Riconoscimento e recupero dell’informazione per bioinformatica

Classificatori discriminativi

Manuele Bicego

Corso di Laurea in Bioinformatica
Dipartimento di Informatica - Università di Verona

Introduzione

- Finora abbiamo visto i classificatori generativi:
  - mirano a modellare le probabilità condizionali delle classi, da mettere assieme con quelle a priori per ottenere le posterior
  - regola di decisione di Bayes
- I classificatori discriminativi mirano ad ottenere "direttamente il confine di decisione"
  - stimando direttamente le posterior
  - o affidandosi a concetti "geometrici"
- Due esempi:
  - Funzioni discriminanti lineari
  - Support Vector Machines

Funzioni discriminanti lineari

(ripasso dalla lezione sulla teoria della decisione di Bayes)

Classificatori, funzioni discriminanti e superfici di separazione

- Per rappresentare un classificatore spesso si utilizza un insieme di ***funzioni discriminanti*** $g_i(\mathbf{x}), i=1...c$

- Il classificatore assegna l'oggetto **x** alla classe ω_i se

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \text{ per ogni } j \neq i$$

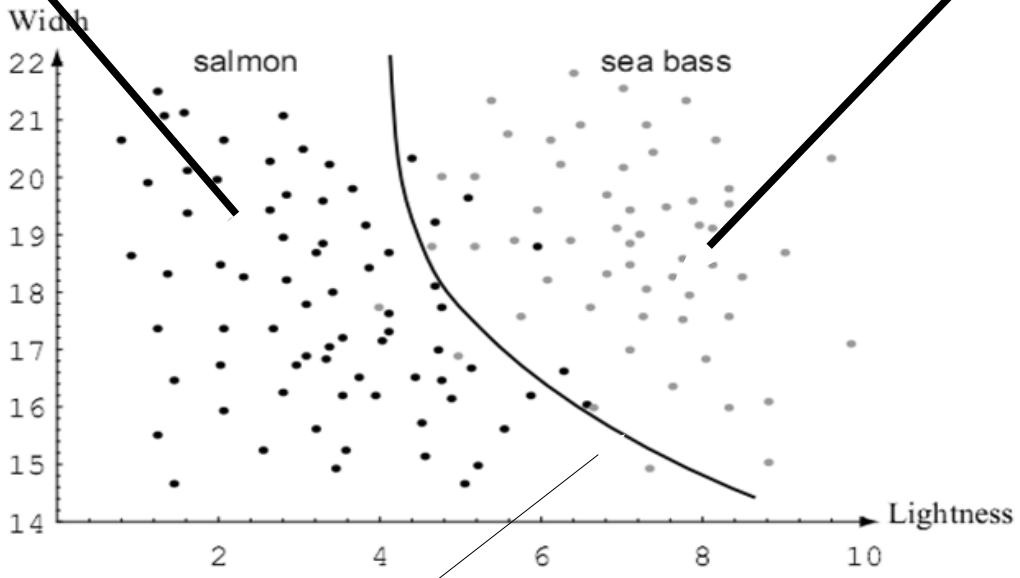
Questo insieme di funzioni suddivide lo spazio delle features in ***Regioni****, separate tra di loro da* ***Confini di decisione*** *(decision boundaries)*

Regione R_1 (tutti i punti che verrebbero classificati come appartenenti alla classe 1)

Punti x per cui $g_1(x) > g_2(x)$

Regione R_2 (tutti i punti che verrebbero classificati come appartenenti alla classe 2)

punti x per cui $g_2(x) > g_1(x)$



Confine di decisione

Punti x per cui $g_1(x) = g_2(x)$

Classificatori, funzioni discriminanti e superfici di separazione

- Un classificatore che utilizza la regola di decisione di Bayes si presta facilmente a questa rappresentazione:

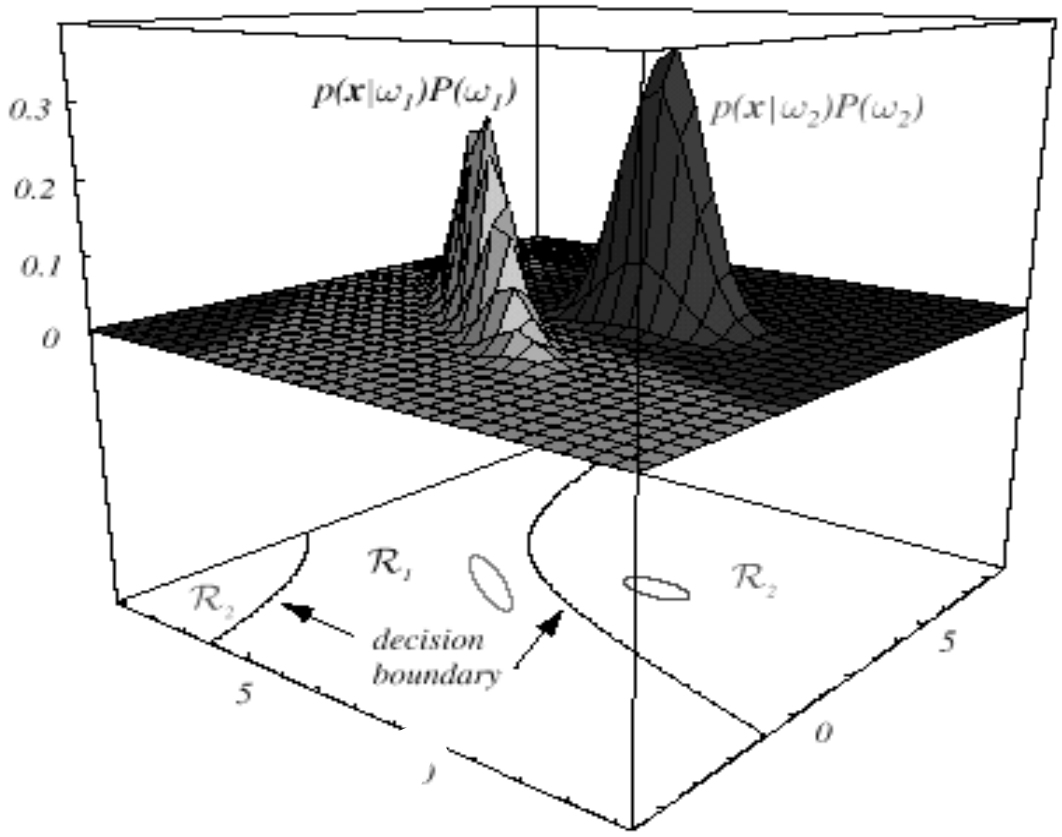
$$g_j(x) = P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)}$$

Nota finale

- Nel caso a *due* categorie ho due funzioni discriminanti, g1,g2 per cui assegno x a ω1 se g1 > g2

- Posso anche definire una sola funzione discriminante g(x) = g1-g2, e classificare quando g(x) > 0 (g1-g2>0, o g1>g2)

$$g(x) = g_1(x) - g_2(x)$$



ho una sola funzione discriminante!

Funzioni discriminanti lineari

Consideriamo il caso a due classi $\omega 1$ e $\omega 2$ con una sola funzione discriminante lineare g(x):

⇨ assegnare x a $\omega 1$ se g(x) >0

DEFINIZIONE DI FUNZIONI DISCRIMINANTI LINEARI:

se g(x) è combinazione lineare delle feature (delle varie componenti di x), più un termine noto, allora si parla di funzioni discriminanti **lineari**

Funzioni discriminanti lineari

Esempio:

- problema bidimensionale, ogni oggetto è rappresentato da due features [x1,x2]
- Una funzione discriminante **lineare** ha la seguente forma:

$$g(x) = ax1 + bx2 + c$$

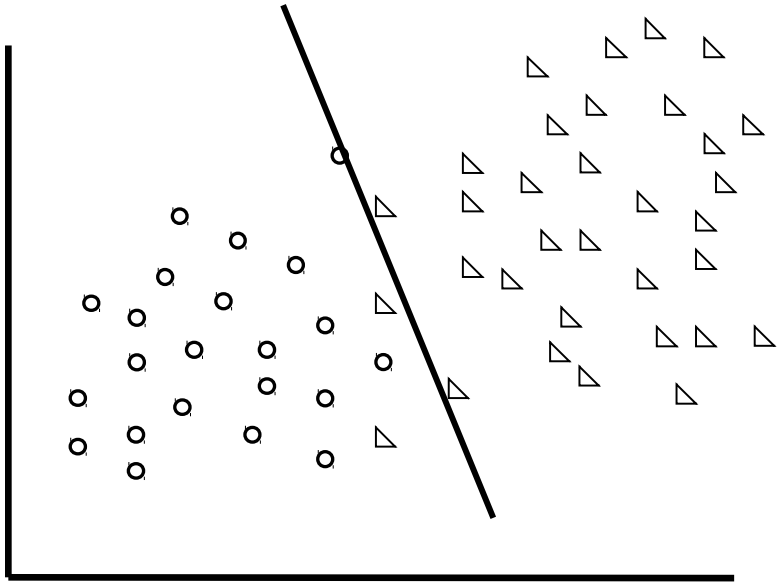
Il confine di decisione, per definizione, si ha quando

$$g(x) = 0$$

Nell'esempio bidimensionale, ax1+bx2+c = 0 rappresenta una retta!

Funzioni discriminanti lineari

- Con le funzioni discriminanti lineari la superficie di separazione è una retta (nel caso bidimensionale) o un iperpiano (nel caso di uno spazio a *d*-dimensioni)



Funzioni discriminanti lineari

Matematicamente: g(x) combinazione lineare delle features di ingresso

$$g(x) = w_0 + \sum_{i=1}^{n} w_i x_i$$

- (caso bidimensionale): g(x) = ax1+bx2+c
  - w1 = a
  - w2 = b
  - w0 = c

Funzioni discriminanti lineari

In forma compatta:

$$g(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} + w_0$$

dove

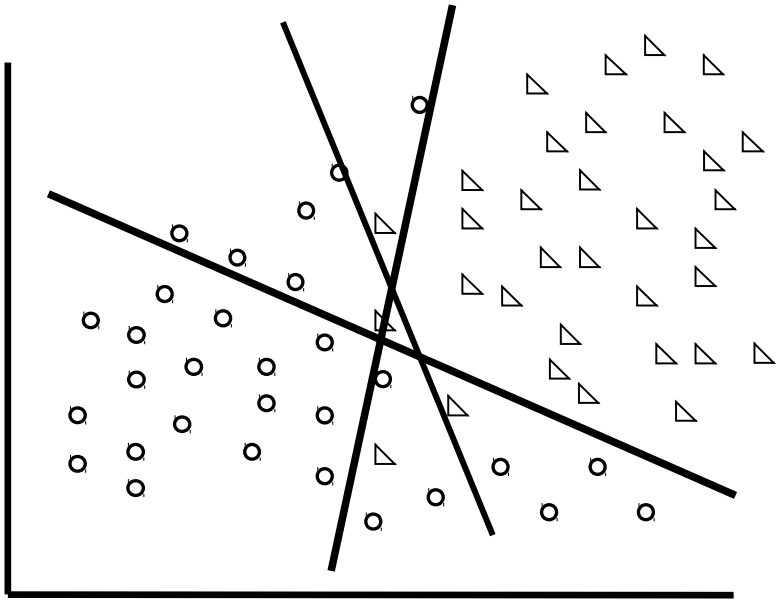
- $\mathbf{w}\cdot\mathbf{x}$ è il prodotto scalare tra $\mathbf{w}$ e $\mathbf{x}$
- $\mathbf{x}=[x_1,\ldots,x_n]$,
- $\mathbf{w}=[w_1,\ldots,w_n]$ (pesi)
- w_0 *bias* (soglia)

