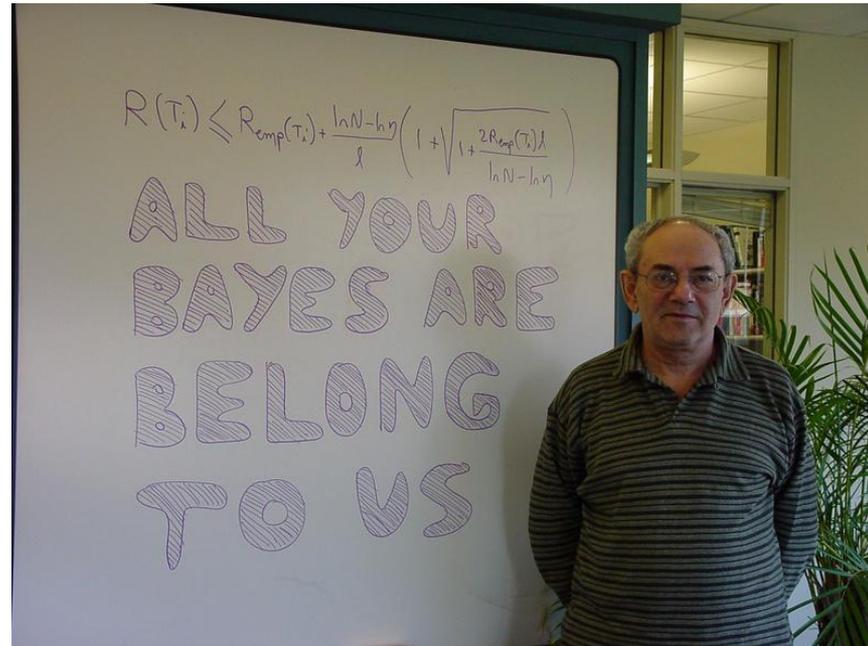


# Sistemi avanzati per il Riconoscimento



<http://www.cs.columbia.edu/~yingbo/misc/vapnik.jpg>

## Support Vector Machines

Dr. Marco Cristani



# Introduzione

- I classificatori *generativi*
  - modellano le probabilità condizionali (likelihood) e a priori delle classi, per ogni classe indipendentemente
  - La regola di decisione di Bayes mi permette di classificare, prendendo la massima posterior
- I classificatori *discriminativi*
  - stimano direttamente il confine di decisione, producendo direttamente una stima di classificazione
  - Sono approcci geometrici (possono essere trovati parallelismi tra questi e la regola di Bayes, ma non è semplice)
- Qui vedremo:
  - **Funzioni discriminanti lineari**
  - **Support Vector Machines**

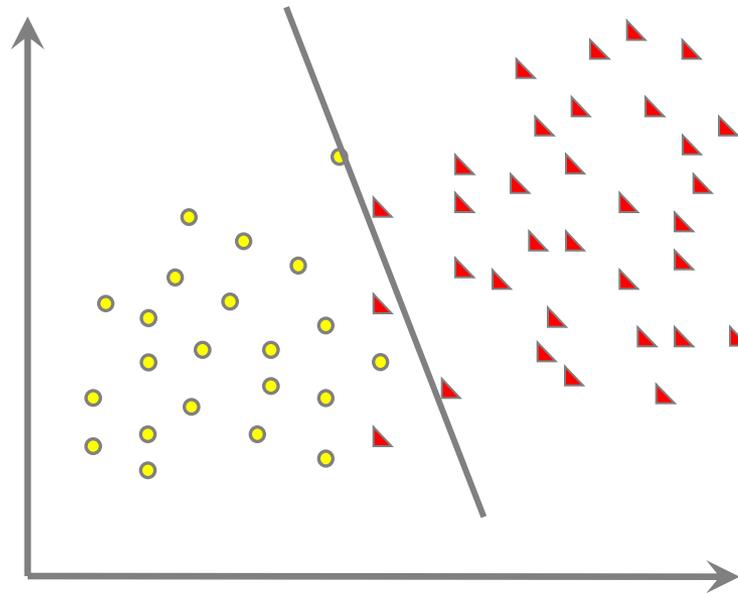


# Funzioni discriminanti lineari



# Funzioni discriminanti lineari

- In un problema binario, l'obiettivo è trovare la retta che separa le due classi



- Visto finora per distribuzioni gaussiane, il compito è ora quello di generalizzare.



# Funzioni discriminanti lineari

- Definiamo la funzione discriminante lineare  $g : \mathcal{R}^d \rightarrow \mathcal{R}$

$$g(\mathbf{x}') = \mathbf{w}'^t \mathbf{x}' + w_0$$

dove  $\mathbf{x}' = [x_1, \dots, x_d]$ ,  $\mathbf{w}' = [w_1, \dots, w_d]$  (pesi) e  $w_0$  *bias* o *termine noto*.

- Un campione  $\mathbf{x}'$  è classificato come appartenente a  $\omega_1$  se  $\mathbf{w}'^t \mathbf{x}' + w_0 > 0$ , a  $\omega_2$  se  $\mathbf{w}'^t \mathbf{x}' + w_0 < 0$*
- Osserviamo geometricamente la funzione discriminante lineare  $g(\mathbf{x}')$



# Funzioni discriminanti lineari

- Settando

$$g(\mathbf{x}') = \mathbf{w}'^t \mathbf{x}' + w_0 = 0$$

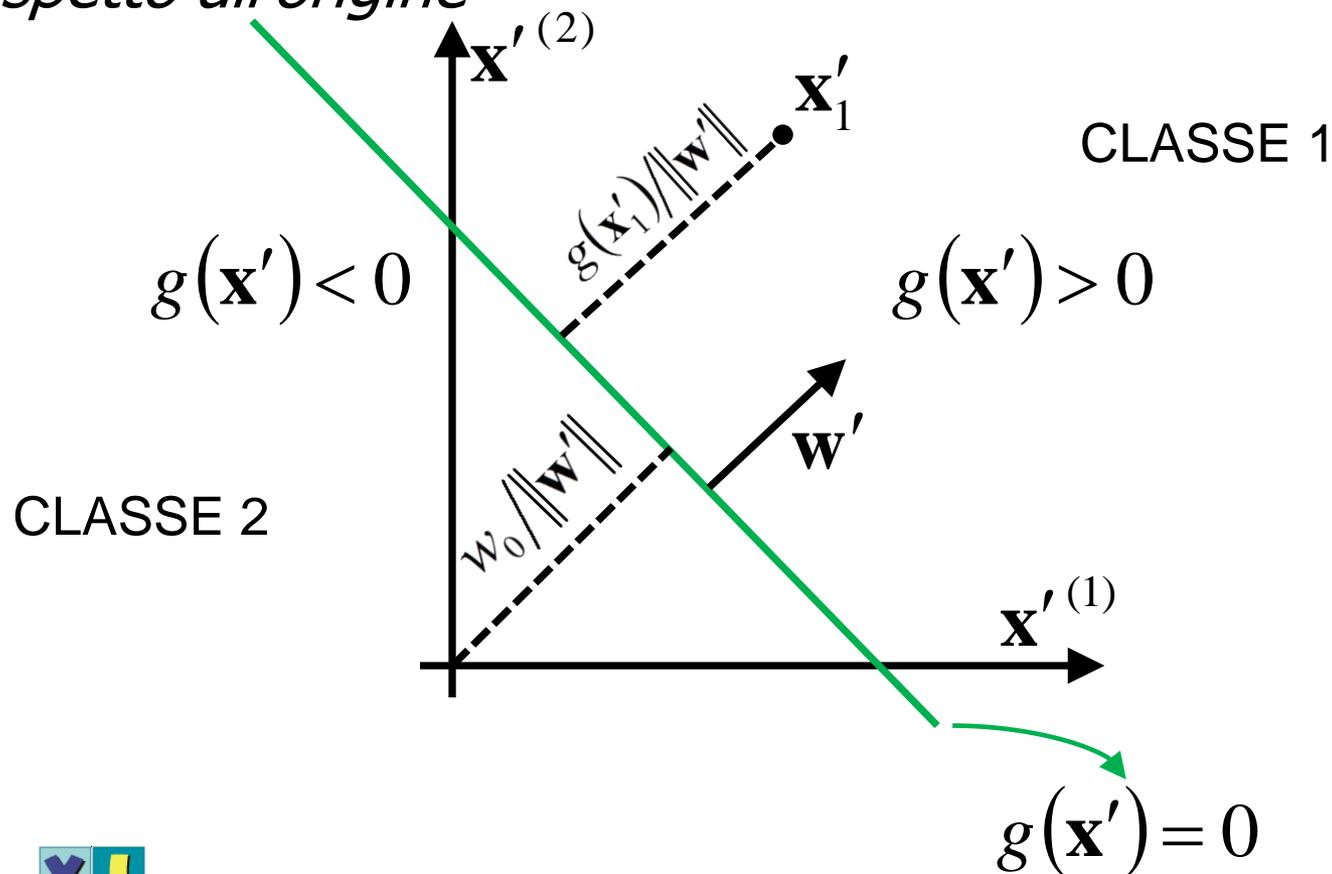
otteniamo un *iperpiano*, o *superficie di separazione*

- Un iperpiano è
  - Un punto in 1D
  - Una linea in 2D
  - Un piano in 3D



# Funzioni discriminanti lineari

- $\mathbf{w}'$  determina l'orientazione del piano
- $w_0$ , assieme a  $\mathbf{w}'$ , determina la distanza del piano rispetto all'origine



# Funzioni discriminanti lineari

- Più genericamente, consideriamo come

$$\mathbf{w} = [w_0, w_1, \dots, w_d]$$

e

$$\mathbf{x} = [1, x_1, \dots, x_d]$$

inserendo così il termine noto.

- La regola di decisione diventa:

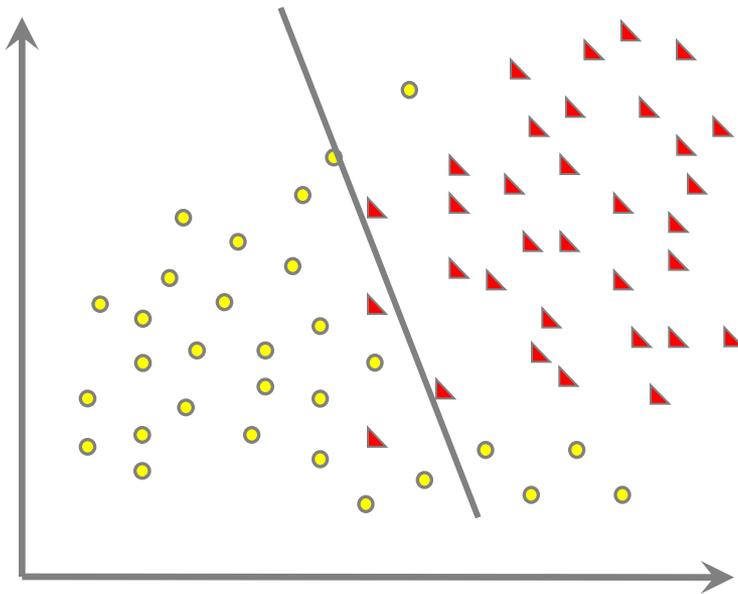
*Un campione  $\mathbf{x}$  è classificato come appartenente a  $\omega_1$  se  $\mathbf{w}^t \mathbf{x} > 0$ , a  $\omega_2$  se  $\mathbf{w}^t \mathbf{x} < 0$*



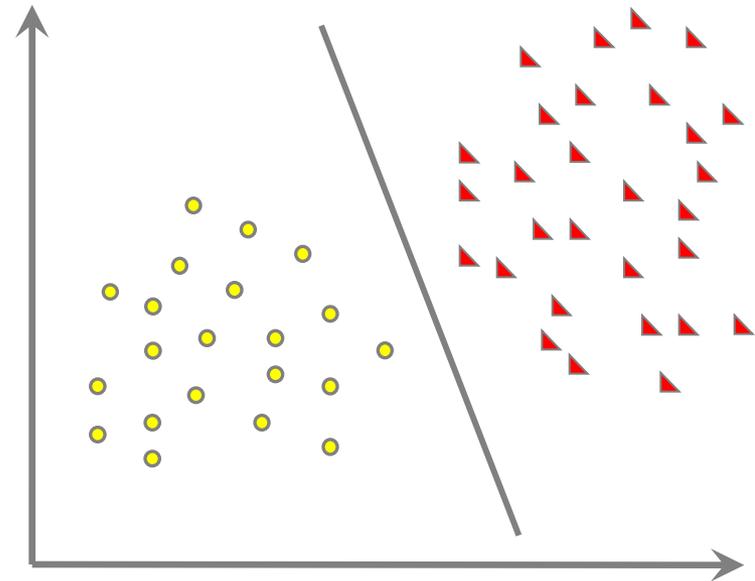
# Funzioni discriminanti lineari

- Training: dato un training set (un insieme di  $N$  campioni, ossia  $\mathbf{x}'_1, \dots, \mathbf{x}'_N$ , alcuni etichettati  $\omega_1$  ed altri etichettati  $\omega_2$ ), si vuole determinare il vettore dei pesi  $\mathbf{w}'$  e  $w_0$  della funzione discriminante lineare.
- Un ragionevole approccio è la ricerca di un vettore di pesi tale che la probabilità di commettere errore sui campioni sia minima.
- Se esiste un vettore di pesi tale da rendere nulla la probabilità di errore, allora i campioni si dicono *linearmente separabili*.





