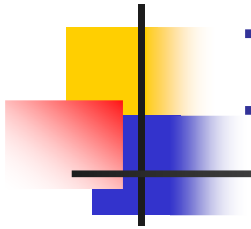
Classificazione e regressione per mezzo di Support Vector Machine

Felice Andrea Pellegrino

Dipartimento di Elettrotecnica Elettronica e Informatica

Università degli Studi di Trieste

Il presente documento è da ritenersi una bozza. Una versione riveduta e corretta sarà disponibile, dopo il 10 luglio, all'indirizzo http://control.units.it/pellegrino

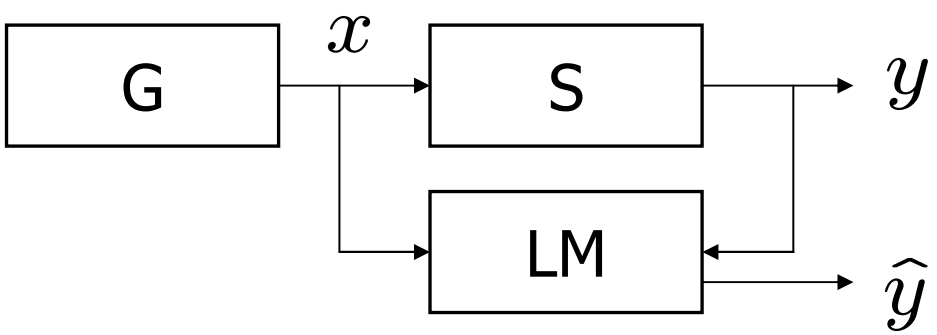


Indice

- Elementi di Teoria dell'Apprendimento Statistico
  - Principio di minimizzazione del rischio empirico
  - Capacità dell'insieme delle ipotesi
  - Principio di minimizzazione del rischio strutturale
- Classificazione binaria:
  - Macchine lineari
  - Percettroni, funzioni potenziale, reti neurali
  - Macchine non lineari
- Regressione

Apprendimento (automatico supervisionato)

- Trovare una dipendenza funzionale a partire da un numero (limitato) di osservazioni
  - Generatore (G)
  - Supervisore (S)
  - Learning machine (LM)

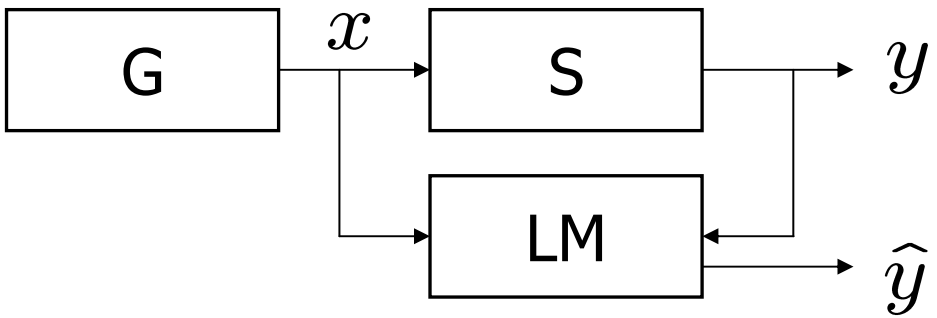


$$x \sim F(x)$$

$$y \sim F(y|x)$$

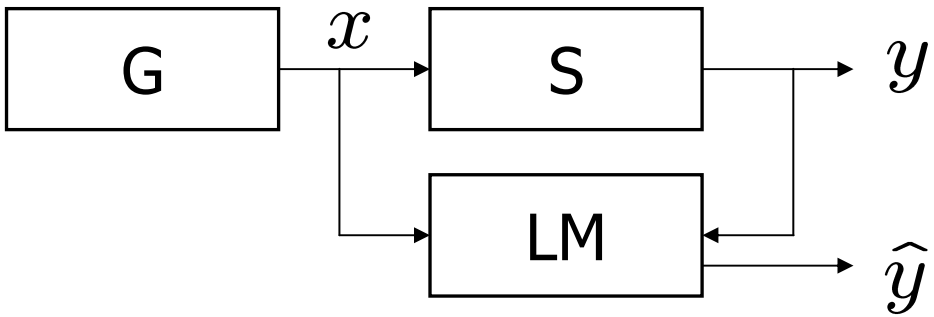
Apprendimento (automatico supervisionato)

- $F(x)$ distribuzione di probabilità di $x \in \mathcal{X} \subseteq R^n$
- $F(y|x)$ distribuzione condizionale di $y \in \mathcal{Y} \subseteq R$ (comprende il caso deterministico $y = f(x)$)
- $F(x, y) = F(x)F(x|y)$ distribuzione congiunta, definita in $\mathcal{X} \times \mathcal{Y}$. E' fissa e incognita.



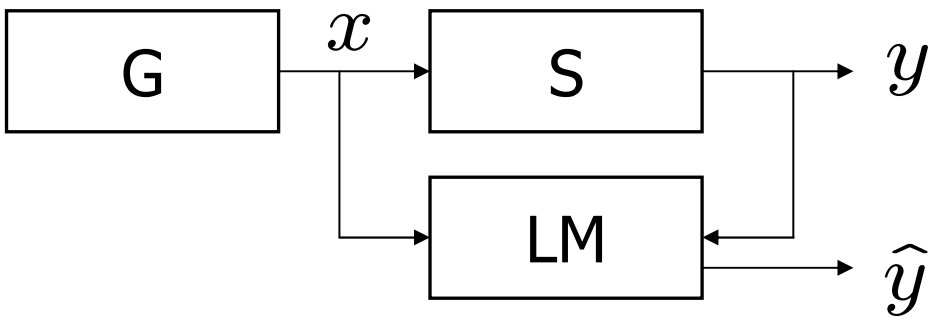
Apprendimento (automatico supervisionato)

- $\mathcal{S} = \{(x_1, y_1), \ldots, (x_l, y_l)\}$ insieme delle coppie (x, y) osservate (*Training Set*)
- $\mathcal{H}$ insieme delle funzioni $h : \mathcal{X} \rightarrow \mathcal{Y}$ che LM è in grado di implementare (*Insieme delle ipotesi*)
  - Esempio: $\mathcal{H} = \{h(x, \alpha), \alpha \in \Lambda\}$



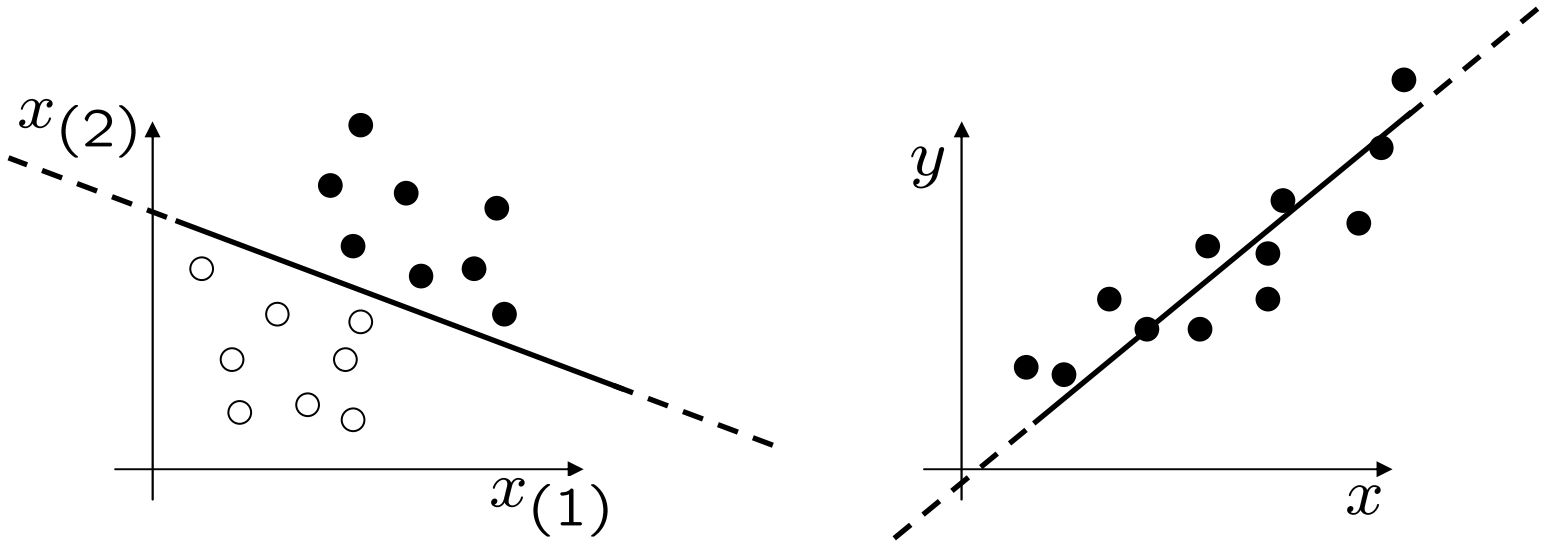
Apprendimento (automatico supervisionato)

- Obiettivo: scegliere *la migliore* fra le funzioni $h \in \mathcal{H}$
  - la funzione migliore è quella che garantisce la massima "capacità di predizione", in un senso che va specificato



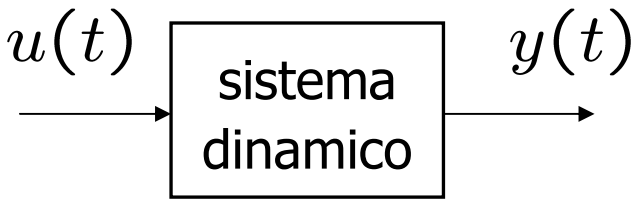
Due tipici problemi

- Due tipici problemi che possono essere espressi nei termini di apprendimento supervisionato:
  - Classificazione binaria: $y \in \mathcal{Y} = \{-1, 1\}$
  - Regressione: $y \in \mathcal{Y} \subseteq R$



Esempio: identificazione a scatola nera

- Anche l'identificazione a scatola nera conduce a un problema di apprendimento supervisionato:



  - Osservazione di $u(t),\ y(t),\ t = 0, \ldots, N$
  - Vettore di regressione, ad esempio:

$$x(t) = [y(t), y(t-1), \ldots, y(t-n_y), u(t), , \ldots, u(t-n_u)]$$
$$x(t) \in R^q$$

Esempio: identificazione a scatola nera

- Training set: $\mathcal{S} = \{(x_1, y_1), \ldots, (x_l, y_l)\}$

dove

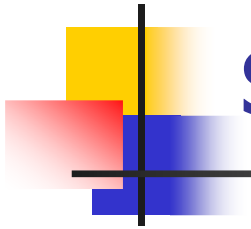
$$x_i = x(i + \max\{n_y, n_u\} - 1)$$
$$y_i = y(i + \max\{n_y, n_u\} - 1)$$
$$l = N - \max\{n_y, n_u\} + 1$$

- Se supponiamo che il legame sia

$$y_i = f(x_i) + e_i$$

si tratta di stimare $f(\cdot)$ sulla base delle osservazioni $\mathcal{S}$

Esempio: identificazione a scatola nera



- … il che è appunto un problema di apprendimento supervisionato: stimare una dipendenza funzionale f sulla base di un insieme (limitato) di osservazioni $\mathcal{S}$

- Si ottiene così un predittore:

$$\hat{y}(t) = h(x(t-1))$$

- Si vuole scegliere $h \in \mathcal{H}$ "il più possibile simile" ad f ovvero quella che genera il migliore predittore.

Il funzionale di rischio (risk functional)

- Quale è la migliore scelta per $h \in \mathcal{H}$?
- Definiamo un *funzionale di rischio*

$$R(h) = \int_{\mathcal{X} \times \mathcal{Y}} L(y, h(x)) dF(x, y)$$

  - $R(h)$ è il valore atteso del *costo* $L(y, h(x))$ in cui si incorre quando, a fronte dell'ingresso x la macchina produce $h(x)$ anziché il valore y prodotto dal supervisore

Esempi di funzione di costo

- Costo 0/1 (0/1 loss function)

$$L(y, h(x)) = \begin{cases} 0 \text{ se } y[h(x)] > 0 \\ 1 \text{ altrimenti} \end{cases}$$

- Costo quadratico (quadratic loss function)

$$L(y, h(x)) = (y - h(x))^2$$

- Costo lineare (linear loss function)

$$L(y, h(x)) = |y - h(x)|$$

Il rischio empirico

- La migliore scelta è allora $h^*(x)$ dove

$$h^* = \arg\min_{h \in \mathcal{H}} R(h) = \arg\min_{h \in \mathcal{H}} \int L(y, h(x)) dF(x, y)$$

  - La distribuzione $F(x, y)$ è però incognita e quindi non è possibile minimizzare direttamente $R(h)$
  - Ciò che invece è noto è il *rischio empirico*

$$R_{emp}(h) = \frac{1}{l} \sum_{i=1}^{l} L(y_i, h(x_i))$$

Minimizzazione del rischio empirico (ERM)

- La minimizzazione del rischio empirico (*Empirical Risk Minimization,* ERM) è un principio induttivo che trova applicazione in metodi ben noti quali:
  - Regressione ai minimi quadrati

$$R_{emp}(h) = \frac{1}{l} \sum_{i=1}^{l} (y_i - h(x_i))^2$$

  - Stima della densità tramite massima verosimiglianza

$$R_{emp}(p) = -\frac{1}{l} \sum_{i=1}^{l} \ln p(x_i)$$

Minimizzazione del rischio empirico

- E' sempre una buona idea minimizzare il rischio empirico?
- Se non lo è, quali sono le proprietà che un dato problema di apprendimento deve possedere affinché abbia senso ricorrere alla minimizzazione del rischio empirico?

Teoria dell'apprendimento statistico
(*Statistical Learning Theory*)

Legge dei grandi numeri e consistenza di ERM (1)

- La legge dei grandi numeri (convergenza delle medie ai valori attesi) assicura che, per h fissata si ha, all'aumentare del numero di osservazioni l :

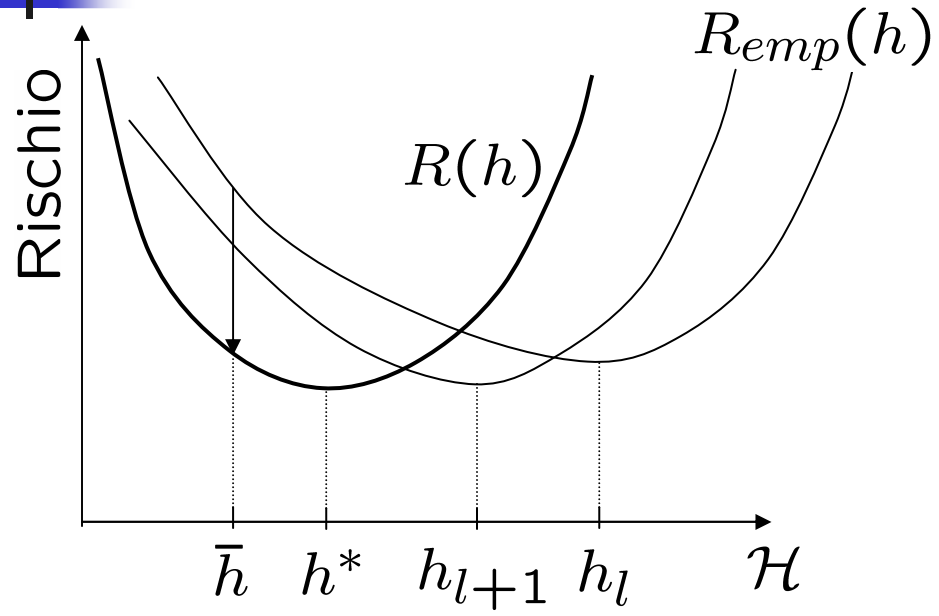
$$R_{emp}(h) \rightarrow R(h)$$

o, più precisamente:

$$\lim_{l \rightarrow \infty} P(|R_{emp}(h) - R(h)| > \epsilon) = 0, \quad \forall \epsilon > 0$$

- Questo fatto però non garantisce la *consistenza* di ERM, ossia che per un infinito numero di osservazioni si ottenga quella funzione $h^* \in \mathcal{H}$ che minimizza il rischio atteso.

Legge dei grandi numeri e consistenza di ERM (2)



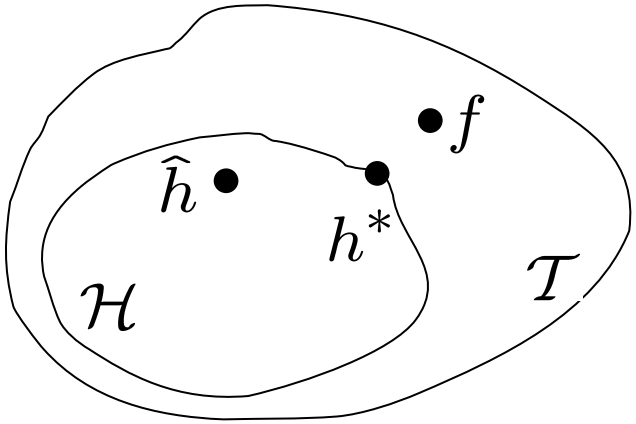
- ERM si dice consistente se la successione $\{h_l\}$ converge ad h^*
- Condizione necessaria e sufficiente: convergenza *uniforme* rispetto alla classe di funzioni che la macchina può implementare

$$\lim_{l\to\infty} P\left(\sup_{h\in\mathcal{H}}(R_{emp}(h) - R(h)) \geq \epsilon\right) = 0, \quad \forall\epsilon > 0$$

- Come si ottiene ciò? Scegliendo in maniera opportuna l'insieme $\mathcal{H}$

Errori di approssimazione, stima e generalizzazione (1)

- $\mathcal{T}$ (target space): lo spazio di funzioni che si assume contenga la funzione "vera" f
- h^*: la migliore scelta possibile fra le funzioni di $\mathcal{H}$
- $\hat{h}$: la funzione scelta dall'algoritmo di apprendimento sulla base del training set



Errori di approssimazione, stima e generalizzazione (2)

- Errore di *approssimazione* (dovuto al fatto che $f \notin \mathcal{H}$)

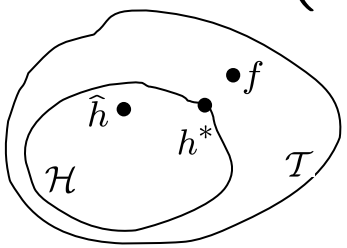
$$R(h^*) - R(f)$$

- Errore di *stima* (dovuto al fatto che $\hat{h} \neq h^*$)

$$R(\hat{h}) - R(h^*)$$

- L'errore di *generalizzazione* è la somma dei due:

$$R(\hat{h}) - R(f) = [R(\hat{h}) - R(h^*)] + [R(h^*) - R(f)]$$

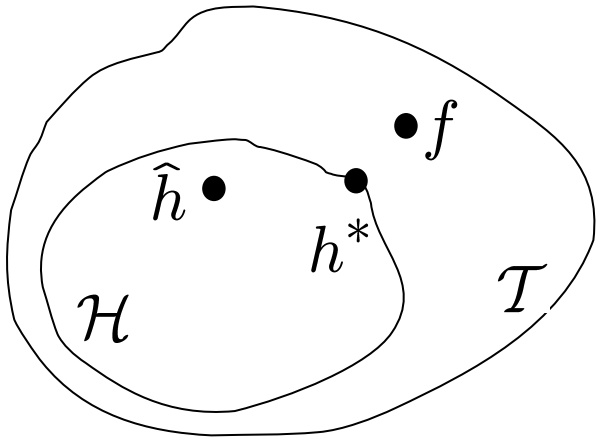


E' ciò che interessa minimizzare!

Minimizzare l'errore di generalizzazione

- Per minimizzare l'errore di generalizzazione bisogna minimizzare i due addendi:

$$R(\hat{h}) - R(f) = [R(\hat{h}) - R(h^*)] + [R(h^*) - R(f)]$$



- E' evidente che il secondo addendo (errore di approssimazione) può essere reso piccolo (e talvolta annullato) prendendo $\mathcal{H}$ "grande"
- D'altra parte, si può dimostrare che prendere $\mathcal{H}$ "grande" rende grande il primo addendo (o addirittura rende inconsistente il principio ERM)

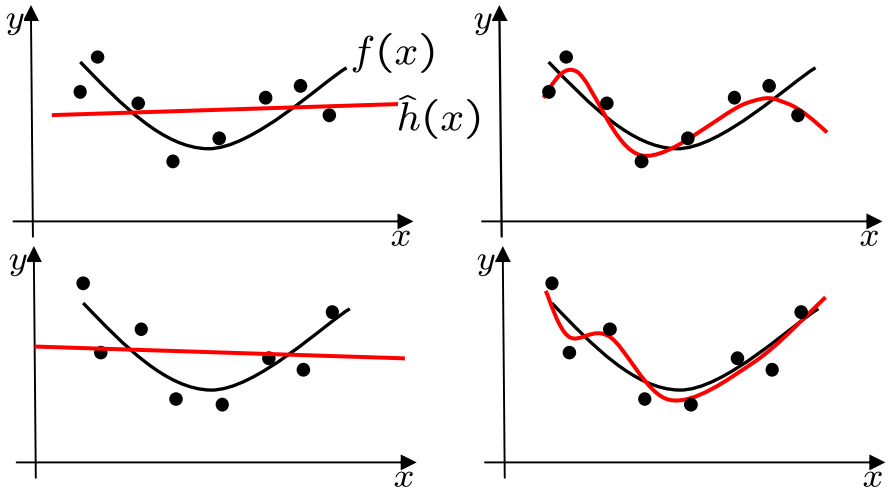
Bias-Variance dilemma

- Regressione ai minimi quadrati della funzione $y = f(x) + e$

  dove e rappresenta un errore di misura

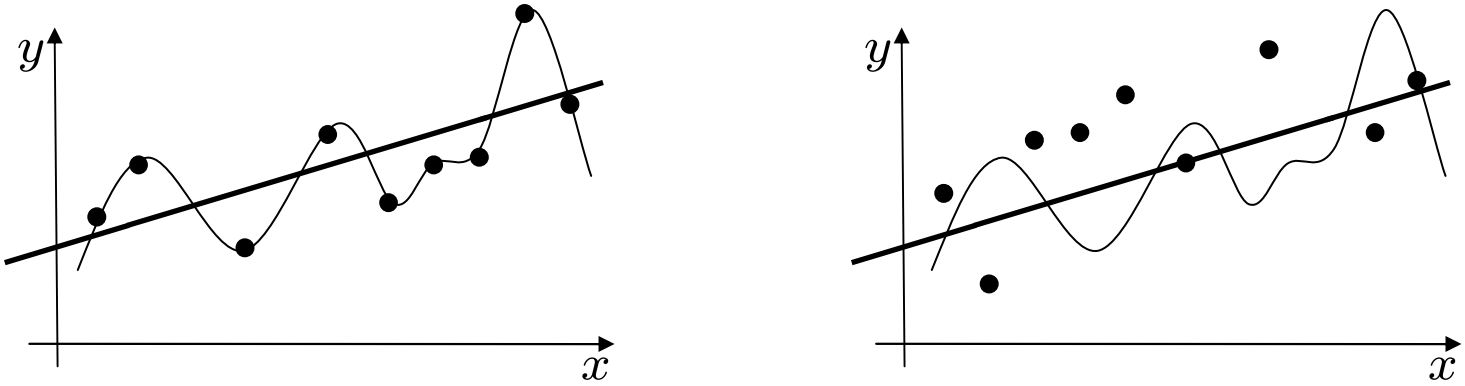
- bias: $E_{\mathcal{S}} R(\hat{h})$

  variance: $E_{\mathcal{S}}[R(\hat{h}) - \text{bias}]^2$



- Se l'insieme delle ipotesi è ristretto alle *funzioni lineari*, la soluzione varia poco al variare del TS (→ variance basso). Ma la classe di funzioni è poco flessibile (→bias alto)
- Se le ipotesi sono *polinomi di grado elevato* aderiscono meglio ai dati (→bias basso) ma la soluzione è più sensibile al rumore e quindi varia al variare del TS (→ variance alto)

Overfitting



- Minimizzare il rischio empirico su un insieme di ipotesi troppo ricco conduce al problema dell'*overfitting*. Un'ipotesi si dice "sovraspecializzata" se esiste in $\mathcal{H}$ un'altra ipotesi che ha rischio minore (e quindi è ad essa preferibile) ma rischio empirico maggiore (e quindi viene scartata).

La capacità dell'insieme delle ipotesi: esempio (1)

- Problema di classificazione binaria:

$$\mathcal{Y} = \{-1, +1\}, \ \mathcal{S} = \{(x_1, y_1), \ldots, (x_l, y_l)\}$$

- Supponiamo che l'insieme $\mathcal{H}$ contenga tutte le funzioni da $\mathcal{X}$ a $\mathcal{Y}$.
- Allora, qualunque sia $\mathcal{S}$, esistono in $\mathcal{H}$ infinite funzioni aventi rischio empirico nullo (tutte quelle per cui $h(x_i) = y_i \ \forall i = 1, \ldots, l$)
- Sulla base del solo training set non è possibile decidere quale sia la migliore. Dunque nessun apprendimento è possibile.