Regola della funzione discriminante lineare

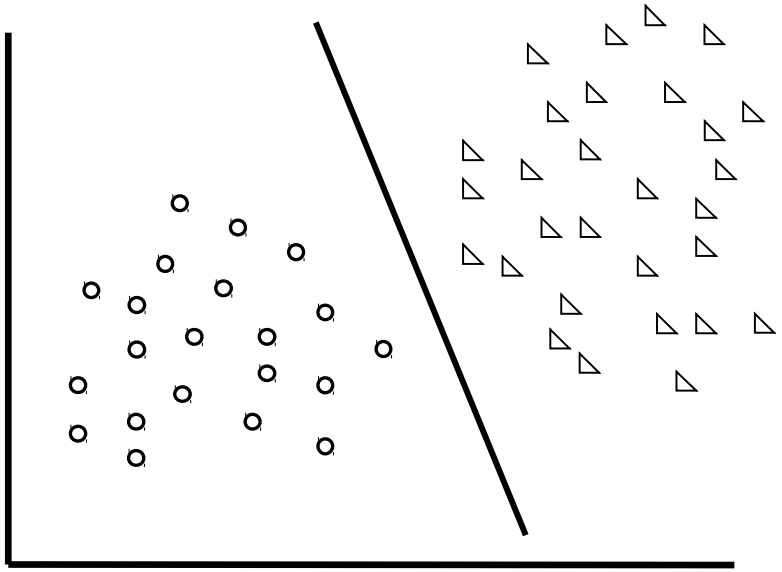
Un campione $\mathbf{x}_i$ è classificato come appartenente a ω_1 se $\mathbf{w}\cdot\mathbf{x}_i+w_0>0$, a ω_2 altrimenti

DEFINIZIONE IMPORTANTE

Se esistono un vettore di pesi **w** e una soglia w_0 tali per cui le due classi risultano separate, allora gli oggetti del dataset si dicono *linearmente separabili*.



non linearmente separabili



linearmente separabili

Non esiste nessun valore di **w** e w_0 tale per cui le due classi siano separate

Funzioni discriminanti lineari

- Nota: il vettore **w** e w_0 non si conoscono
  - si possono stimare dal training set (apprendimento da esempi)
  - L'obiettivo è determinare i migliori valori di **w** e w_0 della funzione discriminante lineare

- Occorre stabilire cosa vuol dire “i migliori valori”, cioè occorre definire il criterio di ottimalità $J(\mathbf{w},w_0)$ da massimizzare

Determinazione di **w** e w_0

Diverse possibilità per il criterio di ottimalità $J(\mathbf{w},w_0)$

⇨ Opzione più semplice: ricercare valori per **w** e w_0 in modo che la funzione discriminante lineare “sbagli” il meno possibile

⇨ Esempio 1: $J(\mathbf{w},w_0)$: numero di campioni mal classificati

⇨ Problema: non tiene conto dell'entità dell'errore

⇨ Esempio 2: $J(\mathbf{w},w_0)$: somma dell'entità degli errori (distanza dal lato giusto del piano)

⇨ Più raffinata

Determinazione di **w** e w_0

- Una volta scelto il criterio di ottimalità $J(\mathbf{w},w_0)$ è necessario ottimizzarlo
  - Esempio: discesa lungo il gradiente

- Diverse altre scelte possibili, sia per il criterio di ottimalità che per la sua ottimizzazione (capitoli 5.4-5.10 del Duda Hart Stork)

  - **Metodo del rilassamento**
  - **Metodo del MSE (minimun square error)**
  - **Metodo del Least MSE o Widrow-Hoff**
  - **Metodo di Ho-Kashyap**

Come si gesticono più classi?

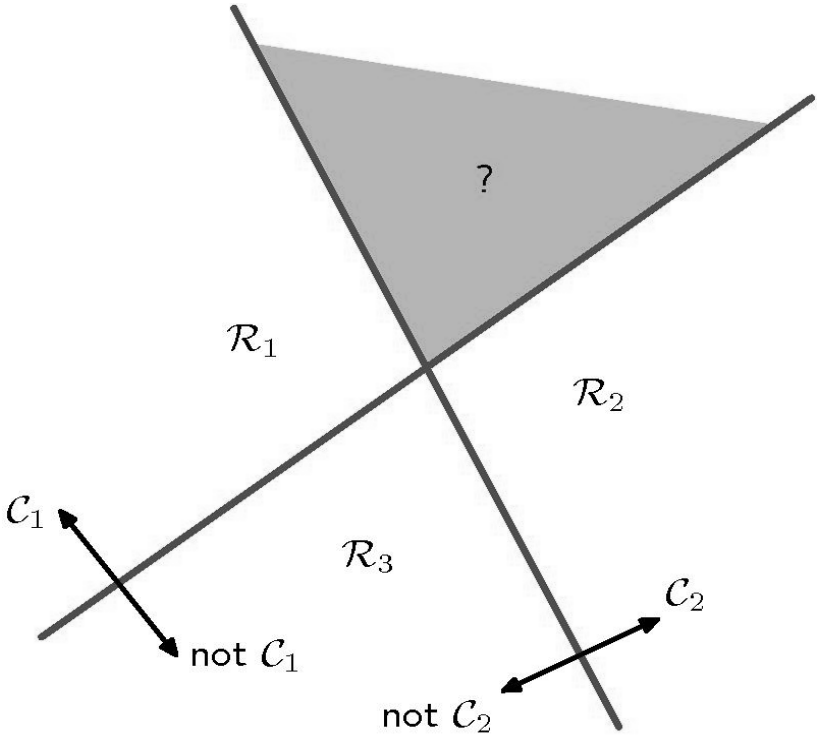
- Problema: le funzioni discriminanti lineari sono classificatori binari (funzionano solo con 2 classi)

- Come si generalizza al caso di C classi?
  - In generale non è banale, il problema rimane ancora aperto nel contesto della pattern recognition

- Le opzioni principali sono 2
  - Opzione *one-vs-rest*
  - Opzione *one-vs-one*

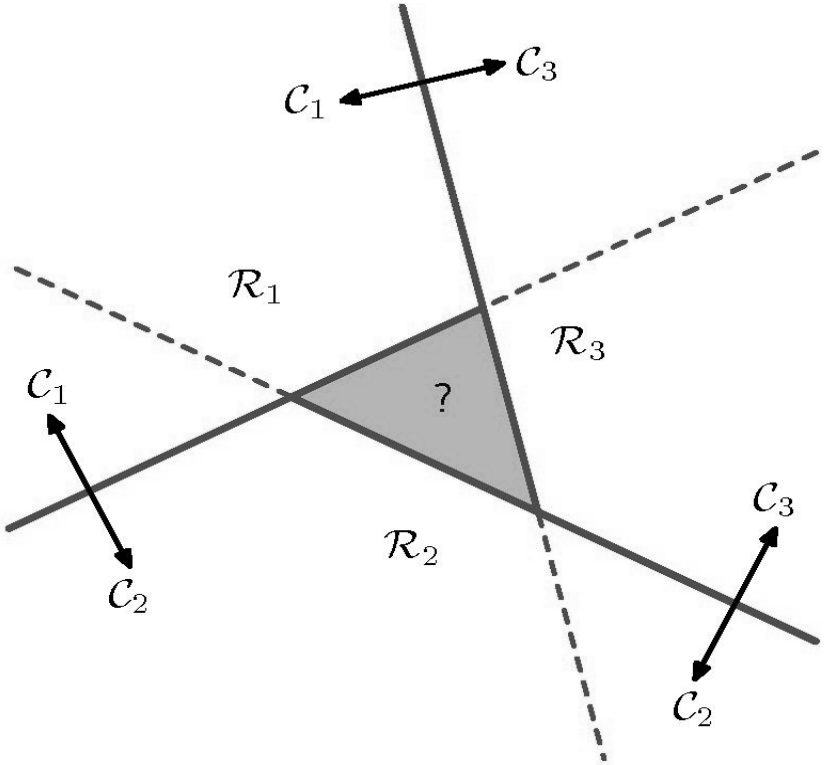
Caso multi classe

- Opzione *one-vs-rest:*
  - TRAINING: si costruiscono C-1 classificatori $g_i(\mathbf{x})$, uno per ogni classe C_i
    - $g_i(x)$ discrimina tra esempi di C_i ed esempi di *"non C_i-"*
  - TESTING: l'oggetto viene classificato con tutti i classificatori e si vede quello che "vince"
- NOTA: rimane comunque una zona di ambiguità



Caso multi classe

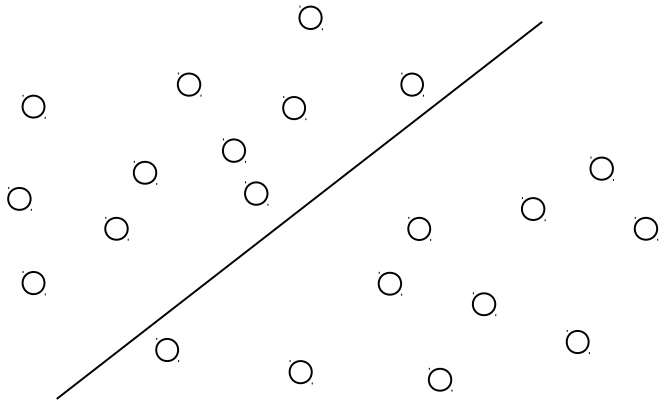
- Opzione *one-vs-one*
  - *TRAINING:* costruire un classificatore binario per ogni coppia di classi
  - TESTING: l'oggetto viene classificato da tutti i classificatori e viene assegnato alla classe che “vince” più volte
  - TESTING (ALTERNATIVO per molte classi): fare un “torneo”
- NOTA: rimane anche qui una zona di ambiguità



Support Vector Machines (SVM)

Introduzione

- Rappresentano i classificatori discriminativi più famosi e utilizzati
  - Concettualmente semplici
  - Potenti ed eleganti
  - Veloci
- Sono classificatori binari (due classi): suddividono lo spazio in due regioni



Introduzione

Applicati in moltissimi campi

Facial recognition, Content–based image retrieval, Facial expression classification, Hand–written text interpretation, 3D object recognition, Texture Classification, Text classification, Traffic prediction, Disease identification, Gene sequencing, Protein folding, Weather forecasting, Earthquake prediction, Automated diagnosis, ...

•Facce, odori, microarray, immagini MRI, EEG, forme 2D, gesti, parlato, spettri NMR, video, oggetti, segnali sismici, tessili, ...

