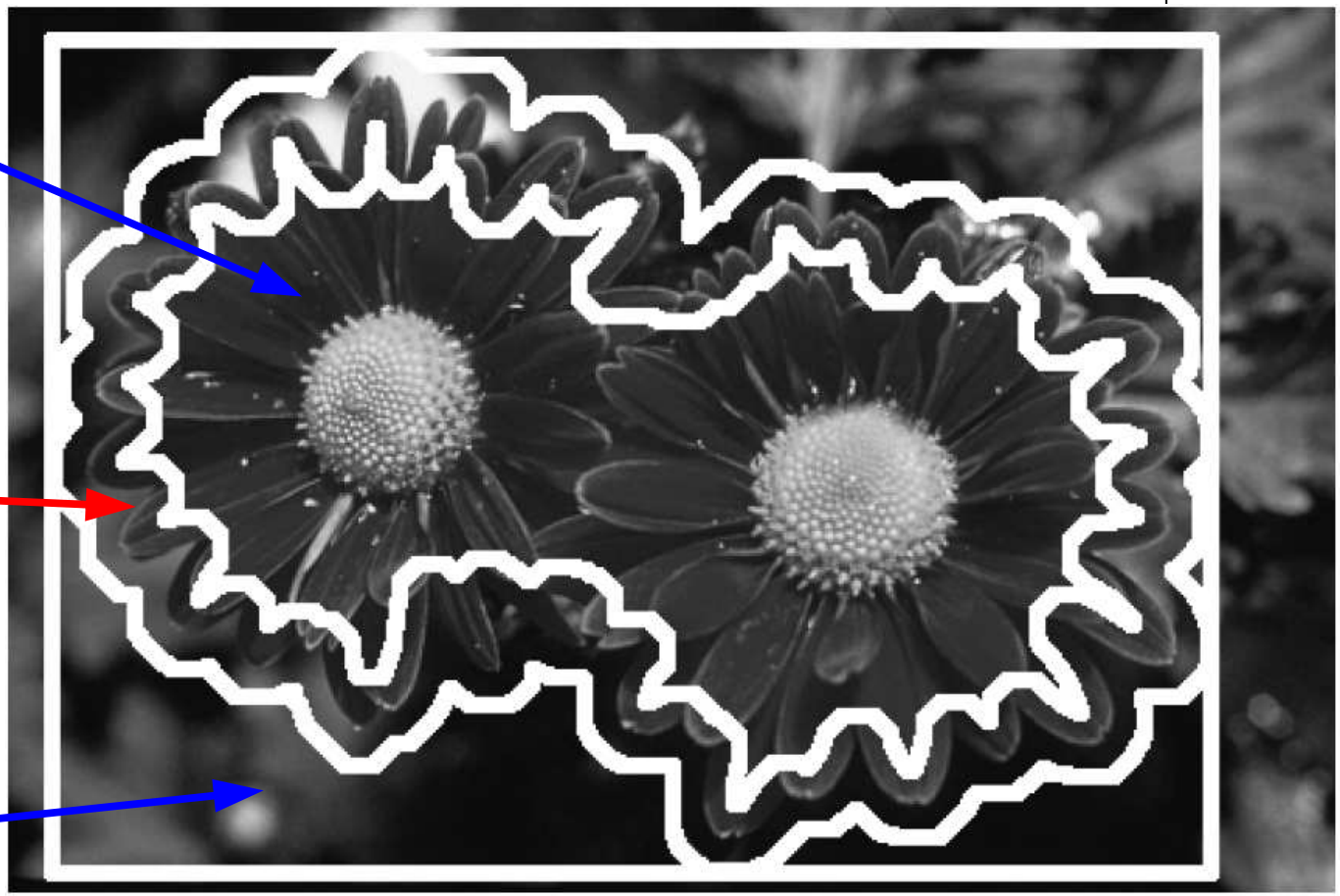


Applicazione: Segmentazione come problema di classificazione



- SpatialBoost (Shai Avidan): algoritmo di segmentazione basato su AdaBoost.
- Features: sia fotometriche che spaziali
- Classi: interno vs. esterno
- Apprendimento e test sulla singola regione

Organizzazione di SpatialBoost



Training set:
Campioni
classe ‘interno’

Test set:
Classificazione
‘interno’ vs.
‘esterno’

Training set:
Campioni
classe ‘esterno’

Risultati