Facoltà di Scienze MM. FF. NN.

Università di Verona

A.A. 2010-11

Teoria e Tecniche del Riconoscimento

Modelli discriminativi: funzioni lineari, Support Vector Machines

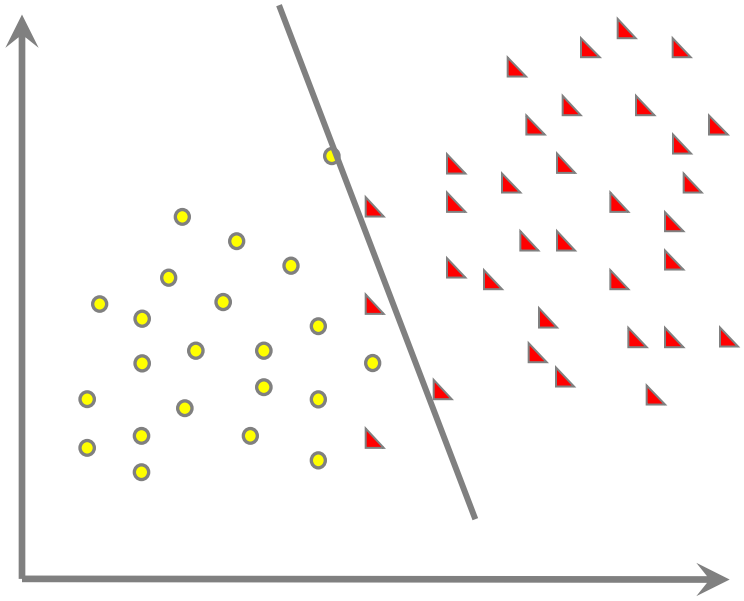
Introduzione

- 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
- Approcci geometrici (possono essere trovati parallelismi tra questi e la regola di Bayes)
  - Funzioni discriminanti lineari
  - Support Vector Machines

Funzioni discriminanti lineari

Funzioni discriminanti lineari

- Obiettivo è trovare la retta che separa le due classi



- L'abbiamo visto finora per distribuzioni gaussiane…

Funzioni discriminanti lineari

- Definiamo la retta generica in R^m

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

dove $\mathbf{x}=[x_1,...,x_m]$, $\mathbf{w}=[w_1,...,w_m]$ (pesi) e w_0 *bias* (soglia).

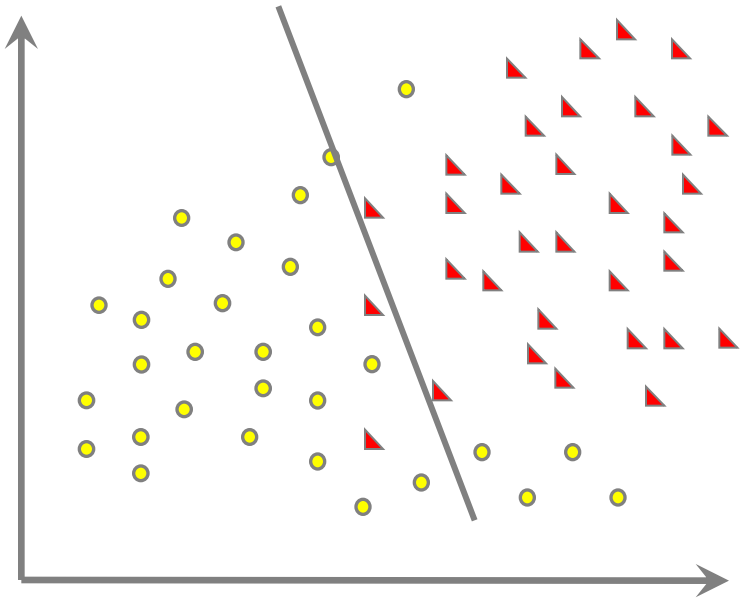
- *Un campione* $\boldsymbol{x}_i$ *è classificato come appartenente a* ω_1 *se* $\boldsymbol{w}^t\boldsymbol{x}_i+w_0>0$, *a* ω_2 *se* $\boldsymbol{w}^t\boldsymbol{x}_i + w_0 < 0$,

- Più genericamente, consideriamo come $\mathbf{w'} = [w_0,w_1,...,w_m]$ e $\mathbf{x'} =[1,x_1,...,x_m]$ per inserire il termine noto, quindi la regola di decisione diventa:

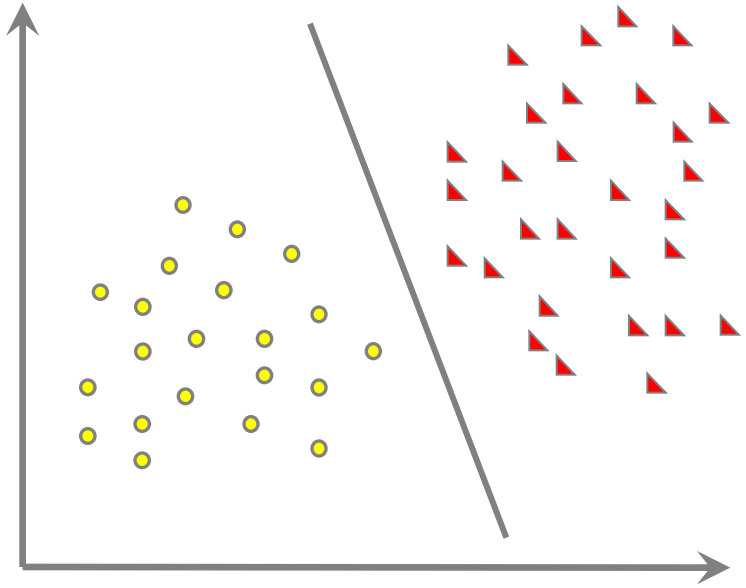
  Un campione $\boldsymbol{x}_i$ *è classificato come appartenente a* ω_1 *se* $\boldsymbol{w'}^t\boldsymbol{x}_i'>0$, *a* ω_2 *se* $\boldsymbol{w'}^t\boldsymbol{x}_i' < 0$