Storia

- Formulazione della fine degli anni 70
- Fine anni 90: algoritmo di training e testing veloce e ottimizzato
  - Ha permesso una loro applicazione su larga scala
- A partire dagli anni 90, impiegate in quasi tutti i problemi di classificazione
  - Hanno sostituito le reti neurali (scatole nere)

Vladimir Vapnik

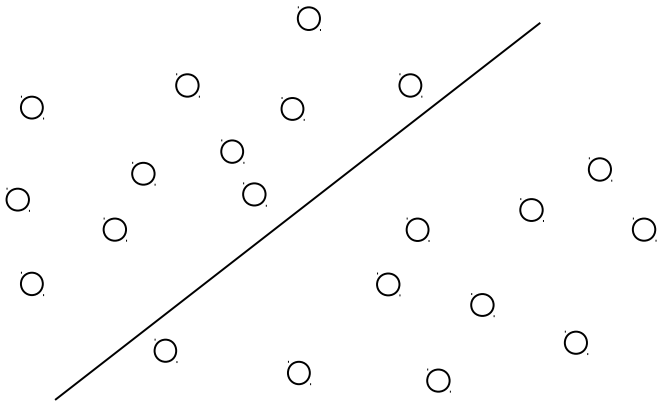
Spiegare le SVM

- Primo punto: le SVM per dati linearmente separabili
- Secondo punto: le SVM per dati non linearmente separabili
- Terzo punto: il trucco del kernel (kernel trick)

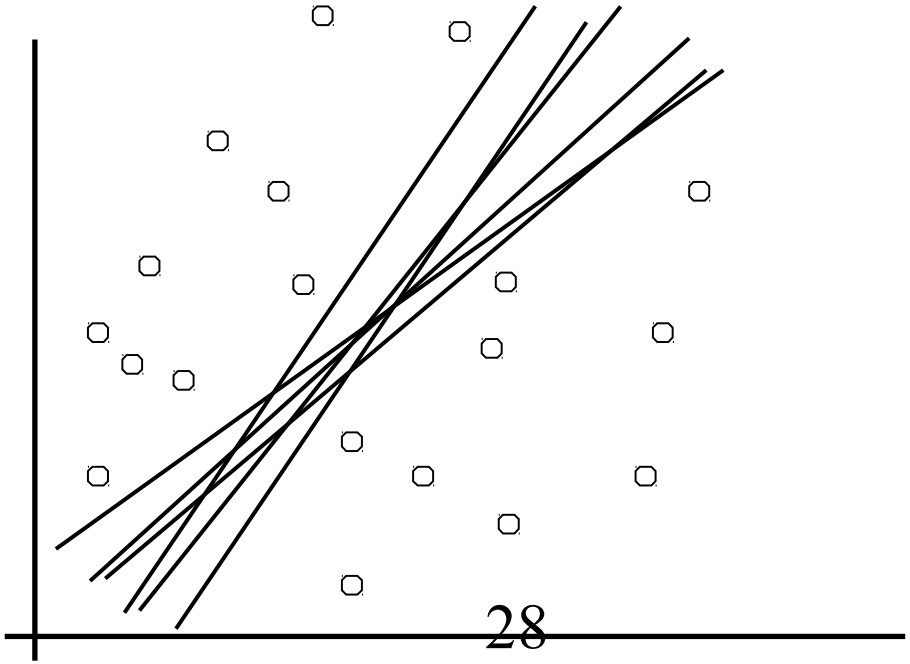
SVM per dati linearmente separabili

SVM (Caso lin. separabili)

- Dati linearmente separabili:
  - Esiste un iperpiano che separa le due classi

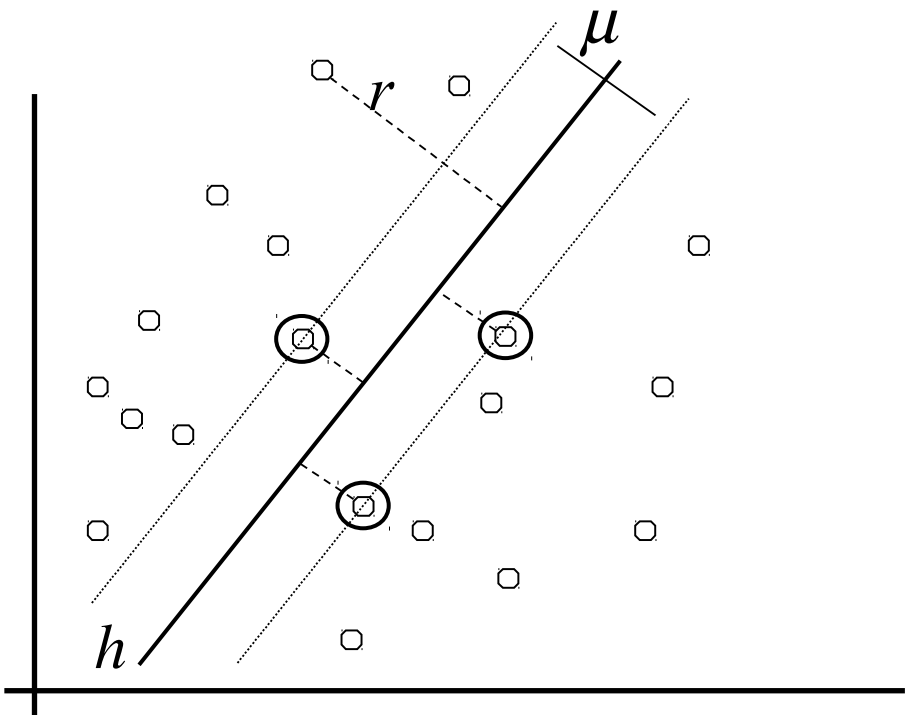


DOMANDA: qual'è il miglior iperpiano?



SVM (Caso lin. separabili)

⇨ Le SVM definiscono l'iperpiano ottimale come quello che massimizza il “margine”

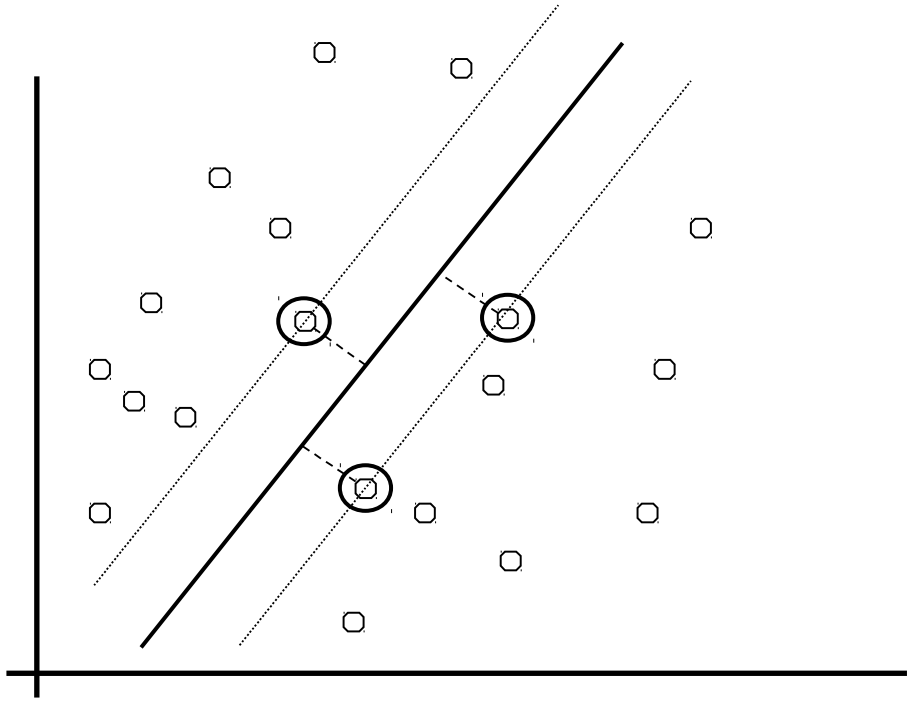


Il margine μ dell’iperpiano di separazione *h* è la distanza <u>minima</u> fra le due classi

SVM (Caso lin. separabili)

- NOTA: solo alcuni degli esempi del training set sono importanti per definire il margine (e quindi l'iperpiano).
- Questi vettori sono detti **vettori di supporto (support vectors)**. Gli altri possono essere ignorati.

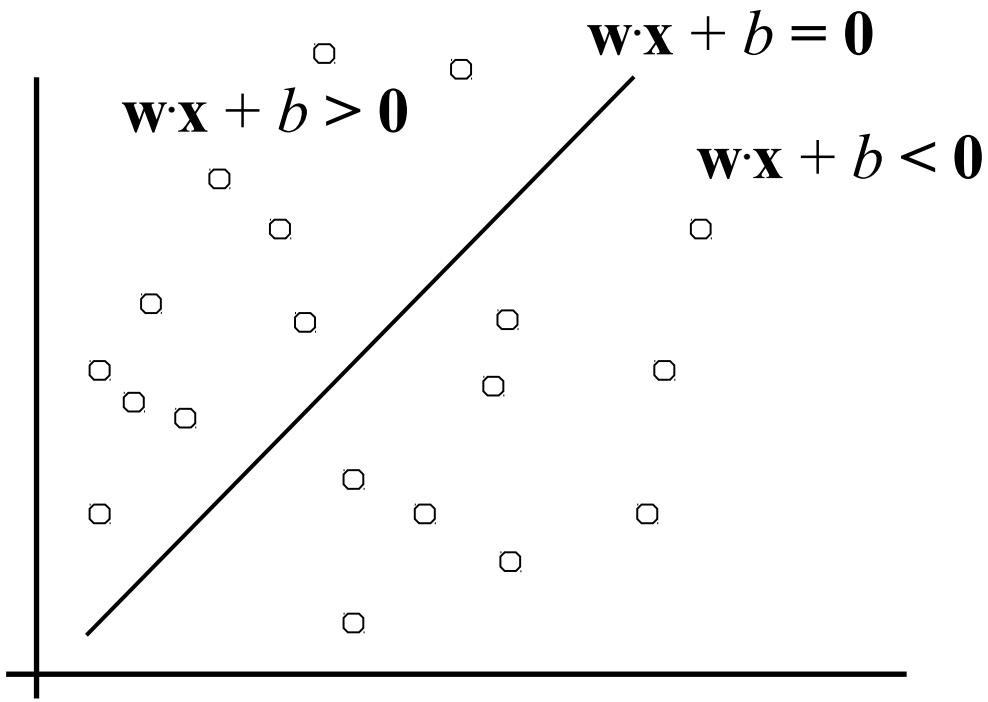
Interpretazione fisica: h è un solido lamellare, i SV sono equivalenti a forze di verso opposto esercitate perpendicolarmente alla superficie per ottenere una condizione di equilibrio



SVM (Caso lin. separabili)