La capacità dell'insieme delle ipotesi: esempio (2)

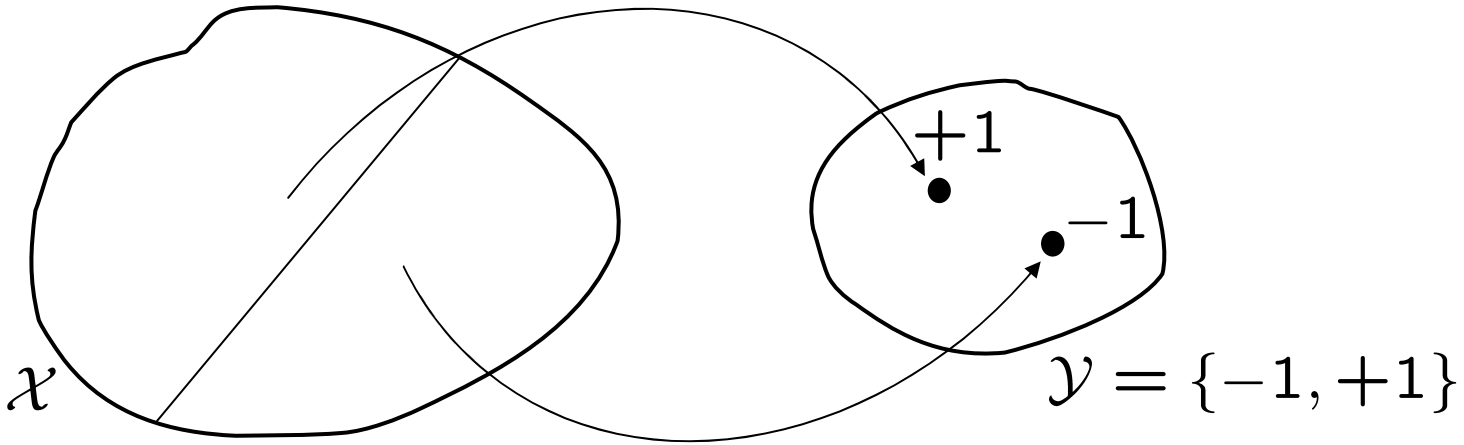
- Dunque è necessario porre delle restrizioni all'insieme delle ipotesi
- Quanto "ricco" è l'insieme? Se è troppo ricco non va bene perché per $l \to \infty$ la sua capacità di spiegare i dati non diminuisce. Ogni nuovo dato non porta alcuna informazione sulla classificazione dei punti ad esso vicini
- Come si quantifica la "ricchezza" dell'insieme delle ipotesi? Una possibilità è ricorrere alla dimensione di Vapnik-Chervonenkis (VC-dimension)

VC-dimension (1)

- Consideriamo per il momento *funzioni indicatrici* (cioè funzioni che assumono solo due valori:

$$h : \mathcal{X} \rightarrow \{-1, +1\}$$

- Le funzioni indicatrici realizzano delle *dicotomie*, cioè suddividono in due classi gli elementi di $\mathcal{X}$



VC-dimension (2)

- Dato $\mathcal{Z} \subseteq \mathcal{X}$, si dice che $\mathcal{Z}$ è frammentato (*shattered*) dall'insieme delle ipotesi $\mathcal{H}$ se $\mathcal{H}$ realizza tutte le possibili dicotomie di $\mathcal{Z}$

- La VC-dimension di un insieme $\mathcal{H}$ di funzioni indicatrici definite su $\mathcal{X}$ è data dalla cardinalità del più grande sottoinsieme di $\mathcal{X}$ che è frammentato da $\mathcal{H}$

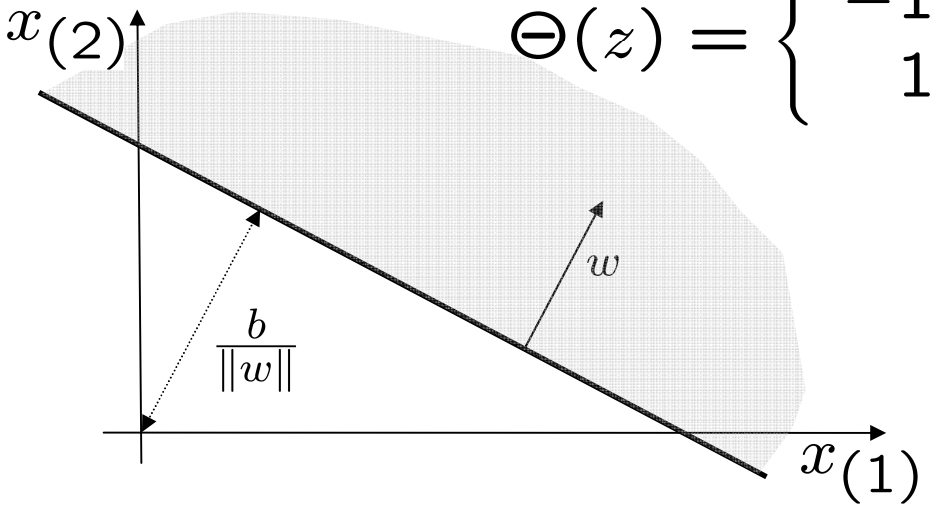
$$VC(\mathcal{H}) = \max_{\mathcal{Z} \subseteq \mathcal{X}} \{|\mathcal{Z}| : \mathcal{Z} \text{ è frammentato da } \mathcal{H}\}$$

VC-dimension, un esempio (1)

- Costruiamo una famiglia parametrica di funzioni indicatrici definite in R^2 componendo una funzione soglia con una funzione lineare (iperpiani separatori in R^2)

$$\mathcal{H} = \{h(x) : h(x) = \Theta(w \cdot x + b),\ w \in R^2, b \in R\}$$

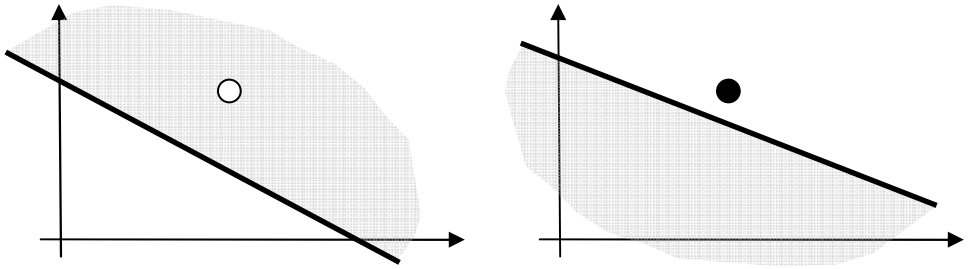
$$\Theta(z) = \begin{cases} -1 & \text{se} \quad z < 0 \\ 1 & \text{se} \quad z \geq 0 \end{cases}$$



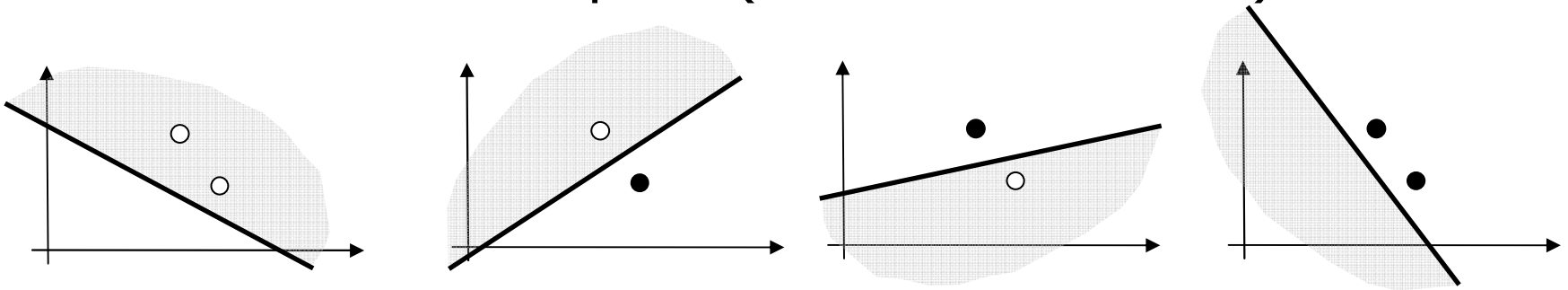
- Quale è la VC-dimension di $\mathcal{H}$?

VC-dimension, un esempio (2)

- Consideriamo un punto ($2^1 = 2$ dicotomie)

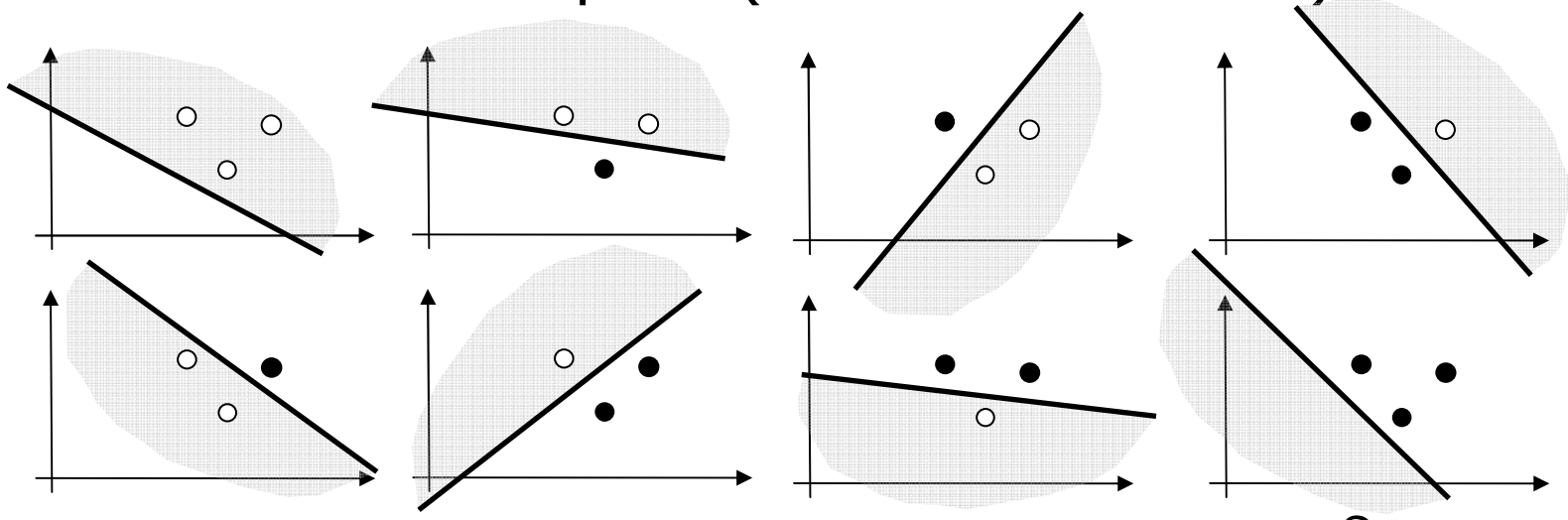


- Consideriamo due punti ($2^2 = 4$ dicotomie)



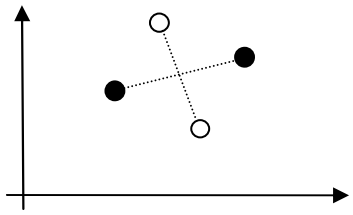
VC-dimension, un esempio (3)

- Consideriamo tre punti ($2^3 = 8$ dicotomie)

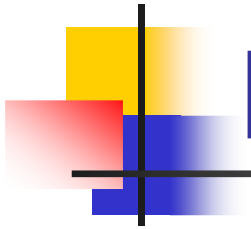


- Nessuna configurazione di quattro punti in R^2 può essere frammentata da piani

$$\Rightarrow \quad VC(\mathcal{H}) = 3$$



VC-dimension e numero di parametri



- In generale, la VC-dimension di un insieme di iperpiani separatori in dimensione n è $n + 1$

- IMPORTANTE: la VC-dimension è in generale diversa dal numero di parametri. Ad esempio, la famiglia seguente ha VC-dimension infinita:

$$\mathcal{H} = \{h(x) : h(x) = \Theta(\sin \alpha x),\ \alpha \in R\}$$

Un legame con la cardinalità dell'insieme delle ipotesi

- Il numero di dicotomie di un insieme di m elementi è 2^m
- quindi $\mathcal{H}$ contiene almeno un numero $2^{VC(\mathcal{H})}$ di funzioni differenti:

$$|\mathcal{H}| \geq 2^{VC(\mathcal{H})}$$

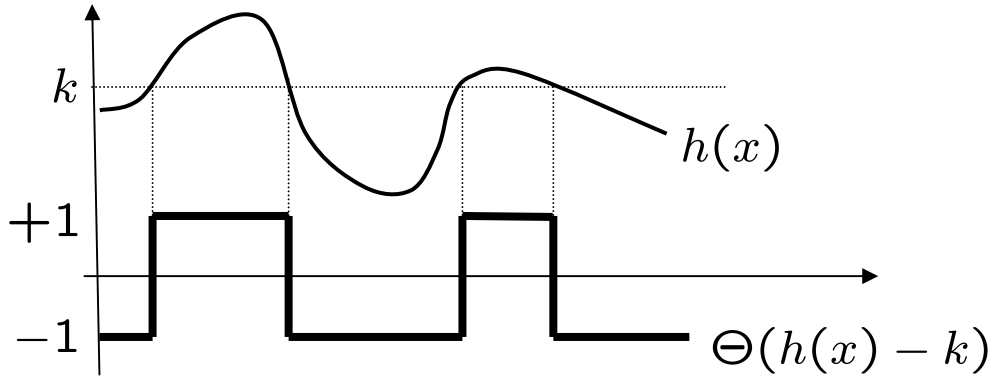
da cui:

$$VC(\mathcal{H}) \leq \log_2 |\mathcal{H}|$$

- Il che mostra il fatto che la VC-dimension è legata alla "ricchezza" dell'insieme delle ipotesi (fatto che si può intuire sulla base della sua stessa definizione)

VC-dimension di funzioni a valori reali

- Una funzione a valori reali $h : \mathcal{X} \rightarrow [A, B] \subseteq R$ può essere individuata tramite l'insieme delle funzioni indicatrici di livello $\mathcal{F}_h = \{\Theta(h(x) - k),\ k \in [A, B]\}$



- L'insieme $\mathcal{F} = \bigcup_{h \in \mathcal{H}} \mathcal{F}_h$ è un insieme di funzioni indicatrici
- Si definisce allora $VC(\mathcal{H}) \doteq VC(\mathcal{F})$

Consistenza di ERM per il problema di classificazione

- Condizione necessaria e sufficiente per la consistenza di ERM indipendentemente dalla distribuzione $F(x)$ è la finitezza della VC-dimension dell'insieme delle ipotesi:

$$d = VC(\mathcal{H}) < \infty \Leftrightarrow \text{ERM consistente su } \mathcal{H}, \forall F(x)$$

- Se vale questa condizione, indipendentemente dalla distribuzione, per $l \to \infty$ il principio di minimizzazione del rischio empirico conduce alla migliore ipotesi nell'insieme $\mathcal{H}$

- Per il problema della regressione, si può formulare un criterio analogo riferito però alla *Vγ-dimension* di funzioni a valori reali

Bound non asintotici

- Cosa si può dire però per un numero finito di osservazioni?
- Grazie alla VC-dimension, è possibile formulare dei bound non asintotici sul rischio indipendenti dalla distribuzione, del tipo seguente:

  Con probabilità $\geq 1-\delta$, per ogni ipotesi $h \in \mathcal{H}$ si ha:

  Rischio Rischio empirico Confidenza

$$R(h) \leq R_{emp}(h) + \phi\left(\frac{d}{l}, \frac{\ln \delta}{l}\right)$$

- ϕ è funzione crescente di d e decrescente di l e δ

Bound non asintotici: esempio

$$R(h) \leq R_{emp}(h) + \sqrt{\frac{d\left(\ln\frac{2l}{d}+1\right) - \ln(\delta/4)}{l}}$$

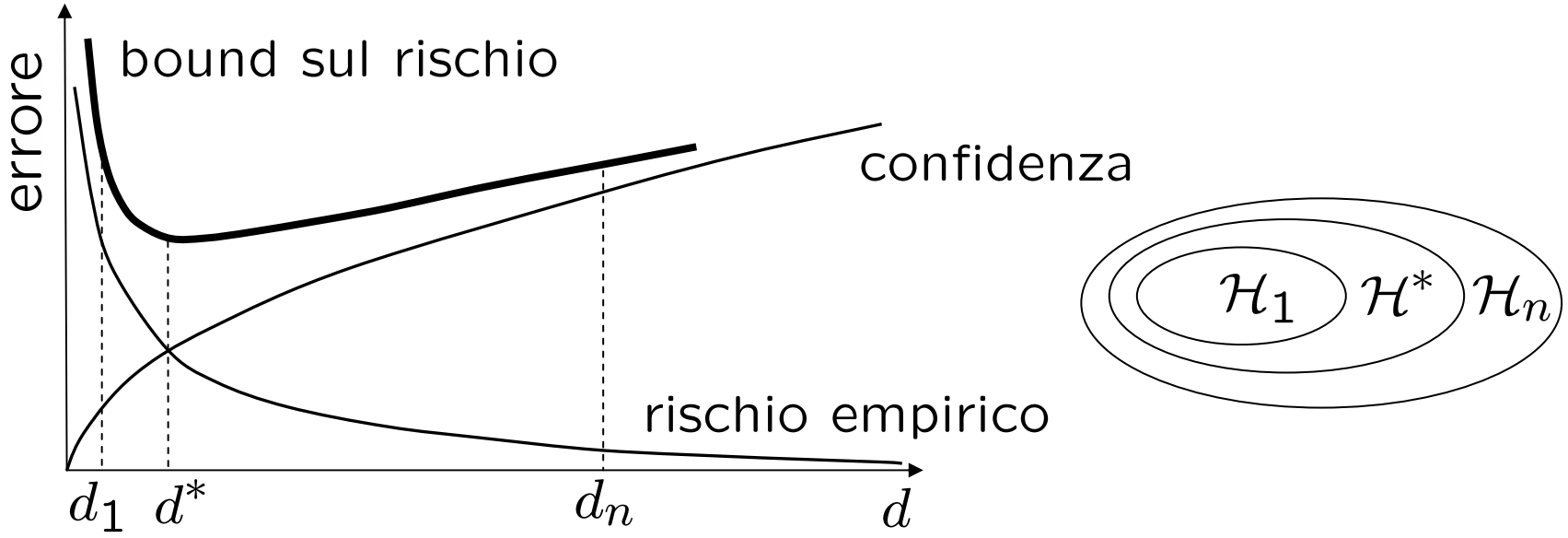
- Questa relazione vale (con probabilità $\geq 1-\delta$) per funzioni indicatrici (problema di classificazione binaria) e costo 0/1
- Per l (numero si osservazioni) fissato, la confidenza è funzione monotona crescente della VC-dimension d dell'insieme delle ipotesi
  - ciò conferma il Bias-Variance dilemma: più "ricco" è l'insieme delle ipotesi, più incerta è la stima della funzione incognita
- IMPORTANTE: il bound è indipendente
  - dalla distribuzione che genera i dati
  - dalla dimensione di $\mathcal{X}$ → vale anche per spazi di dimensione infinita

Osservazioni

$$R(h) \leq R_{emp}(h) + \sqrt{\frac{d\left(\ln\frac{2l}{d}+1\right) - \ln(\delta/4)}{l}}$$

- Questa relazione vale per $\forall h \in \mathcal{H}$ e quindi è valida per ogni algoritmo di apprendimento, anche per quelli che non minimizzano il rischio empirico
- Talvolta la relazione è di scarsa utilità pratica (può fornire bound molto elevati)
- Tuttavia è di grande utilità concettuale perché suggerisce una strategia differente da ERM: minimizzare la somma del rischio empirico e della confidenza

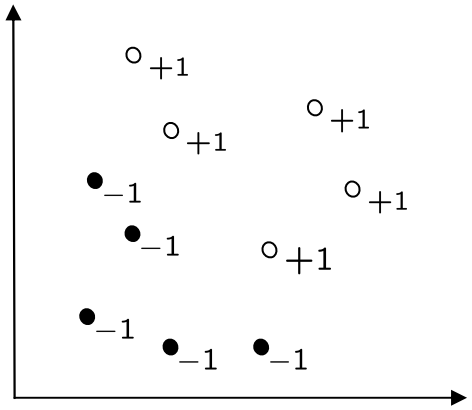
Structural Risk Minimization (SRM) principle



- Perché "strutturale"? Perché si dota l'insieme delle ipotesi di una struttura, cioè si considerano degli insiemi annidati (per i quali la VC-dimension è non decrescente) $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \ldots \subset \mathcal{H}_n \ldots$e ciò permette di minimizzare il bound tenendo conto di entrambi gli addendi.

Il problema della classificazione binaria

- E' dato un training set $\mathcal{S} = \{(x_1, y_1), \ldots, (x_l, y_l)\}$ in cui ogni vettore $x_i \in \mathcal{X} \subseteq R^n$ è assegnato ad una fra due classi, indicata da $y_i \in \mathcal{Y} = \{-1, +1\}$



- Le coppie (x_i, y_i) sono variabili casuali (indipendenti e identicamente distribuite) la cui distribuzione $F(x, y)$ è incognita.

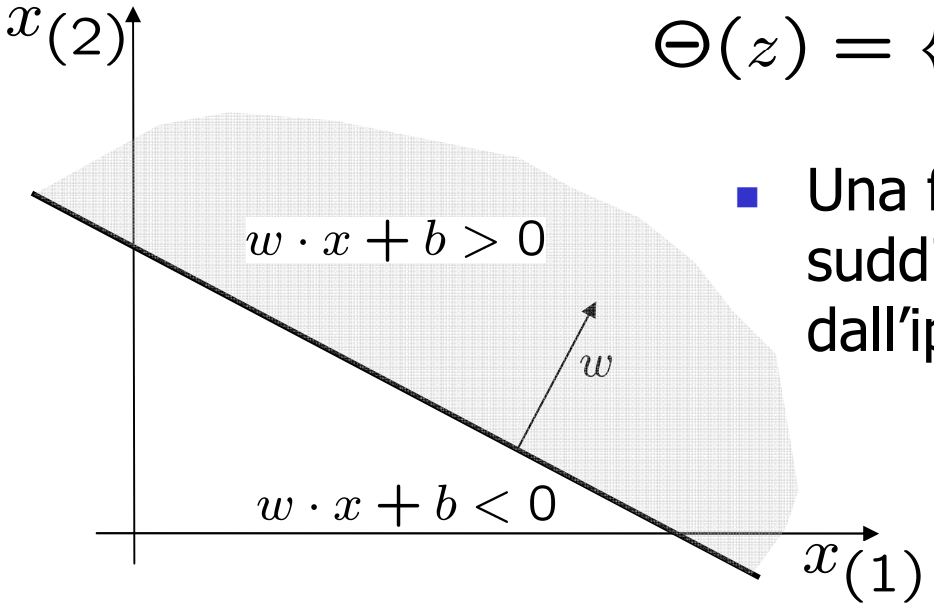
- Si cerca una *funzione di decisione* $h(x) : \mathcal{X} \rightarrow \mathcal{Y}$ in grado di classificare correttamente vettori *nuovi*, cioè vettori non appartenenti al training set

Funzione di decisione lineare (1)

- Chiamiamo (con un certo abuso) *lineare* una funzione di decisione del tipo

$$h(x) = \Theta(w \cdot x + b), \;\; w \in R^n, b \in R$$

$$\Theta(z) = \begin{cases} -1 & \text{se} \quad z < 0 \\ 1 & \text{se} \quad z \geq 0 \end{cases}$$