Funzioni discriminanti lineari

- Training: dato un training set (insieme di *m* campioni $\mathbf{x}_1,\ldots,\mathbf{x}_m$, alcuni etichettati ω_1 ed altri etichettati ω_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

  $\mathbf{w}^t\mathbf{x}_i > 0$ per ogni $\mathbf{x}_i$ appartenente a ω_1

  $\mathbf{w}^t\mathbf{x}_i < 0$ per ogni $\mathbf{x}_i$ appartenente a ω_2
- In quest'ultimo caso (classe 2) si può anche dire che $\mathbf{x}_i$ è classificato correttamente se $\mathbf{w}^t(-\mathbf{x}_i) > 0$.
- 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 che $\mathbf{w}^t\mathbf{x}_i>0$ per tutti i campioni a prescindere dalle classi.
- Questo vettore è chiamato **vettore separatore** o **vettore soluzione.**

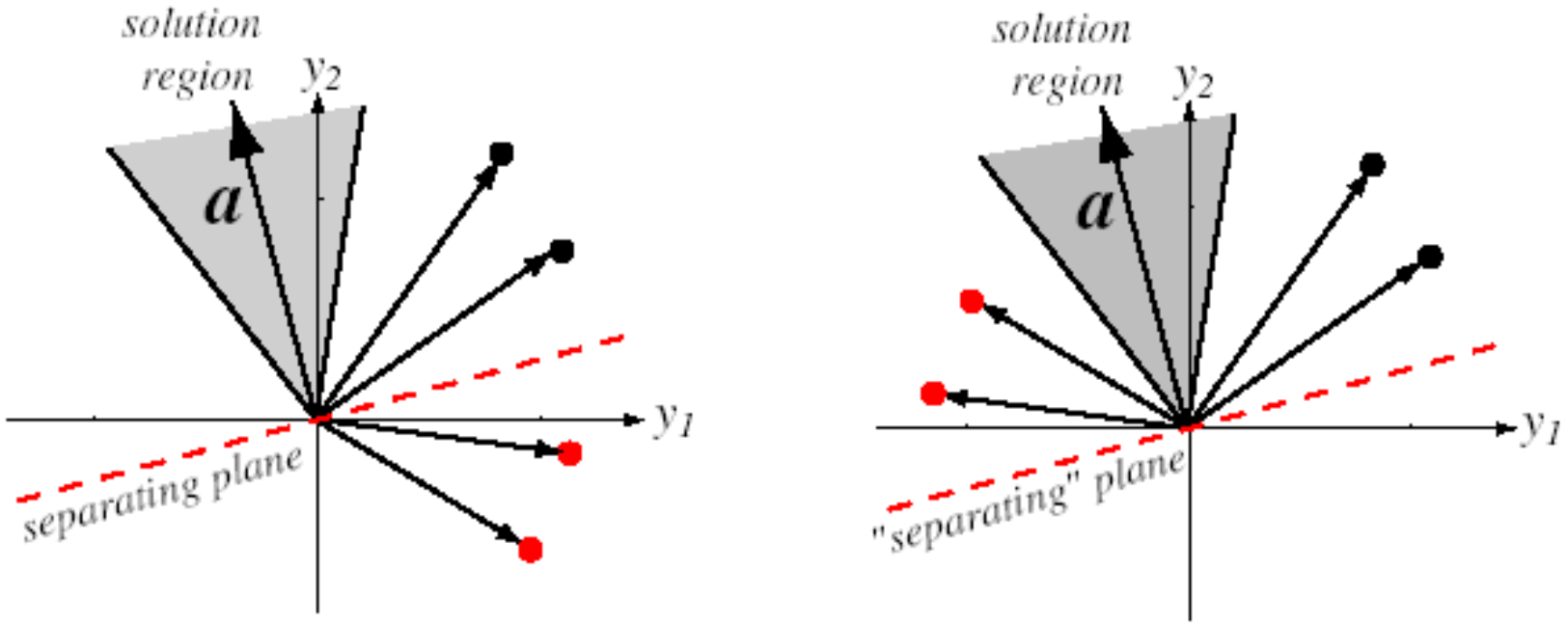


FIGURE 5.8. Four training samples (black for ω_1, red for ω_2) and the solution region in feature space. The figure on the left shows the raw data; the solution vectors leads to a plane that separates the patterns from the two categories. In the figure on the right, the red points have been "normalized"—that is, changed in sign. Now the solution vector leads to a plane that places all "normalized" points on the same side. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- Risulta chiaro, quindi, 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 di lunghezza unitaria che massimizzi la minima distanza dei campioni dal piano separatore.
- Un'altra potrebbe essere quella di cercare il vettore dei pesi a lunghezza minima che soddisfi

$$\mathbf{w}^t \mathbf{x}_i \geq b, \quad \forall i$$

dove *b* è una costante positiva chiamata *margine*.

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 punta nella direzione di massimo scarto di una funzione da massimizzare/minimizzare

Determinazione dei pesi w

- La procedura consiste nell'aggiornare il valore del vettore dei pesi al passo k+1 con un contributo proporzionale al modulo del gradiente stesso 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(**w**) è una funzione di valutazione che deve essere minimizzata.
- J(**w**) viene scelta in modo tale da raggiungere il minimo all'avvicinarsi di **w** alla soluzione ottima, ovvero convessa.