- DETTAGLI: la SVM è un iperpiano (simile ai discriminanti lineari)

$$y(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} + b$$

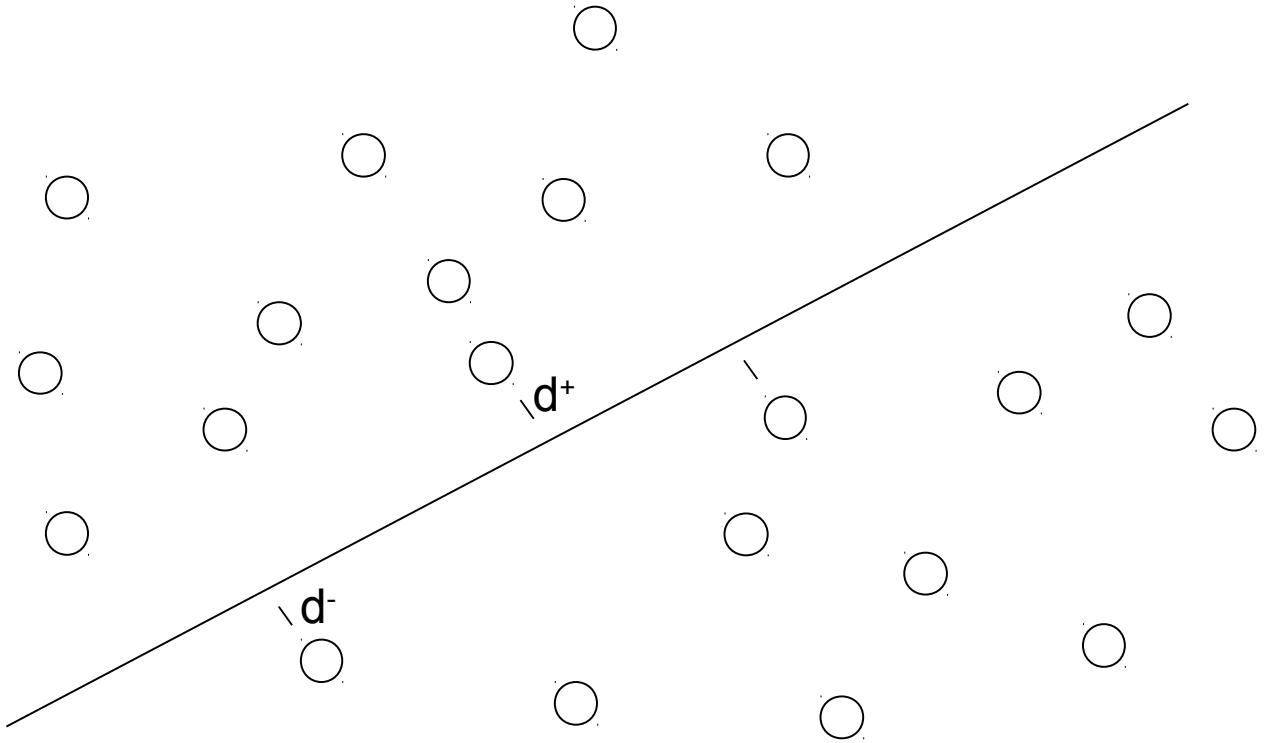


Classificazione di un oggetto: si guarda al segno

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathbf{T}}\mathbf{x} + b) = \begin{cases} +1 \\ -1 \end{cases}$$

SVM (Caso lin. separabili)

- Addestramento della SVM: trovare il vettore w e il parametro b ottimali (che massimizzino il margine), a partire dal training set
- Si può dimostrare che il margine vale $d = d^+ + d^- = \frac{2}{\|w\|}$



$$d^+ = d^- = \frac{1}{\|w\|}$$

SVM (Caso lin. separabili)

- ⇨ VINCOLI: l'iperpiano deve separare le due classi
- ⇨ Dato un insieme di oggetti con le corrispondenti labels

  $\{\mathbf{x}_i, y_i\}$ (yi può valere -1 o +1)

Vincoli:

$$\begin{aligned} (w \cdot x_i + b) &\geq 0, \quad se\ y_i = 1 \\ (w \cdot x_i + b) &< 0, \quad se\ y_i = -1 \end{aligned}$$

In forma più compatta

$$y_i (w \cdot x_i + b) \geq 0, \quad \forall i \in \{1, \ldots, m\}$$

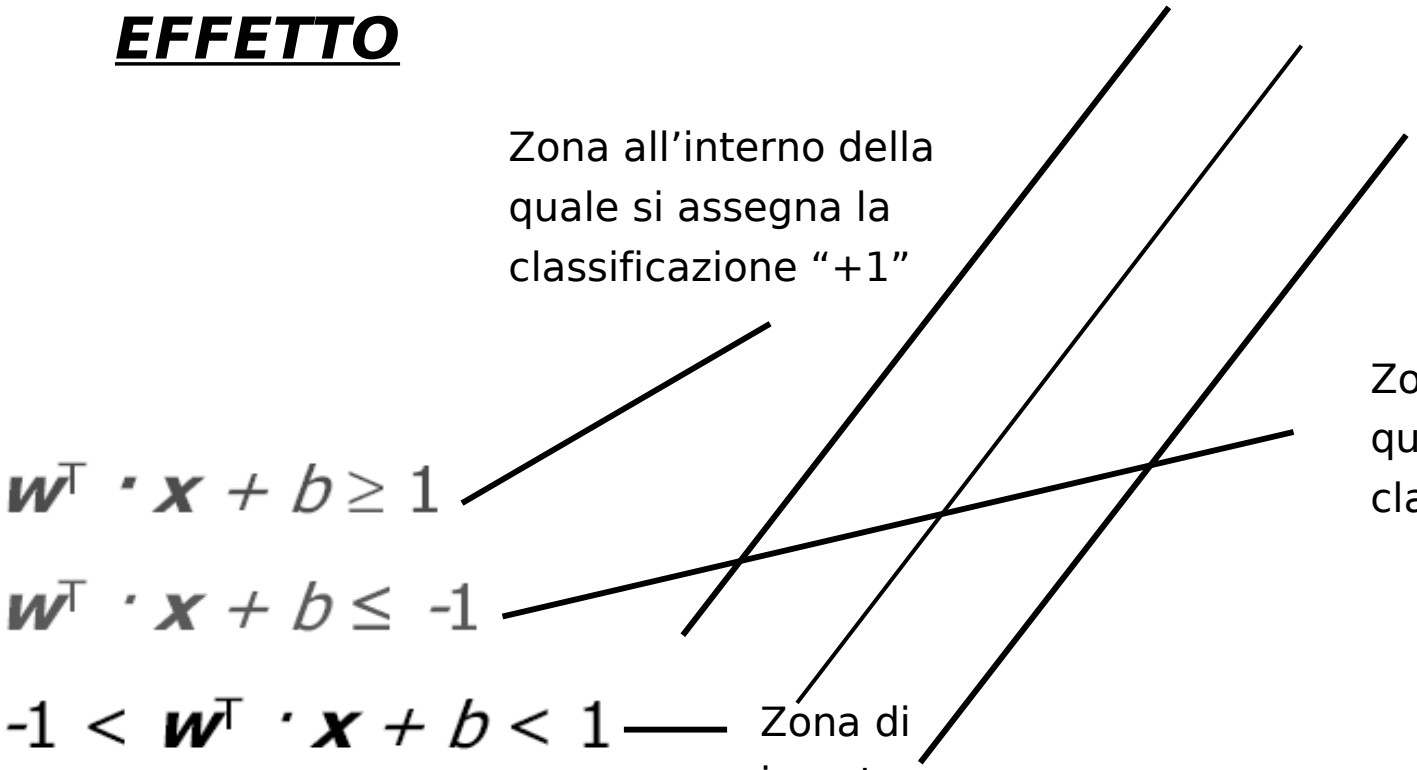
SVM (Caso lin. separabili)

⇨ Si può anche “rinforzare” il vincolo

$$y_i(w \cdot x_i + b) \geq 1, \ \ \forall\, i \in \{1, \ldots, m\}$$

(se troviamo una soluzione per cui vale il maggiore di 1, allora sicuramente vale anche il maggiore di zero)

EFFETTO



SVM (Caso lin. separabili)

⇨ Quindi si ottiene un problema di ottimizzazione quadratica con vincoli

Trovare $\mathbf{w}$ e b che massimizzino

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

VINCOLI: per ogni $\{(\mathbf{x_i}, y_i)\} \in \mathrm{D}$: $\quad y_i(\mathbf{w}\cdot\mathbf{x_i} + b) \geq 1$

Formulazione equivalente

Trovare $\mathbf{w}$ e b che minimizzino

$$\tfrac{1}{2}\,\|\mathbf{w}\| \qquad \text{cioè} \qquad \tfrac{1}{2}\,\mathbf{w}\cdot\mathbf{w}$$

VINCOLI: per ogni $\{(\mathbf{x_i}, y_i)\}$: $\quad y_i(\mathbf{w}\cdot\mathbf{x_i} + b) \geq 1$

SVM (Caso lin. separabili)

- Si deve ottimizzare una funzione quadratica soggetta a vincoli lineari (uno per ogni $\mathbf{x}_i$):

$$y_i\,(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$$

- I problemi di ottimizzazione quadratica sono problemi ben noti, per i quali esistono vari algoritmi.
  - Esiste un'unica soluzione!!

- In questo caso si utilizzano i Lagrangiani
  - I vincoli vengono integrati nella funzione da massimizzare

SVM (Caso lin. separabili)

⇨ Occorre minimizzare il seguente *Lagrangiano*:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1), \;\; \alpha_i \geq 0$$

⇨ dove $\boldsymbol{\alpha}_i$ sono i moltiplicatori di lagrange, uno per ogni vincolo (i.e. uno per ogni punto)

⇨ La soluzione per w:

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

NOTE:

1. w si ottiene come combinazione lineare dei punti del training set
2. solo i punti per cui $\boldsymbol{\alpha}_i$ non è zero contano
   ⇨ Sono i SUPPORT VECTOR!