non linearmente separabili



linearmente separabili



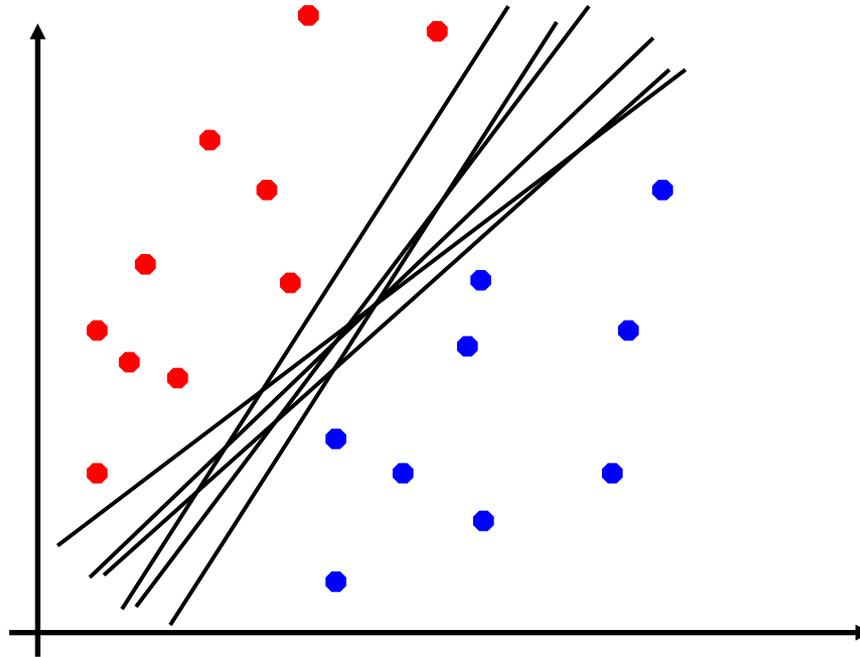
# Due classi

- L'obiettivo è quindi quello di calcolare tali pesi per cui
$$\mathbf{w}^t \mathbf{x} > 0$$
 per ogni  $\mathbf{x}$  appartenente a  $\omega_1$ 
$$\mathbf{w}^t \mathbf{x} < 0$$
 per ogni  $\mathbf{x}$  appartenente a  $\omega_2$
- In quest'ultimo caso (ossia per  $\omega_2$ ) si può anche dire che  $\mathbf{x}$  è classificato correttamente se  $\mathbf{w}^t (-\mathbf{x}) > 0$ .
- Questo suggerisce una *normalizzazione* (cambiare il segno a tutti gli oggetti della classe 2) che semplifica il trattamento nel caso di due diverse classi, ossia il fatto che si possa solo trovare il vettore dei pesi tale per cui si abbia  $\mathbf{w}^t \mathbf{x} > 0$  per tutti i campioni a prescindere dalle classi.



# Separatori Lineari

- Questo vettore  $w$  è chiamato **vettore separatore** o **vettore soluzione**.



- Risulta chiaro che *se il vettore soluzione esiste non è unico*.
- Ci sono diversi modi per imporre requisiti addizionali per vincolare il vettore soluzione.
- Uno potrebbe essere quello di **cercare il vettore dei pesi che massimizzi la minima distanza dei campioni dal piano separatore**.
- Un'altra potrebbe essere quella di cercare il vettore dei pesi che soddisfi

$$\mathbf{w}^t \mathbf{x}_i \geq b, \quad \forall i \text{ degli } N \text{ campioni a disposizione}$$

dove  $b$  è una costante positiva chiamata *margin*.



- Si ricorre a tecniche di *minimizzazione*, che minimizzano un funzionale in funzione di certe quantità
- Una delle tecniche di base è quella di *discesa del gradiente*



# Determinazione dei pesi $w$

## *Tecnica del Gradiente Discendente*

- La tecnica del Gradiente Discendente è uno degli approcci più semplici per il calcolo di una funzione discriminante.
- È un metodo iterativo di assestamento progressivo dei pesi che si basa sulla seguente proprietà:

*il vettore gradiente nello spazio  $W$  (ossia i coefficienti di  $w$ ) punta nella direzione di massimo scarto di una funzione da massimizzare/minimizzare*



# Determinazione dei pesi $w$

- *Tecnica del Gradiente Discendente*

- La procedura consiste nell'aggiornare il valore del vettore dei pesi al passo  $k+1$  con un contributo proporzionale al modulo del gradiente di un particolare funzionale al passo precedente e può essere formulata come:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \rho_k \nabla J(\mathbf{w}) \Big|_{\mathbf{w}=\mathbf{w}(k)}$$

- dove  $J(\mathbf{w})$  è una funzione di valutazione che deve essere minimizzata.
- $J(\mathbf{w})$  viene scelta in modo tale da raggiungere il minimo all'avvicinarsi di  $\mathbf{w}$  alla soluzione ottima, ovvero convessa.



# Determinazione dei pesi $w$

- Il minimo di  $J(\mathbf{w})$  si ottiene spostando  $\mathbf{w}$  in direzione opposta al gradiente.
- $\nabla$  è il simbolo dell'operatore gradiente, dato da:

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \vdots \\ \vdots \\ \frac{\partial J}{\partial w_d} \end{bmatrix}$$

- $\rho_k$  è uno scalare opportuno che varia con l'iterazione,  $k$ , fissando l'“ampiezza” nella correzione.



# Determinazione dei pesi $w$

- Chiaramente occorre scegliere un criterio di ottimalità  $J(\mathbf{w})$
- La scelta più ovvia è quella di definire un funzionale  $\mathcal{J}(\mathbf{w}; \mathbf{x}_1, \dots, \mathbf{x}_N)$  rappresentato dal numero di campioni mal classificati da  $\mathbf{w}$  su un totale di  $N$  campioni (*l'error rate*)
- Siccome tale funzionale è costante a tratti, il metodo della discesa del gradiente non è molto adatto a tale problema.

- Una scelta migliore per  $J$  può essere perciò:

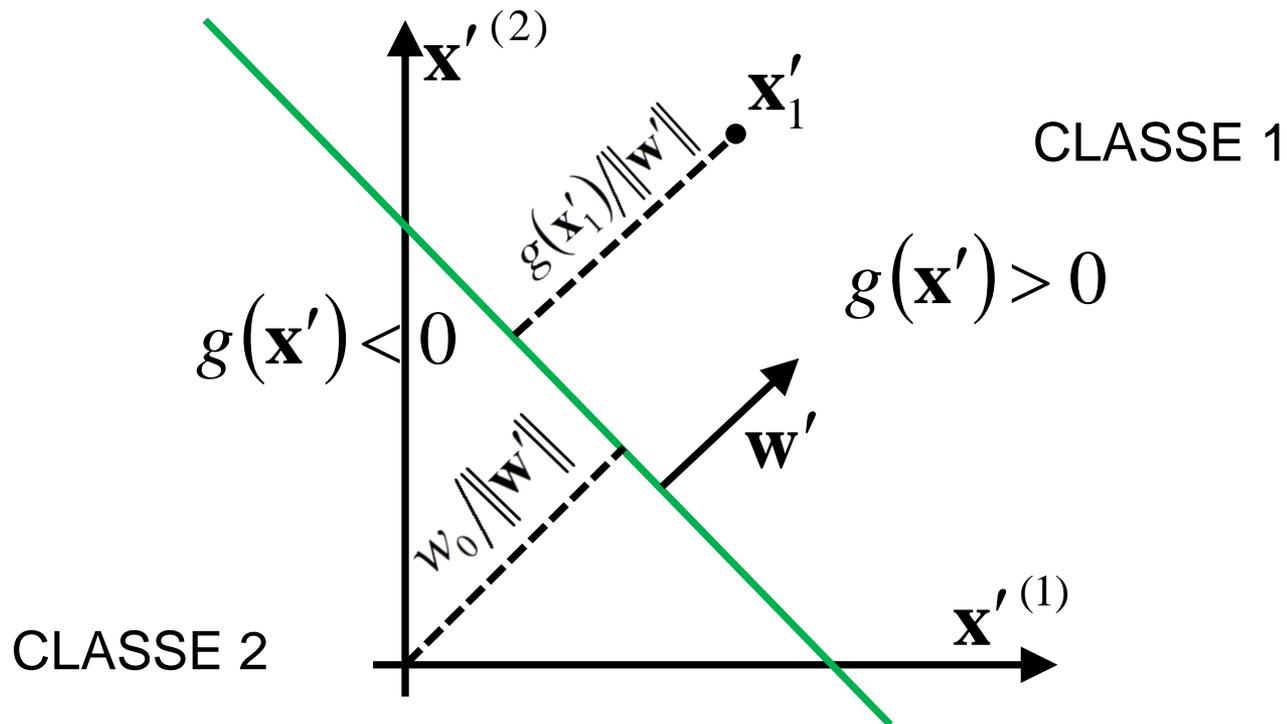
$$J(\mathbf{w}) = - \sum_{i \in X} \mathbf{w}^t \mathbf{x}_i \quad (1)$$

dove  $X$  è l'insieme di punti classificati non correttamente da  $\mathbf{w}$ .



# Determinazione dei pesi $w$

- Geometricamente,  $J(\mathbf{w})$  è proporzionale alla somma delle distanze dei campioni mal classificati dal confine di decisione.



# Determinazione dei pesi $w$

- Siccome la  $i$ -esima componente del gradiente di  $J(\mathbf{w})$  è pari a  $\partial J / \partial w_i$ , si può osservare dall'eq. (1) che:

$$\nabla J = - \sum_{i \in X} \mathbf{x}_i$$

(provare!) e quindi l'algoritmo di discesa del gradiente è

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \rho_k \cdot \sum_{i \in X} \mathbf{x}_i, \quad \text{con} \quad \rho_k = \frac{1}{k}$$



# Determinazione dei pesi $w$

- Questo metodo è stato usato per simulare sia in modo *hardware* che *software* la prima rete neurale ed si è in seguito evoluto a più complesse versioni come, ad esempio il *perceptron* multilivello
- Ci sono molti altri metodi per scegliere  $J(\mathbf{w})$  e per ottimizzarla:
  - **Metodo del rilassamento**
  - **Metodo del MSE (minimun square error)**
  - **Metodo del Least MSE o Widrow-Hoff**
  - **Metodo di Ho-Kashyap**
  - **Stochastic gradient (per deep network)**



# Caso multi classe

- Nel caso di problemi a  $C$  classi, possono essere costruiti  $C-1$  classificatori  $g_j(\mathbf{x})$ , uno per ogni classe  $C_j$  contro  $non-C_j$ , chiamati classificatori *one-vs-rest*
- Oppure si possono costruire  $C(C-1)/2$  classificatori binari (*one-vs-one*), e quindi classificare un punto a maggioranza (fare un torneo)



# Funzioni discriminanti lineari generalizzate

- Ripristiniamo il naturale concetto di

$$\mathbf{x} = [x_1, \dots, x_d]$$

eliminando il termine "1"

- In forma NON vettoriale, la funzione discriminante lineare può anche essere scritta come

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

dove  $w_i$  è la componente  $i$ -esima del vettore dei pesi e  $w_0$  è il solito termine noto.



# Funzioni discriminanti lineari generalizzate

- Possiamo aggiungere altri termini in cui mettiamo i prodotti delle varie componenti di  $\mathbf{x}$
- Esempio: classificatore discriminante quadratico (cosa fa di preciso? *Non ci importa ora*)

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^d w_i x_i + \sum_{i=1}^d \sum_{j=1}^d v_{ij} x_i x_j$$

- Possiamo generalizzare il concetto scrivendo

$$g(\mathbf{x}) = \sum_{k=1}^K a_k \cdot y_k(\mathbf{x})$$

dove  $y_k(\mathbf{x})$ ,  $k=1\dots K$ , sono *funzioni arbitrarie* su  $\mathbf{x}$



# Funzioni discriminanti lineari generalizzate - esempio

- Caso 1D, punti  $x$  su una retta (non ho più il grassetto su  $x$ !)

$$g_{old}(x) = w_0 + wx$$

- Definiamo tre funzioni  $y_k(x)$ ,  $K=3$

$$y_1(x) = 1$$

$$y_2(x) = x$$

$$y_3(x) = x^2$$

- Otteniamo quindi la funzione

$$g(x) = a_1 + a_2x + a_3x^2$$



- Riprendendo la formulazione generale multidimensionale

$$g(\mathbf{x}) = \sum_{k=1}^K a_k \cdot y_k(\mathbf{x})$$

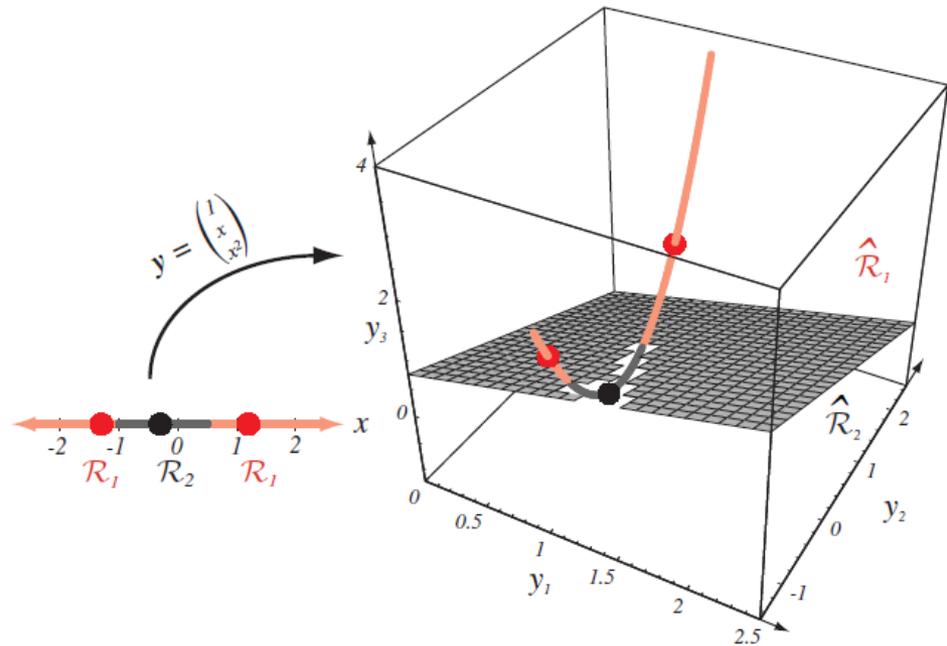
- $y_k(\mathbf{x})$  al variare di  $k$ , possono essere viste direttamente come nuove features
- In altre parole,  $y_k$  possono essere viste come estrattori di features
  - Di solito, la fase di estr. di features è usata per diminuire di dimensionalità (PCA): qui invece **aumenta**
- $g(\mathbf{x})$  si chiama *funzione discriminante lineare generalizzata*
  - lineare rispetto a  $y_k(\mathbf{x})$
  - non lineare nello spazio originale di  $\mathbf{x}$



- Per esempio, con la funzione

$$g'(x) = a_1 + a_2x + a_3x^2$$

si passa da uno spazio di feature monodimensionale ad uno spazio tridimensionale



**FIGURE 5.5.** The mapping  $y = (1, x, x^2)^t$  takes a line and transforms it to a parabola in three dimensions. A plane splits the resulting  $y$ -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.