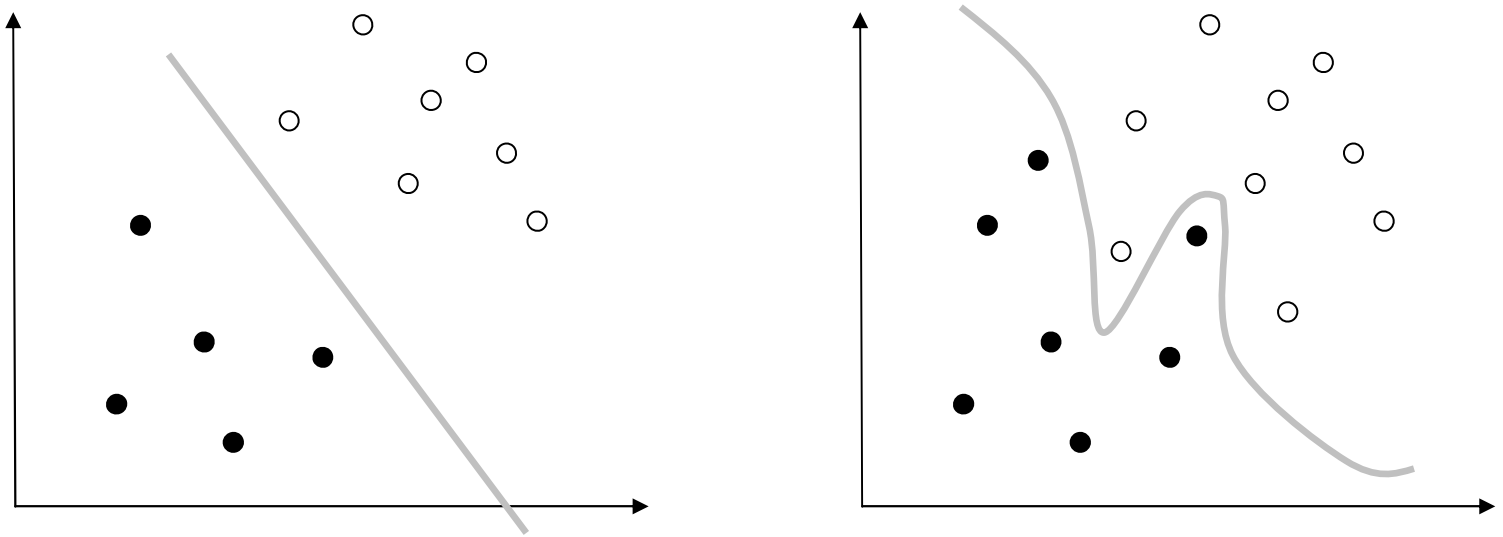


- Una funzione di decisione questo tipo suddivide R^n in due regioni delimitate dall'iperpiano di equazione

$$w \cdot x + b = 0$$

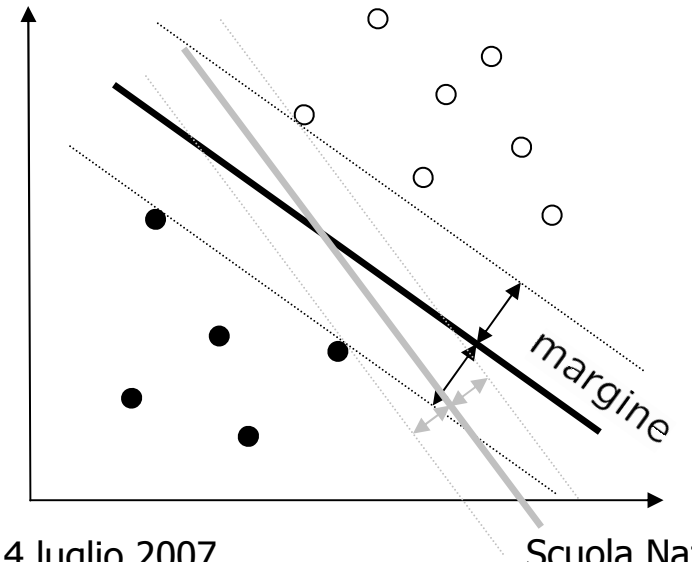
Training set linearmente (non) separabile

- Un training set si dice *linearmente separabile* se esiste un iperpiano che separa correttamente tutti i vettori
- In caso contrario il training set si dice *non linearmente separabile*



Iperpiano di massimo margine (caso separabile)

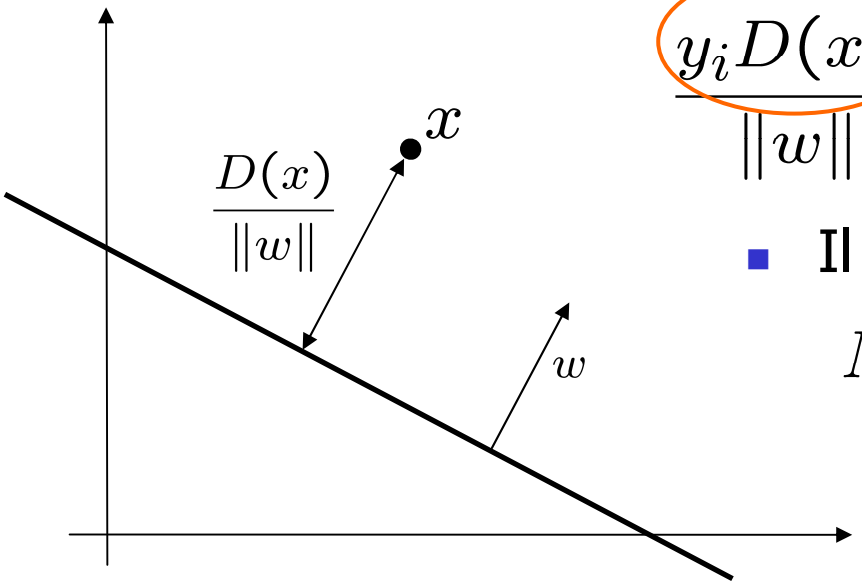
- Consideriamo un training set linearmente separabile
- Tra tutti i possibili iperpiani separatori ce n'è uno che massimizza il *margine*, cioè la distanza dal più vicino esempio (iperpiano di massimo margine)



- Come si calcola?
- Che proprietà ha?

Calcolo dell'iperpiano di massimo margine, caso separabile (1)

- Sia $D(x) \doteq w \cdot x + b = 0$ l'equazione dell'iperpiano
- La distanza (con segno) fra l'iperpiano e il generico x è $\frac{D(x)}{\|w\|}$
- Qualunque iperpiano separatore soddisfa le seguenti:



$$\frac{y_i D(x_i)}{\|w\|} \geq M > 0, \quad \forall i = 1, \ldots, l$$

strettamente positivo per ogni iperpiano separatore

- Il massimo margine è allora:

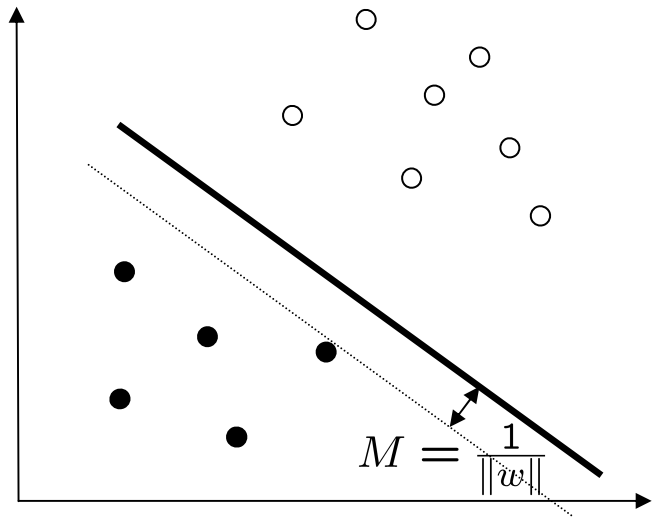
$$M^* = \max_{\|w\|=1, b} \min_i y_i D(x_i)$$

introdotto per risolvere la ridondanza

Calcolo dell'iperpiano di massimo margine, caso separabile (2)

- Anziché ricorrere al vincolo $\|w\| = 1$, dato un iperpiano avente margine M rispetto a un TS, si può scegliere fra le sue infinite rappresentazioni quella per cui $M\|w\| = 1$
- Iperpiani in questa forma si dicono *canonici* e sono tali che

$$\min_{i=1,\ldots,l} y_i D(x_i) = 1$$



- Se si restringe la ricerca agli iperpiani canonici, massimizzare il margine è equivalente a minimizzare $\|w\|$

Calcolo dell'iperpiano di massimo margine, caso separabile (3)

- Il problema diventa quindi

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

s.t.

$$y_i(w \cdot x_i + b) \geq 1, \ \forall i = 1, \ldots, l$$

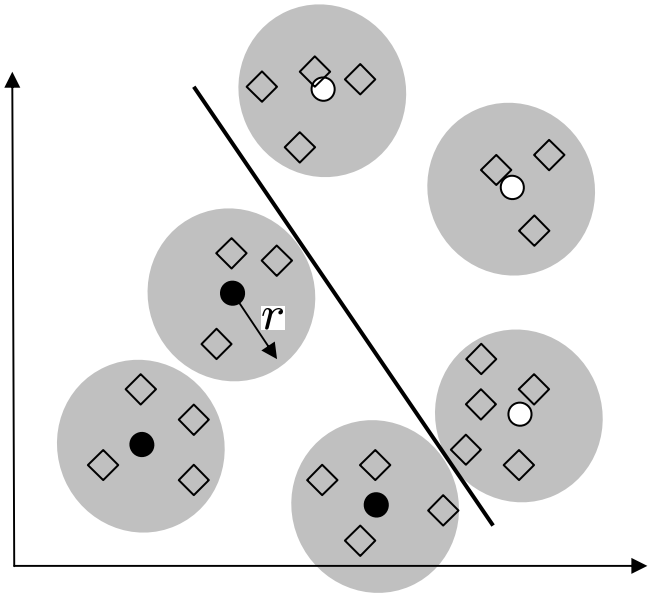
- Il massimo margine vale allora $M^* = \frac{1}{\|w^*\|}$

- Si tratta di un problema di ottimizzazione vincolata, con funzione obiettivo quadratica e vincoli lineari

Margine e rumore (1)

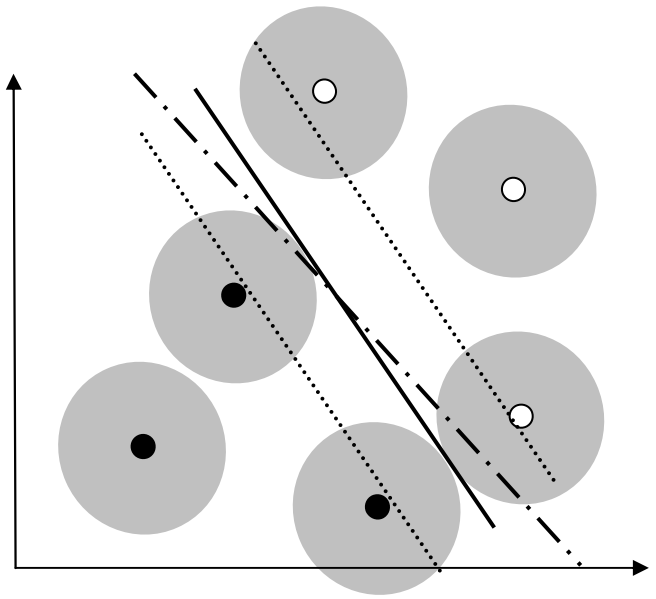
- Vari argomenti giustificano la scelta di massimizzare il margine. Vediamone alcuni.

- Pattern noise: se si suppone che i vettori di test ($\diamond$) siano generati dai vettori di training (x, y) secondo la forma $(x + \Delta x, y)$ dove $\|\Delta x\| \leq r$ allora è chiaro che se il margine è maggiore di r, tutti i vettori di test verranno correttamente classificati



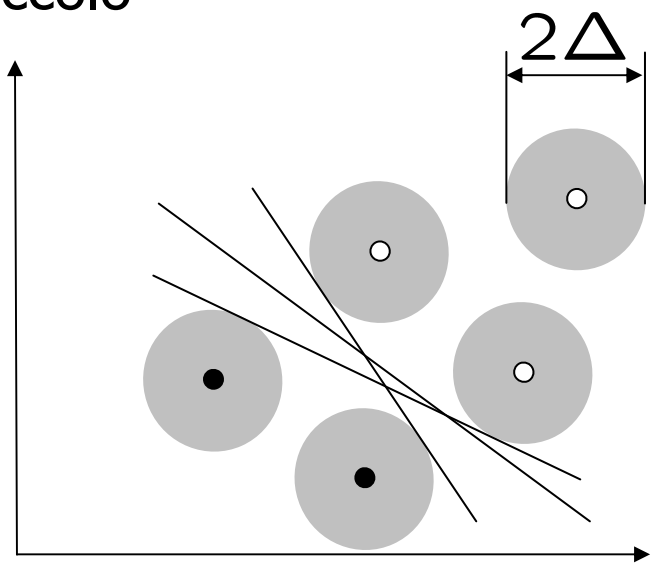
Margine e rumore (2)

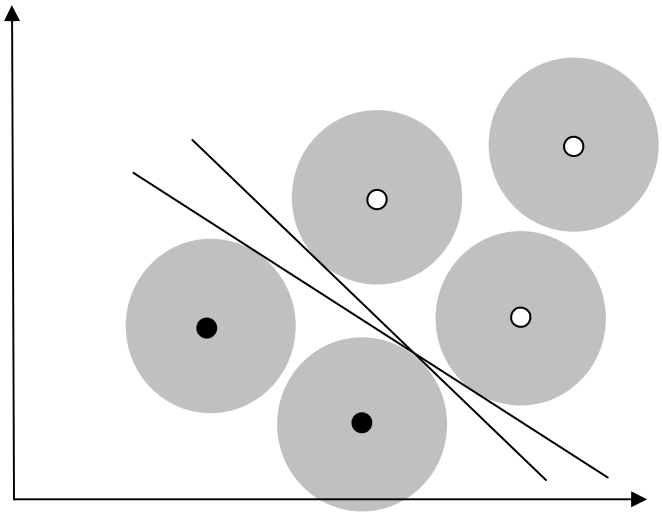
- Parameter noise: l'iperpiano di massimo margine massimizza lo "spessore" della striscia rappresentata in figura. Quindi è lecito aspettarsi che la corretta classificazione sia meno sensibile a perturbazioni (dovute ad esempio a errori numerici) dei parametri w e b rispetto a iperpiani separatori che non massimizzano il margine. (Questo vale solo se $\|x\| \leq r$)



Margine e VC-dimension (1)

- Dato un training set linearmente separabile $\mathcal{S}$ definiamo $\mathcal{H}_{\Delta,\mathcal{S}}$ l'insieme degli iperpiani separatori che hanno margine $M \geq \Delta$
- Al crescere di Δ l'insieme $\mathcal{H}_{\Delta,\mathcal{S}}$ diventa sempre più piccolo



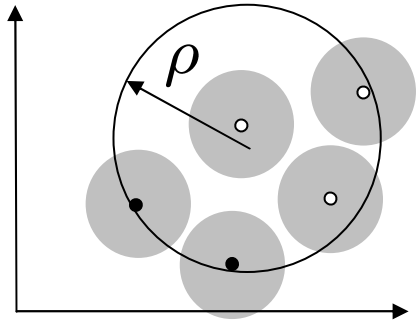


Margine e VC-dimension (2)

- Teorema [6]

  Dato $\mathcal{S} = \{(x_1, y_1), \ldots, (x_l, y_l)\}$, sia ρ il raggio della più piccola sfera contenente i vettori $x_i \in \mathcal{X} \subseteq R^n$. Allora vale disuguaglianza:

$$VC(\mathcal{H}_{\Delta,\mathcal{S}}) \leq \min\left(\left[\frac{\rho^2}{\Delta^2}\right], n\right) + 1$$



- Dunque attraverso il margine è possibile controllare la VC-dimension e quindi applicare il principio di SRM

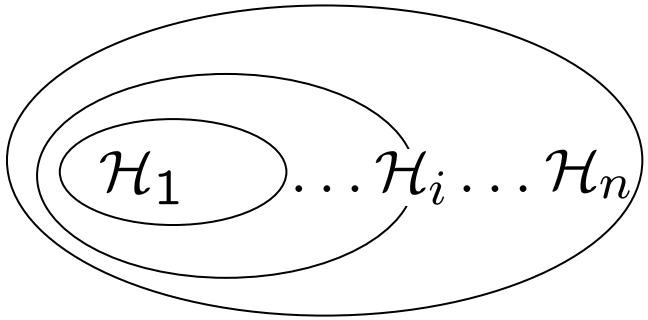
Margine e SRM (1)

- Infatti la famiglia di iperpiani canonici (per un dato TS)

$$\{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_i, \ldots\}$$

$$\mathcal{H}_i = \left\{ w \cdot x + b : \|w\|^2 \leq k_i \right\}, \;\; k_1 < k_2 \;\; \ldots$$

è tale che la VC-dimension è non decrescente e quindi è dotata di una struttura.



Margine e SRM (2)

- Ricordando l'espressione

$$R(h) \leq R_{emp}(h) + \phi\left(\frac{d}{l}, \frac{\ln \delta}{l}\right)$$

nullo se i dati sono linearmente separabili

minimizzato scegliendo l'iperpiano di massimo margine

e osservando che, essendo i dati linearmente separabili, il primo addendo a secondo membro può sempre essere reso nullo, si riconosce che massimizzare il margine (ovvero minimizzare d) significa minimizzare il secondo membro ovvero applicare il principio SRM .

Forma duale (1)

- Costruiamo la funzione Lagrangiana per il problema:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$\text{s.t.}$$
$$y_i(w \cdot x_i + b) \geq 1, \ \forall i = 1, \ldots, l$$

- Indicando con α il vettore dei moltiplicatori di Lagrange si ottiene

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i(w \cdot x_i + b) - 1\right]$$

- Tale funzione va minimizzata rispetto a w e b e massimizzata rispetto ai moltiplicatori $\alpha_i \geq 0$

Forma duale (2)

- Lagrangiana

$$L(w,b,\alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{l} \alpha_i \left[y_i (w \cdot x_i + b) - 1 \right]$$

- Condizioni di stazionarietà

$$\frac{\partial L(w,b,\alpha)}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$\frac{\partial L(w,b,\alpha)}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{l} y_i \alpha_i x_i$$

- Riscriviamo la Lagrangiana per sostituirvi le condizioni appena trovate

$$L(w,b,\alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{l} \alpha_i y_i (w \cdot x_i) - b \sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i$$

Forma duale (3)

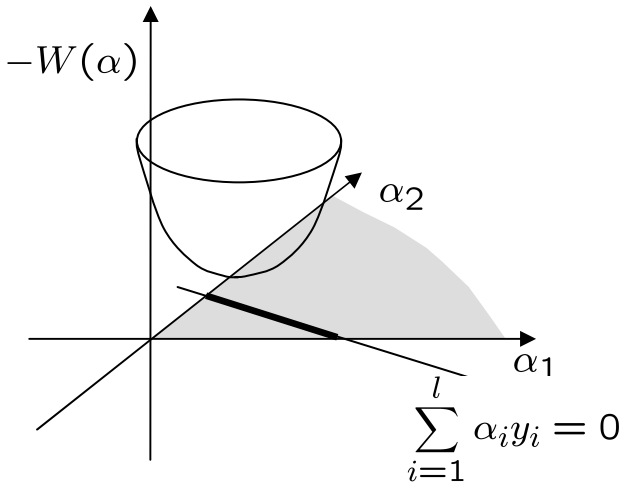
- Si ottiene il problema duale (equivalente al primale):

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

s.t.

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

$$\alpha_i \geq 0, \ \ \forall i = 1, \ldots, l$$



- Si tratta di un problema di programmazione quadratica (funzionale quadratico (convesso) e vincoli lineari)

Osservazioni (1)

- Sia α^* il vettore soluzione del problema duale. La seconda delle condizioni di stazionarietà è

$$w^* = \sum_{i=1}^{l} y_i \alpha_i^* x_i$$

- Cioè w^* è combinazione lineare dei vettori del training set

- Inoltre, dalla teoria dell'ottimizzazione [4], è noto che la soluzione del primale deve soddisfare le condizioni di Karush-Kuhn-Tucker:

$$\alpha_i^* \left[y_i (w^* \cdot x_i + b^*) - 1 \right] = 0, \quad \forall i = 1, \ldots, l$$

vincoli

- Quindi i moltiplicatori corrispondenti a vincoli non attivi sono nulli

⇓

- LA SOLUZIONE E' SPARSA

Osservazioni (2)

- I vettori x_i che corrispondono a moltiplicatori non nulli si dicono *vettori di supporto* (support vectors)

$$SV = \{x_i : \alpha_i^* > 0\}$$

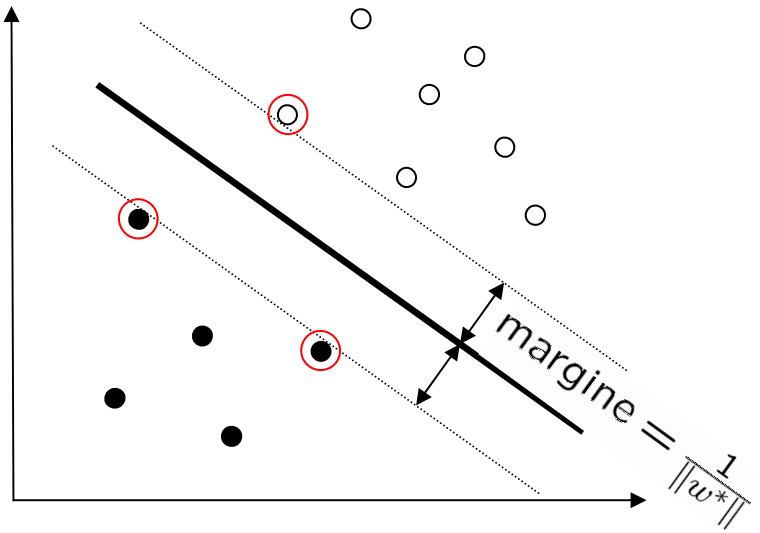
- Il vettore che individua (a meno del termine additivo b^*) l'iperpiano di massimo margine (iperpiano ottimo) è dunque combinazione lineare dei vettori di supporto

$$w^* = \sum_{SV} y_i \alpha_i^* x_i$$

- Per i SV sono attivi i vincoli:

$$y_i(w^* \cdot x_i + b^*) = 1$$

quindi <u>la loro distanza dall'iperpiano è pari al margine</u>

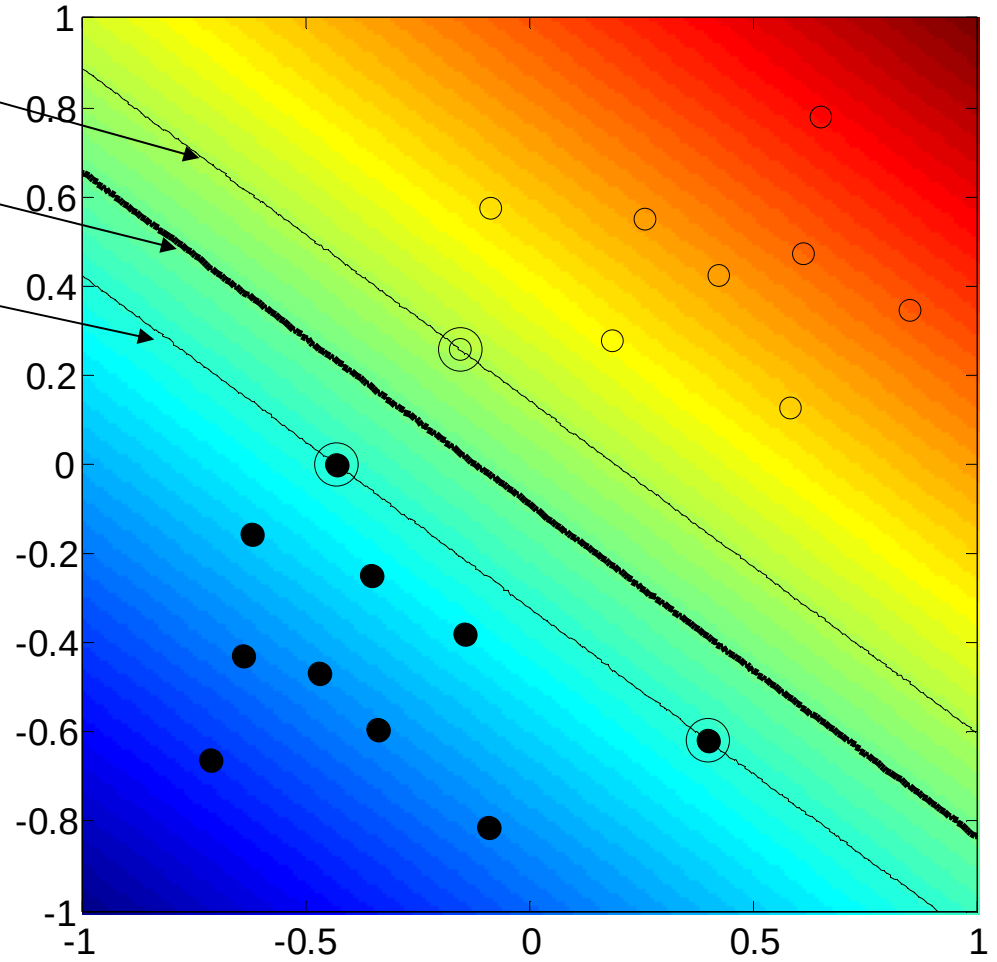


Esempio

$$w^* \cdot x + b^* = 1$$

$$w^* \cdot x + b^* = 0$$

$$w^* \cdot x + b^* = -1$$



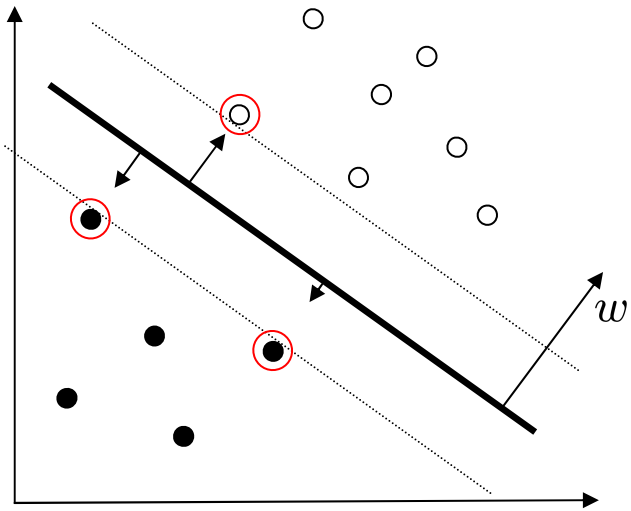
Interpretazione meccanica

- Ogni vettore di supporto esercita una forza di direzione $y_i \frac{w^*}{\|w^*\|}$ e di intensità α_i^*
- L'equilibrio alla traslazione è garantito dai vincoli $\sum_{i=1}^{l} \alpha_i y_i = 0$
- La coppia risultante è:

$$\sum_{SV} b_i \times F_i = \sum_{SV} \left(x_i - y_i \frac{w^*}{\|w^*\|} \right) \times \alpha_i^* y_i \frac{w^*}{\|w^*\|}$$

$$= \underbrace{\sum_{SV} \alpha_i^* y_i x_i}_{w^*} \times \frac{w^*}{\|w^*\|} = 0$$

quindi c'è equilibrio alla rotazione



Osservazioni (3)