SVM (Caso lin. separabili)

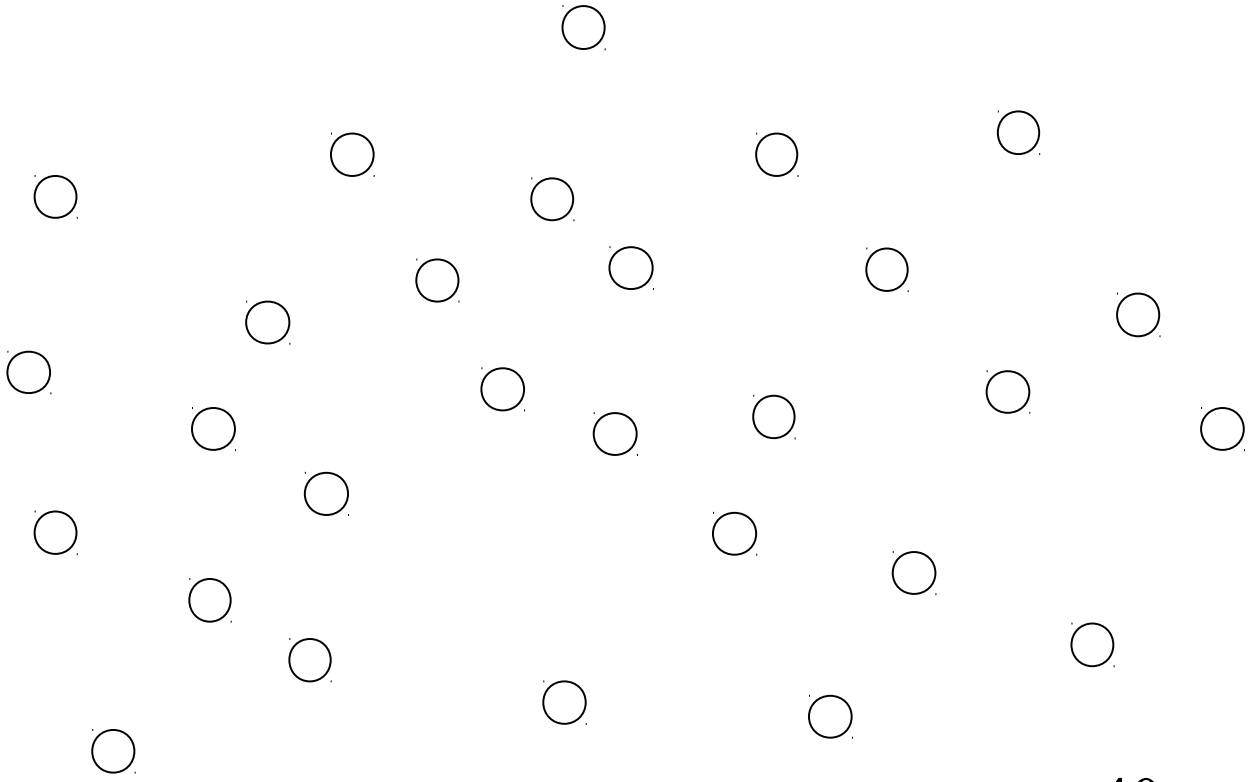
SOMMARIO:

- ⇨ SVM sono classificatori binari
- ⇨ Trovano l'iperpiano che massimizza il margine
- ⇨ L'iperpiano è descritto da un numero ridotto di esempi del training set (i support vector)
- ⇨ La soluzione trovata è ottimale (minimizzazione convessa)
- ⇨ Il metodo non è probabilistico (geometrico)

SVM per dati non linearmente separabili

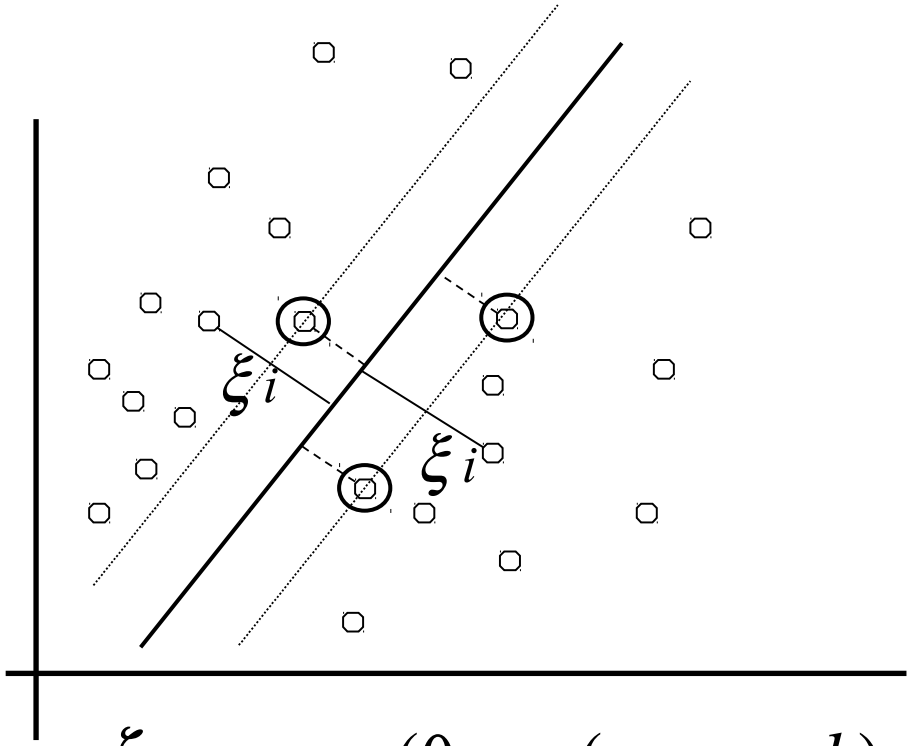
SVM (caso non lin sep.)

- Cosa succede se i dati non sono linearmente separabili?
  - Non c'è modo di soddisfare i vincoli



SVM (caso non lin sep.)

⇨ *Si introducono le slack variables* ξ_i che consentono la classificazione errata di qualche punto.



$$\xi_i = \max\ (0, -y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)$$

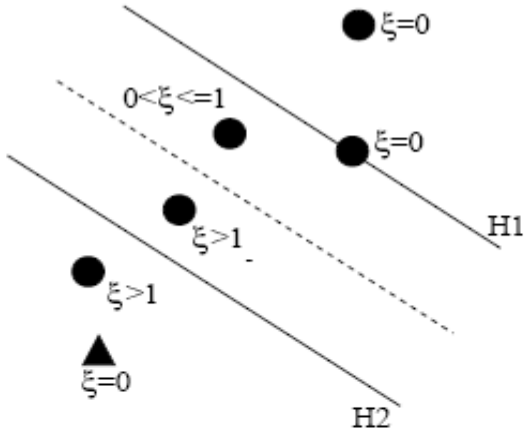
Il vincolo diventa

$$y_i\left(w \cdot x_i + b\right) \geq 1 - \xi_i$$

In questo modo i punti possono attraversare il margine (pagando un prezzo di ξ_i)

SVM (caso non lin sep.)

⇨ Il valore di ξ_i indica la posizione del vettore rispetto all'iperpiano



$\xi_i = 0$ classificazione corretta

$0 < \xi_i <= 1$ classificazione corretta ma nella zona del margine

$\xi_i > 1$ errore di classificazione

SVM (caso non lin sep.)

- Ovviamente è necessario minimizzare il numero di errori (minimizzare il prezzo da pagare)

Quindi....

Trovare $\mathbf{w}$ e b che minimizzino

$$\tfrac{1}{2}\ \mathbf{w}\cdot\mathbf{w} + C\sum \xi_i$$

VINCOLI: $y_i\ (\mathbf{w}\cdot\mathbf{x_i} + b) \geq 1 - \xi_i$ e $\xi_i \geq 0$ per ogni i

Il parametro C “pesa” gli errori (controlla l’overfitting)

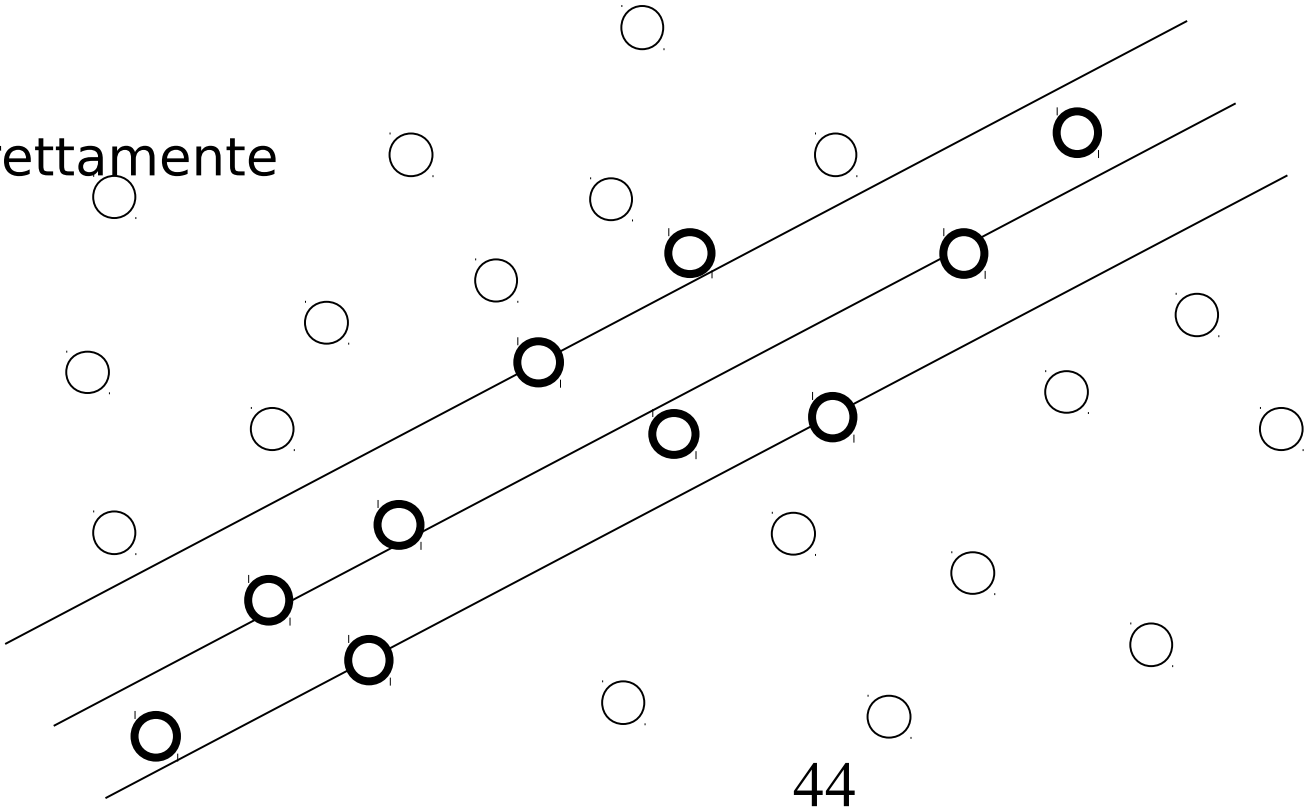
SVM (caso non lin sep.)

⇨ La soluzione è di nuovo $w = \sum_{i=1}^{N} \alpha_i y_i x_i$

Ora i support vectors sono i punti

per cui α_i è diverso da zero

- ⇨ I punti sul margine
- ⇨ I punti classificati correttamente ma oltre il margine
- ⇨ I punti classificati non correttamente



SVM (caso non lin sep.)