Determinazione dei pesi w

- Il minimo di J(**w**) si ottiene spostando **w** in direzione opposta al gradiente.
- ∇ è il simbolo dell'operatore gradiente, dato da:

$$\nabla J = \begin{bmatrix} \dfrac{\partial J}{\partial w_1} \\ \vdots \\ \vdots \\ \vdots \\ \dfrac{\partial J}{\partial w_m} \end{bmatrix}$$

- ρ_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(**w**)
- La scelta più ovvia è quella di definire un funzionale $J(\mathbf{w};\mathbf{x}_1,...,\mathbf{x}_N)$ rappresentato dal numero di campioni mal classificati da **w** su un totale di N campioni
- 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:

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

dove *X* è l'insieme di punti classificati non correttamente da **w**.

Determinazione dei pesi w

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

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

- e quindi l'algoritmo di discesa del gradiente è

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

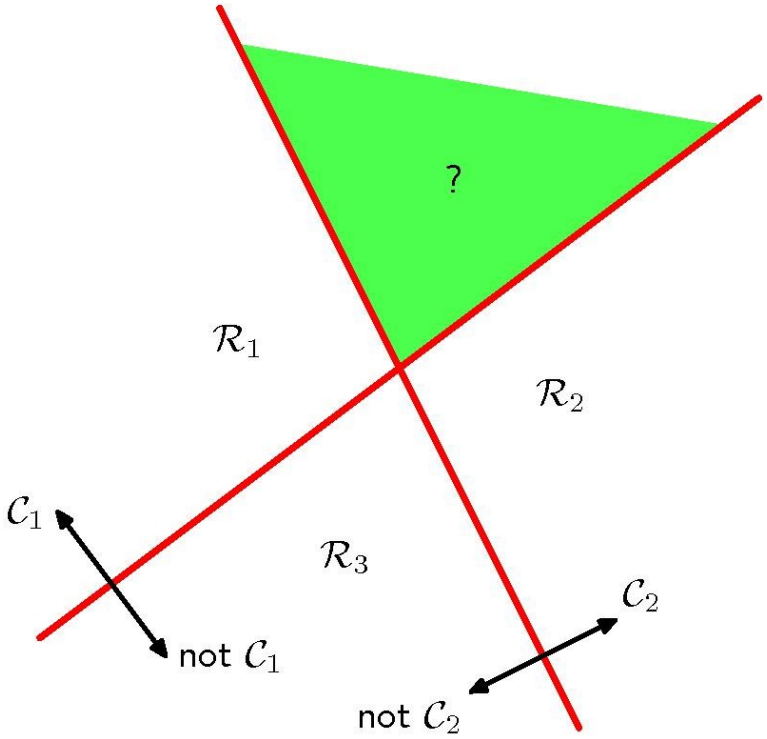
Determinazione dei pesi w

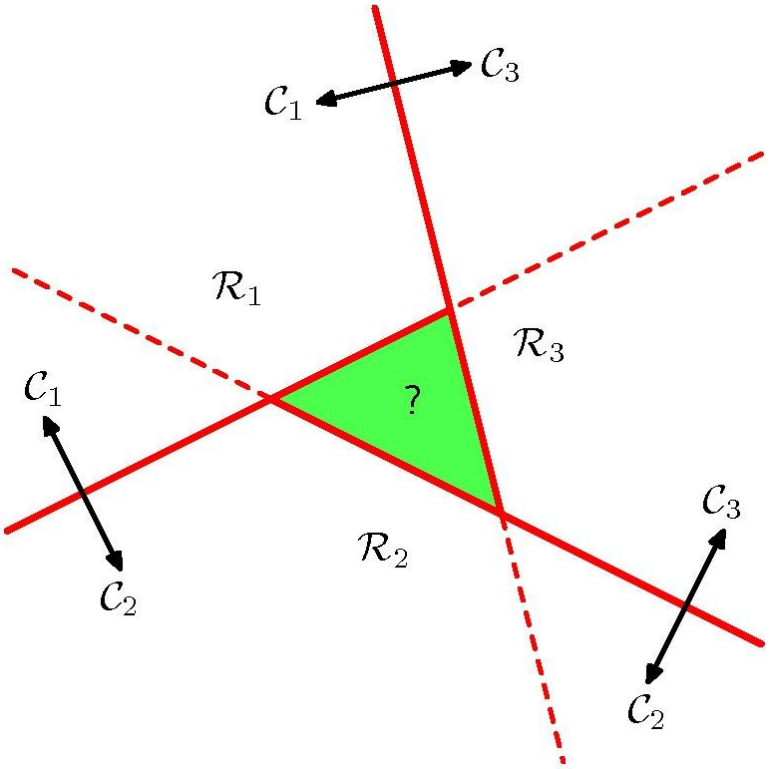
- Questo metodo è stato usato per simulare sia in modo *hardware* che *software* la prima rete neurale ed si è in seguito evoluto a più complesse versioni come, ad esempio il *perceptron* multilivello

- Ci sono molti altri metodi per scegliere J(w) e per ottimizzarla:
  - **Metodo del rilassamento**
  - **Metodo del MSE (minimun square error)**
  - **Metodo del Least MSE o Widrow-Hoff**
  - **Metodo di Ho-Kashyap**

Caso multi classe

- Nel caso di problemi a C classi, possono essere costruiti C-1 classificatori $g_i(\mathbf{x})$, uno per ogni classe C_i contro *non-*C_i, 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)
- Entrambi portano a zone di ambiguità.


?
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_3$
$\mathcal{C}_1$
not $\mathcal{C}_1$
$\mathcal{C}_2$
not $\mathcal{C}_2$


$\mathcal{C}_1$
$\mathcal{C}_3$
$\mathcal{R}_1$
$\mathcal{R}_3$
$\mathcal{C}_1$
?
$\mathcal{C}_3$
$\mathcal{R}_2$
$\mathcal{C}_2$
$\mathcal{C}_2$

Funzioni discriminanti lineari generalizzate

- Abbiamo visto che la funzione discriminante lineare può essere scritta come

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

dove w_i è l'elemento del vettore dei pesi e w_0 è il peso di soglia.