- La funzione di decisione associata all'iperpiano ottimo è

$$h^*(x) = \Theta(w^* \cdot x + b^*)$$

permette classificare il generico nuovo vettore

dove b^* può essere calcolato sfruttando la condizione di KKT corrispondente ad un qualsiasi vettore di supporto x_i:

$$y_i(w^* \cdot x_i + b^*) = 1 \quad \Rightarrow \quad b^* = y_i - \sum_{SV} y_j \alpha_i^* (x_j \cdot x_i)$$

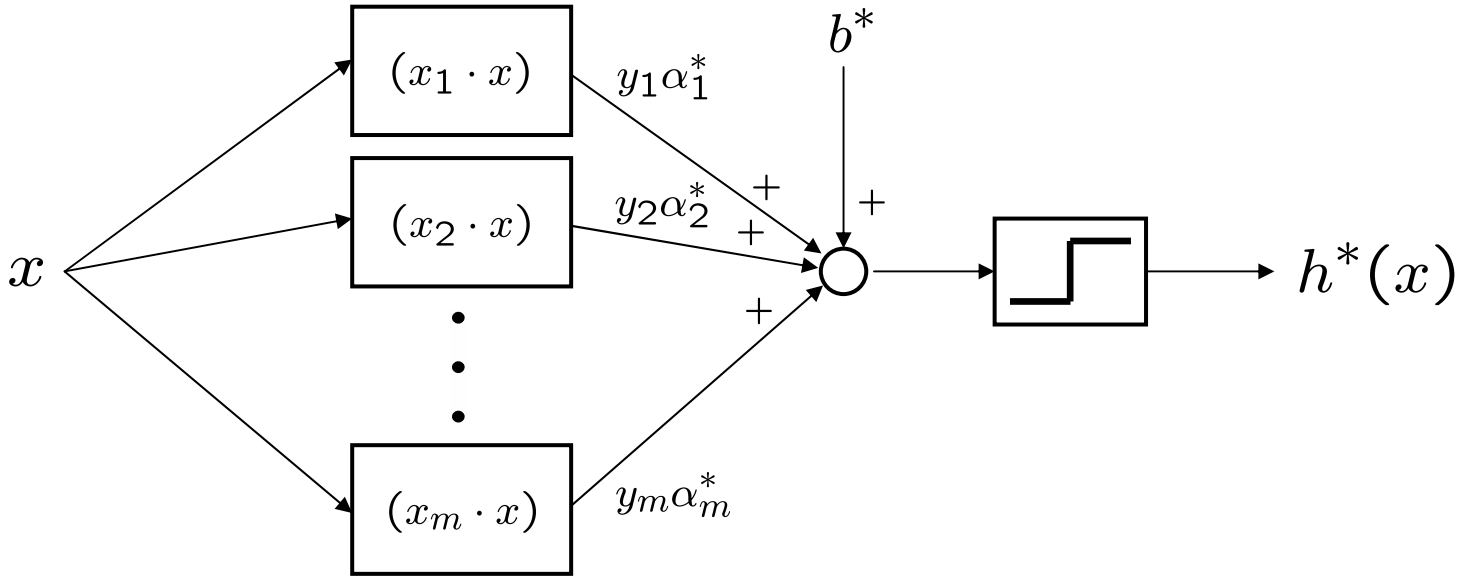
- Sostituendo nella funzione di decisione l'espressione di w^* si trova

$$h^*(x) = \Theta[(\sum_{SV} y_i \alpha_i^* x_i) \cdot x + b^*]$$

$$= \Theta[\sum_{SV} y_i \alpha_i^* (x_i \cdot x) + b^*]$$

Osservazioni (4)

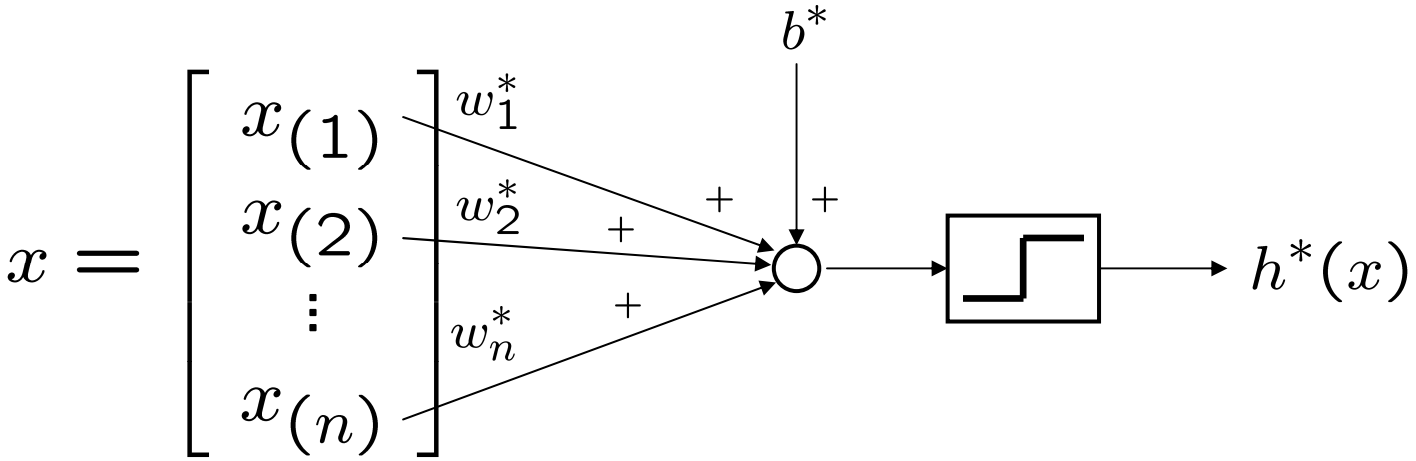
- Schematizzando, la rappresentazione duale è:



$$h^*(x) = \Theta[\sum_{SV} y_i \alpha_i^* (x_i \cdot x) + b^*]$$

Osservazioni (5)

- La rappresentazione primale invece è :



$$h^*(x) = \Theta[(w^* \cdot x) + b^*]$$

Osservazioni (6)

- Sia nel funzionale da minimizzare per risolvere il problema duale

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

che nella funzione di decisione

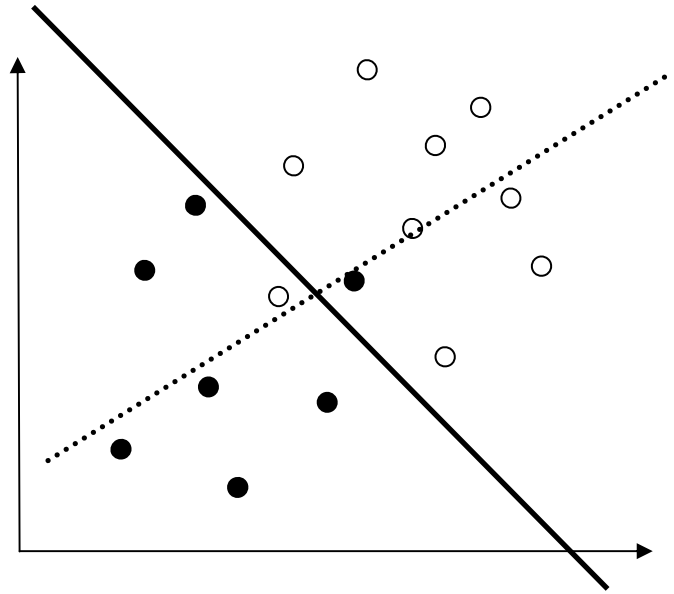
$$h^*(x) = \Theta[\sum_{SV} y_i \alpha_i^* (x_i \cdot x) + b^*]$$

i vettori $x \in \mathcal{X}$ compaiono solo all'interno di un prodotto scalare.

- Come vedremo, questo fatto è di fondamentale importanza per estendere il metodo al caso non lineare

Iperpiano ottimo nel caso non separabile (1)

- Se il TS non è linearmente separabile, nessun iperpiano potrà classificare correttamente tutti i punti (ovvero annullare il rischio empirico) ma è chiaro che vi sono iperpiani migliori di altri.



- Idea: minimizzare il numero di errori di classificazione e simultaneamente massimizzare il margine per i punti correttamente classificati

Iperpiano ottimo nel caso non separabile (2)

- Consentiamo la violazione dei vincoli, introducendo delle variabili di slack $\xi_i \geq 0, \ \forall i = 1, \ldots, l$

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \ \forall i = 1, \ldots, l$$

- Per σ sufficientemente piccolo, il numero di vincoli violati è dato dall'espressione:

$$\sum_{i=1}^{l} \xi_i^{\sigma}$$

- Il problema si può esprimere come segue (C è una costante):

$$\min \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{l} \xi_i^{\sigma}$$

Iperpiano ottimo nel caso non separabile (3)

- In generale, trovare un iperpiano che minimizzi il numero di errori di classificazione in un dato TS è un problema combinatorio difficile (NP-completo). Allora, per mantenere la trattabilità computazionale, si pone $\sigma = 1$

$$\min \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{l} \xi_i$$

- Ciò corrisponde a ricercare l'iperpiano che minimizza la *somma delle deviazioni* dai vincoli (anziché il *numero* di vincoli violati) massimizzando nel contempo il margine per i vettori correttamente classificati (1-*Norm Soft Margin Separating Hyperplane*)

Iperpiano ottimo nel caso non separabile (4)

- Il problema diventa quindi:

$$\min_{w,b} \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{l} \xi_i$$

$$\text{s.t.}$$

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \ \ \forall i = 1, \ldots, l$$

$$\xi_i \geq 0, \ \ \forall i = 1, \ldots, l$$

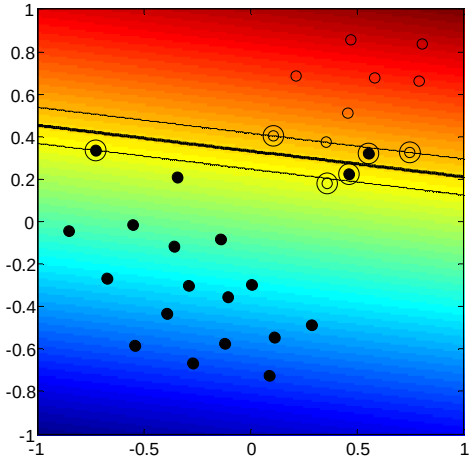
- C è una costante di *regolarizzazione*, che controlla il compromesso fra violazione dei vincoli e capacità dell'insieme delle ipotesi.

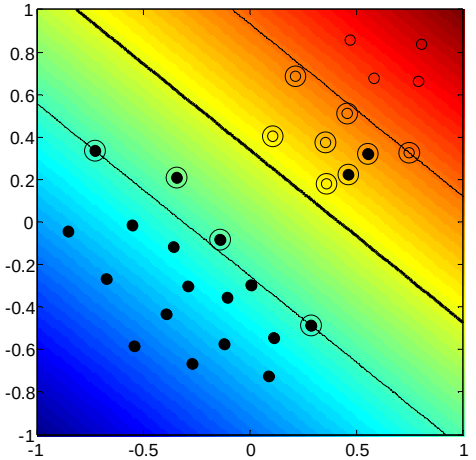
Il ruolo di C

$$\min_{w,b} \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{l} \xi_i$$

- C grande: il primo addendo conta poco, e quindi si otterranno soluzioni con pochi errori di classificazione ma piccolo margine (perché w può essere relativamente elevato)

- C piccolo: il primo addendo conta molto, e quindi si otterranno soluzioni con elevato margine (perché w non può essere elevato) ma molti errori di classificazione.





Formulazione duale

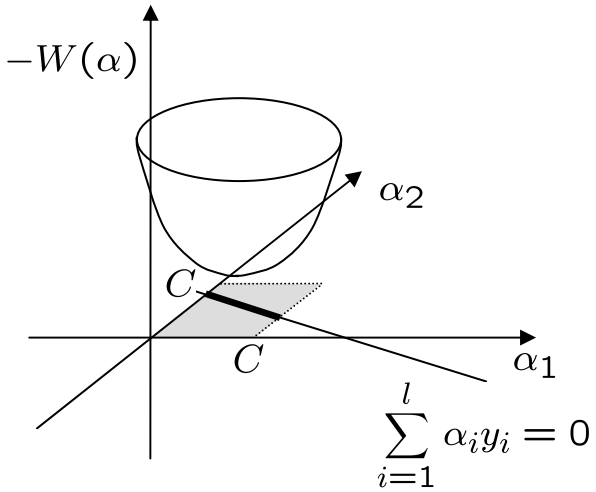
- Il problema duale ha la forma seguente:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

s.t.

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \quad \forall i = 1, \ldots, l$$



- Identico al caso separabile, tranne il vincolo sui moltiplicatori.

(Un)bounded support vectors

- Come nel caso separabile, anche in questo caso, la soluzione è sparsa
- I vettori di supporto sono quelli i cui moltiplicatori sono diversi da zero
- *Bounded support vectors*: quelli per cui $\alpha_i^* = C$
- *Unbounded support vectors*: quelli per cui $0 < \alpha_i^* < C$
- La funzione di decisione ha la stessa forma del caso separabile:

$$h^*(x) = \Theta[\sum_{SV} y_i \alpha_i^* (x_i \cdot x) + b^*]$$

si calcola come nel caso separabile, a partire da un vettore di supporto qualsiasi purché unbounded

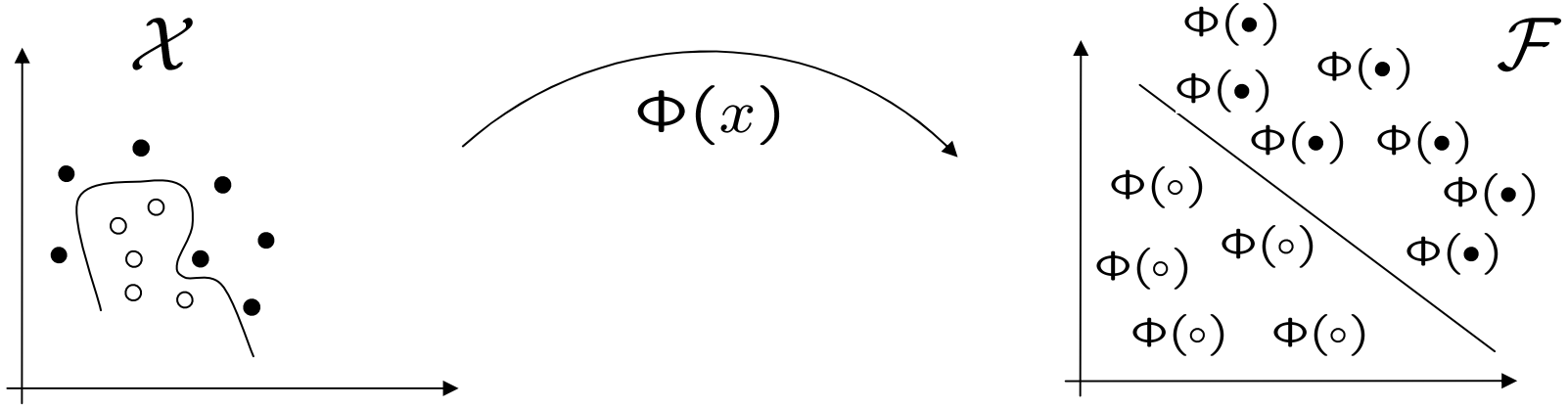
Bound per soft margin

- Esistono bound che giustificano da un punto di vista statistico il ricorso al Soft Margin Separating Hyperplane, ad esempio[5]:

> Dato l'insieme di funzioni $h(x) = \Theta(w \cdot x)$, con $\|w\| \leq \Delta^{-2}$, $\|x\| \leq \rho$. Sia ν la frazione di elementi del training set con margine inferiore a $\lambda/\|w\|$. Allora, indipendentemente dalla distribuzione $F(x, y)$, con probabilità $1 - \delta$, la probabilità che un vettore di test venga classificato erroneamente è minore di
>
> $$\nu + \sqrt{\frac{c}{l}\left(\frac{\rho^2}{\lambda^2\Delta^2}\ln^2 l + \ln(1/\delta)\right)}$$
>
> dove c è una costante.

Macchine non lineari: feature mapping (1)

- In molti casi anche il soft margin è inadeguato poiché il legame fra i dati è non lineare e dunque nessuna legge lineare può "catturare" la regolarità dei dati, come nell'esempio seguente



- E' possibile però che un'opportuna trasformazione renda i dati linearmente separabili

- $\mathcal{X}$ spazio di ingresso (o degli attributi)
- $\mathcal{F}$ spazio delle features

Macchine non lineari: feature mapping (2)

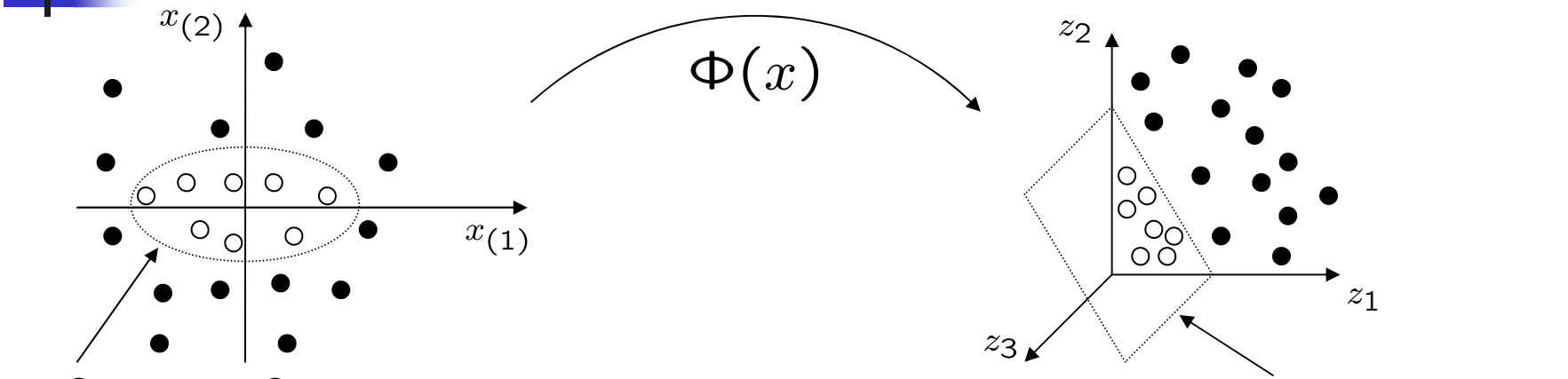
- Allora si può pensare ad un algoritmo in due passi
  - Trasformazione non lineare (feature mapping) $\Phi(x) : \mathcal{X} \rightarrow \mathcal{F}$
  - Classificazione lineare nello spazio delle features

- Cioè ricercare un iperpiano separatore nello spazio $\mathcal{F}$ a partire dal training set trasformato:

$$\mathcal{S} = \{(\Phi(x_1), y_1), \ldots, (\Phi(x_l), y_l)\}$$

$$\text{dove } (\Phi(x_i), y_i) \in \mathcal{F} \times \mathcal{Y}$$

Esempio



$$ax_{(1)}^2 + bx_{(2)}^2 + cx_{(1)}x_{(2)} = k$$

$$az_1 + bz_2 + cz_3 = k$$

$$\Phi(x) = \Phi(x_{(1)}, x_{(2)}) = (z_1, z_2, z_3) = (x_{(1)}^2, x_{(2)}^2, x_{(1)}x_{(2)})$$

- L'ellisse in due dimensioni viene mappata in un piano in 3 dimensioni. Il coefficiente c è in questo caso pari a zero perché l'ellisse ha gli assi paralleli agli assi coordinati quindi tutti i punti giacciono sul piano (z_1, z_2)

Macchine polinomiali (1)

- Nell'esempio precedente, le features sono costituite da tutti i monomi di secondo grado che si possono ottenere moltiplicando fra loro le componenti del vettore di ingresso.

$$\Phi(x) = \Phi(x_{(1)}, x_{(2)}) = (z_1, z_2, z_3) = (x_{(1)}^2, x_{(2)}^2, x_{(1)}x_{(2)})$$

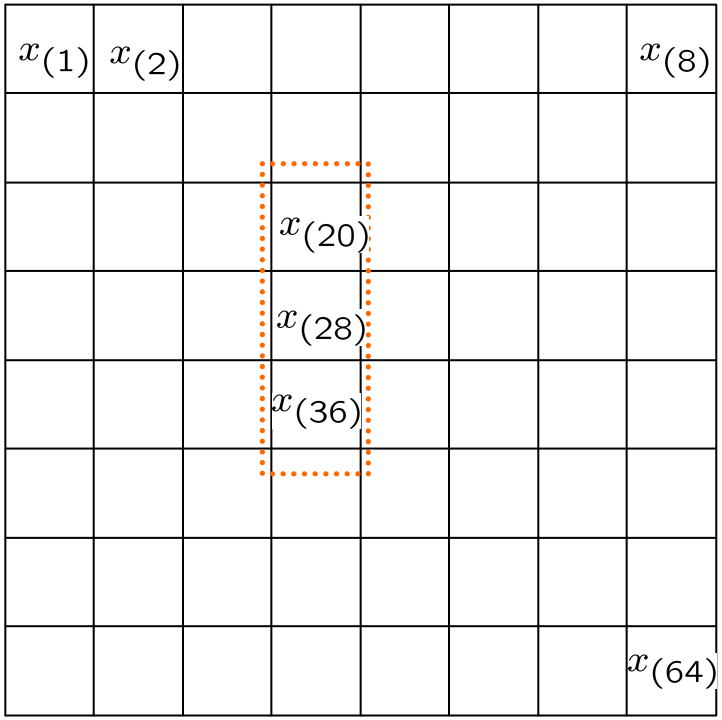
- Un iperpiano nello spazio delle features corrisponde, nello spazio di ingresso, ad una superficie di livello di un polinomio:

$$az_1 + bz_2 + cz_3 = k \qquad ax_{(1)}^2 + bx_{(2)}^2 + cx_{(1)}x_{(2)} = k$$

- Macchine di questo tipo si dicono *classificatori polinomiali*.

Macchine polinomiali (2)

- Perché usare i monomi come features?
- Perché il prodotto fra due o più componenti del vettore di ingresso può essere interpretato come una sorta di AND logico.



- Nel caso dell'immagine 8*8 qui a sinistra, il monomio $x_{(20)}x_{(28)}x_{(36)}$ avrà un valore elevato solo se tutti i tre pixel hanno alta intensità e quindi questa feature codifica la condizione "tutti i tre pixel 20,28,36 hanno alta intensità". Questa condizione è tanto più vera per un dato ingresso $\bar{x} \in [0,1]^{64}$ quanto più $\bar{x}_{(20)}\bar{x}_{(28)}\bar{x}_{(36)}$ ha valore elevato.

Macchine polinomiali (3)

- Se $x \in R^n$, il numero di monomi di grado d ottenuti moltiplicando fra loro le componenti di x è pari a

$$\binom{n+d-1}{d}$$

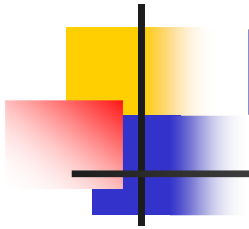
- Il numero di monomi di grado $\leq d$ ottenuti moltiplicando fra loro le componenti di x è pari a

$$\binom{n+d}{d}$$

possibili difficoltà computazionali nel risolvere un problema di ottimizzazione in uno spazio così grande!

- Esempio non irrealistico: immagine 16*16, monomi di quinto grado:

$$\binom{256+5-1}{5} \approx 10^{10}$$



Due esigenze da soddisfare

- Ricorrere a una trasformazione non lineare (feature mapping) può essere efficace a patto di soddisfare due esigenze:
  - Garantire buona capacità di generalizzazione
  - Mantenere la trattabilità computazionale

- La prima esigenza viene soddisfatta attraverso la massimizzazione del margine (i bound sul rischio non dipendono dalla dimensione dello spazio)
- La seconda attraverso una trasformazione implicita ("kernel trick")

Trasformazione implicita (1)

- Tenendo conto delle trasformazione $\Phi(\cdot)$, la funzione di decisione assume la forma

$$h(x) = \Theta(w \cdot \Phi(x) + b) = \Theta(\sum_{i=1}^{N} w_i \Phi_i(x) + b)$$

dove N indica la dimensione dello spazio delle features ($w \in R^N$)

- Il problema primale è un problema di ottimizzazione in uno spazio di dimensione N

$$\min_{w,b} \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{l} \xi_i$$

s.t.

$$y_i(w \cdot \Phi(x_i) + b) \geq 1 - \xi_i, \ \forall i = 1, \ldots, l$$

$$\xi_i \geq 0, \ \forall i = 1, \ldots, l$$

  - Se N è molto grande sorgono difficoltà computazionali sia per il problema di ottimizzazione, sia per valutare funzione di decisione

Trasformazione implicita (2)

- La formulazione duale è invece:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \underbrace{(\Phi(x_i) \cdot \Phi(x_j))}_{\text{prodotto scalare in } \mathcal{F}}$$

s.t.