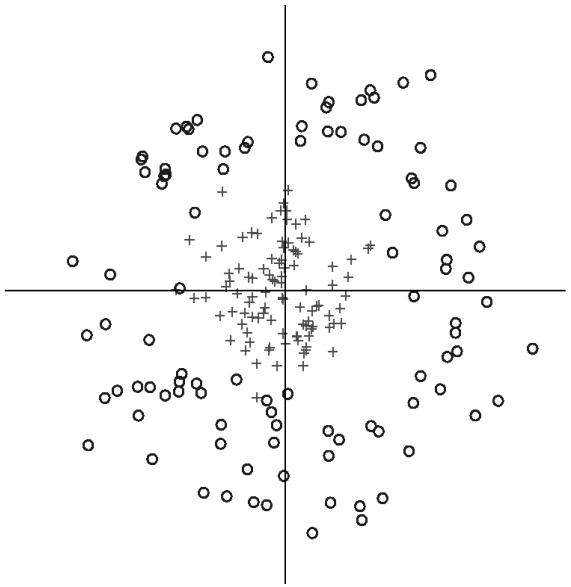
SOMMARIO

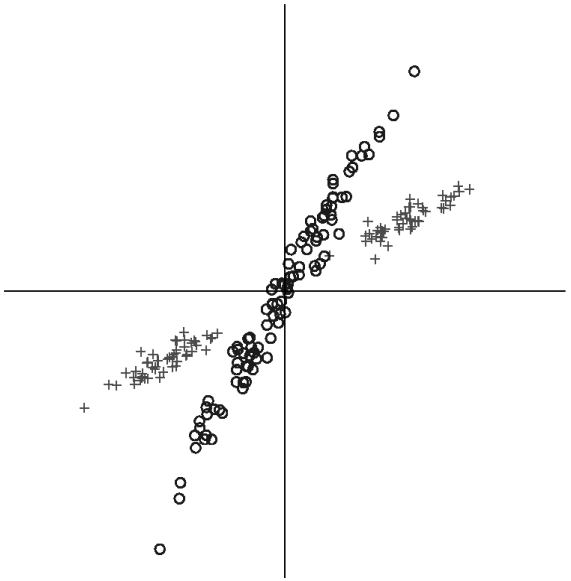
- In caso di dati non linearmente separabili vengono introdotte le variabili slack per permettere ad alcuni punti di attraversare il margine
- Occorre settare il parametro C, che definisce il costo della classificazione errata
- I support vector ora sono i punti sul margine e *oltre* il margine

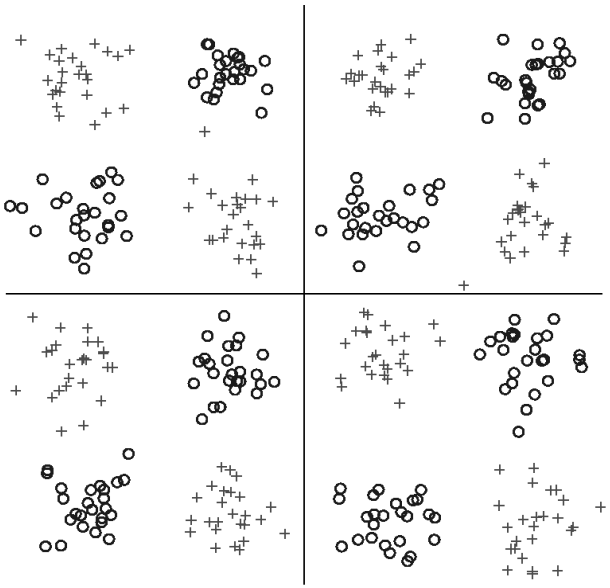
Il trucco del kernel
(kernel trick)

Kernel trick

- Ci sono casi in cui un iperpiano rappresenterebbe una soluzione troppo semplicistica

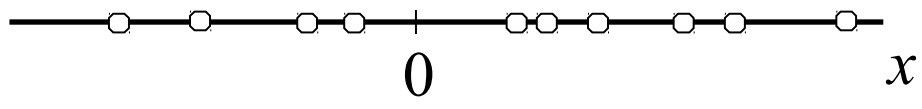


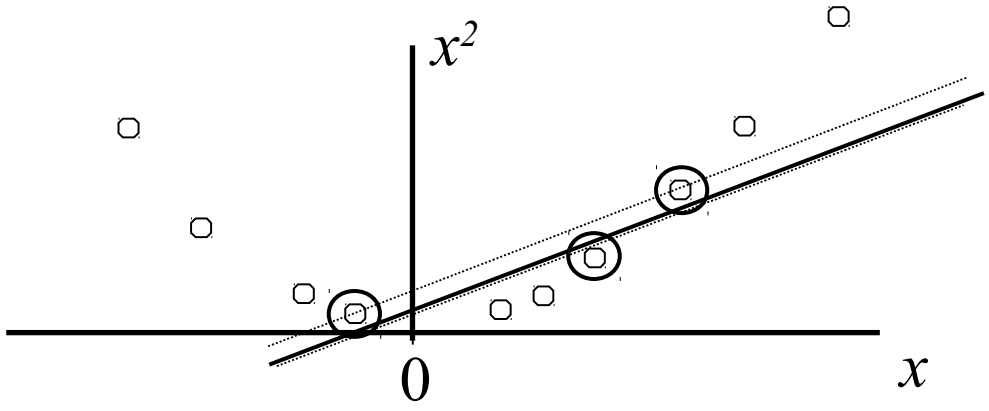




Kernel trick

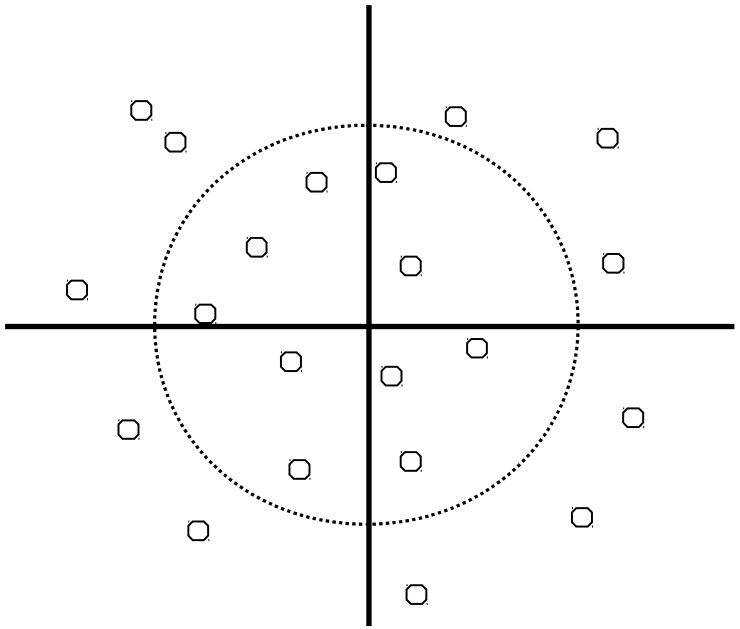
- Possibile soluzione: proiettare i punti in uno spazio a dimensione maggiore dove i punti possano essere separati più facilmente





Kernel trick

⇨ Esempio 1

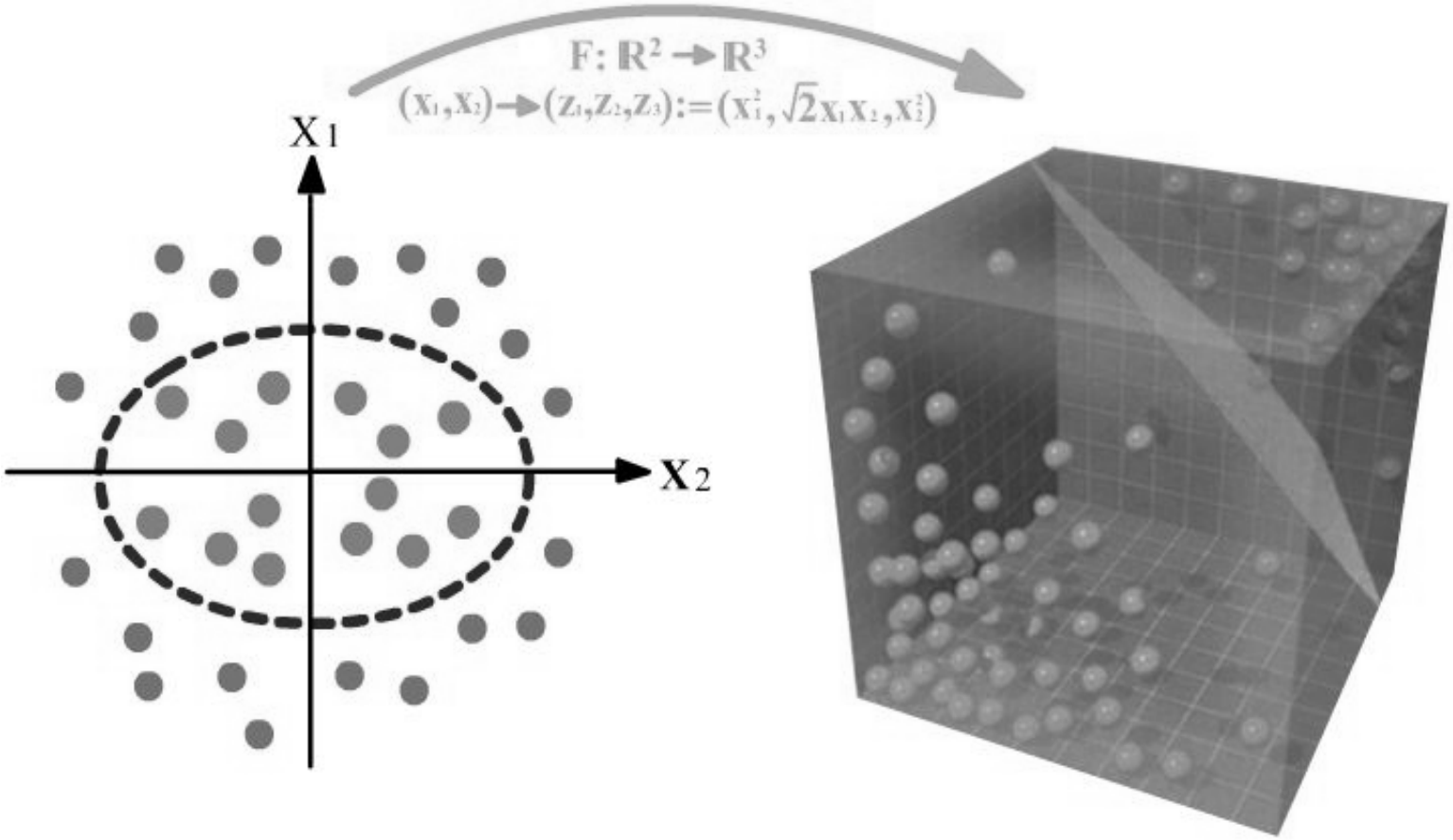


Proposta: applicare il seguente mapping:

$$\phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{pmatrix}$$

Si passa da uno spazio a dimensione 2 ad uno spazio a dimensione 3

Kernel trick



Nel nuovo spazio i punti sono linearmente separabili

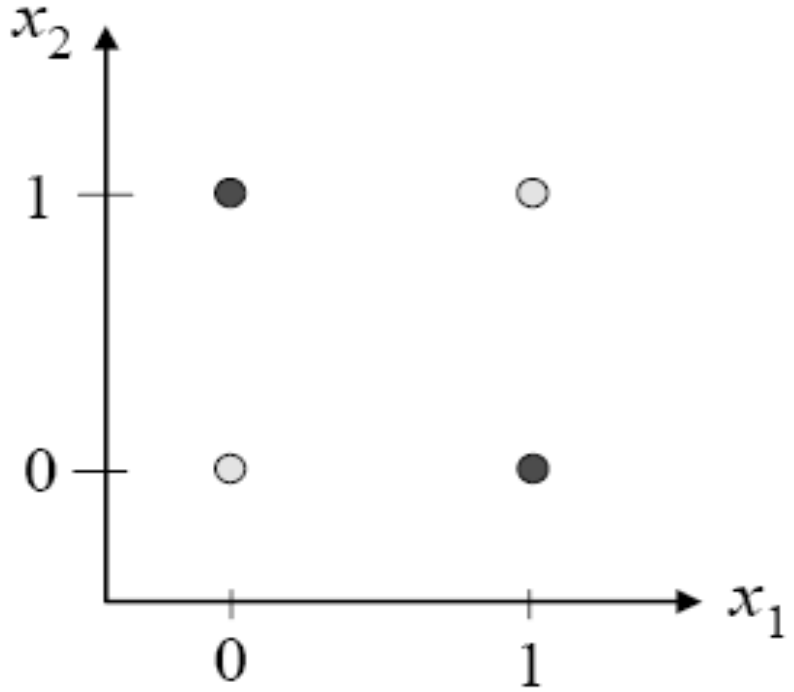
Kernel trick

p	q	p XOR q
0	0	0
0	1	1
1	0	1
1	1	0

- Esempio 2: la funzione logica XOR
  - “p XOR q” è vera se e solo se uno dei due è vero