- Possiamo aggiungere altri termini in cui mettiamo i prodotti delle varie componenti di x
- Esempio: classificatore discriminante quadratico

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

- Possiamo generalizzare il concetto scrivendo

$$g(\mathbf{x}) = \sum_{i=1}^{d1} a_i \cdot y_i(\mathbf{x})$$

  dove $y_i(\mathbf{x})$ sono funzioni arbitrarie su **x**
- In questo caso, g(**x**) si chiama funzione discriminante lineare generalizzata
  - lineare rispetto a $y_i(\mathbf{x})$
  - non lineare nello spazio originale di **x**
- $y_i(\mathbf{x})$ rappresenta una sorta di “feature extraction”
  - solo che di solito si sale di dimensionalità, non si scende (come abbiamo visto nel caso della PCA)

Esempio:

- consideriamo la funzione quadratica

$$g(\mathbf{x}) = a_1 + a_2\mathbf{x} + a_3\mathbf{x}^2$$

- Nella formula del discriminante lineare generalizzato, questo equivale a definire tre funzioni $y_i(\mathbf{x})$

$$y_1(\mathbf{x}) = 1$$

$$y_2(\mathbf{x}) = \mathbf{x}$$

$$y_3(\mathbf{x}) = \mathbf{x}^2$$

- Quindi si passa da uno spazio monodimensionale ad uno spazio tridimensionale

- Adesso la funzione g(**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 **x**

- Idea alla base delle Support Vector Machines

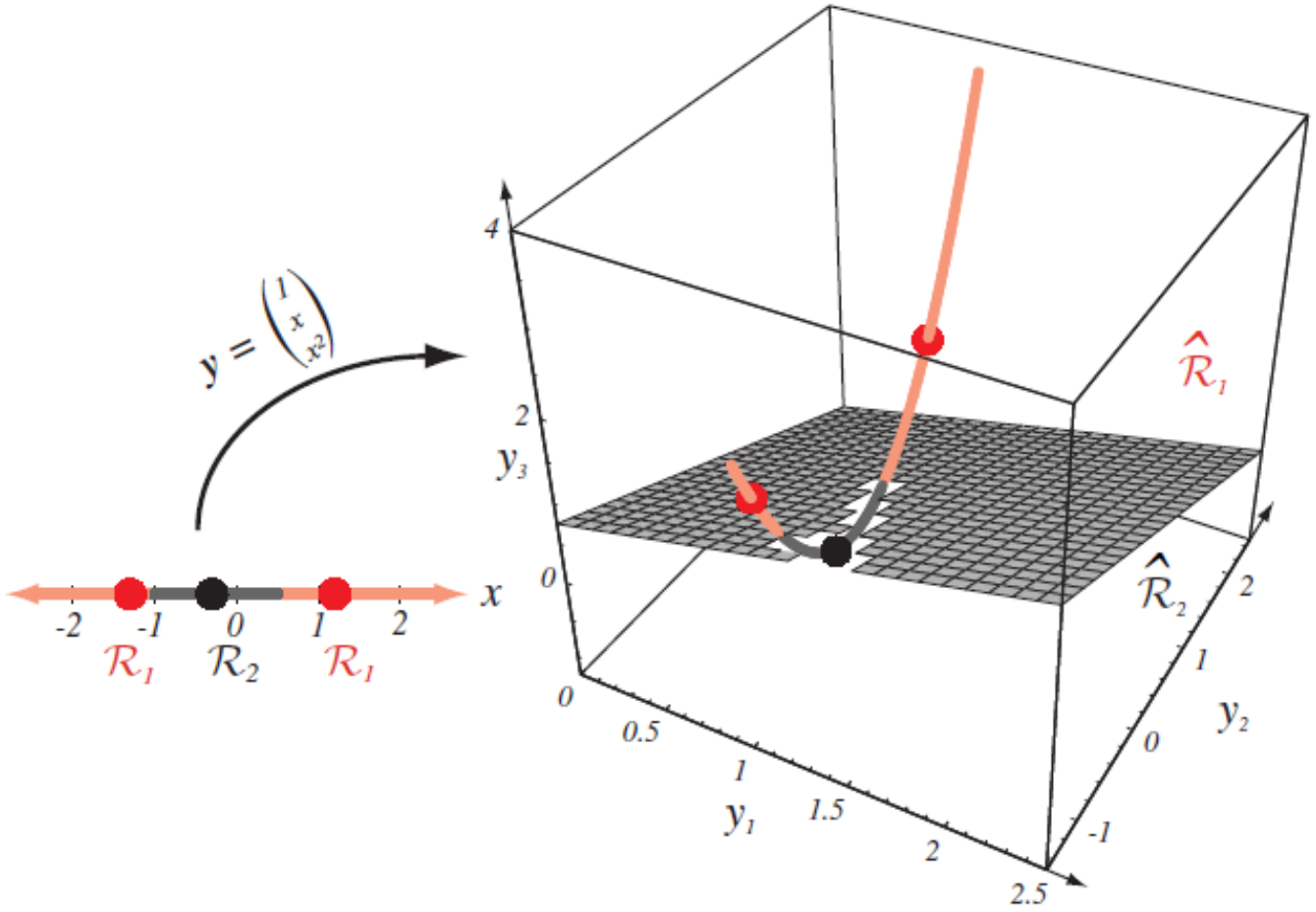


FIGURE 5.5. The mapping $\mathbf{y} = (1, x, x^2)^t$ takes a line and transforms it to a parabola in three dimensions. A plane splits the resulting $\mathbf{y}$-space into regions corresponding to two categories, and this in turn gives a nonsimply connected decision region in the one-dimensional x-space.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