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \ \forall i = 1, \ldots, l$$

- Le immagini dei vettori di ingresso secondo la mappa $\Phi(\cdot)$ compaiono solo entro prodotti scalari.
- Se è possibile calcolare tali prodotti direttamente (senza calcolare i vettori di $\mathcal{F}$) è possibile aggirare le difficoltà computazionali

- La funzione di decisione è

$$h(x) = \Theta[\sum_{i=1}^{l} y_i \alpha_i \underbrace{(\Phi(x_i) \cdot \Phi(x))}_{\text{prodotto scalare in } \mathcal{F}} + b]$$

Kernel (1)

- Si dice *kernel* una funzione $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{R}$ tale che

$$k(x, x') = \Phi(x) \cdot \Phi(x'), \quad \forall x, x' \in \mathcal{X}$$

  dove Φ è una mappa da $\mathcal{X}$ a uno spazio $\mathcal{F}$ (dotato di prodotto scalare)

- Il kernel definisce in maniera implicita la trasformazione Φ ed è tutto ciò che serve per individuare l'iperpiano ottimo nello spazio $\mathcal{F}$.
- L'uso del kernel permette di evitare il calcolo esplicito delle features, ovvero i vettori $\Phi(x)$ (→ "kernel trick")

Esempio

- Esempio classico: $x = (x_{(1)}, x_{(2)})$ $z = (z_{(1)}, z_{(2)})$

$$
\begin{aligned}
(x \cdot z)^2 &= [(x_{(1)}, x_{(2)}) \cdot (z_{(1)}, z_{(2)})]^2 \\
&= [x_{(1)} z_{(1)} + x_{(2)} z_{(2)}]^2 \\
&= x_{(1)}^2 z_{(1)}^2 + 2 x_{(1)} x_{(2)} z_{(1)} z_{(2)} + x_{(2)}^2 z_{(2)}^2 \\
&= (x_{(1)}^2, \sqrt{2} x_{(1)} x_{(2)}, x_{(2)}^2) \cdot (x_{(1)}^2, \sqrt{2} x_{(1)} x_{(2)}, z_{(2)}^2) \\
&= \Phi(x) \cdot \Phi(z)
\end{aligned}
$$

features monomiali di grado 2 (pesate)

dove $\Phi : (x_{(1)}, x_{(2)}) \rightarrow (x_{(1)}^2, \sqrt{2} x_{(1)} x_{(2)}, x_{(2)}^2)$

Kernel (2)

- Dato $k(x, x')$, l'iperpiano ottimo nello spazio delle features si trova risolvendo:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

s.t.

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \ \forall i = 1, \ldots, l$$

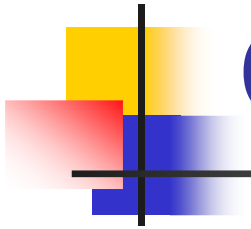
- La funzione di decisione è

$$h(x) = \Theta[\sum_{i=1}^{l} y_i \alpha_i k(x_i, x) + b]$$

- NB: la matrice

  $[K_{ij}] = [k(x_i, x_j)], \ i, j = 1, \ldots, l$

  si dice matrice Grammiana ed è il nucleo della forma quadratica che compare nel funzionale di costo.
- se $k(x, x')$ è un kernel, si può dimostrare che la matrice Grammiana è semidefinita positiva



Caratterizzazione

- In alcuni casi, il kernel si trova a partire dalla mappa Φ (come nel caso, trattato più avanti, di kernel che generano splines)
- Nella maggior parte dei casi invece si fissa direttamente il kernel, definendo implicitamente la trasformazione Φ (che al limite può essere anche incognita)
- Per far ciò è utile avere delle condizioni che permettano di dire se una funzione $k(x, x')$ è un kernel

Insieme degli ingressi finito (1)

- Anzitutto, poiché il prodotto scalare è commutativo, il kernel deve essere simmetrico:

$$k(x,z) = k(z,x), \ \forall x,z \in \mathcal{X}$$

- Sia $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$
- Data la funzione simmetrica $k(x,z)$ resta definita una matrice (simmetrica) Grammiana

$$K = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ \vdots & \vdots & \vdots & \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix} \in R^{n \times n}$$

Insieme degli ingressi finito (2)

- Dato un insieme degli ingressi finito $\mathcal{X}$, si può dimostrare che una funzione simmetrica

$$k(x,z) : \mathcal{X} \times \mathcal{X} \rightarrow R$$

è un kernel se e solo se la matrice Grammiana

$$K = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ \vdots & \vdots & \vdots & \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix} \in R^{n \times n}$$

è semidefinita positiva (ha tutti gli autovalori non negativi)

Insieme degli ingressi finito (3)

- Il motivo si può intuire sulla base di quanto segue:
  per la simmetria di K , vale la seguente, con $\Lambda = \mathrm{diag}\{\lambda_i\}$

$$K = V^T \Lambda V$$

$$K(x_i, x_j) = K_{ij} = \sum_{p=1}^{n} (V^T)_{ip} \Lambda_{pp} V_{pj} = \sum_{p=1}^{n} V_{pi} \lambda_p V_{pj} =$$

(se $\lambda_i \geq 0$, ovvero K semidefinita positiva)

$$= (\underbrace{\sqrt{\lambda_1} V_{1i}, \sqrt{\lambda_2} V_{2i}, \ldots, \sqrt{\lambda_n} V_{ni}) \cdot (\sqrt{\lambda_1} V_{1j}, \sqrt{\lambda_2} V_{2j}, \ldots, \sqrt{\lambda_n} V_{nj}}) =$$

$$= \Phi(x_i) \cdot \Phi(x_j)$$

E' un prodotto scalare fra features, a patto di definire la mappa delle features così:

$$\Phi(x_i) = (\sqrt{\lambda_1} V_{1i}, \sqrt{\lambda_2} V_{2i}, \ldots, \sqrt{\lambda_n} V_{ni}),\ i = 1, \cdots, n$$

Teorema di Mercer

- Una funzione continua e simmetrica $k(x,z) : \mathcal{X} \times \mathcal{X} \rightarrow R$ con $\mathcal{X}$ sottoinsieme compatto di R^n può essere espansa nella serie a coefficienti positivi $\lambda_i > 0$

$$k(x,z) = \sum_{i=1}^{\infty} \lambda_i \Phi(x) \Phi(z)$$

se e solo se vale la condizione

$$\int_{\mathcal{X} \times \mathcal{X}} k(x,z) g(x) g(z) dx dz > 0 \qquad \forall g : g \in \mathcal{L}_2, \ g \neq 0$$

- Dunque questa condizione garantisce che $k(x,z)$ sia il prodotto scalare in un qualche spazio delle features.

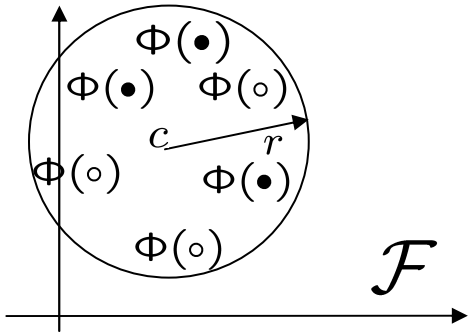
Matrice Grammiana e convessità

- Si può dimostrare che la condizione del teorema precedente è equivalente a richiedere che la matrice Grammiana formata a partire da qualsiasi sottoinsieme finito $\{x_1, x_2, \ldots, x_p\}$ dell'insieme $\mathcal{X}$ sia semidefinita positiva (il che garantisce la convessità del problema di ottimizzazione)

- Attraverso il kernel si possono calcolare in maniera implicita varie quantità nello spazio delle feature. Una particolarmente interessante è il raggio della più piccola sfera contenente le features (perché legata al bound sulla VC-dimension).

- Per approfondimenti su kernel: [5] e riferimenti in esso contenuti

Calcolo della più piccola sfera contenente le features (1)



- Costruiamo la funzione Lagrangiana per il problema:

$$\min_{r,c} \|r\|_2^2$$

s.t.

$$\|r\|_2^2 - \|\Phi(x_i) - c\|_2^2 \geq 0, \ \ \forall i = 1, \ldots, l$$

- Indicando con $\lambda_i \geq 0$ i moltiplicatori di Lagrange si ottiene

$$L(r, c, \lambda) = \|r\|_2^2 - \sum_{i=1}^{l} \lambda_i (\|r\|_2^2 - \|\Phi(x_i) - c\|_2^2)$$

Calcolo della più piccola sfera contenente le features (2)

- Lagrangiana

$$L(\rho, c, \lambda) = \|\rho\|_2^2 - \sum_{i=1}^{l} \lambda_i (\|\rho\|_2^2 - \|\Phi(x_i) - c\|_2^2)$$

- Condizioni di stazionarietà

$$\frac{\partial L(\rho, c, \lambda)}{\partial \rho} = 0 \quad \Rightarrow \quad \sum_{i=1}^{l} \lambda_i = 1$$

$$\frac{\partial L(\rho, c, \lambda)}{\partial c} = 0 \quad \Rightarrow \quad c = \sum_{i=1}^{l} \lambda_i \Phi(x_i)$$

- Riscriviamo la Lagrangiana per sostituirvi le condizioni appena trovate

$$L(\rho, c, \lambda) = (\rho \cdot \rho) \left(1 - \sum_{i=1}^{l} \lambda_i\right) + \sum_{i=1}^{l} \lambda_i [(\Phi(x_i) - c) \cdot (\Phi(x_i) - c)]$$

Calcolo della più piccola sfera contenente le features (3)

- Si ottiene il problema (duale)

$$\max_{\lambda} W(\lambda) = \sum_{i=1}^{l} \lambda_i (\underbrace{\Phi(x_i)\cdot\Phi(x_i)}_{k(x_i,x_i)}) - \sum_{i,j} \lambda_i \lambda_j y_i y_j (\underbrace{\Phi(x_i)\cdot\Phi(x_j)}_{k(x_i,x_j)})$$

s.t.

$$\sum_{i=1}^{l} \lambda_i = 1$$

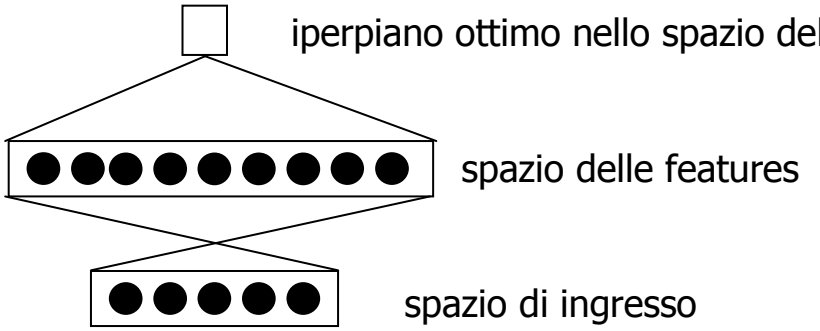
$$\lambda_i \geq 0, \ \ \forall i = 1, \ldots, l$$

- Si risolve attraverso il "kernel trick" senza usare Φ e poi si calcola il raggio da un vincolo attivo attraverso KKT (e kernel)

$$\rho^2 = (\Phi(x_i) - c) \cdot (\Phi(x_i) - c)$$ ← si può scrivere in funzione del kernel!

Support Vector Machines

- Mappano i vettori di ingresso x in uno spazio $\mathcal{F}$ di dimensione tipicamente molto elevata (la mappa è fissa e implicitamente definita dal kernel)
- Individuano in tale spazio l'iperpiano ottimo

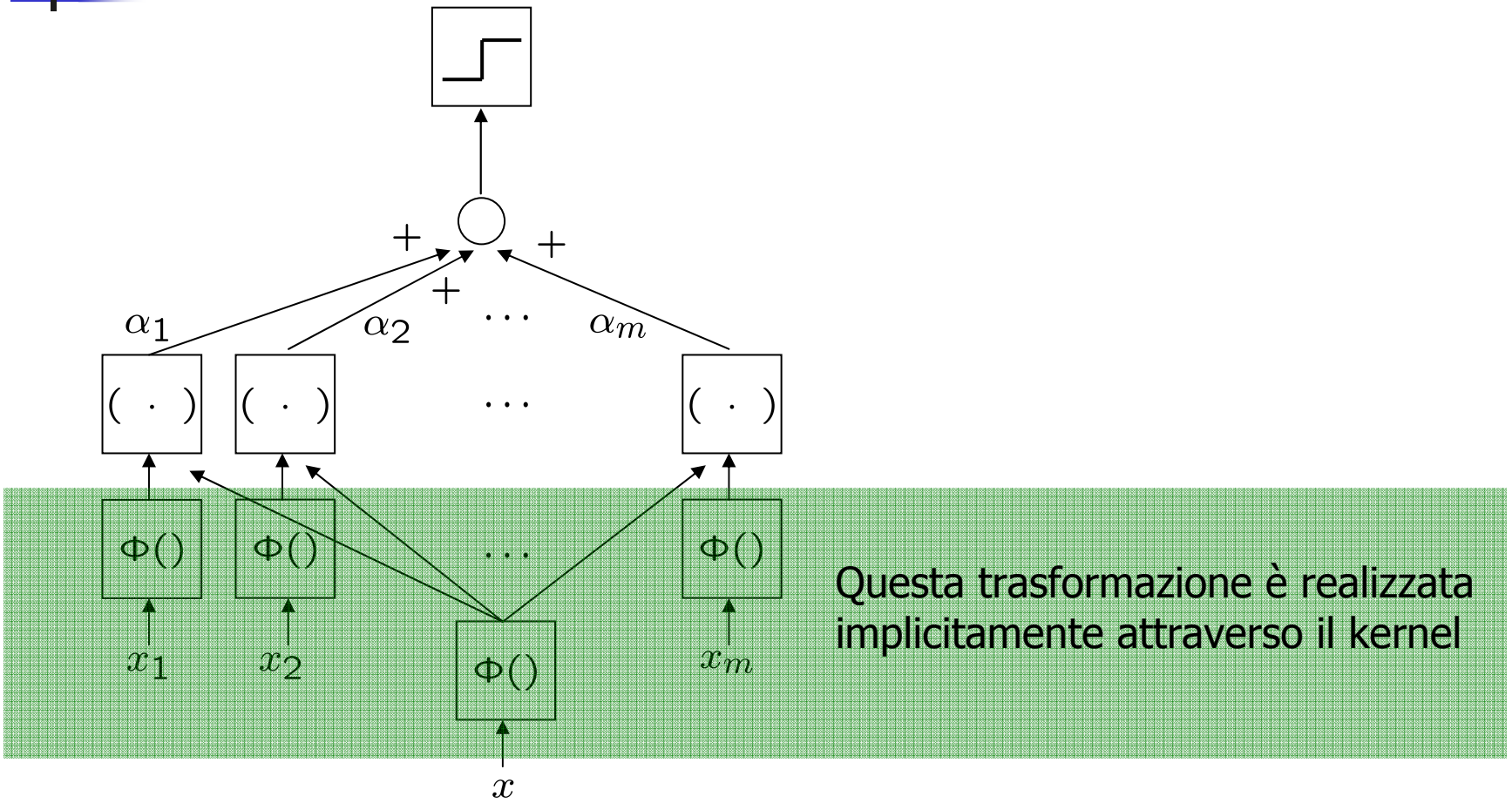


si riferiscono allo spazio delle features!

$$d \leq \min\left(\left[\frac{\rho^2}{\Delta^2}\right], n\right) + 1$$

- Capacità di generalizzazione garantita dalla massimizzazione del margine
- Trattabilità computazionale garantita dal ricorso al kernel

Architettura



Generalizzazione e #SV (1)

- Una tecnica utilizzata per stimare la capacità di generalizzazione è il metodo detto "leave-one-out". Consiste in:
  - rimuovere dal training set il pattern x_i
  - effettuare l'addestramento
  - testare il classificatore ottenuto sul pattern rimosso
  - ripetere la procedura per ogni i
  - Stimare l'errore di test attraverso la media degli errori ottenuti

- Per la particolare struttura delle SVM, l'esito di un test basato sul criterio leave-one-out è legato al numero di vettori di supporto

Generalizzazione e #SV (2)

- Nel caso di un classificatore a vettori di supporto, rimuovendo un pattern possono verificarsi i casi seguenti:

  a) il pattern rimosso non era SV del classificatore "completo": la soluzione, cioè l'iperpiano separatore, non cambia e il pattern viene classificato correttamente (errore = 0)

  b) il pattern rimosso era SV ma la soluzione non cambia (errore = 0)

  c) il pattern rimosso era SV e la soluzione cambia; in questo caso il vincolo $y_i(w \cdot \Phi(x_i) + b) \geq 1$ non è più rispettato e quindi può accadere che il pattern sia classificato scorrettamente (errore = 1)

- Nel caso pessimo, la rimozione di ciascuno dei SV comporta un errore e quindi $\frac{\#SV}{l}$ è un indice della capacità di generalizzazione.

  Più precisamente: $EP_{err} \leq E\left(\frac{\#SV}{l}\right)$

Esempi di kernel per SVM (1)

- Qualunque funzione che soddisfi il criterio di Mercer può fungere da kernel. Alcuni kernel molto usati sono i seguenti:
  - Polinomiale omogeneo: le features sono tutti e soli i monomi di grado d ottenuti moltiplicando fra loro le componenti del vettore di ingresso (pesate)

$$k(x, x') = (x \cdot x')^d$$

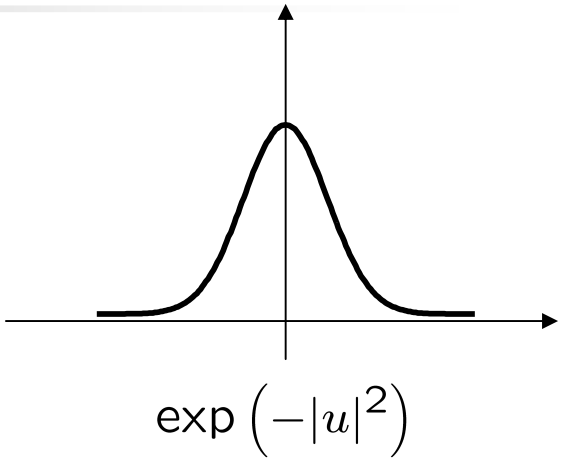
  - Polinomiale omogeneo: le features sono tutti e soli i monomi di grado $\leq d$ ottenuti moltiplicando fra loro le componenti del vettore di ingresso (pesate)

$$k(x, x') = (x \cdot x' + c)^d$$

Esempi di kernel per SVM (2)

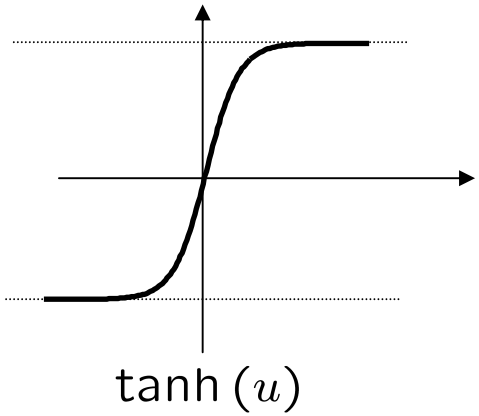
- Gaussiano

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{c}\right)$$



- Sigmoidale (non per tutti valori di γ e θ soddisfa le condizioni del teorema di Mercer)

$$k(x, x') = \tanh(\gamma(x \cdot x') + \theta)$$



Kernel a base radiale

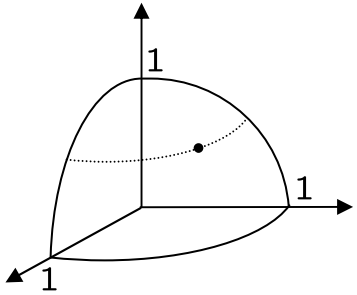
- Il kernel Gaussiano è un caso particolare di kernel a base radiale:

$$k(x, x') = f\left(d(x, x')\right)$$

dove $d(x, x')$ è una metrica e $f : \mathcal{X} \times \mathcal{X} \rightarrow R^{+}$

- I kernel a base radiale sono invarianti per traslazione:

$$k(x - x_0, x' - x_0) = k(x, x')$$



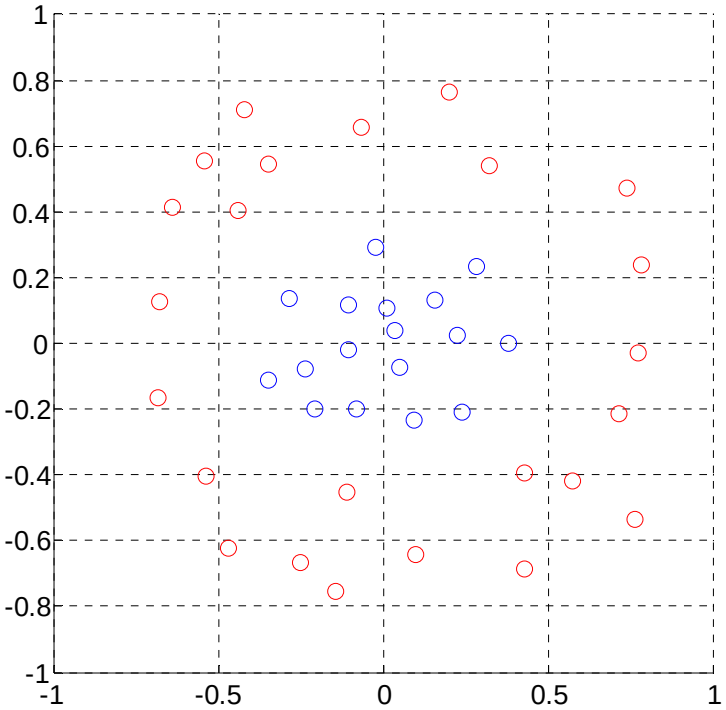
- In aggiunta, il kernel Gaussiano gode delle proprietà:
  - lunghezza unitaria dei vettori immagine: $k(x, x) = 1 \ \forall x$
  - collocazione nello stesso ortante: $k(x, x') > 0 \ \forall x, x'$
  - dimensione infinita dello spazio delle features:

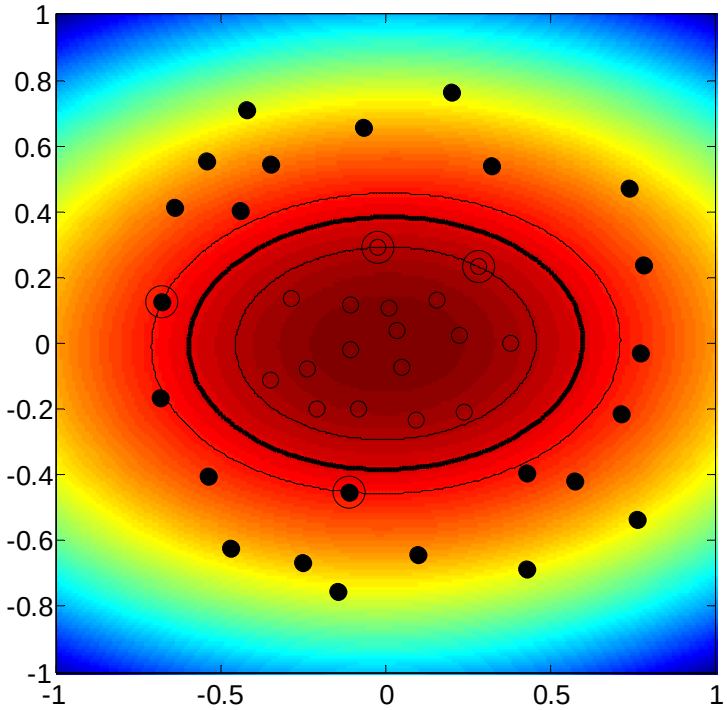
$$[k(x_i, x_j)] = [\Phi(x_i) \cdot \Phi(x_j)] \text{ ha rango } \max \forall \{x_1, x_2, \ldots, x_m\}, \ \forall m$$