- Adesso la funzione  $g(\mathbf{x})$  è lineare nello spazio di  $y$  (cioè è un iperpiano nello spazio tridimensionale)
  - funzione semplice nello spazio di  $y$
  - funzione complessa nello spazio originale di  $\mathbf{x}$
  - Bisogna quindi calcolare gli  $\{\alpha\}$ , che svolgono la funzione dei coefficienti di  $\mathbf{w}$
- *Idea alla base delle Support Vector Machines*

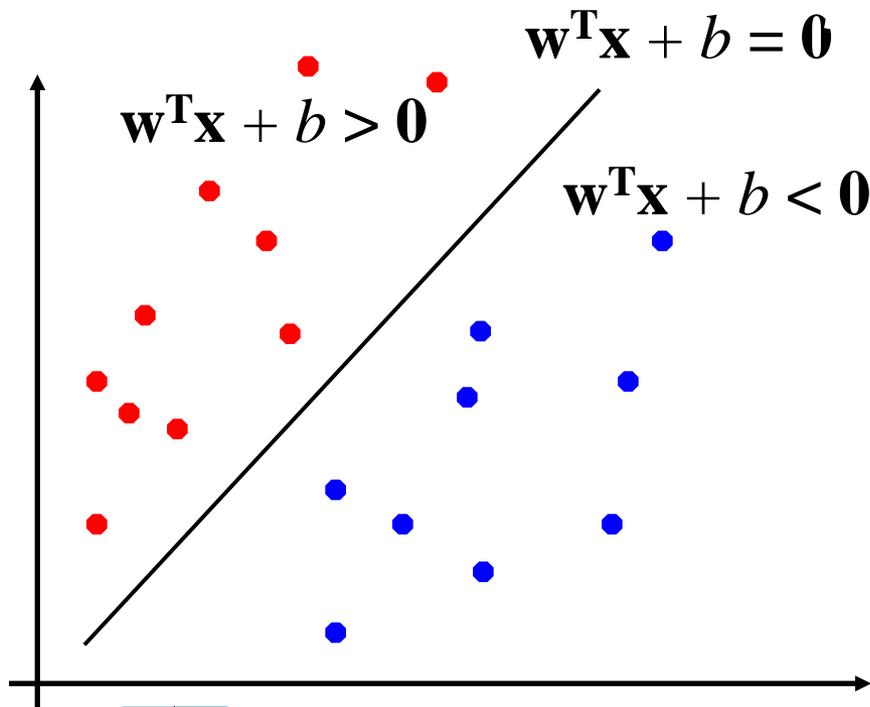


# Support Vector Machines



# SVM come separatori lineari

- SVM possono essere visti come separatori lineari, ereditando la notazione vista finora, con  $b = w_0$



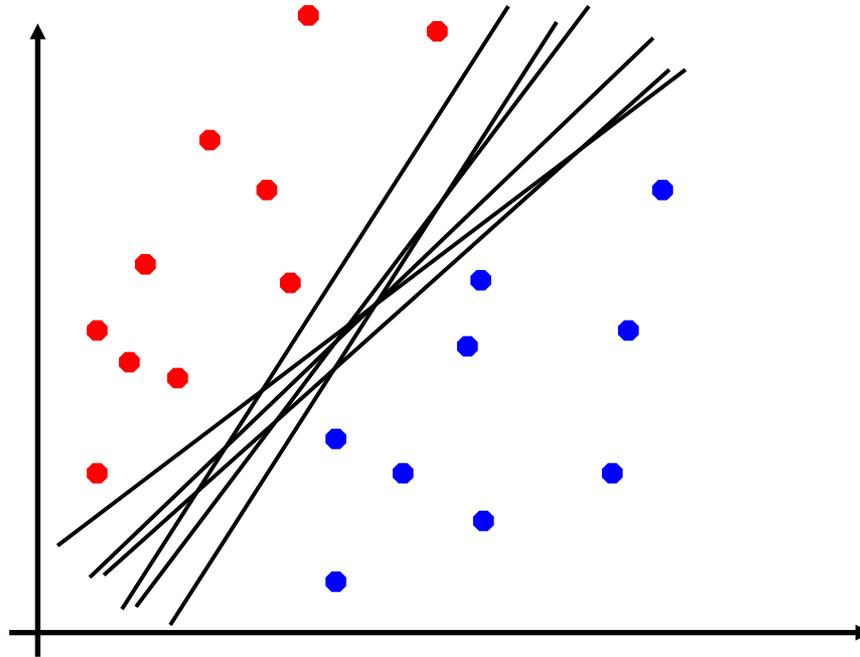
Per classificare applico il segno alla funzione discriminante

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



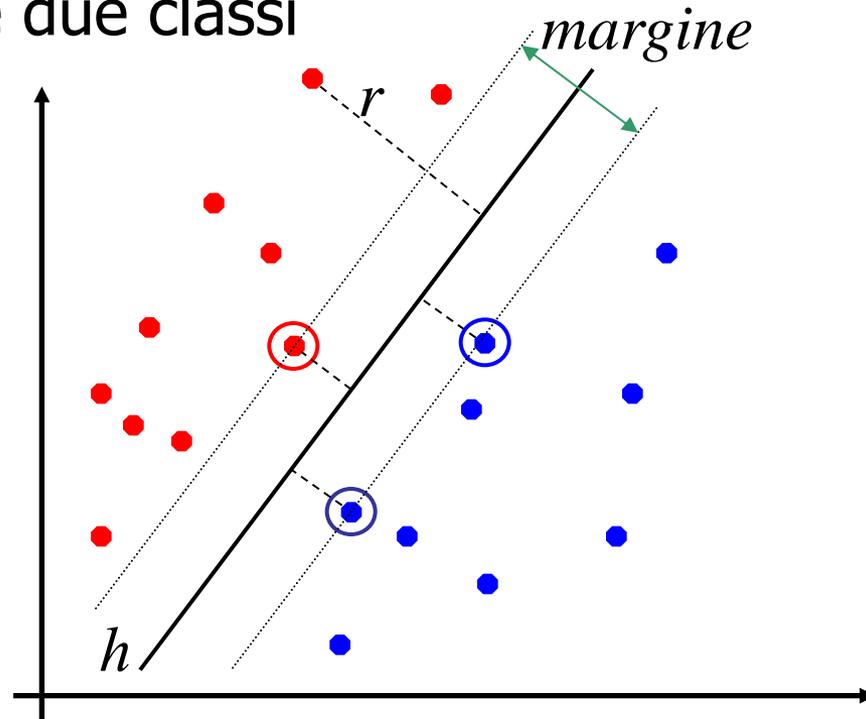
# SVM come separatori lineari

- Quale separatore è ottimo?



# SVM come separatori lineari

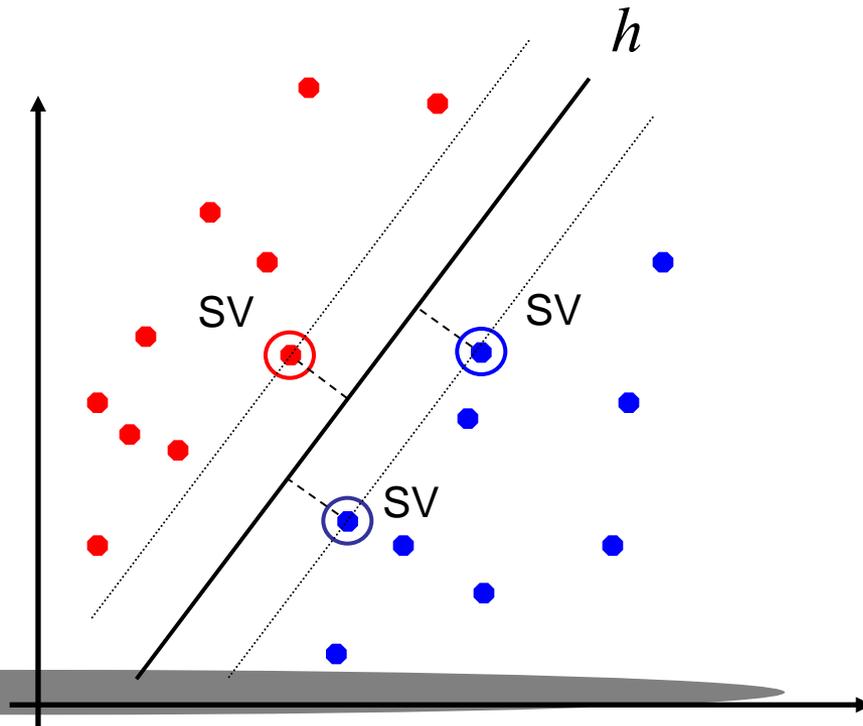
- Chiamo la distanza di un campione dall'iperpiano  $r$
- Gli esempi più vicini all'iperpiano si chiamano **support vectors**.
- **Il margine** dell'iperpiano  $h$  di separazione è la distanza minima fra le due classi



# SVM come separatori lineari

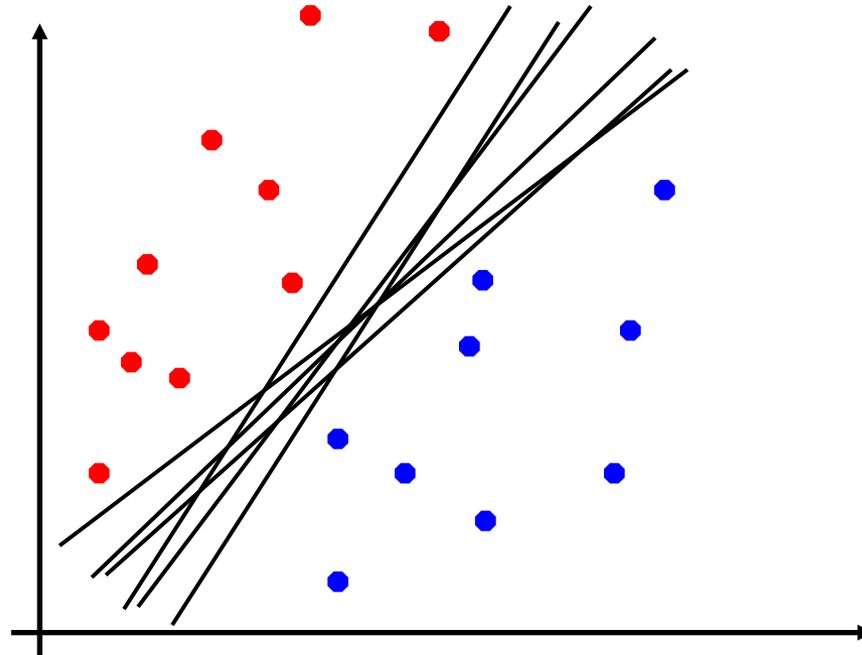
- Individuare l'iperpiano ottimo  $h$  corrisponde a **massimizzare il margine**
- Questo implica che solo alcuni esempi sono importanti per l'apprendimento, i **vettori di supporto**. Gli altri possono essere ignorati.