Può essere codificato come un problema di classificazione:

- Features: Falso → 0, Vero → 1
- Classi: (x1,x2) → x1 XOR x2
  (vero classe rossa, falso classe gialla)



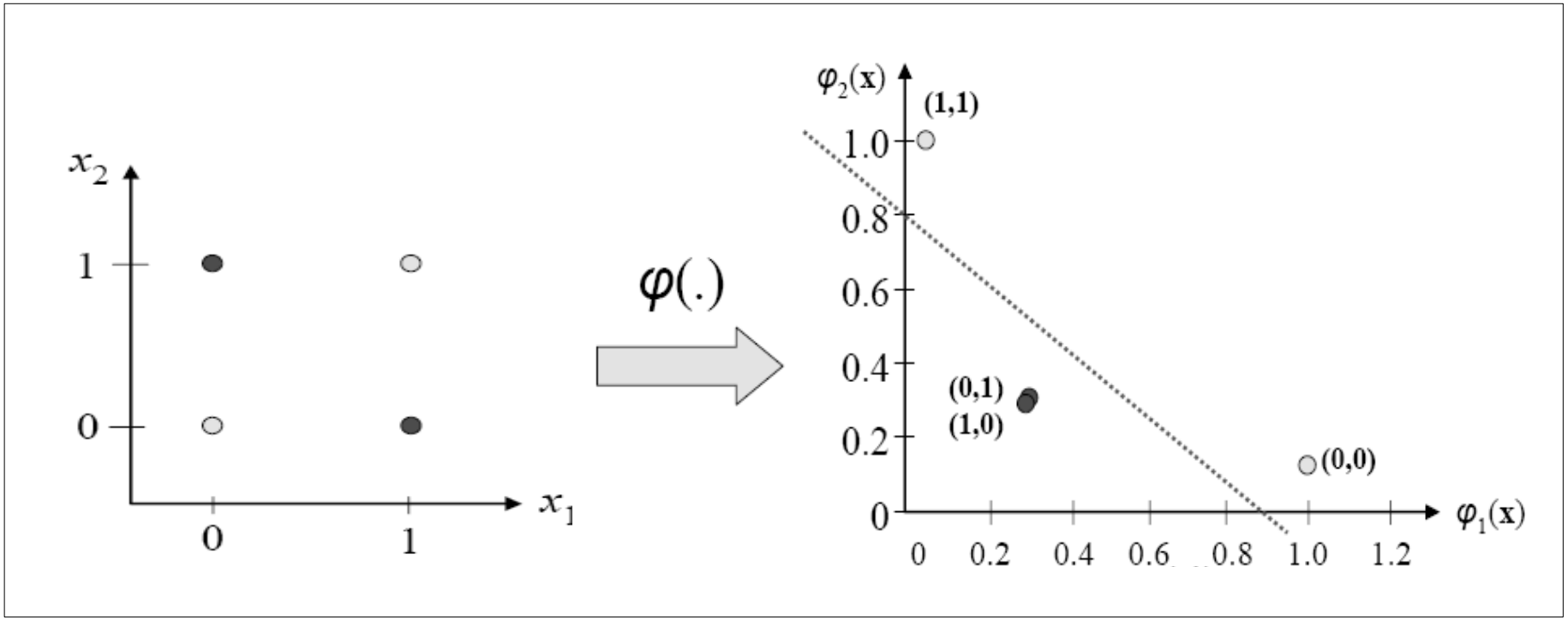
Problema di classificazione non linearmente separabile!

Kernel trick

- Possiamo trovare un mapping Φ(x) tale che il problema nel nuovo spazio sia separabile? SI'!

$$x=[x_1, x_2]$$

$$\phi_1(x)=e^{-\|x-[1,1]^T\|} \qquad \phi_2(x)=e^{-\|x-[0,0]^T\|}$$



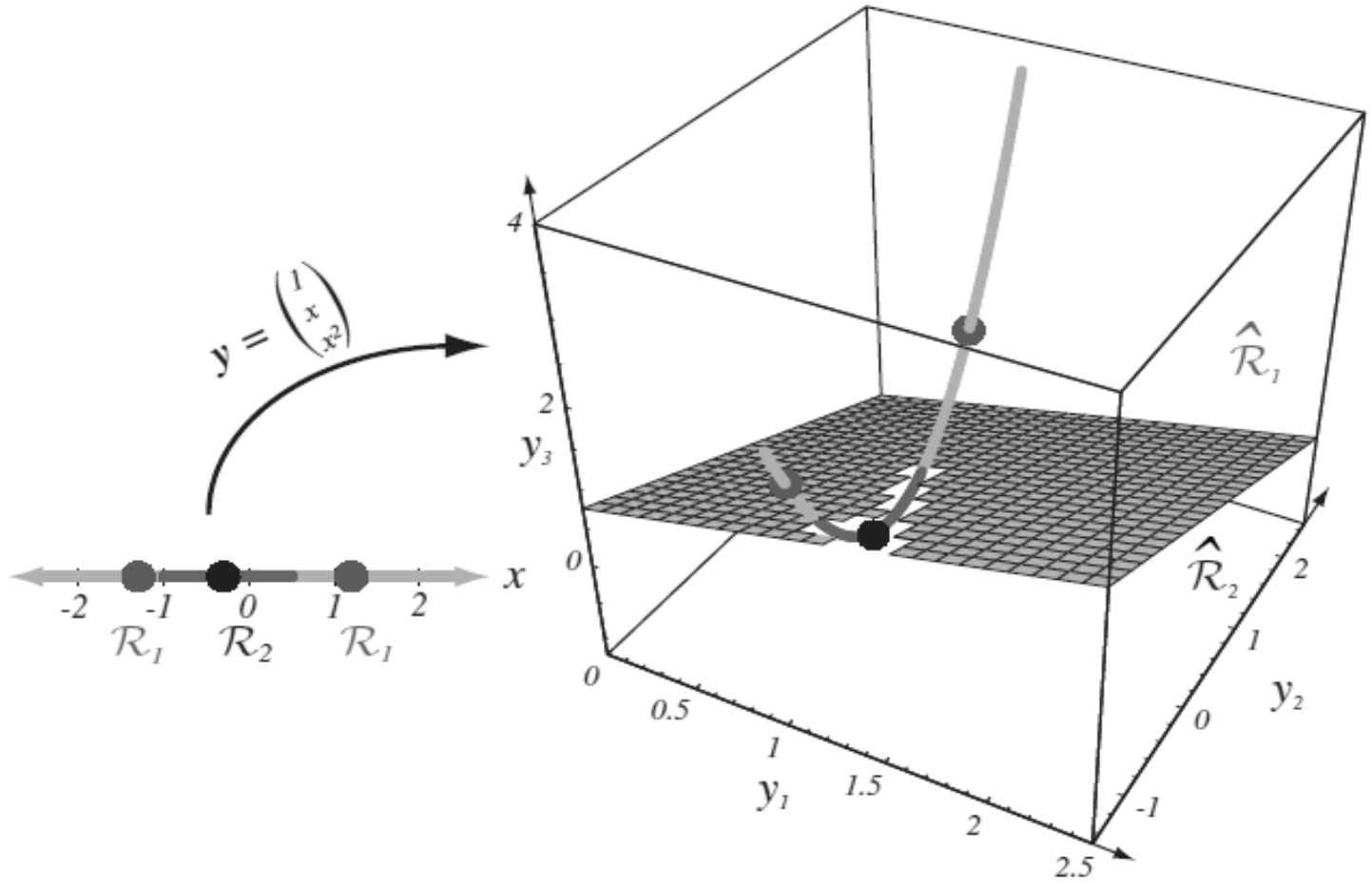


FIGURE 5.5. The mapping $\mathbf{y} = (1, x, x^2)^t$ takes a line and transforms it to a parabola in three dimensions. A plane splits the resulting $\mathbf{y}$-space into regions corresponding to two categories, and this in turn gives a nonsimply connected decision region in the one-dimensional x-space. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Kernel trick

QUINDI: Proiettare i punti in uno spazio a dimensione maggiore potrebbe migliorare le capacità discriminative...

MA....

⇨ Come si gestisce la curse of dimensionality?

⇨ Come si sceglie il mapping?

SOLUZIONE: kernel trick

Kernel trick

OSSERVAZIONE: il problema da risolvere per addestrare una SVM

Trova **w** e b che minimizzino

1/2 w·w

VINCOLI: per ogni $\{(\mathbf{x}_i, y_i)\}$: $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$

può essere riscritto come

$$\underset{\alpha_i}{maximize}\left(\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i \cdot x_j\right)$$

VINCOLI $\sum_{i=1}^{N}\alpha_i y_i = 0 \quad \alpha_i \geq 0 \,\forall\, i$

Kernel trick

In fase di testing, la classificazione di un punto avviene con:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathbf{T}}\mathbf{x} + b) = \begin{cases} +1 \\ -1 \end{cases}$$

Per classificare un punto $\mathbf{x}$ occorre quindi calcolare $\mathbf{w}\cdot\mathbf{x}$

Ricordando che

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

abbiamo

$$w \cdot x = \sum_{i=1}^{N} \alpha_i y_i x_i \cdot x$$

Kernel trick

QUINDI: i punti del training set, nel training e nel testing di una SVM, non appaiono mai da soli ma sempre in prodotto scalare con altri punti

$$\underset{\alpha_i}{maximize}\left(\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \boxed{x_i\cdot x_j}\right) \qquad w\cdot x=\sum_{i=1}^{N}\alpha_i y_i \boxed{x_i\cdot x}$$

Applicando il mapping Φ(x) per proiettare i punti in uno spazio a dimensionalità maggiore si ottiene

$$\underset{\alpha_i}{maximize}\left(\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \phi(x_i)\cdot\phi(x_j)\right) \qquad w\cdot x=\sum_{i=1}^{N}\alpha_i y_i \phi(x_i)\cdot\phi(x)$$

Kernel trick

KERNEL TRICK: definire una funzione K (chiamata kernel) t.c.