Esempio 1

- Kernel polinomiale omogeneo di secondo grado

$$k(x, x') = (x \cdot x')^2$$

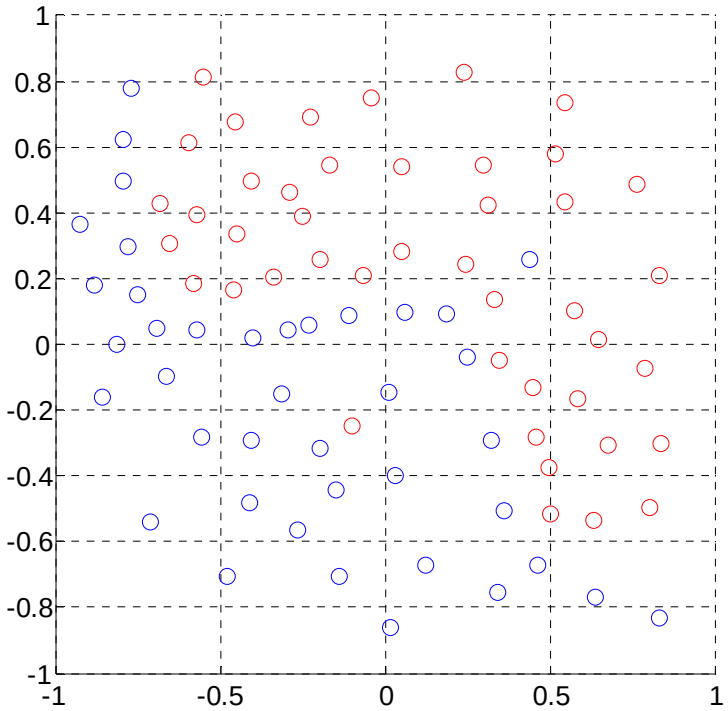


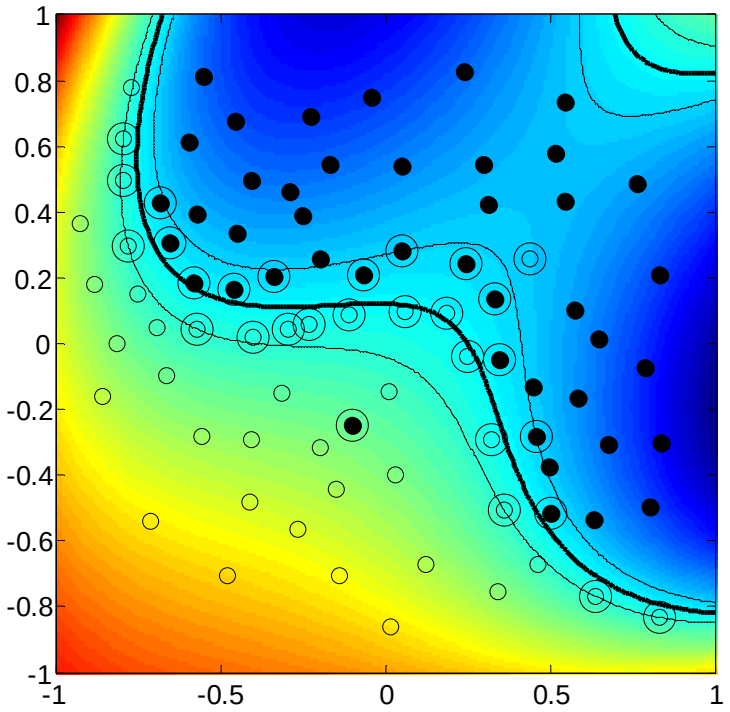


Esempio 2

- Kernel polinomiale non omogeneo di quarto grado

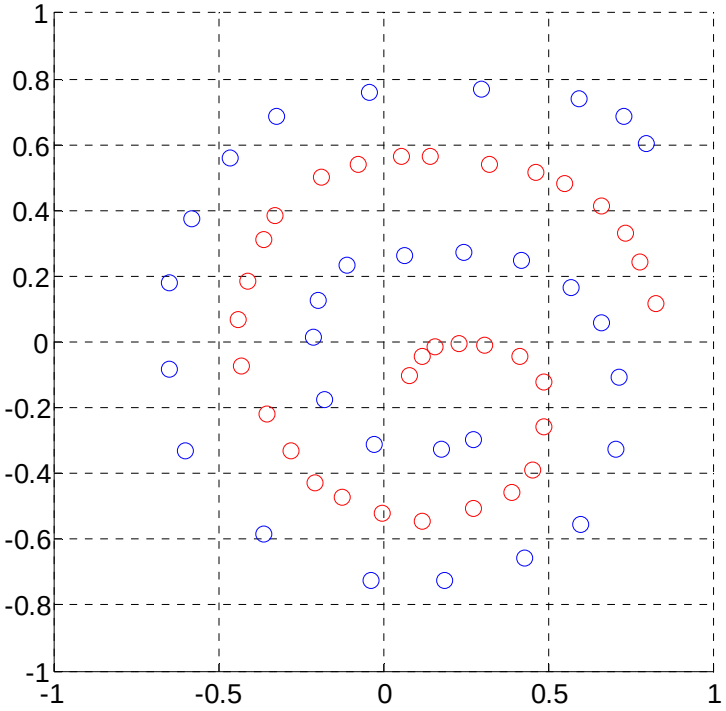
$$k(x, x') = (x \cdot x' + 1)^4$$

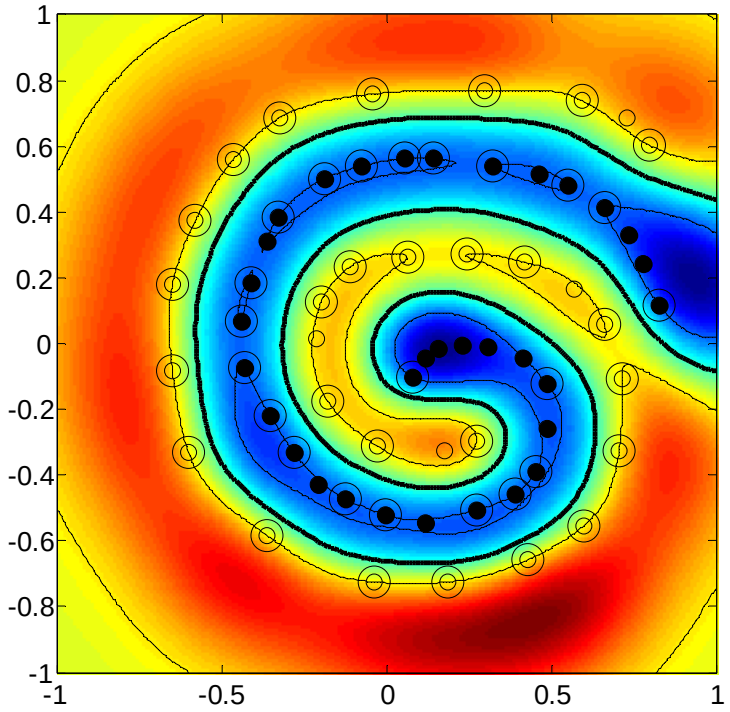




Esempio 3

- Kernel gaussiano $k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{\sigma^2}\right)$

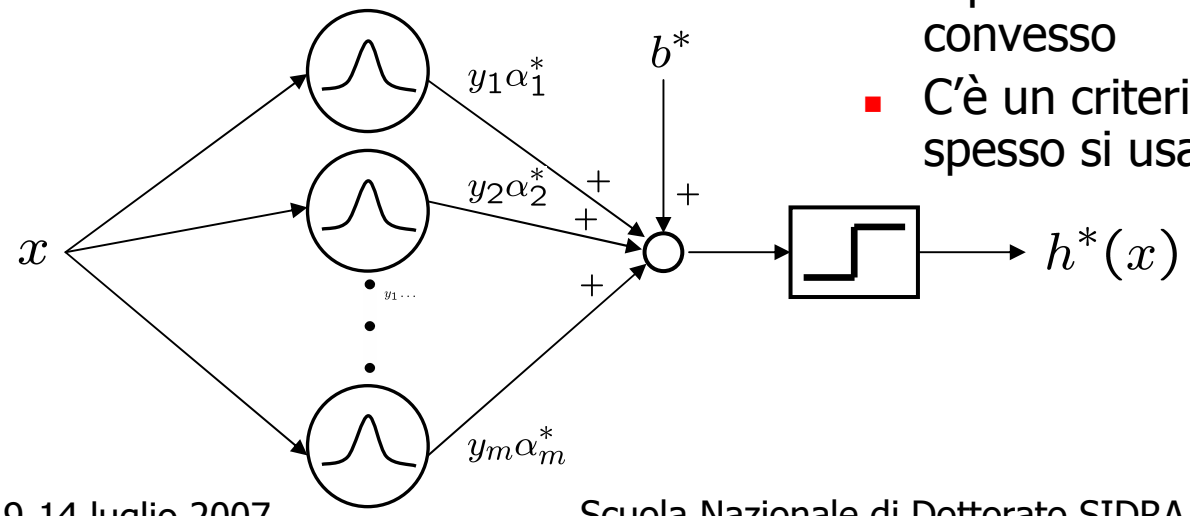




Legame con reti neurali RBF

- Una SVM con kernel gaussiano è equivalente ad una rete neurale a base radiale, con funzione di attivazione gaussiana, tuttavia:
  - Il numero effettivo di neuroni (che sono i SV) è automaticamente scelto dall'algoritmo.
  - La localizzazione dei "centri" è automaticamente scelta dall'algoritmo
  - Il problema di addestramento è convesso
  - C'è un criterio per la scelta di σ (ma spesso si usa cross-validazione)

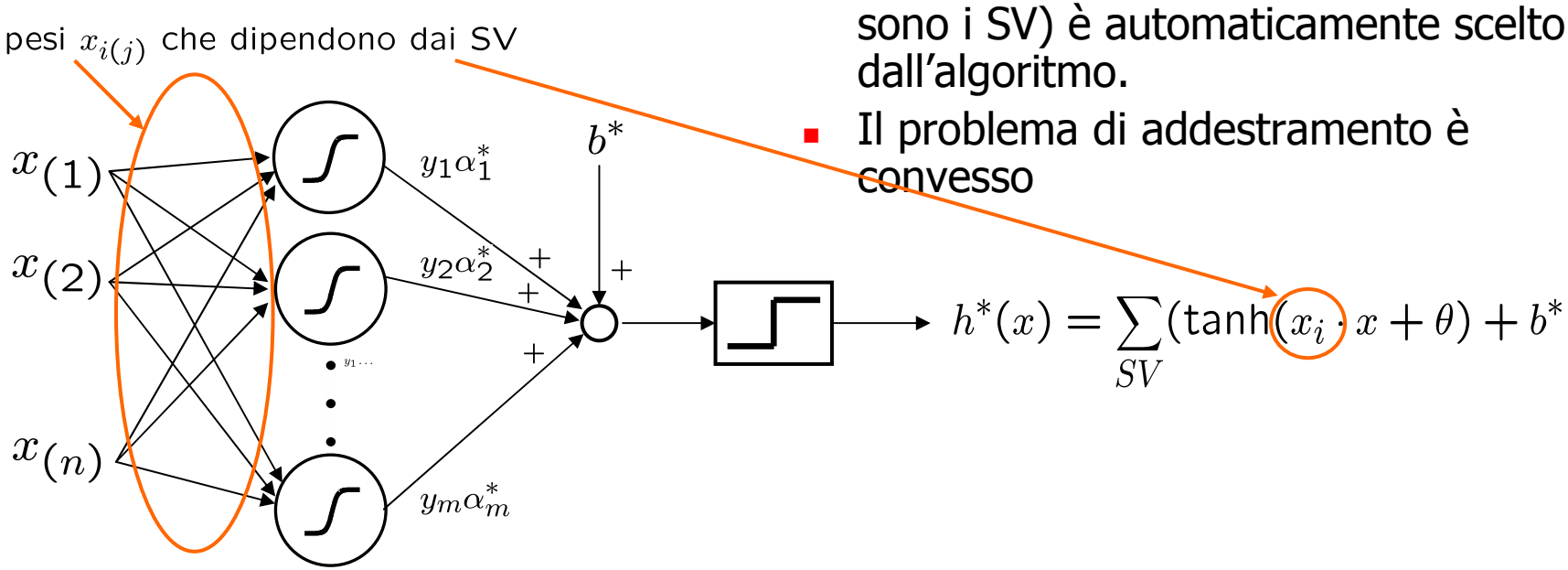
$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{\sigma^2}\right)$$



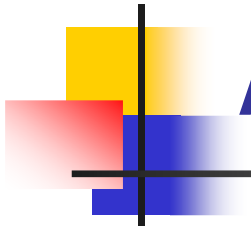
Legame con reti neurali con attivazione sigmoidale

- Una SVM con kernel sigmoidale è equivalente ad una rete neurale a con funzione di attivazione sigmoidale, con un singolo strato nascosto tuttavia:

$$k(x, x') = \tanh(x \cdot x' + \theta)$$

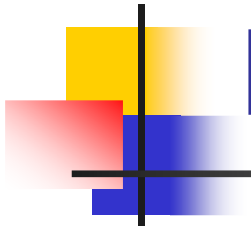


- Il numero effettivo di neuroni (che sono i SV) è automaticamente scelto dall'algoritmo.
- Il problema di addestramento è convesso



Altro

- Ci sono innumerevoli estensioni e affinamenti del metodo base appena descritto, fra i quali
  - Formulazioni che fanno uso di norme diverse dalla norma-1 per penalizzare la violazione dei vincoli
  - Diverse e più significative parametrizzazioni per la regolarizzazione (ν-SVM)
  - Formulazioni che conducono a problemi LP
  - Varie estensioni alla classificazione multiclasse
  - …



Regressione

- Finora abbiamo trattato il problema della classificazione, ovvero la stima di funzioni indicatrici

$$f : \mathcal{X} \rightarrow \{-1, 1\}$$

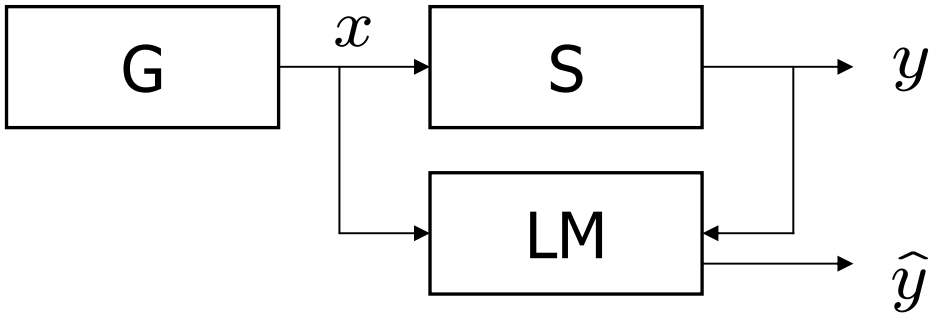
- Ora consideriamo il problema della stima di funzioni a valori reali

$$f : \mathcal{X} \rightarrow R$$

- Ci riferiremo a questo problema con il termine di "regressione"

Regressione come problema di apprendimento supervisionato

- $F(x)$ distribuzione di probabilità di $x \in \mathcal{X} \subseteq R^n$
- $F(y|x)$ distribuzione condizionale di $y \in \mathcal{Y} \subseteq R$ (comprende il caso deterministico $y = f(x)$)
- $F(x, y) = F(x)F(x|y)$ distribuzione congiunta, definita in $\mathcal{X} \times \mathcal{Y}$. E' fissa ma incognita.



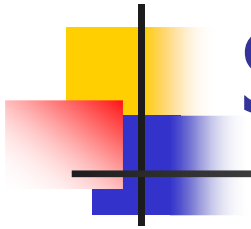
Ingredienti

- Un insieme di osservazioni $\mathcal{S} = \{(x_1, y_1), \ldots, (x_l, y_l)\}$
- Un insieme di ipotesi $\mathcal{H}$
- Un costo (*loss function*) $L(y, h(x))$ attraverso il quale si definisce il rischio

$$R(h) = \int_{\mathcal{X} \times \mathcal{Y}} L(y, h(x)) dF(x, y)$$

- Un criterio per scegliere, sulla base di $\mathcal{S}$, una funzione in $\mathcal{H}$ che si vorrebbe quanto più vicina possibile alla

$$h^* = \arg\min_{h \in \mathcal{H}} R(h) = \arg\min_{h \in \mathcal{H}} \int L(y, h(x)) dF(x, y)$$

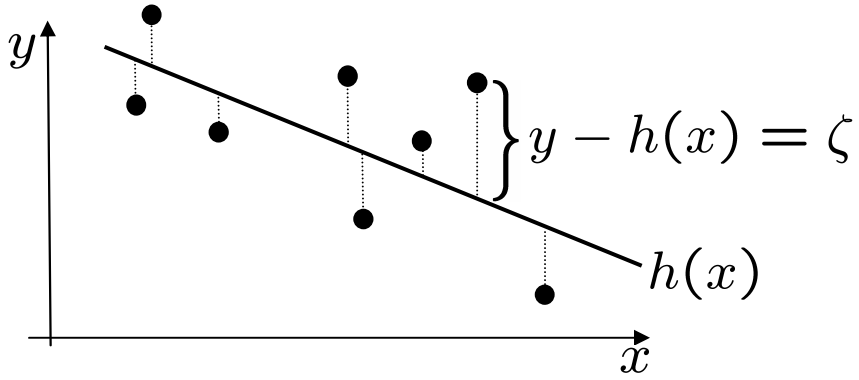


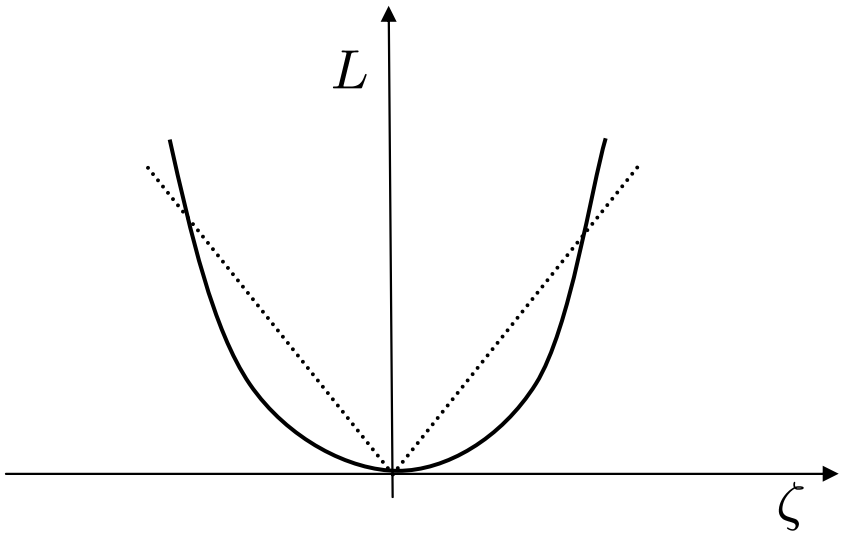
SV Regression

- Support Vector Regression (SVR), analogie con la classificazione:
  - Una macchina lineare apprende una dipendenza non lineare
  - Lavora in uno spazio delle features implicitamente definito dal kernel
  - La capacità dell'insieme delle ipotesi è sotto controllo (attraverso $\|w\|$)
  - Il problema si riconduce alla minimizzazione di un funzionale convesso
  - La soluzione è sparsa

Funzioni di costo

- La funzione di costo attribuisce un costo alla differenza fra la predizione e il valore osservato.

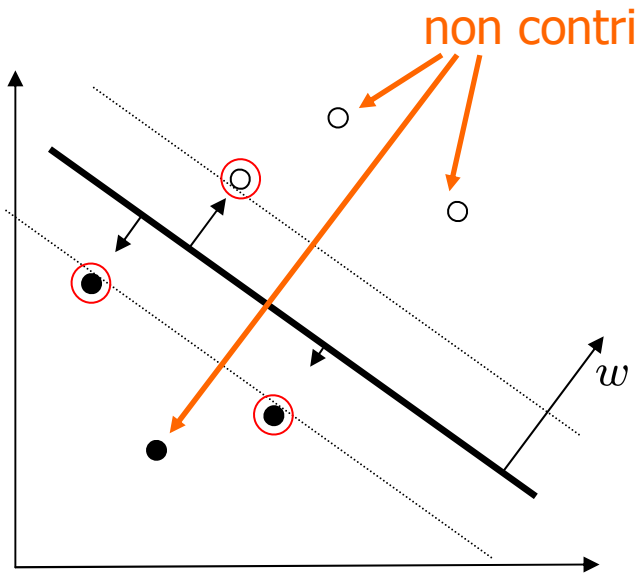




- Esempi di funzioni di costo:
  - Costo quadratico: $L(\zeta) = \zeta^2$
  - 1-norm loss: $L(\zeta) = |\zeta|$

Un passo indietro

- Da cosa dipende la sparsità della soluzione nel caso della classificazione? Dal fatto che il costo è definito in modo tale che solo alcuni pattern vi contribuiscono (in particolare, quelli che distano dall'iperpiano meno del margine)

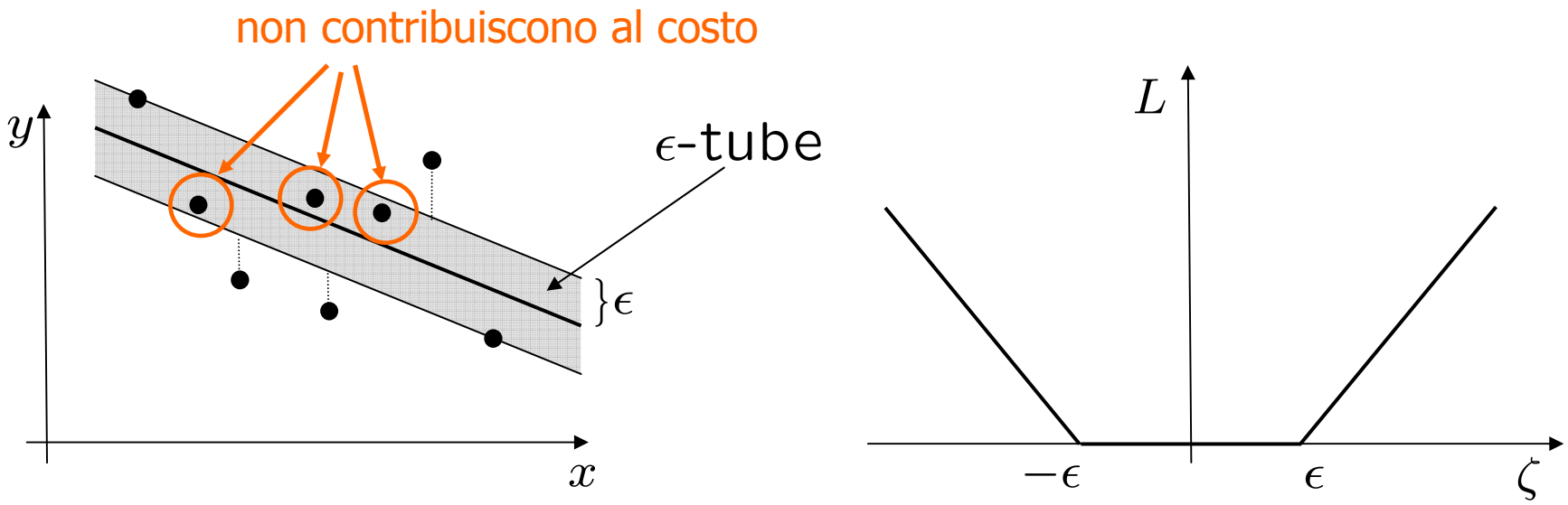


- I pattern più distanti dall'iperpiano non contribuiscono al costo e quindi non hanno alcuna influenza nel formare la funzione di decisione. Se anche fossero rimossi, la soluzione non cambierebbe

Funzione di costo di Vapnik (1)

- Per garantire la sparsità della soluzione, Vapnik introdusse una famiglia di funzioni di costo dette "ε– insensitive". Nel caso lineare (norma-1) si ha

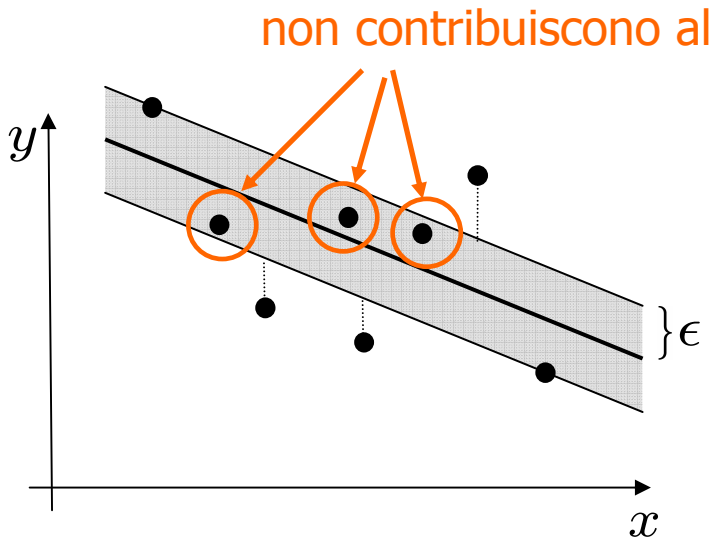
$$L_1^{\epsilon}(\zeta) = \begin{cases} 0 \text{ se } |\zeta| \leq \epsilon \\ |\zeta| - \epsilon \text{ altrimenti} \end{cases}$$

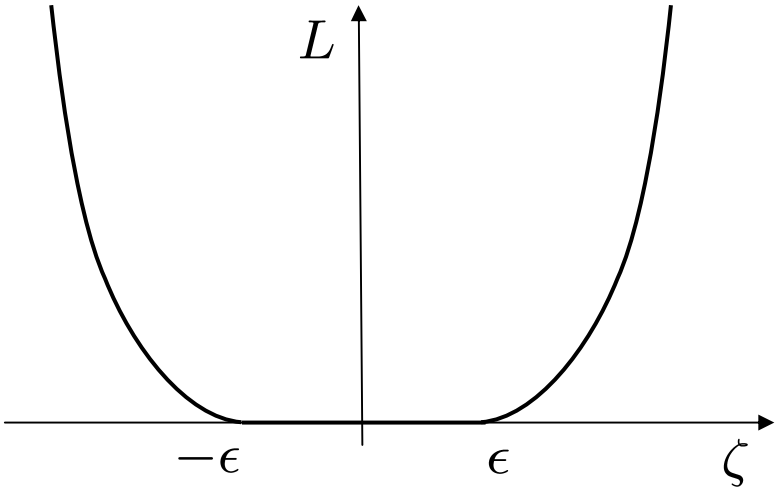


Funzione di costo di Vapnik (2)

- Nel caso quadratico (norma-2) si ha invece

$$L_2^{\epsilon}(\zeta) = \begin{cases} 0 \text{ se } |\zeta| \leq \epsilon \\ (|\zeta| - \epsilon)^2 \text{ altrimenti} \end{cases}$$





Formulazione del problema (1)

- Consideriamo dapprima il caso lineare, cioè un insieme delle ipotesi costituito da funzioni affini:

$$\mathcal{H} = \{h(x) = (w \cdot x) + b\}$$

- Il problema SVR si può formulare (norma-1) come segue:

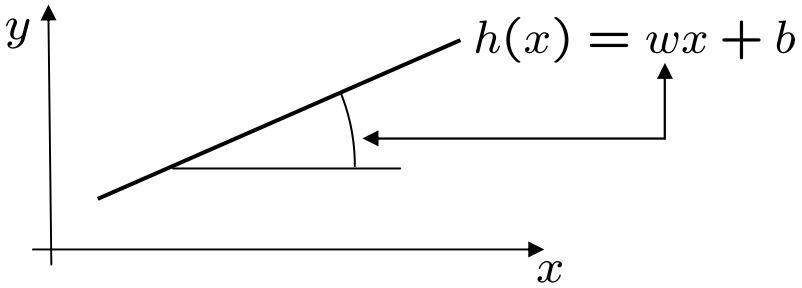
$$\min_{w,b} \underbrace{\frac{1}{2}(w \cdot w)}_{?} + \underbrace{C\frac{1}{l}\sum_{i=1}^{l} L_1^{\epsilon}(|y_i - (w \cdot x_i) - b|)}_{\text{rischio empirico}}$$

- Che è molto simile al problema formulato per la classificazione. Ma per quale motivo minimizzare il modulo del vettore w ?