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



# SVM come separatori lineari

- Addestramento della SVM: trovare il vettore  $\mathbf{w}$  e il parametro  $b$  ottimali (che massimizzino il margine), a partire dal training set



# SVM come separatori lineari

- Si hanno i dati di addestramento  $D \{(\mathbf{x}_i, y_i)\}$  ( $\{y\}$  label binarie, 1,-1) a distanza almeno 1 dall'iperpiano, allora valgono le seguenti condizioni per  $\mathbf{x}_i$  in  $D$ :

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \quad \text{se } f(\mathbf{x}_i) = +1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \quad \text{se } f(\mathbf{x}_i) = -1$$

- Per i **vettori di supporto**, la disequaglianza diventa una eguaglianza, perchè stanno esattamente a distanza 1 dall'iperpiano separatore



- Evidenzio in verde le variabili *incognite* del mio problema

Zona all'interno della  
quale si assegna la  
classificazione "+1"

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

Zona all'interno della  
quale si assegna la  
classificazione "-1"

$$\mathbf{w}^T \cdot \mathbf{x} + b \geq 1$$

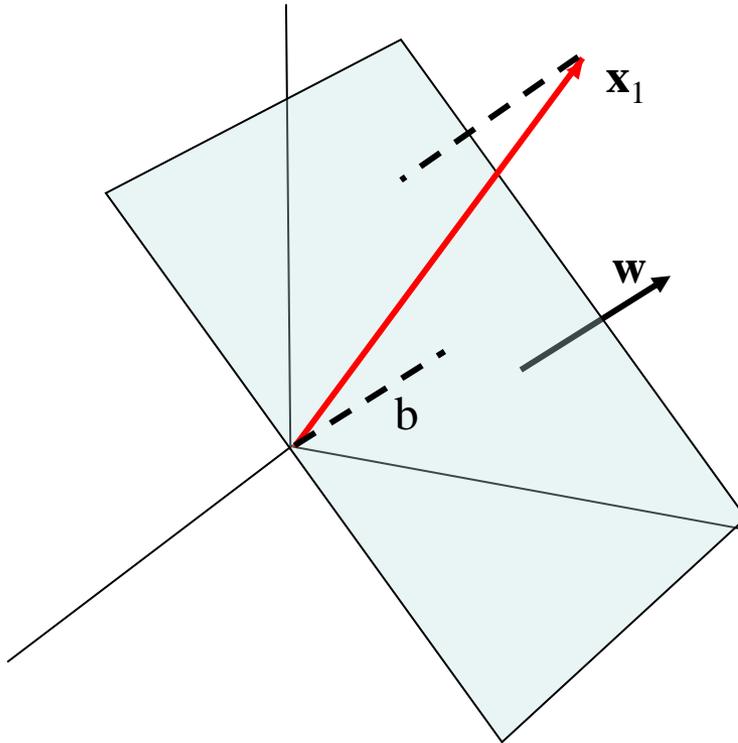
$$\mathbf{w}^T \cdot \mathbf{x} + b \leq -1$$

$$-1 < \mathbf{w}^T \cdot \mathbf{x} + b < 1$$

Zona di margine



$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + b = 0$$



- $\mathbf{w}$  è il vettore perpendicolare al piano di separazione
- Suppongo per comodità  $\mathbf{w}$  versore
- $b/\|\mathbf{w}\| = b$  è la distanza del piano dall'origine
- $|g(\mathbf{x}_1)/\|\mathbf{w}\|| = |g(\mathbf{x}_1)|$  è la distanza del punto  $\mathbf{x}_1$  dal piano separatore



- Indichiamo con  $\rho$  la distanza fra i piani

$$P+: \mathbf{w}^T \mathbf{x}_i + b = 1 \text{ e}$$

$$P-: \mathbf{w}^T \mathbf{x}_i + b = -1 \text{ e}$$

- Sia  $\mathbf{x}^+$  un punto di  $P+$  e  $\mathbf{x}^-$  un punto di  $P-$  a distanza minima da  $\mathbf{x}^+$

- $\rho = |\mathbf{x}^+ - \mathbf{x}^-|$ , dove posso anche scrivere:

$$(\mathbf{x}^+ - \mathbf{x}^-) = \lambda \mathbf{w}$$



Se  $\mathbf{x}^+$  e  $\mathbf{x}^-$  sono a distanza minima, muoversi da  $\mathbf{x}^+$  a  $\mathbf{x}^-$  corrisponde ad un percorso nella direzione di  $\mathbf{w}$  (suppongo  $\mathbf{w}$  versore)

$$\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$$

di lunghezza  $|\lambda \mathbf{w}|$

Per riassumere:

$$\mathbf{w}^T \mathbf{x}^+ + b = +1$$

$$\mathbf{w}^T \mathbf{x}^- + b = -1$$

$$\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$$



# Mettendo assieme:

- Isolando  $\lambda$  e  $\mathbf{w}$ , vogliamo capire chi sia  $\rho$ :

$$\mathbf{x}^+ - \mathbf{x}^- = \lambda \mathbf{w} \Rightarrow \mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$$

$$\mathbf{w}^T \cdot \mathbf{x}^+ + b = 1, \quad \mathbf{w}^T \cdot \mathbf{x}^- + b = -1$$

$$\Rightarrow (\mathbf{w}^T \cdot (\mathbf{x}^- + \lambda \mathbf{w})) + b = 1 \Rightarrow (\mathbf{w}^T \cdot \mathbf{x}^- + b) + \lambda \mathbf{w}^T \cdot \mathbf{w} = 1$$

$$-1 + \lambda \mathbf{w}^T \cdot \mathbf{w} = 1 \Rightarrow \lambda = \frac{2}{\mathbf{w}^T \cdot \mathbf{w}}$$

$$\rho = \|\lambda \cdot \mathbf{w}\| = \left\| \frac{2}{\mathbf{w}^T \cdot \mathbf{w}} \mathbf{w} \right\| = \left\| \frac{2}{\mathbf{w}^T} \right\| = \frac{2}{\|\mathbf{w}\|}$$

- Per massimizzare il margine, dobbiamo minimizzare  $\|\mathbf{w}\|$



# SVM lineare (2)

- Il problema di ottimizzazione quadratica che ne risulta é:

Trova  $\mathbf{w}$  e  $b$  tali che

$$\rho = \frac{2}{\|\mathbf{w}\|} \quad \text{è massimo; e per ogni } \{(\mathbf{x}_i, y_i)\} \in D$$

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ se } y_i = 1; \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1 \text{ se } y_i = -1$$

- La formulazione duale che se ne può ricavare é:

Trova  $\mathbf{w}$  e  $b$  t.c.

$$\Phi(\mathbf{w}) = 1/2 \mathbf{w}^T \mathbf{w} \quad \text{è minimizzata;}$$

$$\text{E per ogni } \{(\mathbf{x}_i, y_i)\} \in D : \quad y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$



# Risolvere il problema di ottimizzazione

- Si deve ottimizzare una funzione quadratica soggetta a vincoli lineari (uno per ogni  $\mathbf{x}_i$ ):  
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$
- I problemi di ottimizzazione quadratica sono problemi di programmazione matematica ben noti, per i quali esistono vari algoritmi.
- La soluzione comporta l'utilizzo di moltiplicatori di Lagrange



# Torniamo al problema di SVM

- Minimizzare il seguente *Lagrangiano*:

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

$\langle \mathbf{x}_i, y_i \rangle$  learning set

- Imponiamo dapprima:  $\frac{\partial(L)}{\partial b} = 0, \quad \frac{\partial(L)}{\partial \mathbf{w}} = 0$  da cui:

$$(1) \sum_{i=1}^N \alpha_i y_i = 0 \quad (2) \mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i$$

- La condizione  $\alpha \geq 0$  porta a selezionare solo un sottoinsieme di vettori (punti  $\mathbf{x}$ ), per i quali questa condizione è verificata, detti Support Vectors, da cui:

$$\mathbf{w} = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$



# RIASSUMIAMO

1) Formulazione del problema di ottimizzazione:

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

$$\text{soggetto a: } \forall i \in \{1, \dots, N\} : y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \geq 0$$

2) Espressione del problema con un Lagrangiano:

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

3) Ottieni la soluzione (teorema di Kuhn-Tucker)

$$\begin{aligned} \frac{\partial L(\vec{w}, b, \alpha)}{\partial \vec{w}} = 0 &\Leftrightarrow \vec{w}^* = \sum_{i=1}^N y_i \alpha_i^* \vec{x}_i \quad \text{E1} \\ \frac{\partial L(\vec{w}, b, \alpha)}{\partial b} = 0 &\Leftrightarrow \sum_{i=1}^N y_i \alpha_i^* = 0 \quad \text{E2} \end{aligned}$$

Ricorda, i vettori per cui  $\alpha^* > 0$  sono i vettori di supporto.

4) Ottieni la formulazione duale eliminando le variabili primarie  $w, b$  in 2) (formule E1 e E2)

$$\begin{aligned} \max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j \\ \text{soggetto a: } \forall i \in \{1, \dots, N\} : \alpha_i \geq 0 \text{ e } \sum_{i=1}^N y_i \alpha_i = 0. \end{aligned}$$



# RIASSUMIAMO

1) Formulazione del problema di ottimizzazione:

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

$$\text{soggetto a: } \forall i \in \{1, \dots, N\} : y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \geq 0$$

2) Espressione del problema con un Lagrangiano:

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

3) Ottieni la soluzione

$$\frac{\partial L(\vec{w}, b, \alpha)}{\partial \vec{w}} = 0 \Leftrightarrow \vec{w}^* = \sum_{i=1}^N y_i \alpha_i^* \vec{x}_i \quad \text{E1}$$

$$\frac{\partial L(\vec{w}, b, \alpha)}{\partial b} = 0 \Leftrightarrow \sum_{i=1}^N y_i \alpha_i^* = 0 \quad \text{E2}$$

$$\min \left( \frac{1}{2} w \cdot w - \sum (\alpha_i y_i \mathbf{x}_i) \cdot w - b \sum \alpha_i y_i + \sum a_i \right) =$$

$$\min \left( \frac{1}{2} \sum (\alpha_i y_i \mathbf{x}_i) \sum (\alpha_i y_i \mathbf{x}_i) - \sum (\alpha_i y_i \mathbf{x}_i) \sum (\alpha_i y_i \mathbf{x}_i) - b \cdot 0 + \dots \right)$$

$$\max \left( \sum \alpha_i - \frac{1}{2} \sum_{i,j=1..n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \right)$$

# Formulazione finale

Trova  $\alpha_1 \dots \alpha_N$  t.c.

$$Q(\alpha) = \sum \alpha_i - 1/2 \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \text{ é}$$

massimizzata e

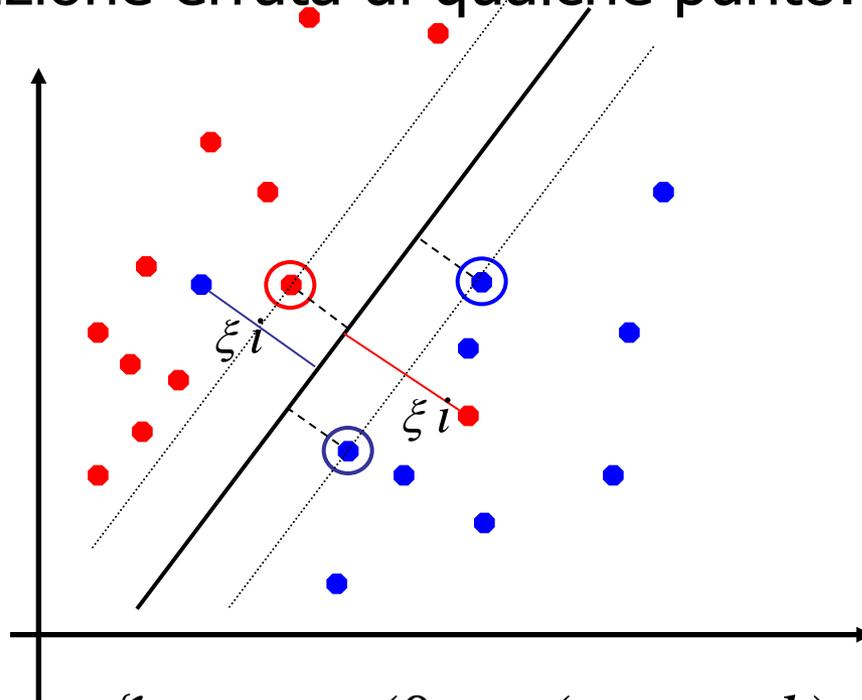
$$(1) \sum \alpha_i y_i = 0$$

$$(2) 0 \leq \alpha_i \text{ per ogni } \alpha_i$$



# Margini "Soft"

- Se il set di addestramento **non è linearmente separabile?**
- *Si introducono le slack variables  $\xi_i$  che consentono la classificazione errata di qualche punto.*



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



# Soft Margin Classification

- Problema lineare:

Trova  $\mathbf{w}$  e  $b$  t.c.

$\Phi(\mathbf{w}) = 1/2 \mathbf{w}^T \mathbf{w}$  è minimizzata e per ogni  $\{(\mathbf{x}_i, y_i)\}$   
 $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- Con le slack variables:

Trova  $\mathbf{w}$  e  $b$  t.c.

$\Phi(\mathbf{w}) = 1/2 \mathbf{w}^T \mathbf{w} + C \sum \xi_i$  è minimizzata, e per ogni  $\{(\mathbf{x}_i, y_i)\}$   
 $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$  e  $\xi_i \geq 0$  per ogni  $i$

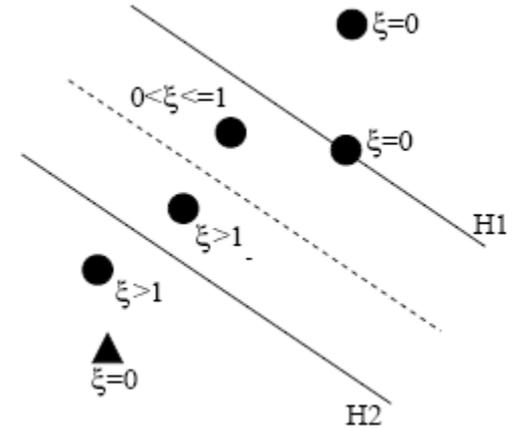
- Il parametro  $C$  pesa gli errori e controlla l'overfitting ( $C$  piccolo, *ammetto errori*).