Support Vector Machines

Separatori Lineari (Percettrone)

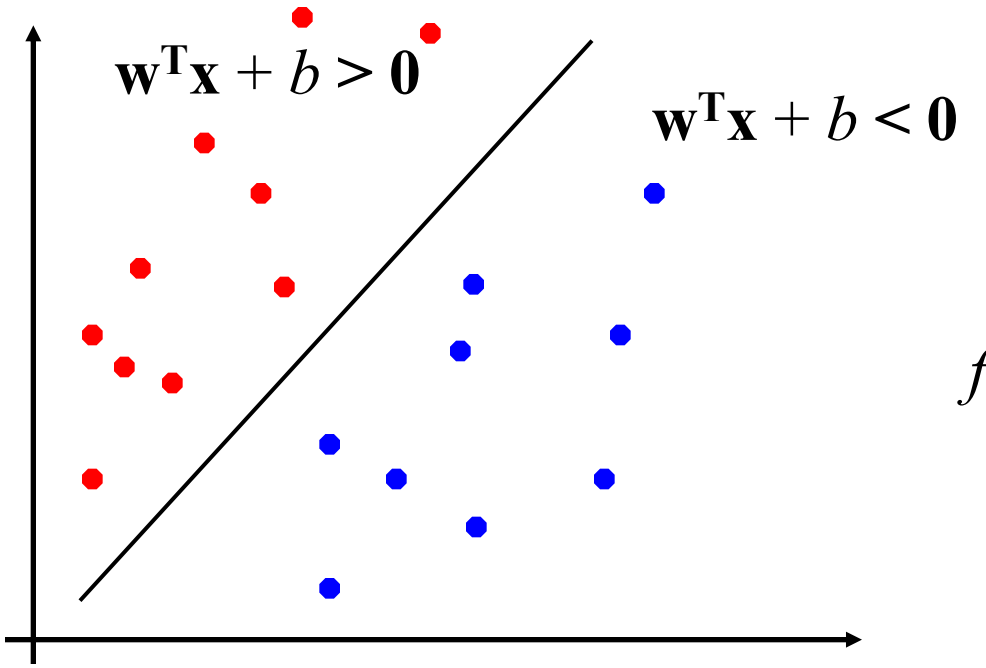
- La classificazione binaria può essere vista come un problema di separazione di classi nello spazio delle *features*

$$y(\mathbf{x}) = \sum_{j=1}^{m} w_j x_j + b = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$$

w vettore dei pesi, $\mathbf{w}^{\mathbf{T}}$ trasposta di **w**

x vettore associato ad una istanza di X

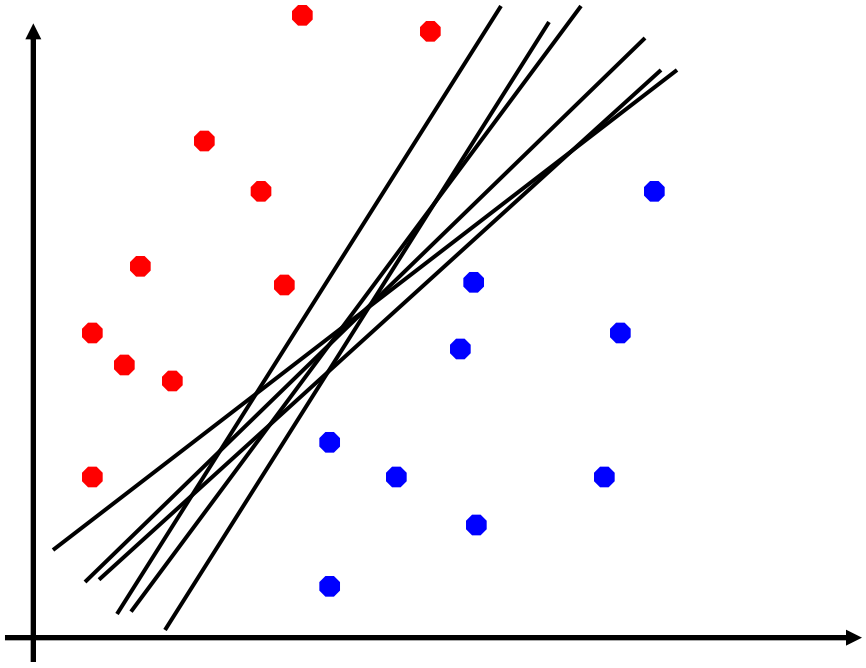
$\mathbf{w}^{\mathbf{T}}\mathbf{x}$ prodotto scalare