Formulazione del problema (2)

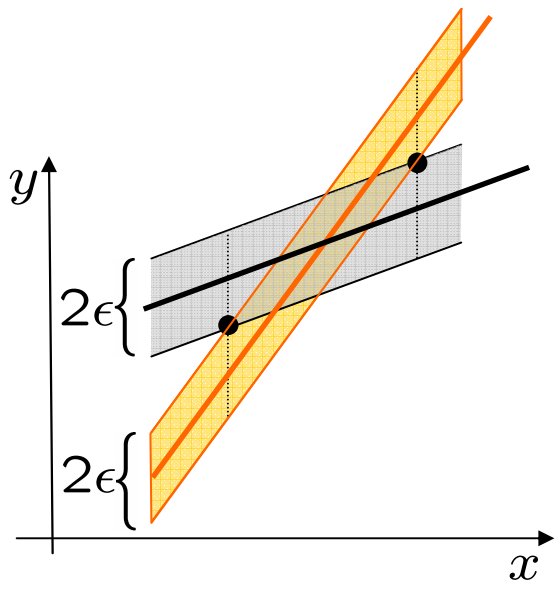
$$\min_{w,b} \frac{1}{2}(w \cdot w) + C\frac{1}{l}\sum_{i=1}^{l} L_1^{\epsilon}(|y_i - (w \cdot x_i) - b|)$$

- La quantità $\|w\|$ rappresenta nel caso della classificazione, l'inverso del margine e quindi minimizzarla significa massimizzare il margine
- In questo caso invece rappresenta la *pendenza* di $h(x)$
- Dunque si richiede di minimizzare nel contempo il rischio empirico e massimizzare la "piattezza" (flatness) della funzione



Flatness

- Esistono dei bound (non asintotici e indipendenti dalla distribuzione) che mostrano che riducendo $||w||$ si migliora la capacità di generalizzazione.
- La cosa si può intuire dall'esempio elementare che segue:



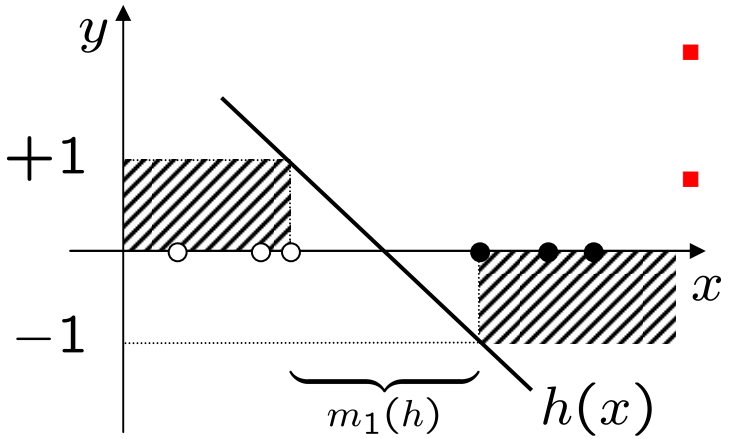
- Dati due soli punti e fissato ε>0, entrambe le rette realizzano rischio empirico nullo (i punti stanno al confine della zona di insensitività)
- E' sensato, in mancanza di informazioni sulla funzione "vera", preferire la retta a pendenza minore perché ad essa corrispondono, per perturbazioni di x, scostamenti minori di y rispetto al valore assunto nei punti del TS
- Significa "dare credito" al TS: principio analogo alla massimizzazione del margine per la classificazione (cfr. slide sul "pattern noise")

Flatness e margine

- E' possibile definire un *margine* che viene massimizzato attraverso la minimizzazione di $\|w\|$

dati $h : \mathcal{X} \rightarrow R$ e $\epsilon \geq 0$

$$m_\epsilon(h(x)) \doteq \inf \left\{ \|x - x'\|, x, x' \in \mathcal{X} : \|h(x) - h(x')\| \geq 2\epsilon \right\}$$



- Problema unidimensionale di classificazione:
  - Il classificatore a vettori di supporto minimizza $\|w\|$ con il vincolo di canonicità
  - Ciò corrisponde a massimizzare il margine m_ϵ con $\epsilon = 1$

Problema primale

- Assorbendo il fattore $\frac{1}{l}$ nella costante C e introducendo delle variabili di slack il problema diventa:

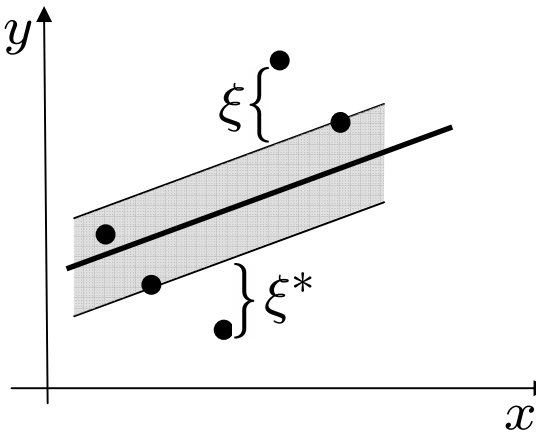
$$\min_{w,b} \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{l} (\xi_i + \xi_i^*)$$

s.t.

$$y_i - (w \cdot x_i) - b \leq \epsilon + \xi_i, \ \ \forall i = 1, \ldots, l$$

$$(w \cdot x_i) + b - y_i \leq \epsilon + \xi_i^*, \ \ \forall i = 1, \ldots, l$$

$$\xi_i, \xi_i^* \geq 0, \ \ \forall i = 1, \ldots, l$$



- C è una costante che controlla il compromesso fra deviazioni dalla zona di insensitività e flatness.

Problema duale (1)

- Costruiamo la funzione Lagrangiana, introducendo i moltiplicatori $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0$

$$L(w, b, \alpha, \alpha^*, \eta, \eta^*) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) -$$
$$- \sum_{i=1}^{l}\alpha_i\left(\epsilon + \xi_i - y_i + (w \cdot x_i) + b\right) -$$
$$- \sum_{i=1}^{l}\alpha_i^*\left(\epsilon + \xi_i^* + y_i - (w \cdot x_i) - b\right) -$$
$$- \sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*)$$

Problema duale (2)

- Le condizioni di stazionarietà forniscono

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0$$

$$\frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*)\, x_i$$

$$\frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad \eta_i = C - \alpha_i$$

$$\frac{\partial L}{\partial \xi_i^*} = 0 \quad \Rightarrow \quad \eta_i^* = C - \alpha_i^*$$

è una combinazione lineare dei vettori del training set.

$$h(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*)\, (x_i \cdot x) + b$$

Problema duale (3)

- Le condizioni di stazionarietà, sostituite nella Lagrangiana, forniscono

$$\min \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(x_i \cdot x_j) - \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \\ + \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*)$$

s.t.

$$\sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0$$

$$\alpha_i, \alpha_i^* \in [0, C], \ \forall i = 1, \ldots, l$$

- Minimizzare un funzionale quadratico convesso sotto vincoli lineari → problema di ottimizzazione quadratica

Osservazioni (1)

- Le condizioni di KKT permettono di studiare alcune caratteristiche della soluzione

$$\alpha_i \left(\epsilon + \xi_i - y_i + (w \cdot x_i) + b\right) = 0$$
$$\alpha_i^* \left(\epsilon + \xi_i^* + y_i - (w \cdot x_i) - b\right) = 0$$

$$(C - \alpha_i)\, \xi_i = 0$$
$$(C - \alpha_i^*)\, \xi_i^* = 0$$

$$|(w \cdot x_i) + b - y_i| < \epsilon \Rightarrow \alpha_i, \alpha_i^* = 0$$

- Cioè i moltiplicatori sono nulli per tutti i punti all'interno di ε-tube: la soluzione è sparsa

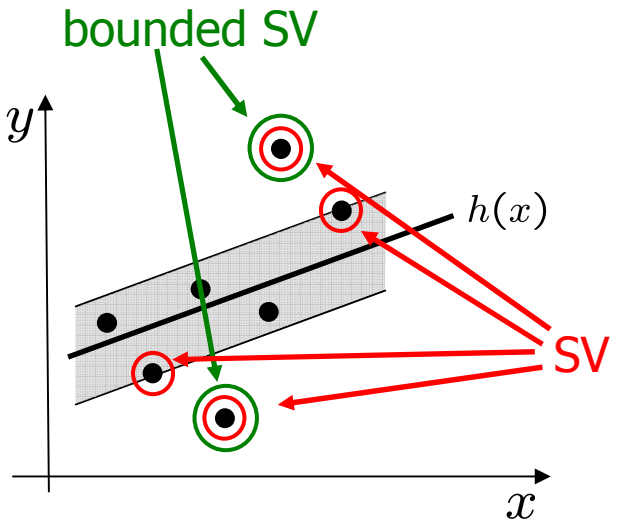
$$\xi_i^{(*)} > 0 \Rightarrow \alpha_i^{(*)} = C$$

- Cioè i punti al di fuori di ε-tube sono bounded support vectors

Osservazioni (2)

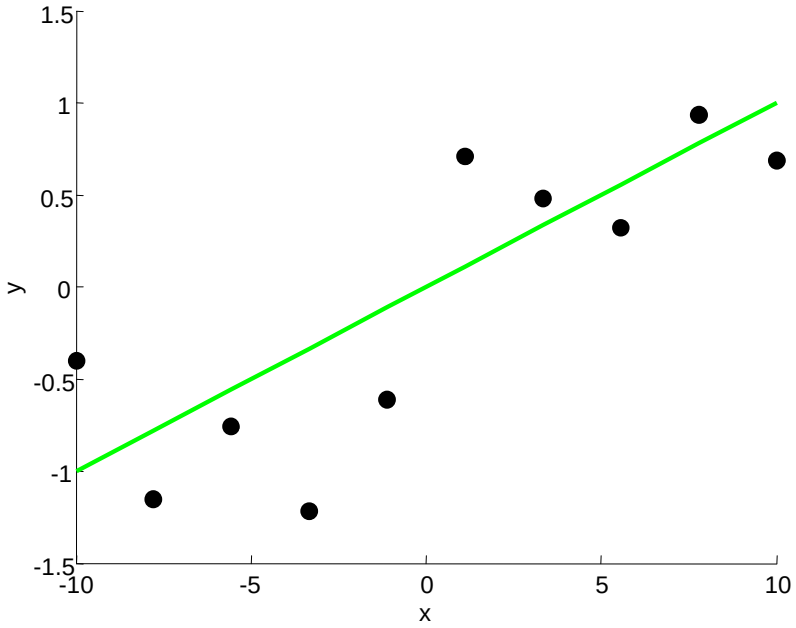
- La funzione può essere espressa in funzione dei soli vettori di supporto

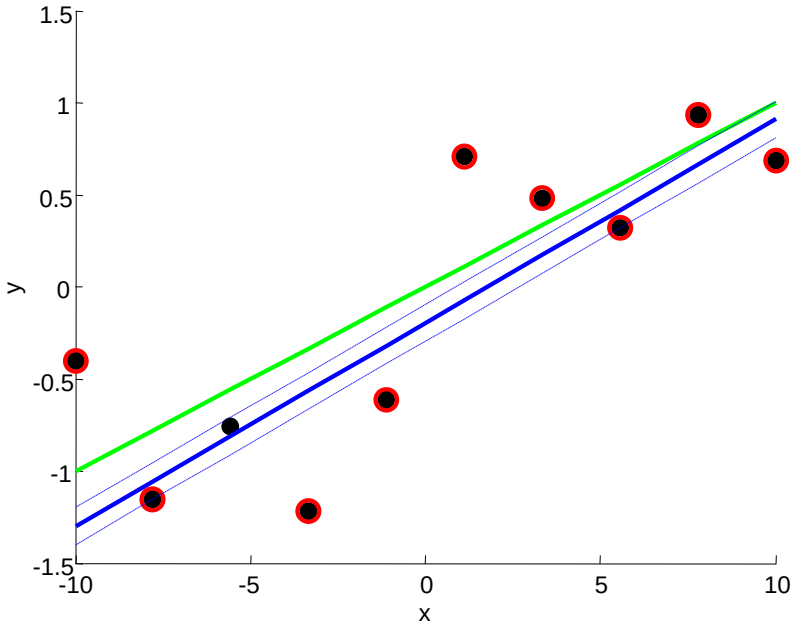
$$h(x) = \sum_{SV} (\alpha_i - \alpha_i^*)\,(x_i \cdot x) + b$$



- Si calcola sfruttando opportunamente le condizioni di KKT

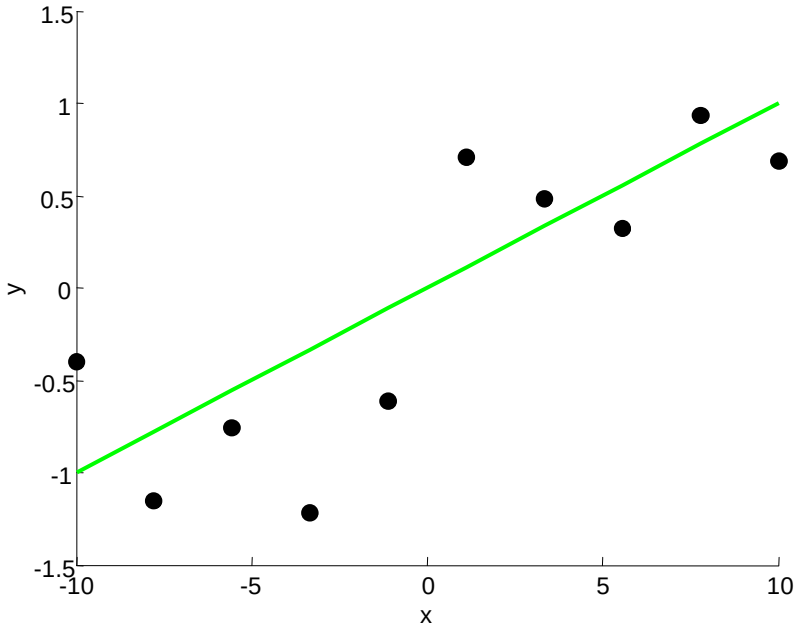
Esempio (1)

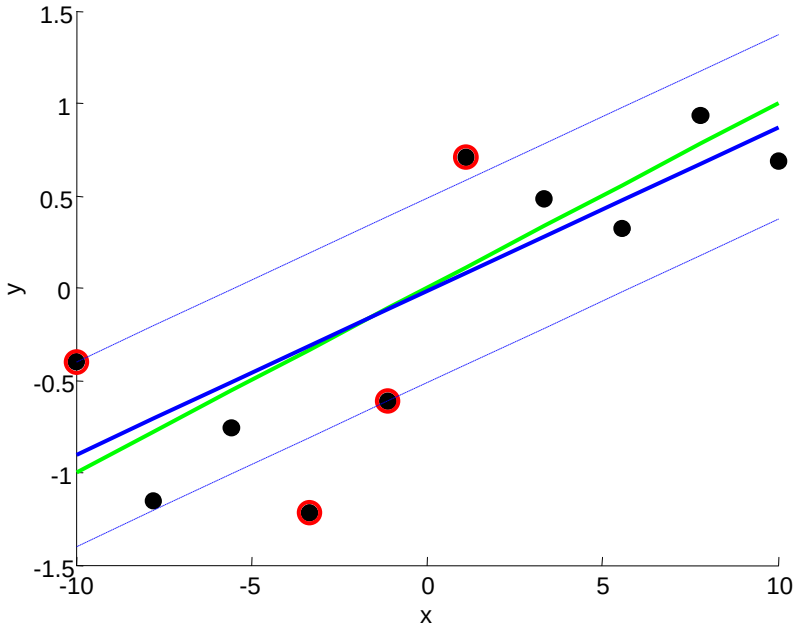




- `y = 0.1*x+randn(size(x));`
- `epsilon = 0.1;`

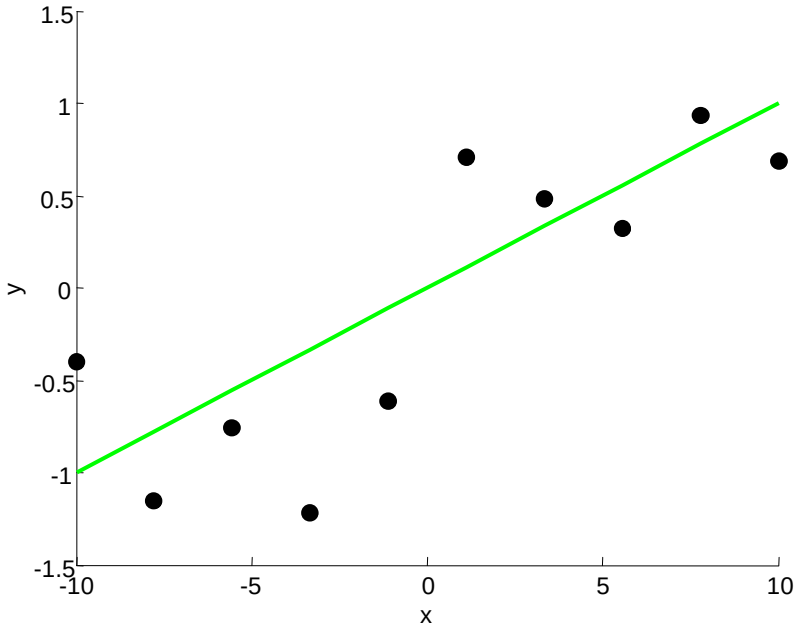
Esempio (2)

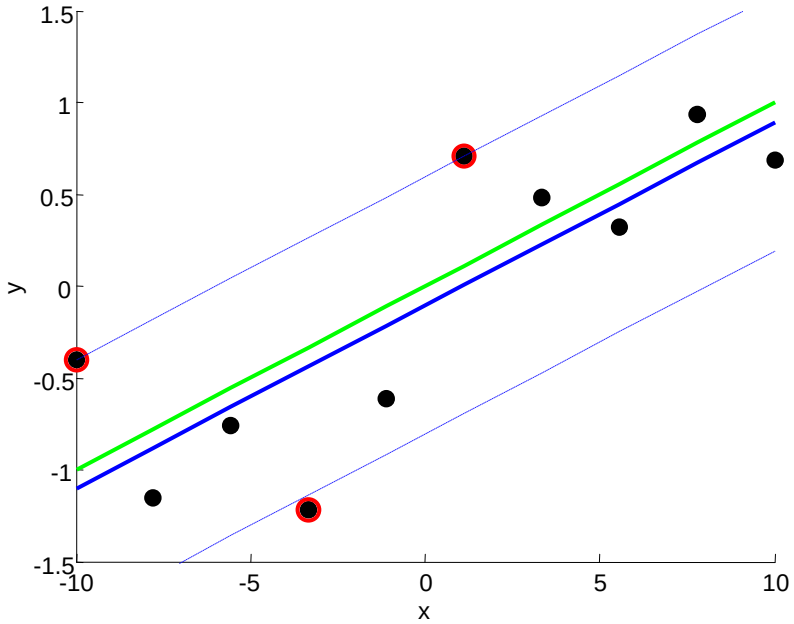




- `y = 0.1*x+randn(size(x));`
- `epsilon = 0.5;`

Esempio (3)





- `y = 0.1*x+randn(size(x));`
- `epsilon = 0.7;`

Kernel trick

- Sia nel problema di ottimizzazione che nell'espressione della funzione soluzione, i vettori del TS compaiono solamente all'interno di prodotto scalari.

$$\min \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(x_i \cdot x_j) - \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*)$$

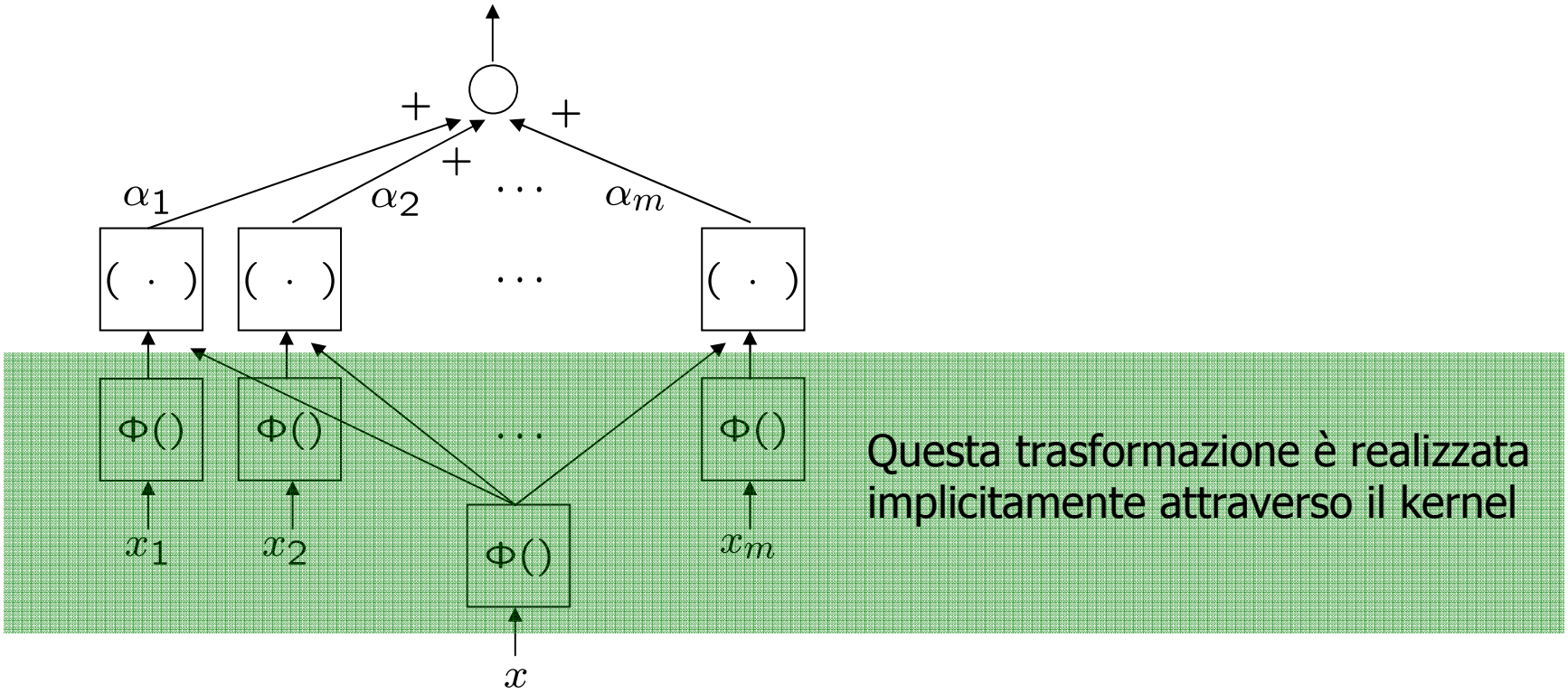
$$h(x) = \sum_{SV} (\alpha_i - \alpha_i^*)(x_i \cdot x) + b$$

- Quindi attraverso un kernel si può estendere l'algoritmo ad uno spazio delle features $\mathcal{F}$ in cui i vettori originali vengono mappati attraverso una funzione Φ implicitamente definita dal kernel

→ esattamente come si è fatto nel caso della classificazione!