# Soluzione per dati NLS

Il sistema vincolato viene risolto ancora nella sua forma duale, ed il risultato è ancora

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



I valori di  $\xi$  si interpretano:

$\xi_i = 0$  classificazione corretta

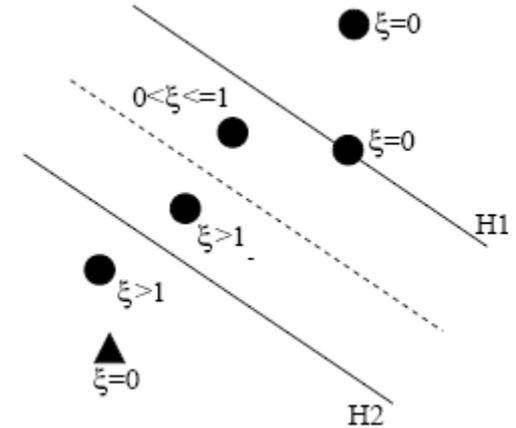
$0 < \xi_i < 1$  classificazione corretta ma fuori  $H_i$

$\xi_i \geq 1$  errore di classificazione



# Soluzione per dati NLS

- Ora i support vectors sono i punti per cui
  - $\alpha_i$  è diverso da zero
  - I punti sul margine
  - I punti classificati correttamente ma oltre il margine
  - I punti classificati non correttamente



# Sommario di SVM lineare

- Il classificatore (funzione obiettivo) è un iperpiano di separazione.
- I "punti" (esempi) più importanti sono i vettori di support ("sostengono" l'iperpiano, mantendolo in equilibrio)
- Algoritmi di ottimizzazione quadratica identificano quali punti rappresentano il "supporto".
- Nella formulazione del problema appaiono i prodotti

scalari :

Trova  $\alpha_1 \dots \alpha_N$  t.c.

$Q(\alpha) = \sum \alpha_i - 1/2 \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  é  
massimizzata e

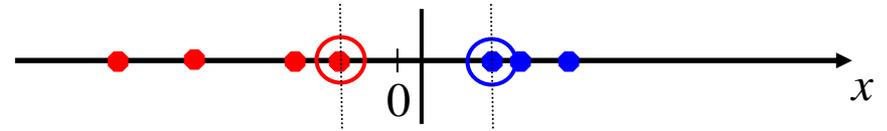
$$(1) \sum \alpha_i y_i = 0$$

$$(2) 0 \leq \alpha_i \text{ per ogni } \alpha_i$$

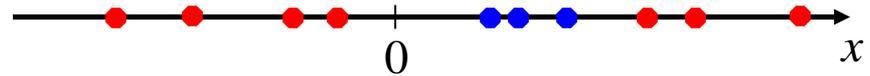


# Non-linear SVMs

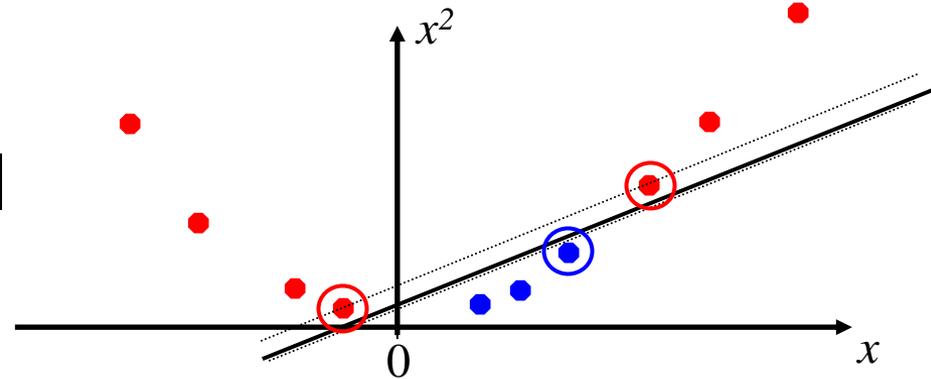
- Se i dataset sono separabili le cose funzionano:



- Altrimenti?



- Si può proiettare il problema

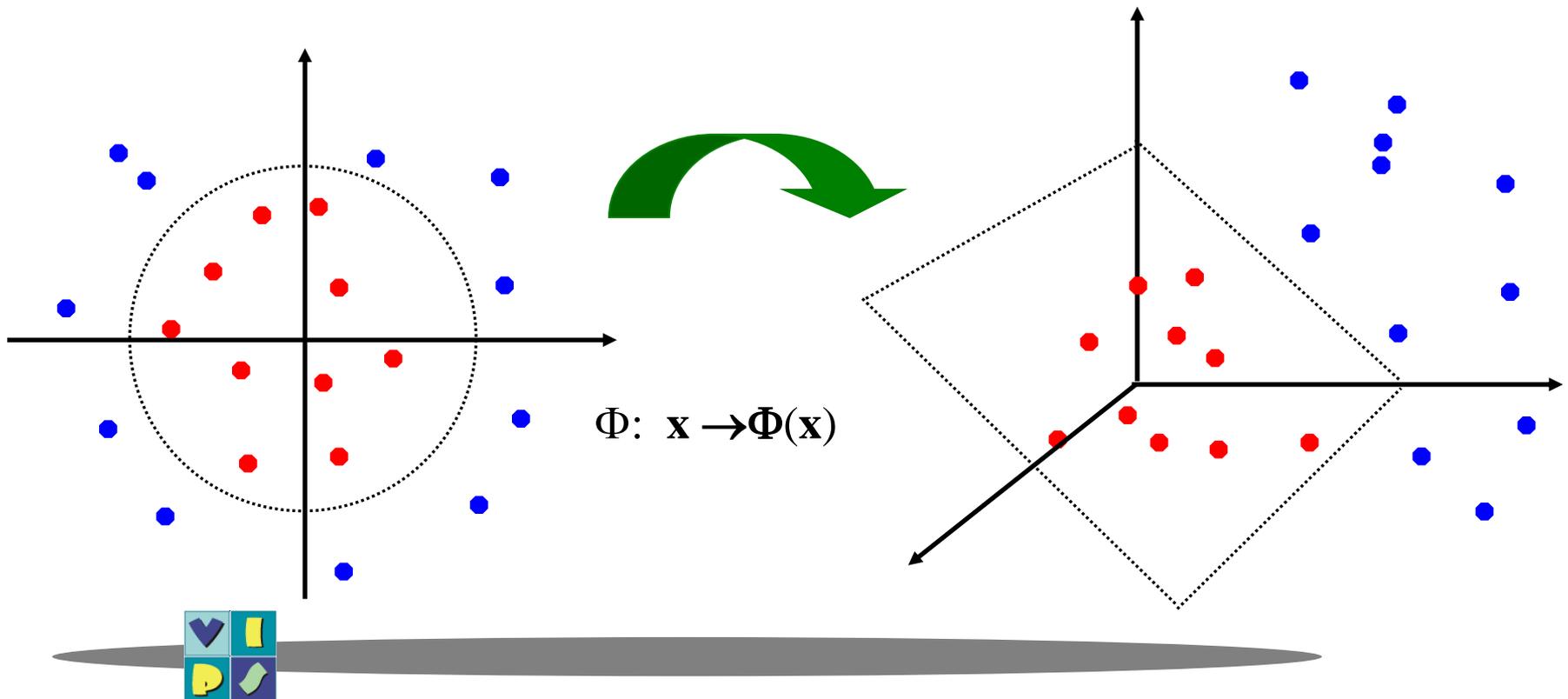


- in uno spazio di dimensioni maggiori:



# Non-linear SVMs: Feature spaces

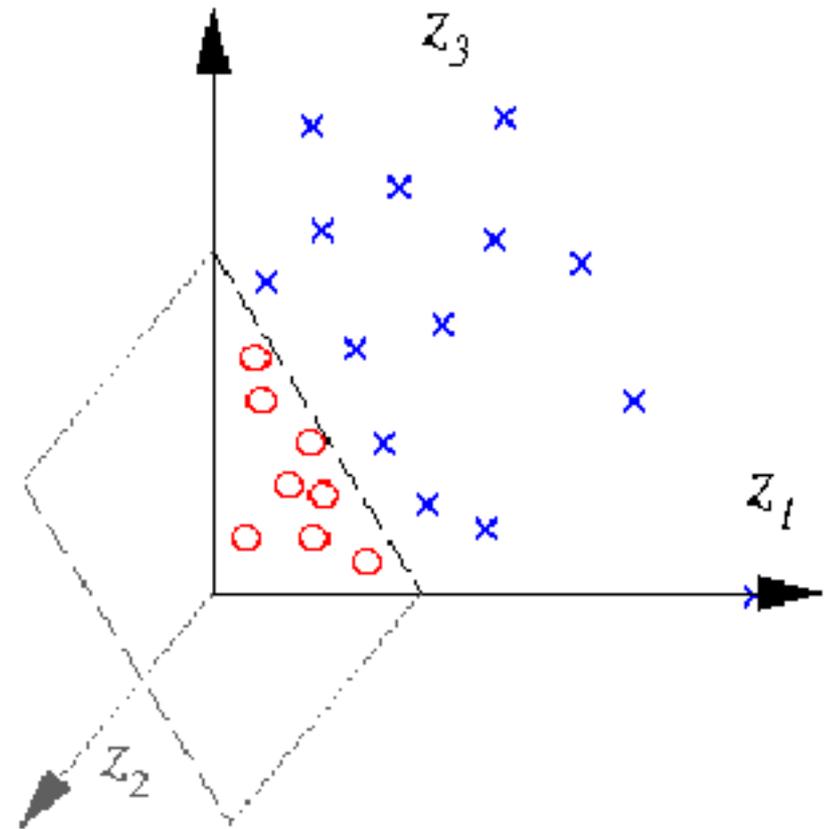
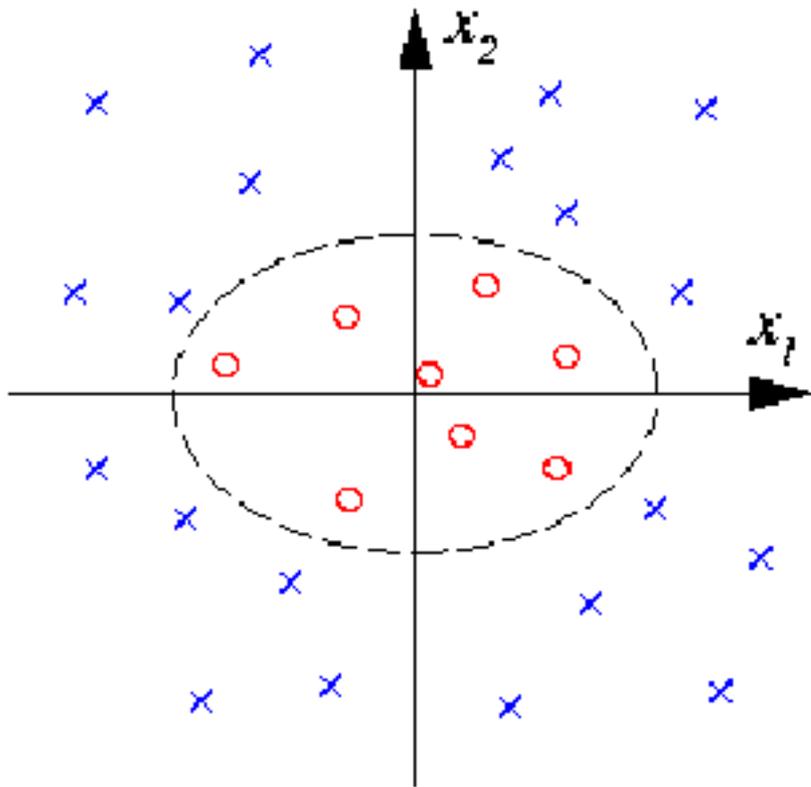
- Proiettare in uno spazio nel quale i dati risultino separabili:



$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

Esempio di funzione  $\Phi$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$$



# Proiezione ad alta dimensionalità: problemi

- Proiettare i punti in uno spazio a dimensionalità maggiore può portare alla curse of dimensionality
- Problemi di complessità computazionale
- Spazio quasi vuoto
- Potrebbero in teoria esserci anche mapping in grado di proiettare i punti in uno spazio a dimensionalità infinita
- Scegliere il mapping corretto potrebbe non essere facile



# Kernel Trick: ripasso

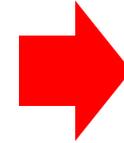
1.

Trova  $\alpha_1 \dots \alpha_N$  t.c.