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

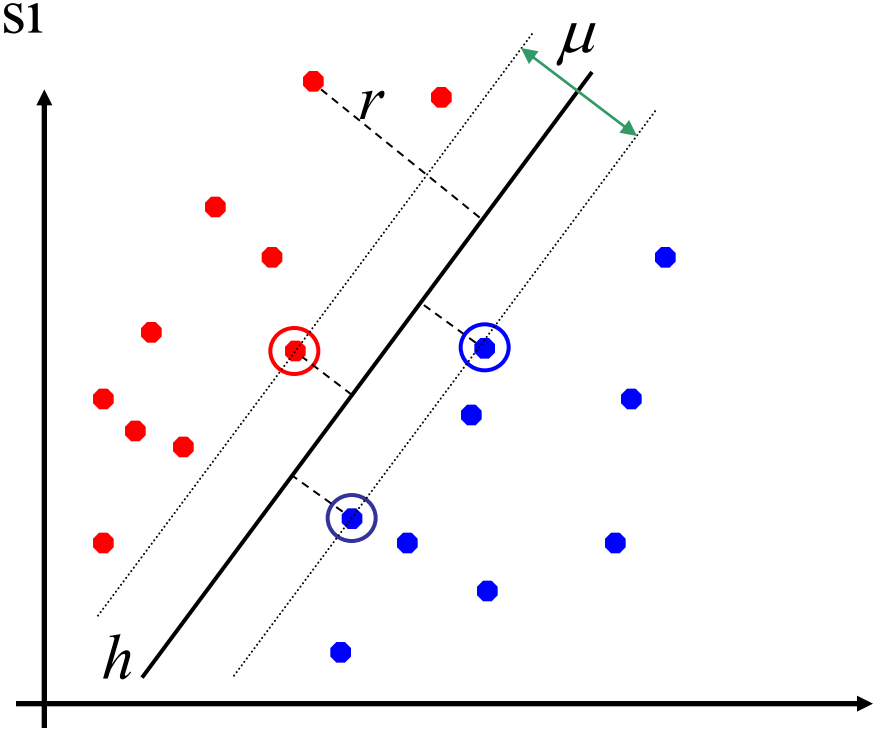
Separatori Lineari

- Quale separatore è ottimo?



Classification Margin

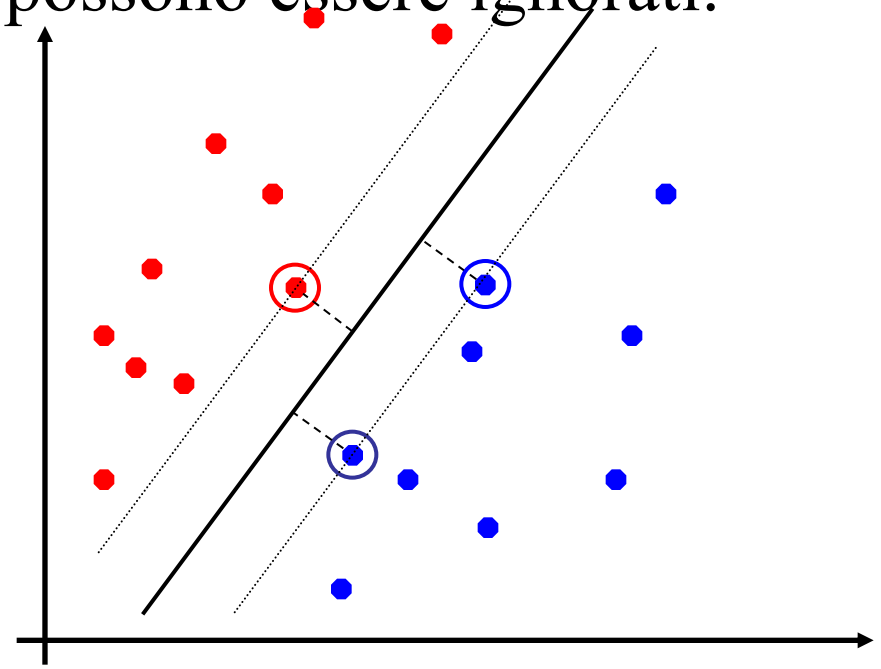
- La distanza di un esempio dall'iperpiano é r
- Gli esempi più vicini all'iperpiano si chiamano ***support vectors***.
- ***Il margine*** μ dell'iperpiano di separazione *h* è la distanza minima fra le due classi



Classificazione con il margine massimo

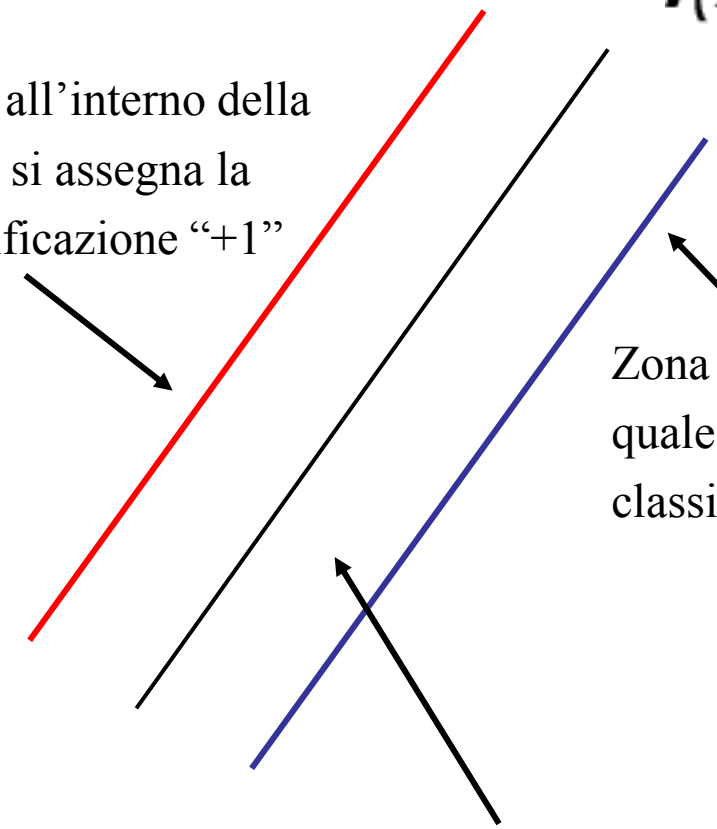
- Massimizzare il margine corrisponde ad individuare l'iperpiano ottimo *h*.
- Questo implica che solo alcuni esempi sono importanti per l'apprendimento, i **vettori di supporto**. Gli altri possono essere ignorati.