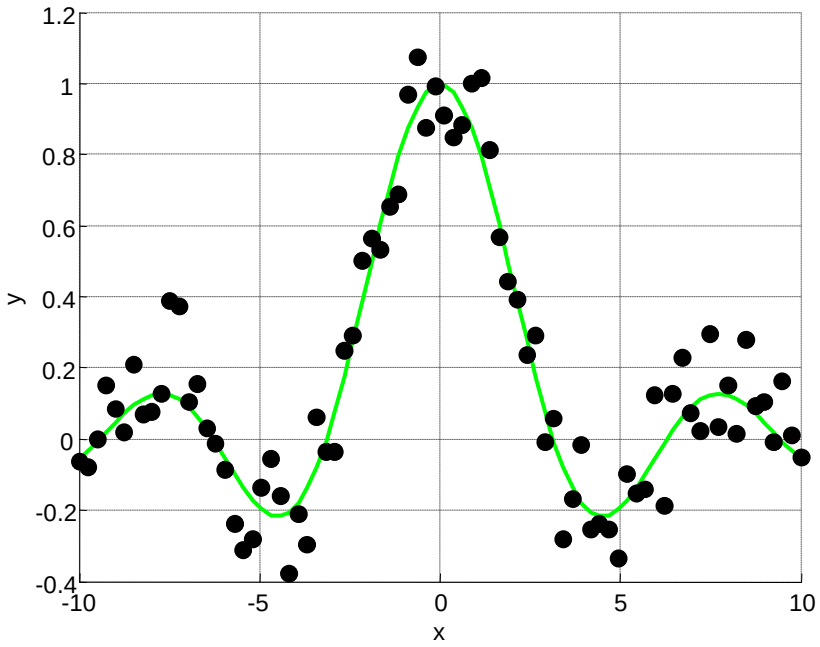
Architettura

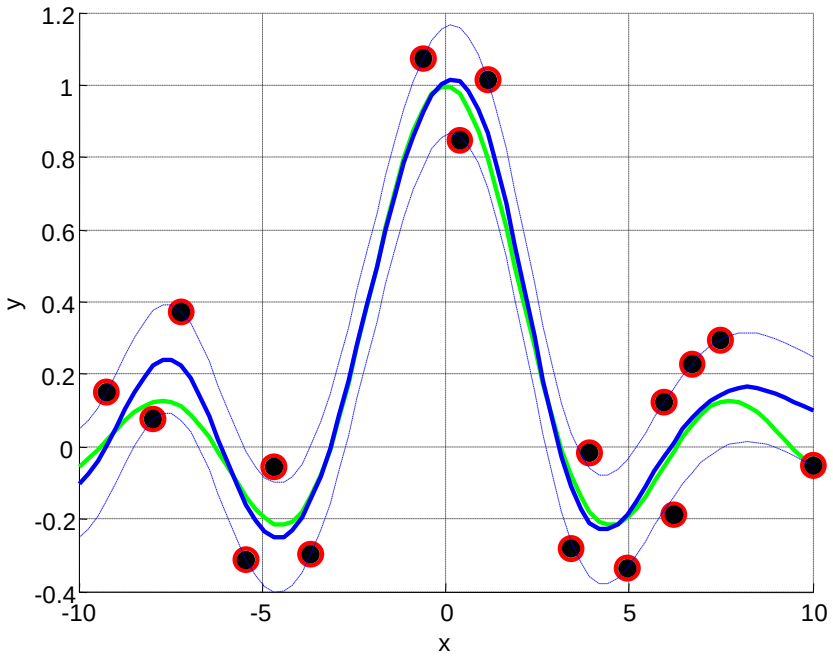
- ...



Esempio (1)

- kernel Gaussiano, $\sigma=2$

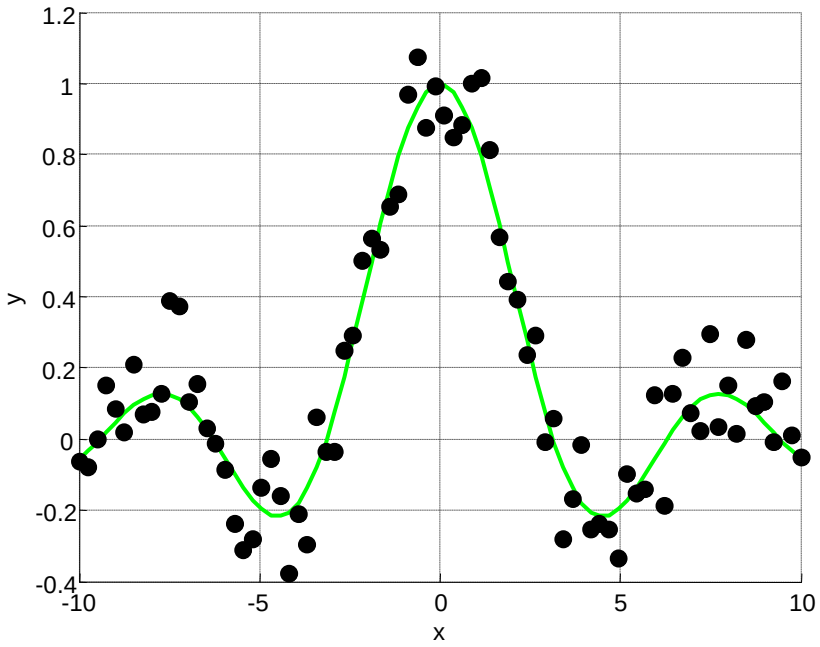


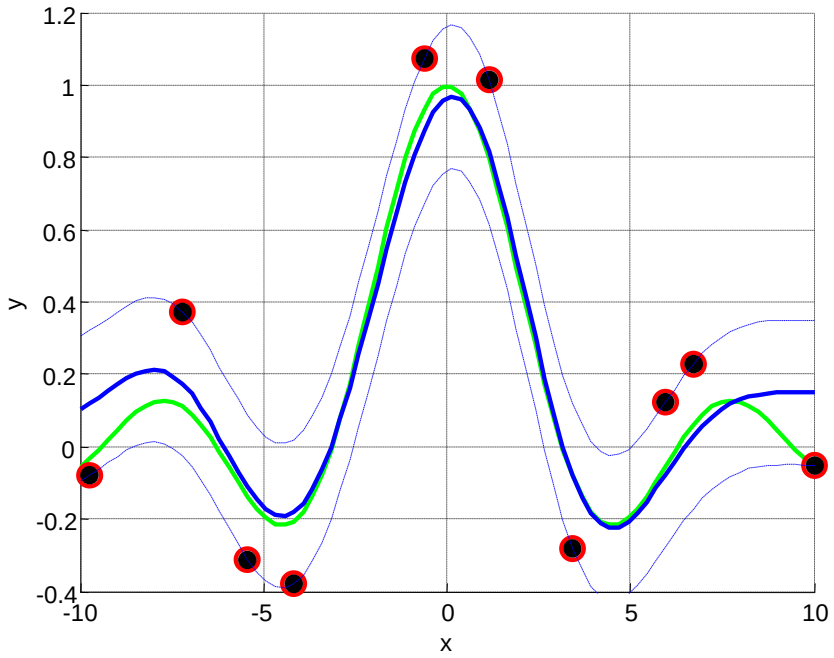


- `y = sinc(x/pi)+randn(size(x));`
- `epsilon = 0.15;`

Esempio (2)

- kernel Gaussiano, $\sigma=2$

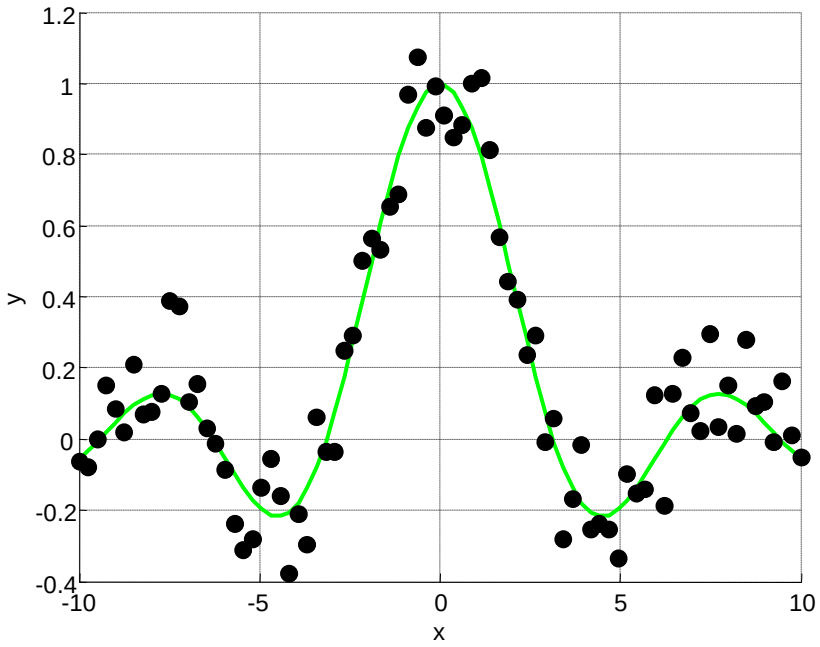


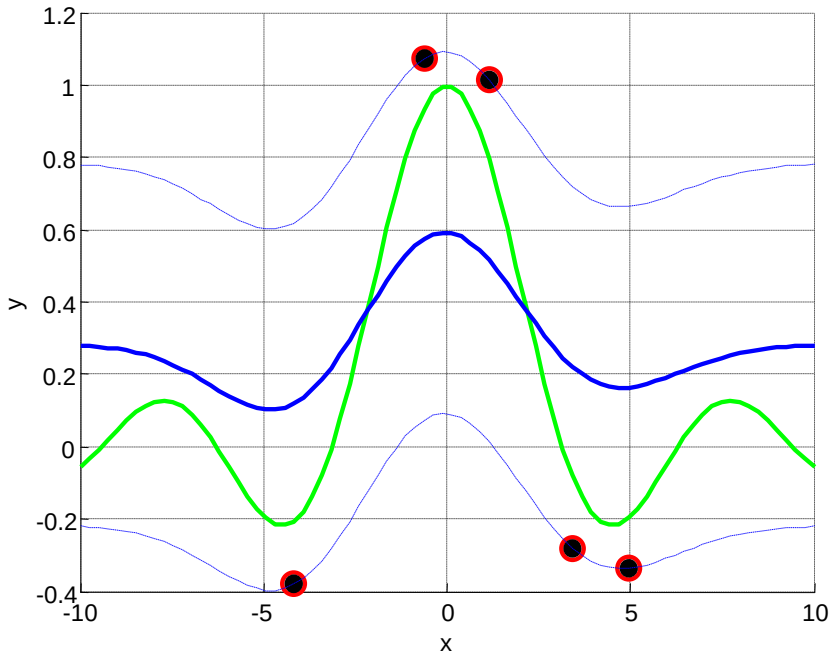


- `y = sinc(x/pi)+randn(size(x));`
- `epsilon = 0.2;`

Esempio (3)

- kernel Gaussiano, $\sigma=2$

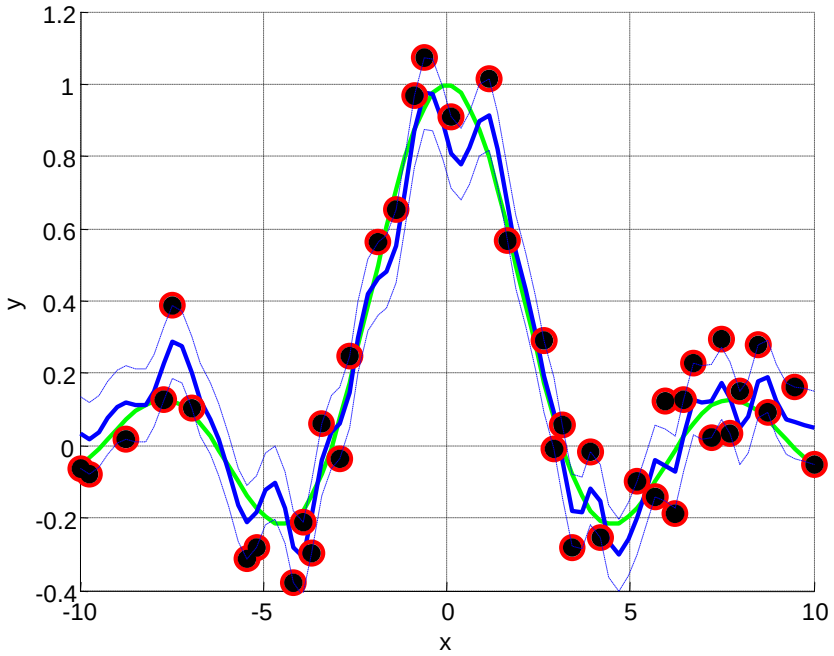




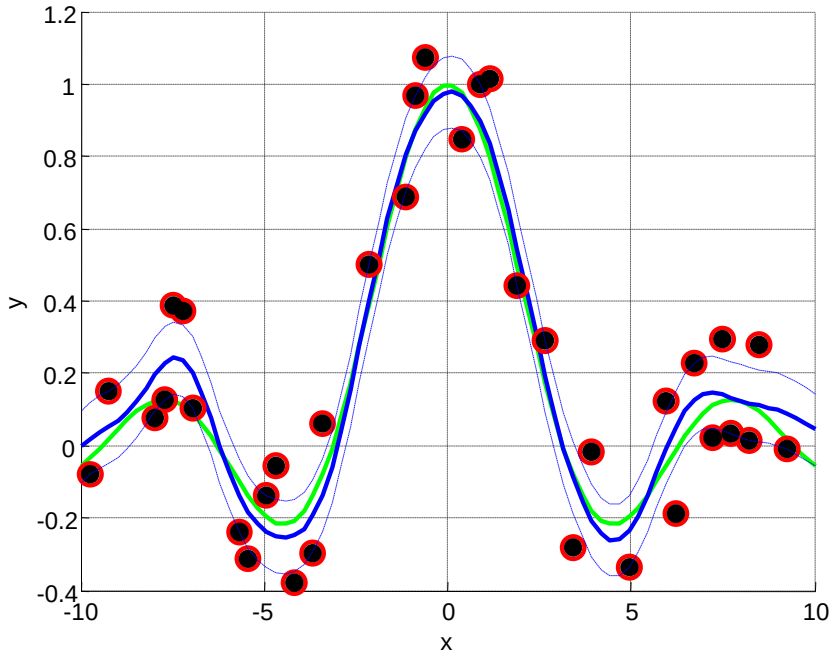
- `y = sinc(x/pi)+randn(size(x));`
- `epsilon = 0.5;`

Scelta del kernel

- Ci vuole un po' di "sensibilità" nella scelta di σ



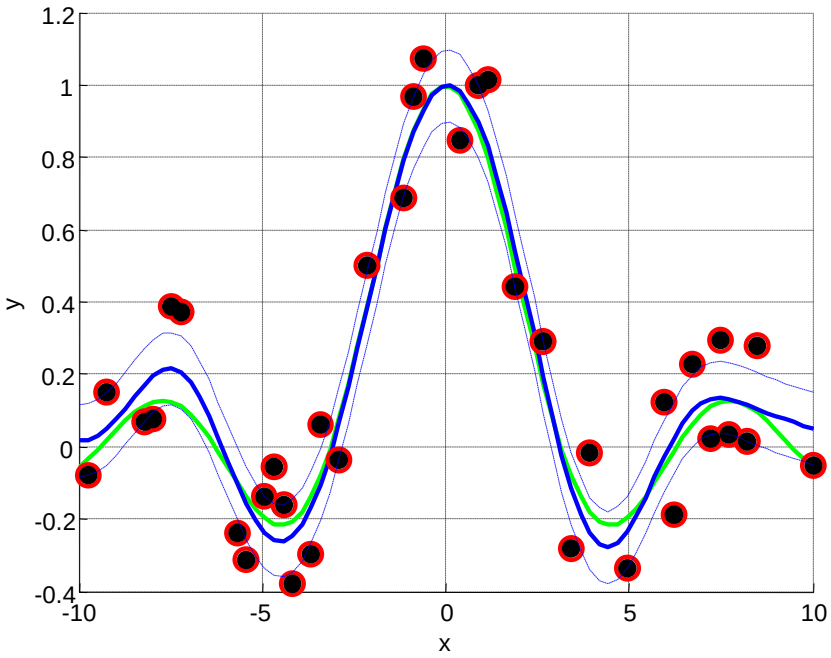
- `sigma=0.5;`

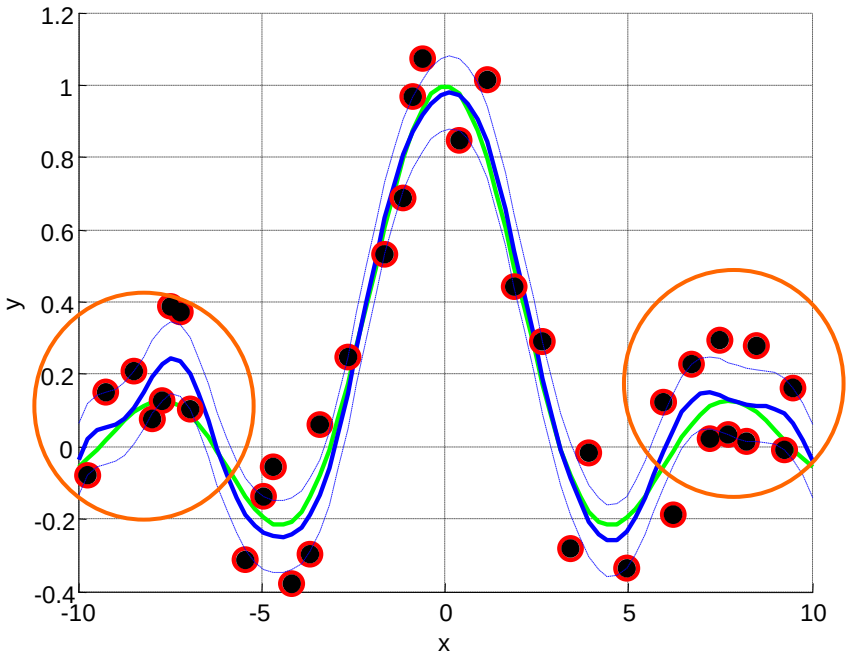


- `sigma=1.5;`

Ruolo di C

- kernel Gaussiano, $\sigma=2$

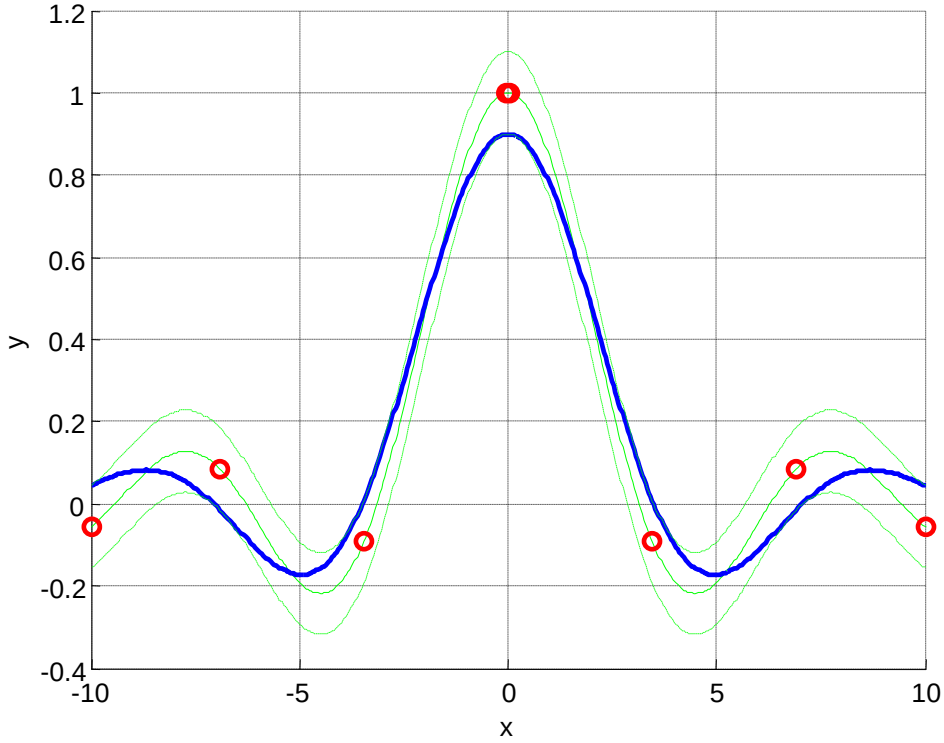




- `C=10;`
- `C=1000;`

Utile anche per approssimare funzioni note

- kernel Gaussiano, $\sigma=2$



- 8 vettori di supporto

⇓

- La funzione viene descritta come la somma di 8 Gaussiane

- Se non ci sono vettori di supporto bounded allora l'approssimazione (blu) sta all'interno dell'ε-tube attorno alla funzione da approssimare (verde)

Altre funzioni di costo (1)

- Il problema considerato è del tipo:

$$\min R_{reg}(h) \doteq \lambda \|w\| + R_{emp}(h)$$

cioè consiste nel minimizzare un *rischio regolarizzato* che è dato dalla somma del rischio empirico e di un termine che serve a controllare la capacità (la ricchezza delle ipotesi).

- In linea di principio qualunque funzione di costo ε – insensitive, può essere utilizzata e fornire una soluzione sparsa. Ciò ovviamente nel caso in cui dia luogo ad un problema di ottimizzazione trattabile.

- La funzione di costo "lineare" $L_1^{\epsilon}(\zeta)$ fu utilizzata per prima in virtù delle sue proprietà di robustezza (è infatti simile alla funzione di costo di Huber).

Altre funzioni di costo (2)

- Se però si fanno delle assunzioni (o si hanno informazioni) sul modello che genera i dati, altre funzioni di costo possono risultare preferibili.
- Ad esempio, se i dati sono generati da una dipendenza funzionale f fissa più un rumore additivo di densità $p(\xi)$

$$y_i = f(x_i) + \xi_i$$

allora si può mostrare che la funzione di costo ottima nel senso della massima verosimiglianza è

$$L(y - h(x)) = -\ln p(y - h(x))$$

- Nulla assicura tuttavia che ciò conduca ad un problema convesso né tantomeno quadratico

Altre funzioni di costo (3)

- Ad esempio, se il rumore è Gaussiano:

$$p(\xi) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{\xi^2}{2})$$

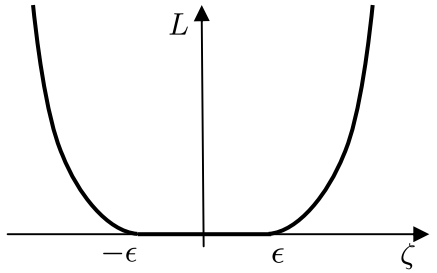
la funzione di costo ottima nel senso della massima verosimiglianza è

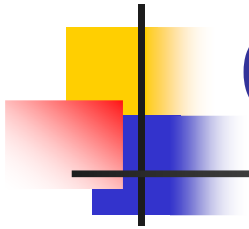
$$L(y-h(x)) = -\ln p(y-h(x)) = \frac{[y-h(x)]^2}{2} + \ln\sqrt{2\pi}$$

ovvero, a meno di una costante, una funzione di costo quadratica.

- Dotandola di una zona di insensitività, si ottiene

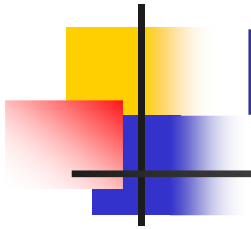
$$L_2^{\epsilon}(y-h(x))$$





Considerazioni finali

- I principali vantaggi dei metodi visti sono
  - Solidi fondamenti teorici
  - Pochi(ssimi) parametri
  - No minimi locali
  - Nessuna euristica
  - Estrema semplicità d'uso
- Suggerimento
  - Non scrivere il codice da sé ma usare qualche buon toolbox per Matlab come ad esempio
    `http://asi.insa-rouen.fr/~arakotom/toolbox/index.html`



Per approfondire

[1] M. Aizerman, E. Braverman and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning", *Automation and Remote Control*, vol. 25, pp. 821 - 837, 1964.

[2] B. E. Boser and I. M. Guyon and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers", *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, Pittsburgh, PA, pp. 144–152, 1992.

[3] N. Cristianini and J. Shawe–Taylor, *An introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[4] D. G. Luenberger, *Linear and Nonlinear Programming 2nd ed.*, Springer, 2003.

[5] B. Schölkopf and A. Smola, *Learning with Kernels*, MIT Press Cambridge, MA, 2002.

[6] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 1999.