$\mathbf{Q}(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$  é  
massimizzata e

(1)  $\sum \alpha_i y_i = 0$

(2)  $0 \leq \alpha_i$  per ogni  $\alpha_i$



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

2. In fase di testing, la classificazione avviene con

$$\begin{aligned} f(\mathbf{x}) &= \text{sign}(\mathbf{w}^T \mathbf{x} + b) = \text{sign} \left( \left[ \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i \right]^T \mathbf{x} + b \right) \\ &= \text{sign} \left( \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b \right) \end{aligned}$$



# Kernel Trick: ripasso

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

$$\max_{\alpha_i} \left( \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right) \quad \mathbf{w}^T \mathbf{x} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i^T \mathbf{x}$$

Applicando il mapping  $\Phi(\mathbf{x})$  per proiettare i punti in uno spazio a dimensionalità maggiore si ottiene

$$\max_{\alpha_i} \left( \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i^T) \Phi(\mathbf{x}_j) \right) \quad \mathbf{w}^T \mathbf{x} = \sum_{i=1}^N \alpha_i y_i \Phi(\mathbf{x}_i^T) \Phi(\mathbf{x})$$



# Kernel Trick: Funzioni Kernel

- Introduco una funzione *kernel*, che corrisponde ad un prodotto scalare in uno spazio esteso
- Il classificatore lineare si basa sul prodotto scalare fra vettori dello spazio delle istanze  $X$  (quindi, non esteso):  
 $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Se ogni punto di  $D$  è traslato in uno spazio di dimensioni maggiori attraverso una trasformazione  $\Phi: \mathbf{x} \rightarrow \Phi(\mathbf{x})$  il prodotto scalare diventa:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) = \mathbf{x}'_i{}^T \mathbf{x}'_j$$

dove  $x'$  e  $y'$  indicano trasformazioni non lineari



# Kernel Trick: ripasso

QUINDI:

$$\max_{\alpha_i} \left( \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right)$$

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



$$\max_{\alpha_i} \left( \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i^T) \Phi(\mathbf{x}_j) \right)$$

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



$$\max_{\alpha_i} \left( \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right)$$

$$\mathbf{w}^T \mathbf{x} = \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$$



# Kernel Trick: ripasso

In pratica le funzioni kernel mi permettono di evitare di calcolare il prodotto in uno spazio ad elevata dimensionalità, quindi eseguendo esplicitamente delle proiezioni

$$\max_{\alpha_i} \left( \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \quad \mathbf{w}^T \mathbf{x} = \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$$



# Funzioni kernel

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

Sia  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$ ,

Dobbiamo mostrare che  $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ :

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 =$$

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

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

$\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ , dove

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



# Quali funzioni sono Kernels?

- Per alcune funzioni  $K(\mathbf{x}_i, \mathbf{x}_j)$  verificare che  $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$  è complesso.
- Il Teorema di Mercer enuncia le caratteristiche necessarie affinché una funzione qualsiasi sia un kernel:

***Ogni funzione  
semi-definita positiva  
è un kernel***



# Esempi di funzioni kernel

- Lineare:  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polinomiale potenza di  $p$ :  
$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$$
- Gaussiana (*radial-basis function network*):  
$$K(\mathbf{x}_i, \mathbf{x}_j) = \frac{e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}}{e}$$
- Percettrone a due stadi:
- $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$



# Applicazioni

- SVMs sono attualmente fra i migliori classificatori in una varietà di problemi (es. testi e genomica).
- Il tuning dei parametri SVMs è un'arte: la selezione di uno specifico kernel e i parametri viene eseguita in modo empirico (tenta e verifica, test and trial)



# 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, meno è meglio)
- In moltissimi contesti applicativi funzionano decisamente bene. La teoria delle SVM (Statistical learning theory) è elegante e solida



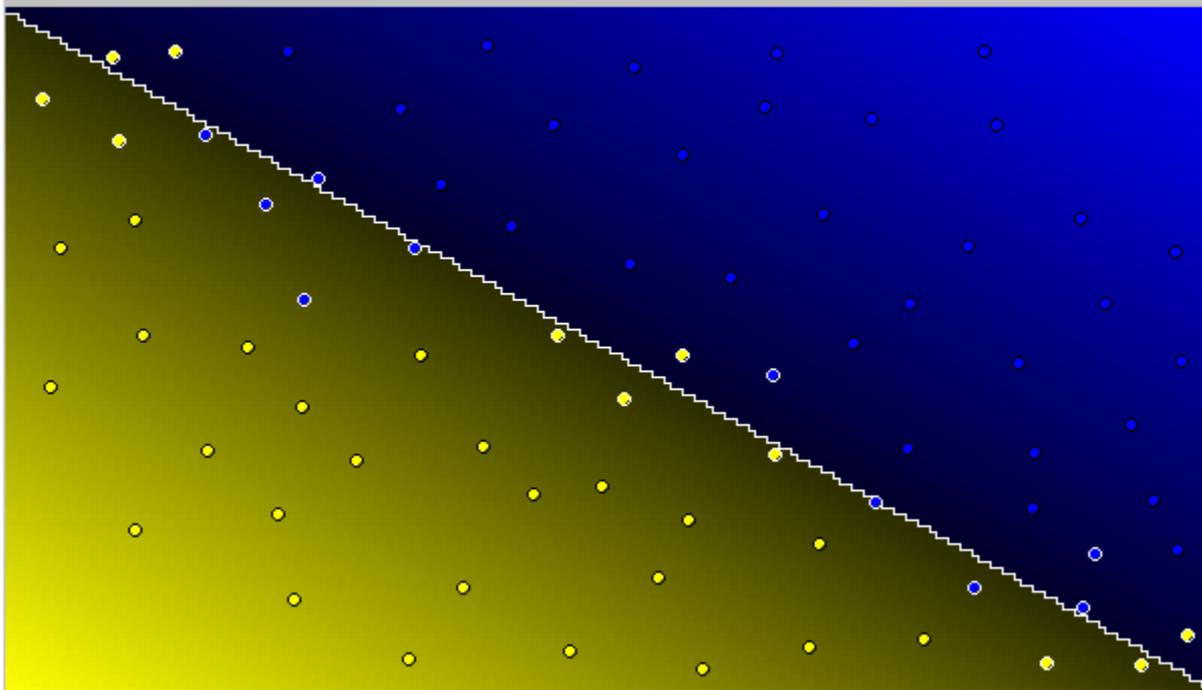
# Svantaggi

- 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)



# Classification examples

Number of Support Vectors: 21 (-ve: 10, +ve: 11) Total number of points: 75



## Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

Polinomiale, grado 3

Polinomiale, grado 4

Polinomiale, grado 3, C=100

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

RBF, sigma 1

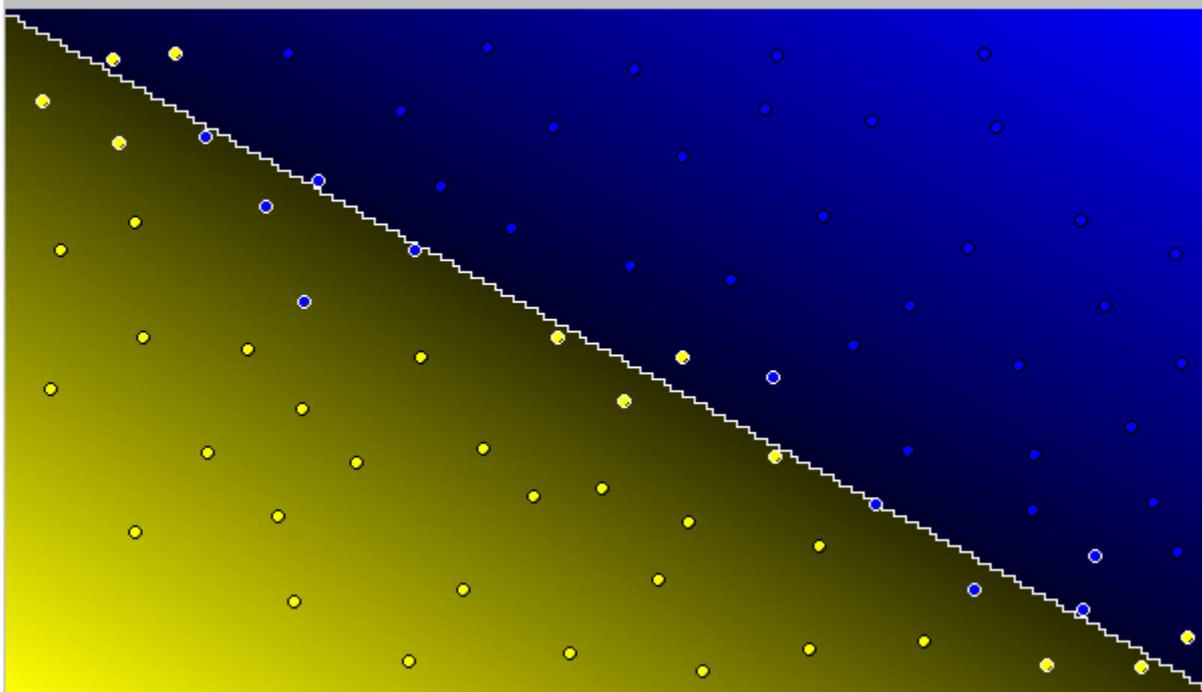
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: 21 (-ve: 10, +ve: 11) Total number of points: 75



Kernel lineare

**Polinomiale, grado 1**

Polinomiale, grado 2

Polinomiale, grado 3

Polinomiale, grado 4

Polinomiale, grado 3, C=100

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

RBF, sigma 1

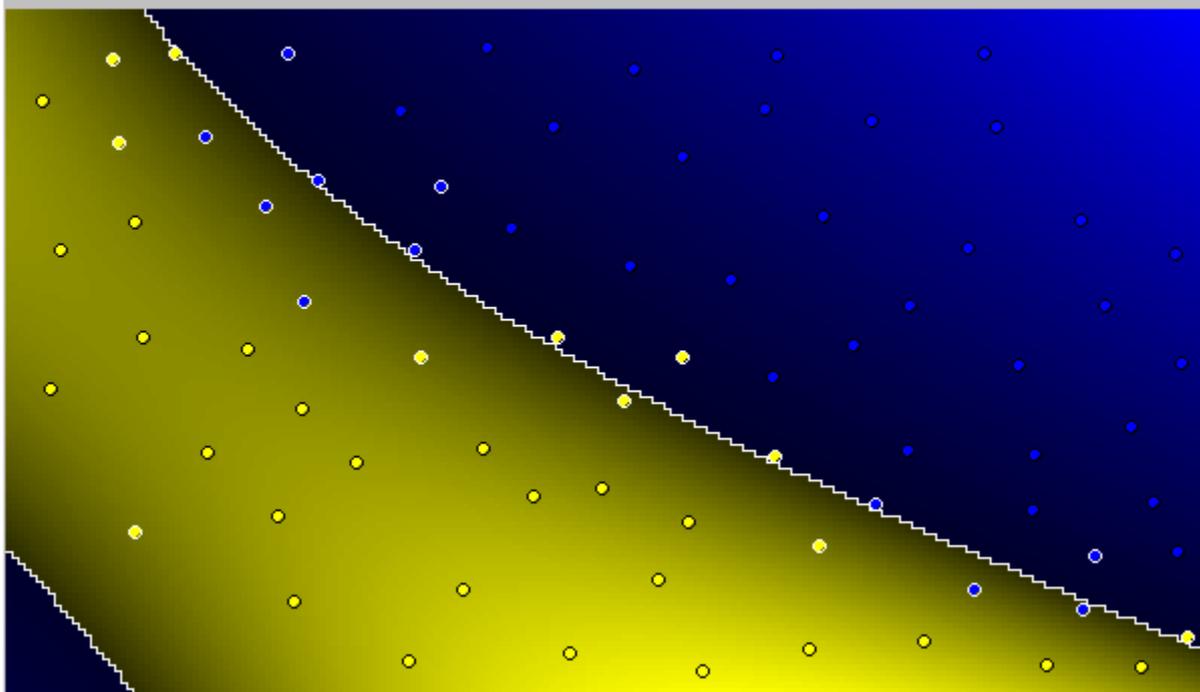
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: 22 (-ve: 11, +ve: 11) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

**Polinomiale, grado 2**

Polinomiale, grado 3

Polinomiale, grado 4

Polinomiale, grado 3, C=100

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

RBF, sigma 1

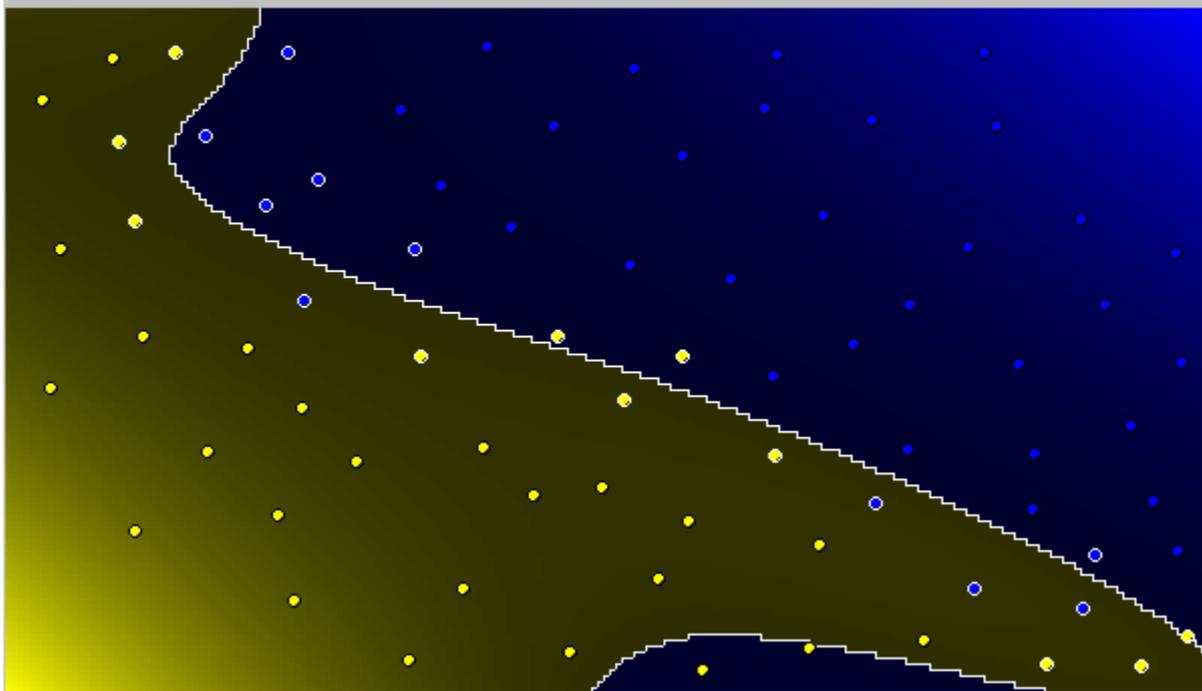
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: 21 (-ve: 10, +ve: 11) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