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



- Evidenzio in verde le variabili incognite del mio problema

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



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

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

Zona di incertezza

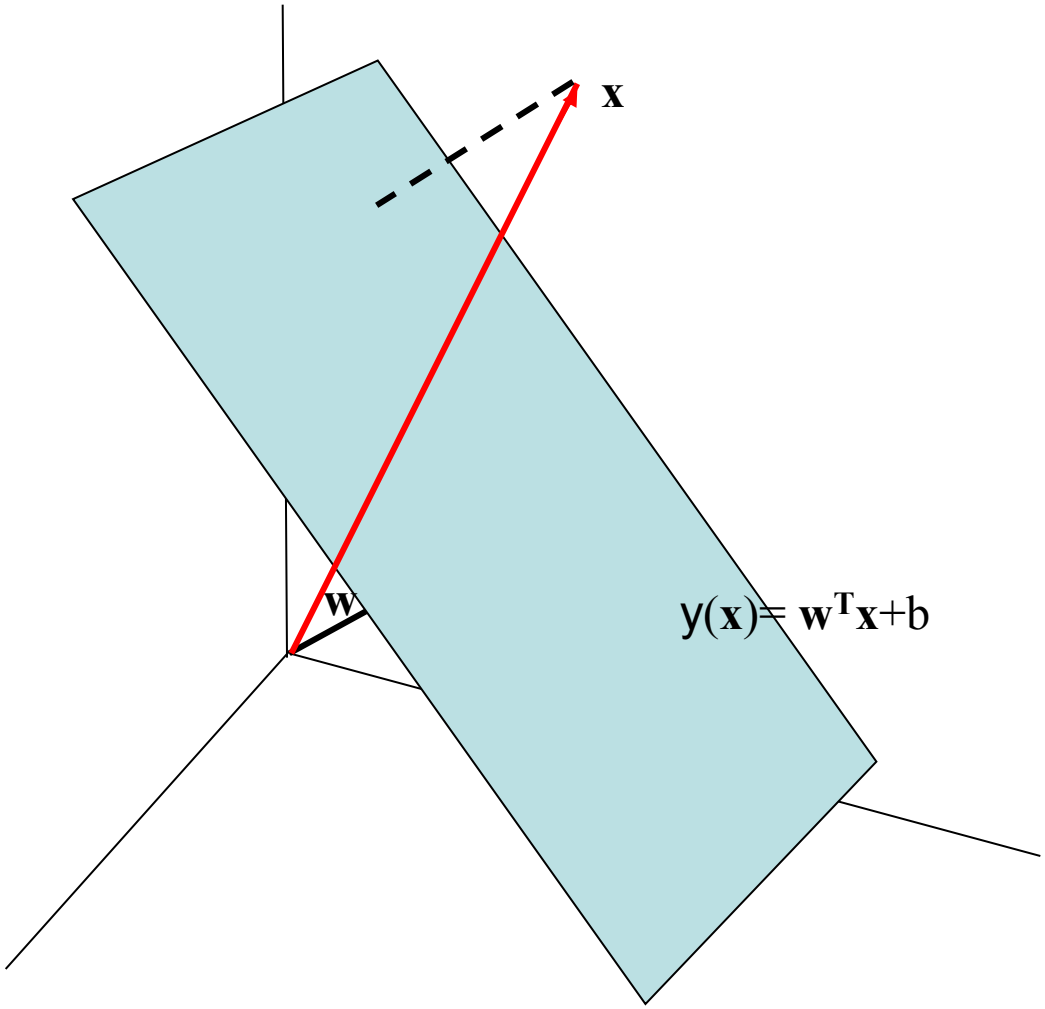
$$\mathbf{w}^T \cdot \mathbf{x} + b \geq 1$$