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

In questo modo si può addestrare e testare una SVM senza calcolare in maniera esplicita il mapping Φ(x)

$$\underset{\alpha_i}{maximize} \left(\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \right)$$

$$w \cdot x = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x)$$

Kernel trick

Esempio: vettori a 2 dim. $\mathbf{x}=[x_1 \; x_2]$;

$\Phi(\mathbf{x}) = [1 \; x_1^2 \; \sqrt{2}\, x_1 x_2 \; x_2^2 \; \sqrt{2}x_1 \; \sqrt{2}x_2]$

$\Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j}) =$

$= [1 \; x_{i1}^2 \; \sqrt{2}\, x_{i1}x_{i2} \; x_{i2}^2 \; \sqrt{2}x_{i1} \; \sqrt{2}x_{i2}]^T [1 \; x_{j1}^2 \; \sqrt{2}\, x_{j1}x_{j2} \; x_{j2}^2 \; \sqrt{2}x_{j1} \; \sqrt{2}x_{j2}]$

$= 1 + x_{i1}^2 x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$

$= (1 + \mathbf{x_i} \cdot \mathbf{x_j})^2$

Possiamo quindi definire $K(\mathbf{x_i},\mathbf{x_j}) = \Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j}) = (1 + \mathbf{x_i} \cdot \mathbf{x_j})^2$

Kernel trick

- In realtà non è necessario calcolare $\Phi(x)$, mi basta definire una funzione K che rappresenti il prodotto scalare di due punti in uno spazio ad una dimensionalità elevata

TEOREMA DI MERCER: enuncia le caratteristiche necessarie affinché una funzione qualsiasi sia un kernel

Kernel trick

VANTAGGI

1. Usando un kernel, una SVM può lavorare in uno spazio a dimensionalità enorme (potenzialmente infinita) utilizzando sempre gli stessi algoritmi
   - L'iperpiano nello spazio a dimensionalità grande diventa una superficie complessa nello spazio originale
2. Il mapping $\Phi(x)$ non viene mai calcolato esplicitamente, quindi può essere complicato quanto si vuole
3. Non ci sono problemi di curse of dimensionality, perchè l'algoritmo è indipendente dalla dimensione dello spazio finale (lavora nello spazio originale dei punti)

Occorre scegliere la funzione kernel!

Kernel trick

Esempi di kernel

Lineare $K(x,z)=x\cdot z$

Polinomiale $K(x,z)=(x\cdot z+1)^p$

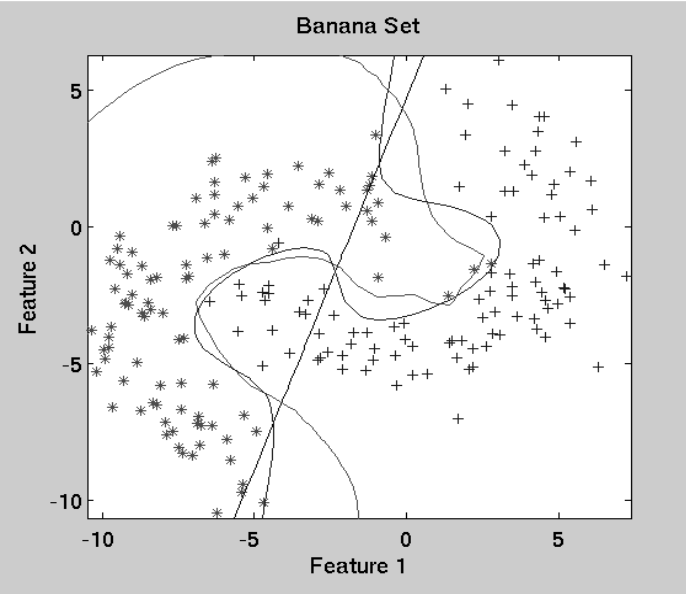
Radial basis function $K(x,z)=e^{-\frac{\|x-z\|^2}{2\sigma^2}}$

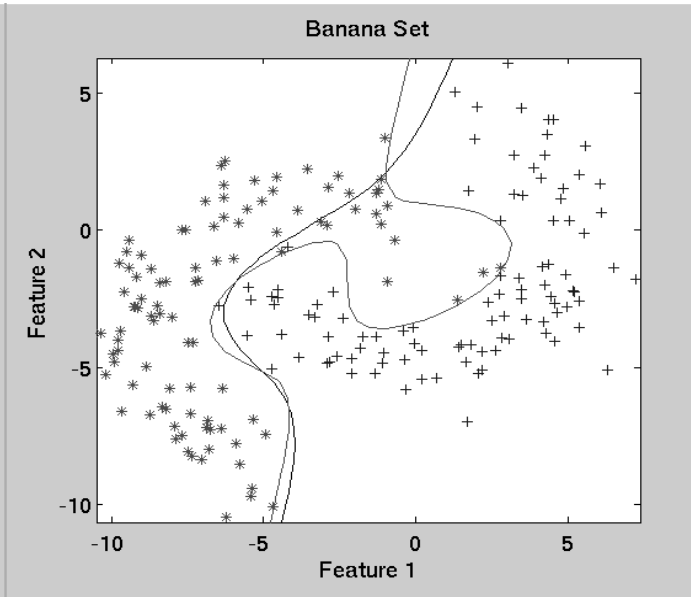
Sigmoide $K(x,z)=\tanh(a\,x\cdot z+b)$

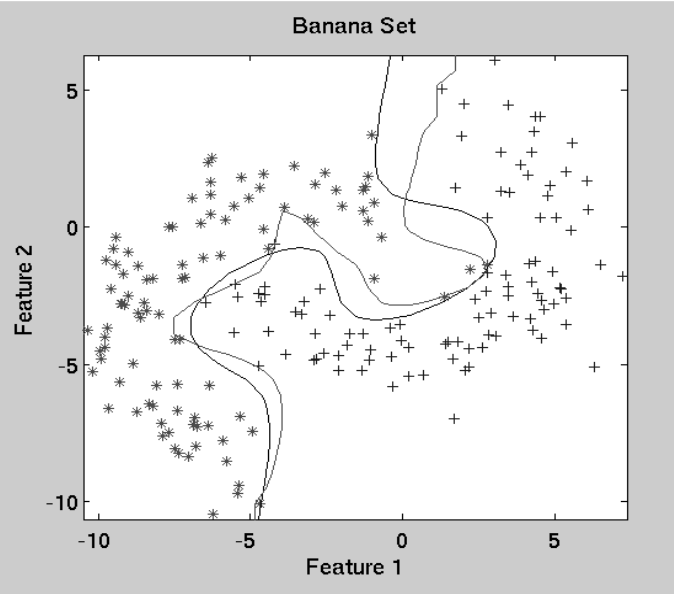
Nota

- Nella versione più flessibile delle SVM (con la formulazione per dati non separabili e i kernel) occorre fare diverse scelte:
  - Il tipo di kernel
  - Il parametro (o i parametri) del kernel
  - Il parametro C (costo delle mis-classificazioni)

- Queste scelte portano a superfici di separazione diverse







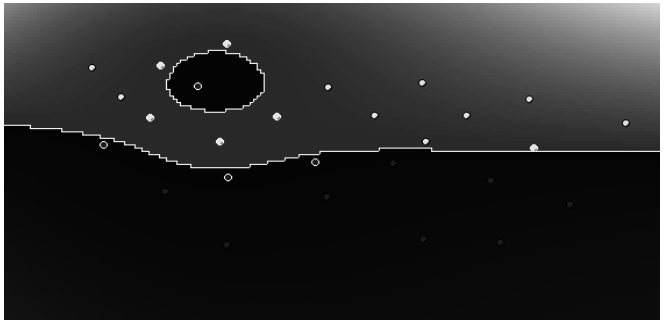
Kernel lineare
Kernel polinomiale
Kernel rbf

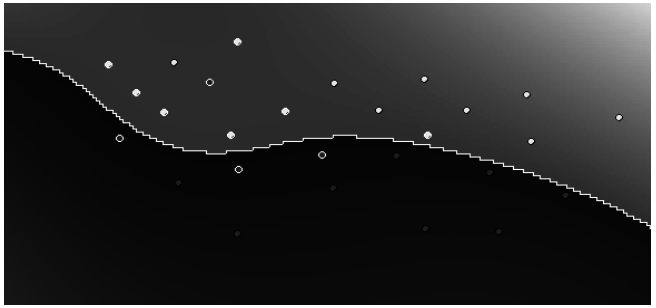
Kernel polinomiale:
C = 0.0001
C = 100

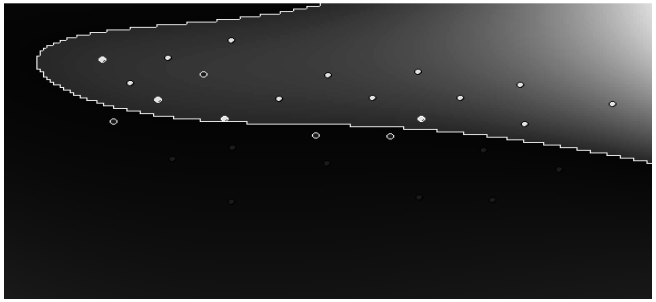
Kernel polinomiale:
P = 3
P = 5

Commenti

- La selezione del kernel (e dei suoi parametri) è critica
  - Kernel lineare se lo spazio delle feature è grande (e.g. Testi)
  - Kernel rbf tipicamente è il migliore
- Il parametro C ha un impatto molto forte sulla capacità di generalizzazione
- Il numero di support vector fornisce un bound dell'errore di generalizzazione







SVM: conclusioni

- Classificatore binario
- Idee principali
  - Massimizzazione del margine
  - Slack variables
  - Kernel trick
- Addestramento: ottimizzazione convessa con vincoli
  - Soluzione ottimale SEMPRE
- Parametri liberi:
  - Tipo e parametri del kernel
  - Costo C

Vantaggi

- SVM hanno una interpretazione geometrica semplice
- Il kernel trick permette di ottenere in modo efficiente un classificatore non lineare senza incorrere nella curse of dimensionality
- La soluzione del training è ottimale
- I support vectors danno una rappresentazione compatta del training set (il loro numero fornisce un'idea della capacità di generalizzazione)
- In moltissimi contesti applicativi funzionano decisamente bene
- (non vista) La teoria delle SVM (Statistical learning theory) è elegante e solida

Svantaggi

- La scelta di kernel e parametri è cruciale
- A volte il training può essere oneroso in termini di tempo
- Il training non è incrementale (arriva un nuovo oggetto occorre rifare da capo l'addestramento)
- La gestione del caso multiclasse non è ottimale (sia in termini di efficienza che in termini di soluzione proposta)