**Polinomiale, grado 3**

Polinomiale, grado 4

Polinomiale, grado 3, C=100

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

RBF, sigma 1

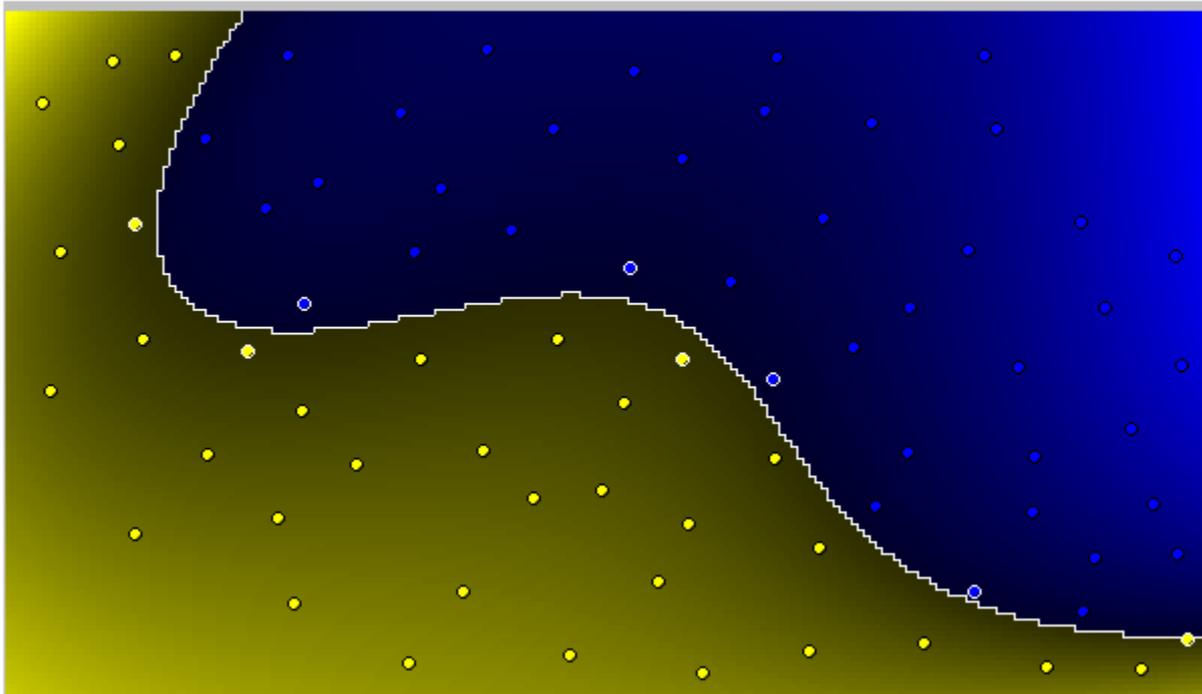
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: 8 (-ve: 4, +ve: 4) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

Polinomiale, grado 3

**Polinomiale, grado 4**

Polinomiale, grado 3, C=100

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

RBF, sigma 1

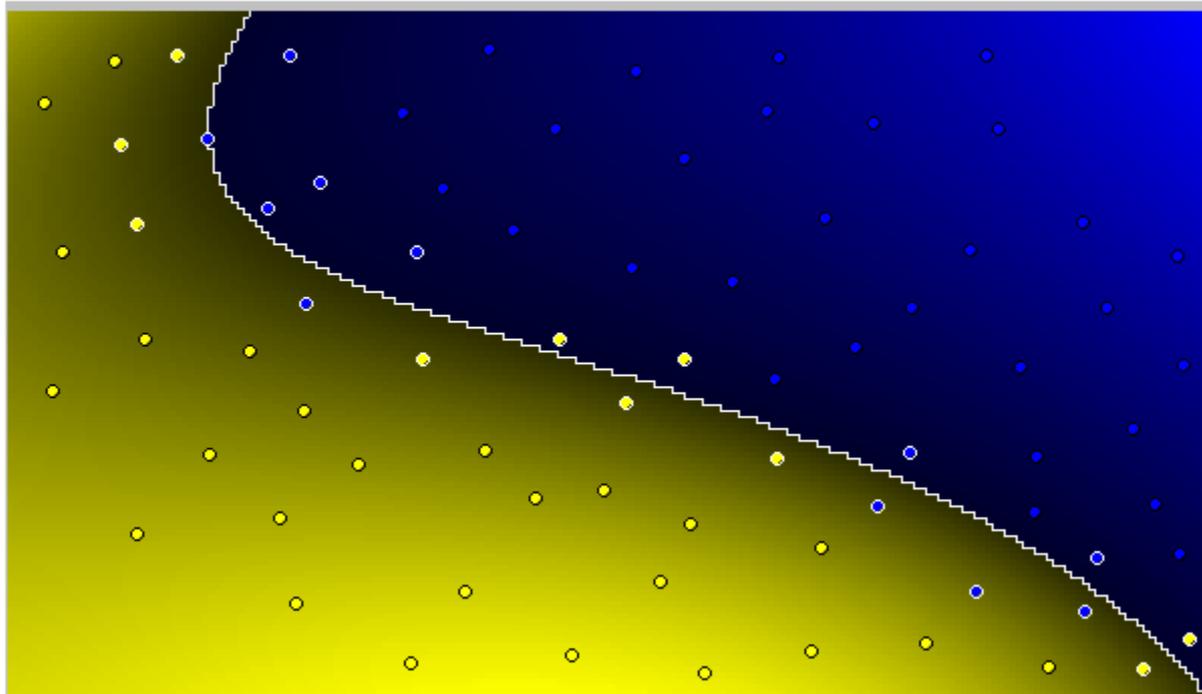
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: 21 (-ve: 11, +ve: 10) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

Polinomiale, grado 3

Polinomiale, grado 4

**Polinomiale, grado 3, C=100**

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

RBF, sigma 1

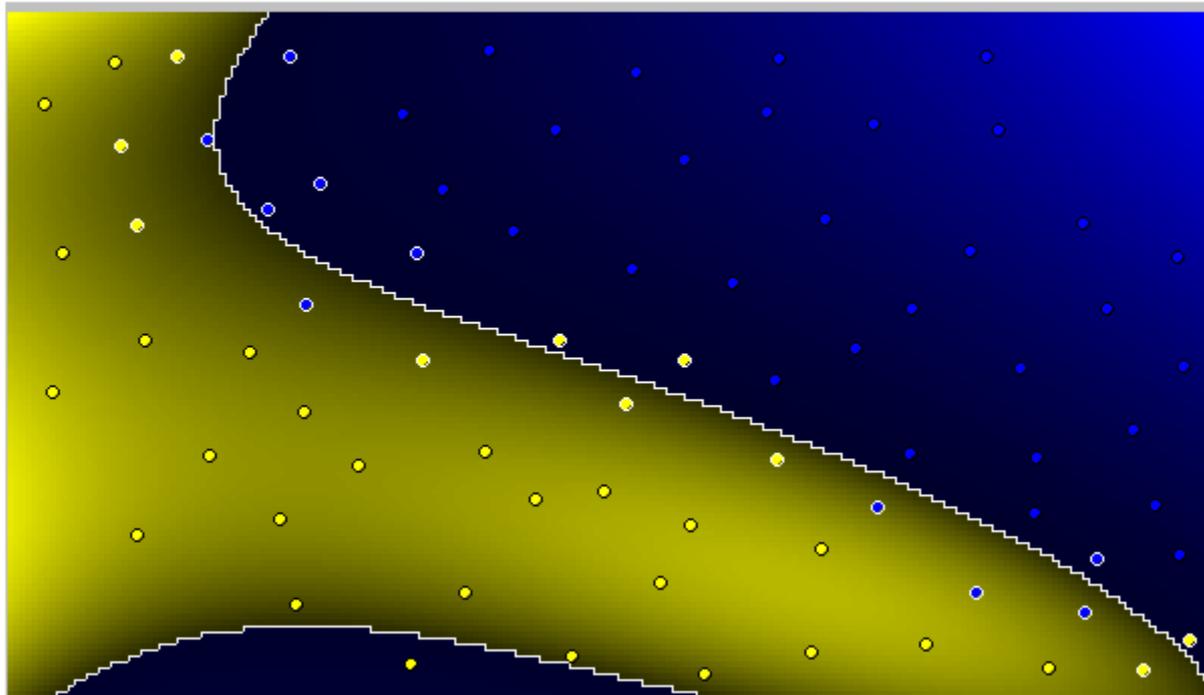
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: **20** (-ve: 10, +ve: 10) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

Polinomiale, grado 3

Polinomiale, grado 4

Polinomiale, grado 3, C=100

**Polinomiale, grado 3, C=1000**

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

RBF, sigma 1

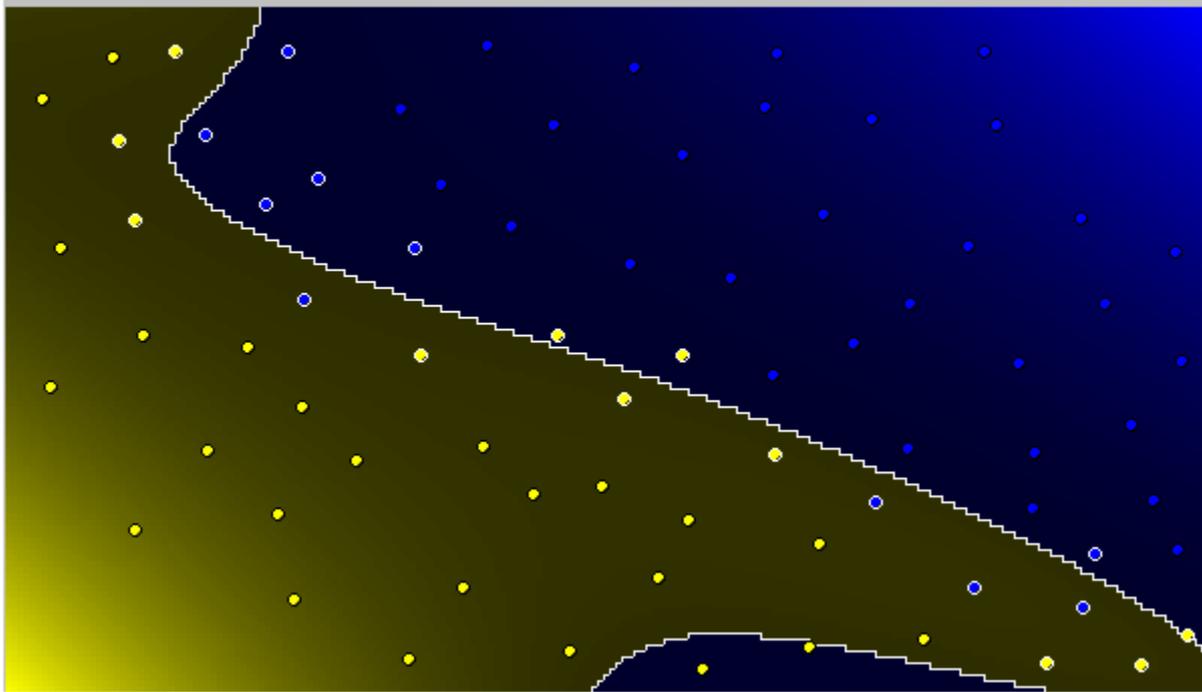
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: 21 (-ve: 10, +ve: 11) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

Polinomiale, grado 3

Polinomiale, grado 4

Polinomiale, grado 3,  $C=100$

Polinomiale, grado 3,  $C=1000$

**Polinomiale, grado 3,  
 $C=10000$**

RBF, sigma 0.5

RBF, sigma 1

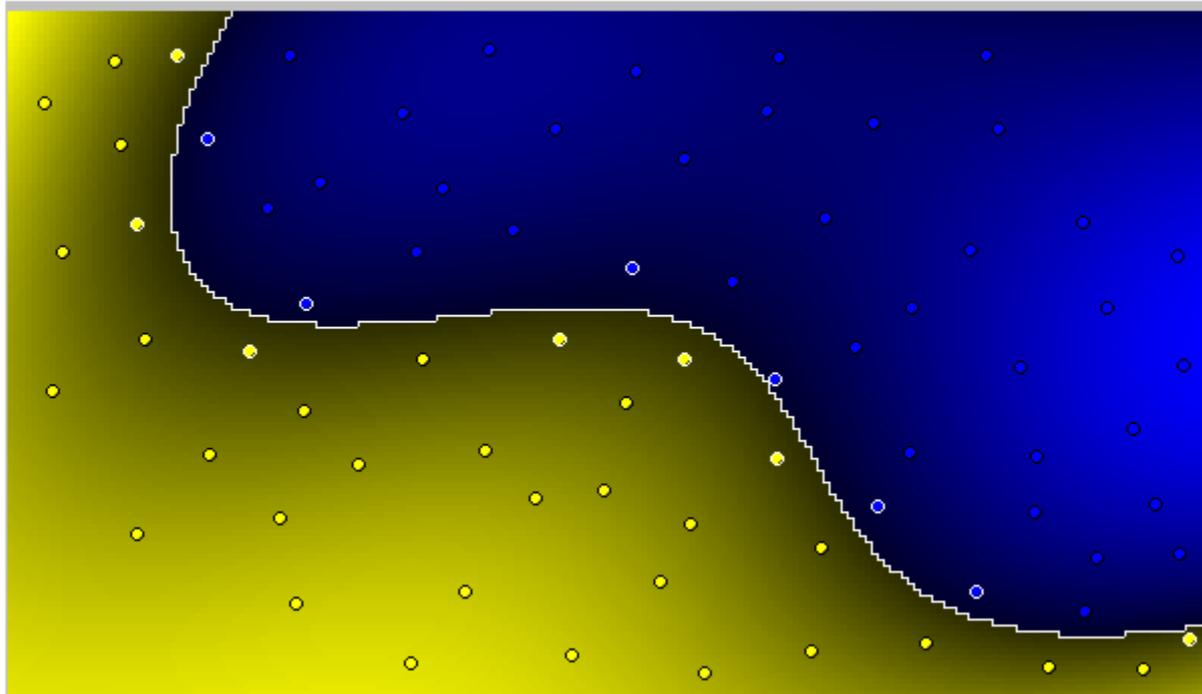
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: 13 (-ve: 6, +ve: 7) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

Polinomiale, grado 3

Polinomiale, grado 4

Polinomiale, grado 3, C=100

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

**RBF, sigma 0.5**

RBF, sigma 1

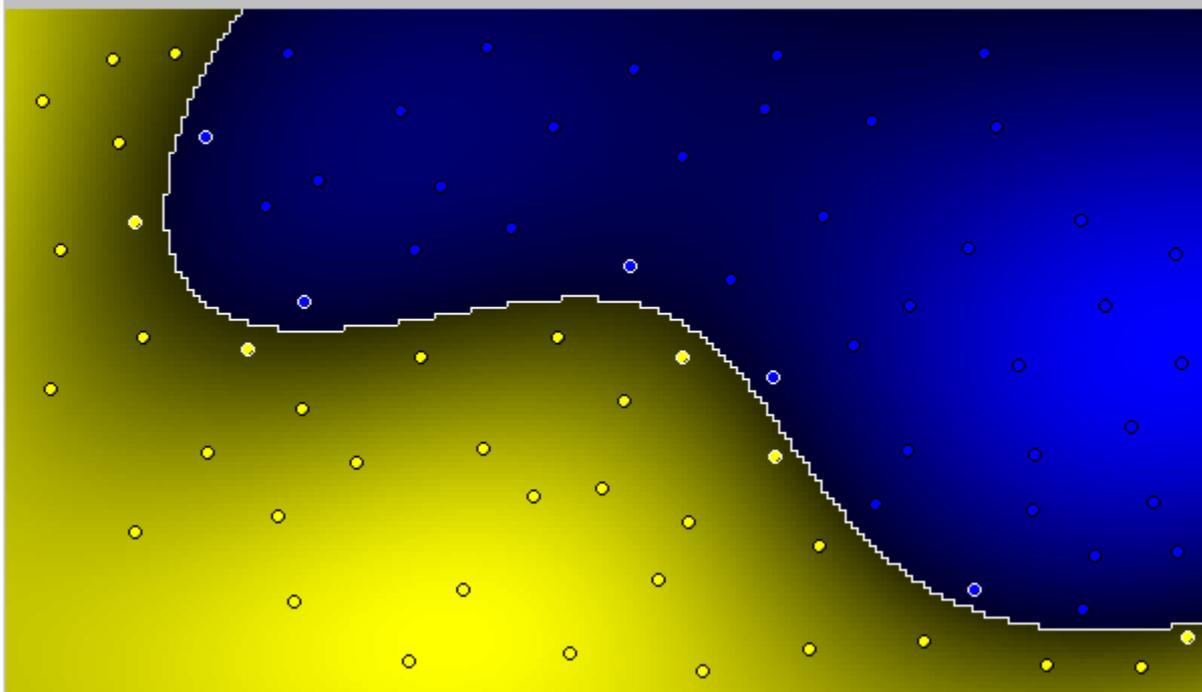
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: **10** (-ve: 5, +ve: 5) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

Polinomiale, grado 3

Polinomiale, grado 4

Polinomiale, grado 3, C=100

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

**RBF, sigma 1**

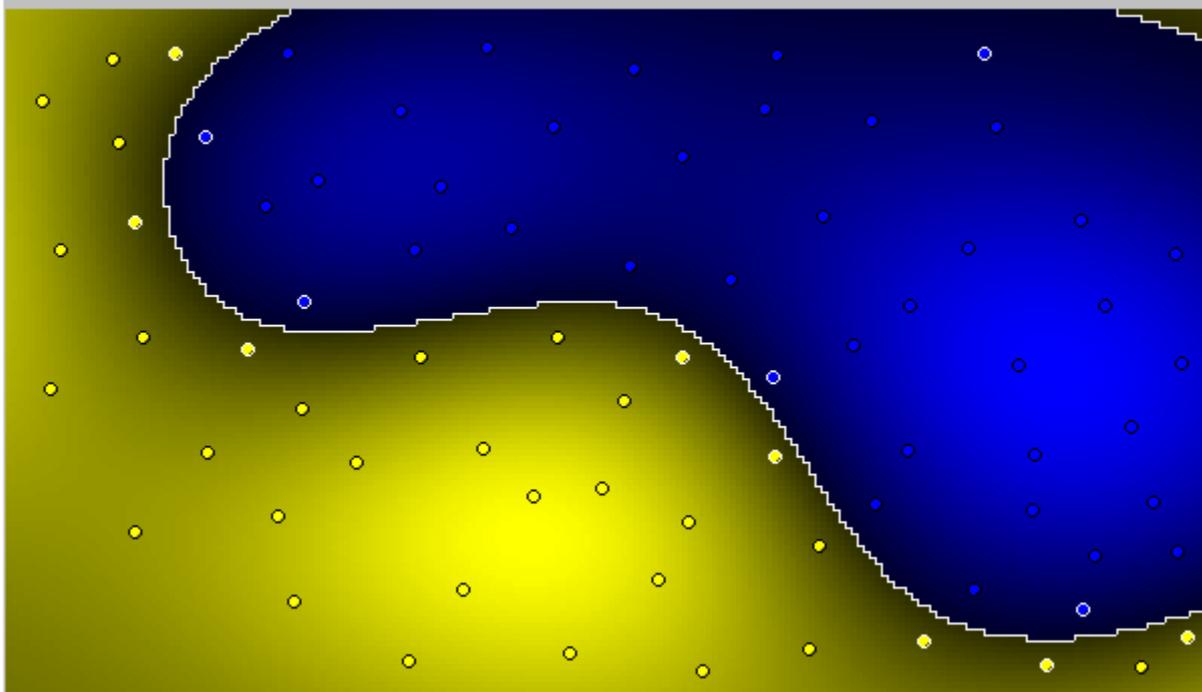
RBF, sigma 2

RBF, sigma 7



# Classification examples

Number of Support Vectors: 13 (-ve: 5, +ve: 8) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

Polinomiale, grado 3

Polinomiale, grado 4

Polinomiale, grado 3, C=100

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

RBF, sigma 1

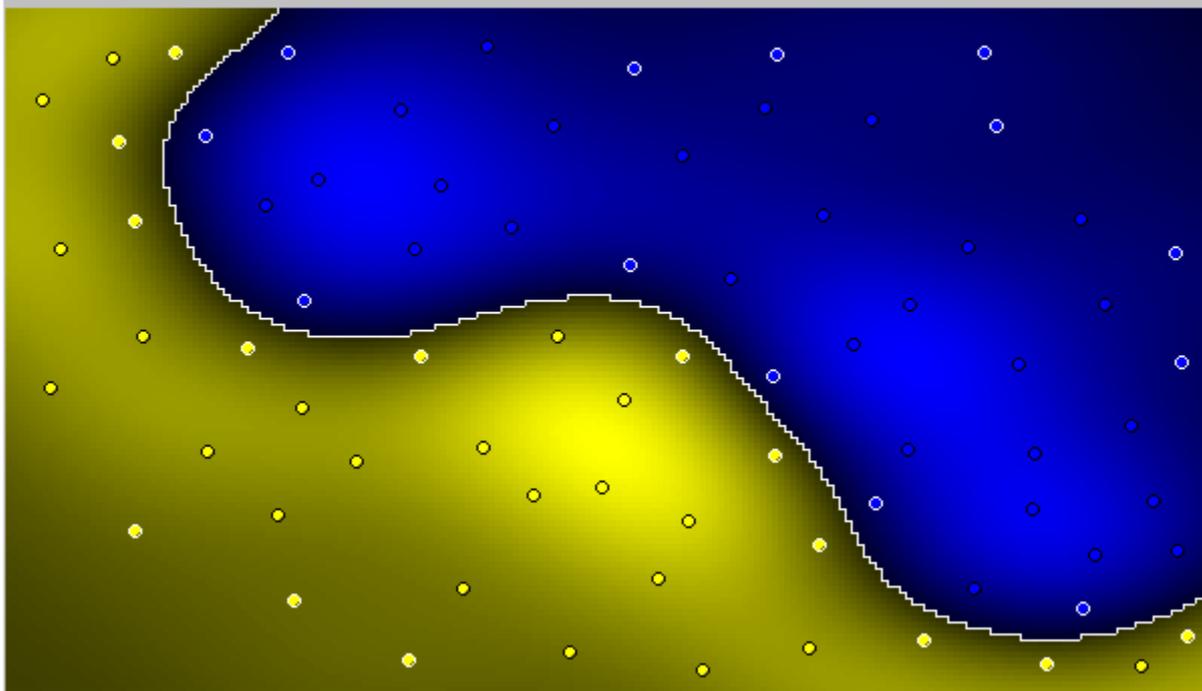
**RBF, sigma 2**

RBF, sigma 7



# Classification examples

Number of Support Vectors: 27 (-ve: 13, +ve: 14) Total number of points: 75



Kernel lineare

Polinomiale, grado 1

Polinomiale, grado 2

Polinomiale, grado 3

Polinomiale, grado 4

Polinomiale, grado 3, C=100

Polinomiale, grado 3, C=1000

Polinomiale, grado 3, C=10000

RBF, sigma 0.5

RBF, sigma 1

RBF, sigma 2

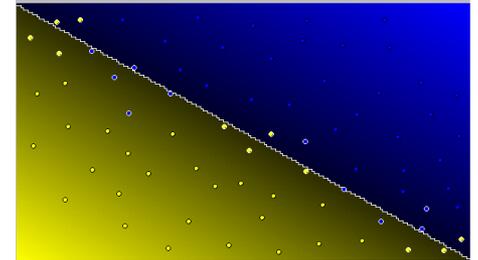
**RBF, sigma 7**



# Selezione del kernel

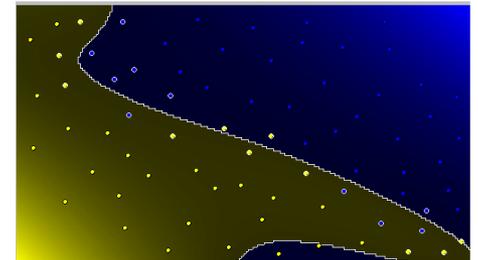
## Kernel Lineare

- Used when the feature space is huge (for example in text classification, which uses individual word counts as features)
- Shown to be a special case of the RBF kernel
- No additional parameters



## Polinomiale

- Has numerical difficulties approaching 0 or infinity
- A good choice for well known and well conditioned tasks
- One additional parameter (degree  $p$ )



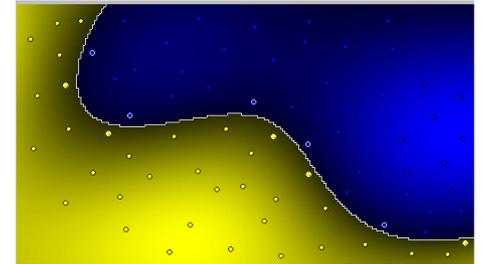
# Kernel selection

## Radial Basis Function

- Indicated in general as the best choice in the literature
- One additional parameter (sigma  $\sigma$ )

## Sigmoide

- Two additional parameters (a and b)
- For some values of a and b, the kernel doesn't satisfy the Mercer condition
- From neural networks
- Not recommended in the literature



*Scegliere il giusto kernel è un'arte...*



# Cookbook approach

1. Conduct simple scaling on the data
2. Consider RBF kernel
3. Use cross-validation to find the best parameter  $C$  and  $\sigma$
4. Use the best  $C$  and  $\sigma$  to train the whole training set
5. Test



# Cookbook problems

- Parameter search can be very time consuming  
→ Solution: conduct parameter search hierarchically
- RBF kernels are sometimes subject to overfitting  
→ Solution: use high degree polynomials kernels
- Parameter search must be repeated for every chosen features; there no reuse of computations  
→ Solution: compare features on random subsets of the entire dataset to contain computational cost
- Search ranges for C and  $\sigma$  are tricky to choose  
→ Solution: literature suggests using exponentially growing values like  $C = 2^{[-5..15]}$  and  $\sigma = 2^{[-15..5]}$



# Materiale aggiuntivo

- Il libro sulle SVM
  - Cristianini, Nello, and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.
- Una più semplice introduzione
  - Simple Intro to SVM.pdf
- Slides toste sulle SVM
  - Heavy slides on SVM.pdf