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

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

• **w** è il vettore perpendicolare al piano di separazione

• b è la distanza dall'origine



SVM lineare

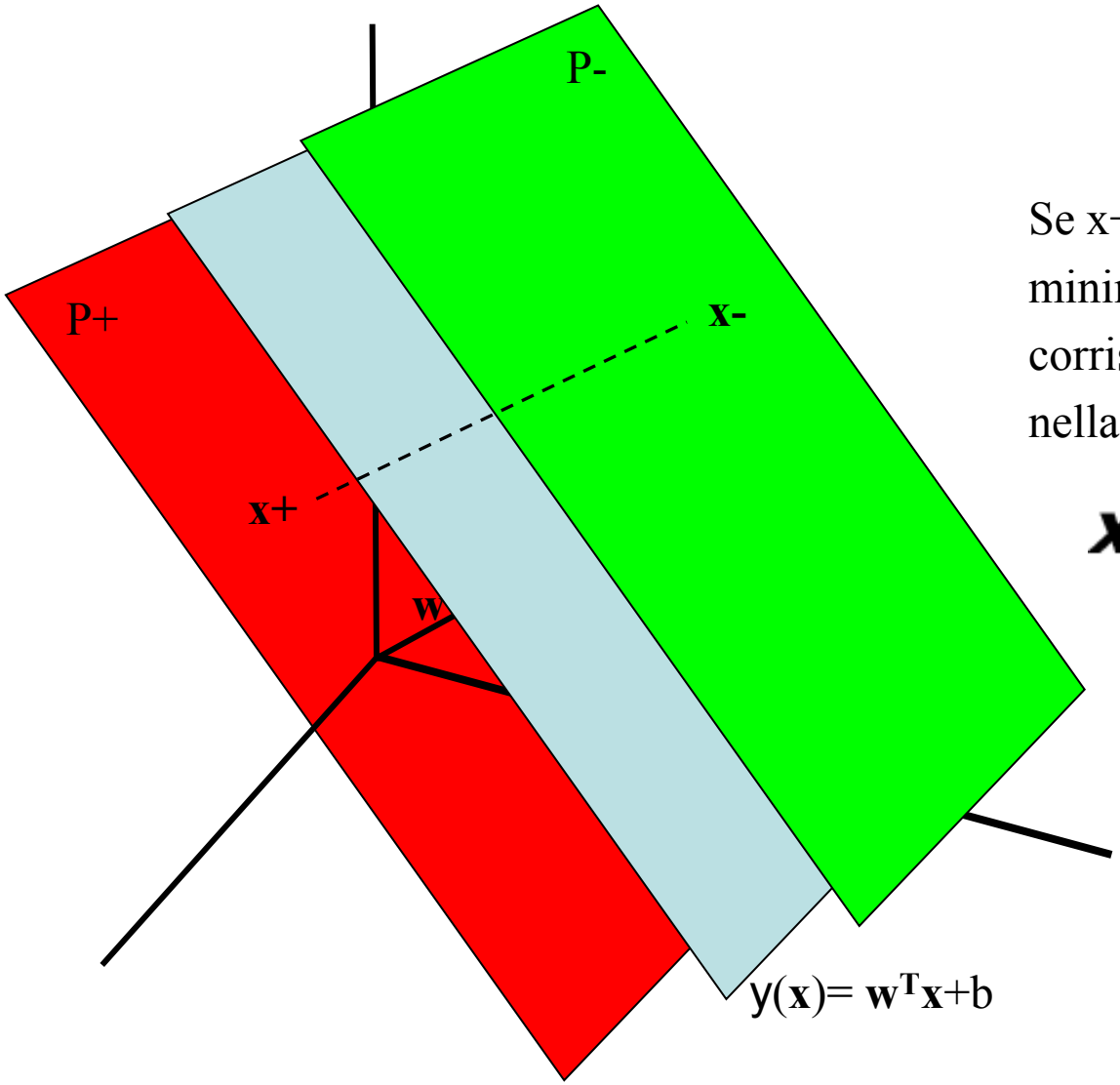
- Se assumiamo che i dati di addestramento D $\{(\mathbf{x_i}, y_i)\}$ si trovino a distanza almeno 1 dall'iperpiano, allora valgono le seguenti condizioni per $\mathbf{x_i}$ in D:

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

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

- Per i vettori di supporto, la diseguaglianza diventa una eguaglianza;
- Indichiamo con ρ la distanza fra i piani P+: $\mathbf{w^T x}_i + b = 1$ e P-: $\mathbf{w^T x}_i + b = -1$ e
- Sia x^+ un punto di P+ e x^- un punto du P- a distanza minima da x+
- $\rho = \| \mathbf{x}^+ - \mathbf{x}^- \|$ che si può scrivere anche: $(\mathbf{x}^+ - \mathbf{x}^-) = \lambda \mathbf{w}$

Perché?



Se x+ e x- sono a distanza minima , muoversi da x+ a x- corrisponde ad un percorso nella direzione di **w**

$$x^+ = x^- + \lambda w$$

Per riassumere:

$$w^T x^+ + b = +1$$

$$w^T x^- + b = -1$$

$$x^+ = x^- + \lambda w$$

Mettendo assieme:

- Isolando λ e $\mathbf{w}$, giungiamo a:

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

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

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

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

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

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

SVM lineare (2)

- Il problema di ottimizzazione quadratica che ne risulta é:

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

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

$\mathbf{w^T x_i} + b \geq 1$ se $y_i = 1$; $\mathbf{w^T x_i} + b \leq -1$ se $y_i = -1$

- Normalizzando come visto per le funzioni lineari, una formulazione migliore é:

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

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

E per ogni $\{(\mathbf{x_i}, y_i)\}$: $y_i\ (\mathbf{w^T x_i} + b) \geq 1$

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 Lagrangiani

Lagrangiani

- Data una funzione da ottimizzare $\mathcal{F}$ ed un insieme di condizioni $f_1..f_n$, un *Lagrangiano* è una funzione $\mathcal{L}(\mathcal{F},f_1,..f_n, \alpha_1,.. \alpha_n)$ che "incorpora" le condizioni nel problema di ottimizzazione
- Es: $\mathcal{F}$: $w^2/2$ e $f_1 : wx-1 \geq 0$

Lagrangiano:

$$L(w,\alpha) = \frac{w^2}{2} - \alpha(wx-1), \quad \alpha \geq 0$$

- α è detto *moltiplicatore di Lagrange*, ne esiste uno per ogni vincolo. I vincoli vengono <u>sottratti</u> alla funzione.
- Si calcolano le *derivate* rispetto alle variabili del lagrangiano (w e α in questo esempio) e si impone siano =0
- Risolvendo il sistema di equazioni ottenuto, si ricavano i valori che soddisfano il problema

Calcola le derivate parziali di α

$$L(w,\alpha) = \frac{w^2}{2} - \alpha(wx - 1), \quad \alpha \geq 0$$

$$\frac{\partial(L)}{\partial(w)} = w - \alpha x = 0 \quad \Rightarrow w = \alpha x$$

$$\frac{\partial(L)}{\partial(\alpha)} = 0$$

Torniamo al problema di SVM

- Minimizzare il seguente *Lagrangiano*:

<$\mathbf{x}_i$,y_i> learning set

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

- 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 sotto-insieme di vettori, 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:

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

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

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)

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

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

$$\alpha_i^*[y_i(\vec{w}^*\cdot\vec{x}_i + b^*) - 1] = 0 \;\; i = 1,\ldots,n$$

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)

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \vec{x}_i\cdot\vec{x}_j$$

soggetto a: $\forall i \in \{1,\ldots,n\} : \alpha_i \geq 0$ e $\sum_{i=1}^{n} y_i\alpha_i = 0.$

RIASSUMIAMO

1) Formulazione del problema di ottimizzazione:

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

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

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 Tuck

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

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

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

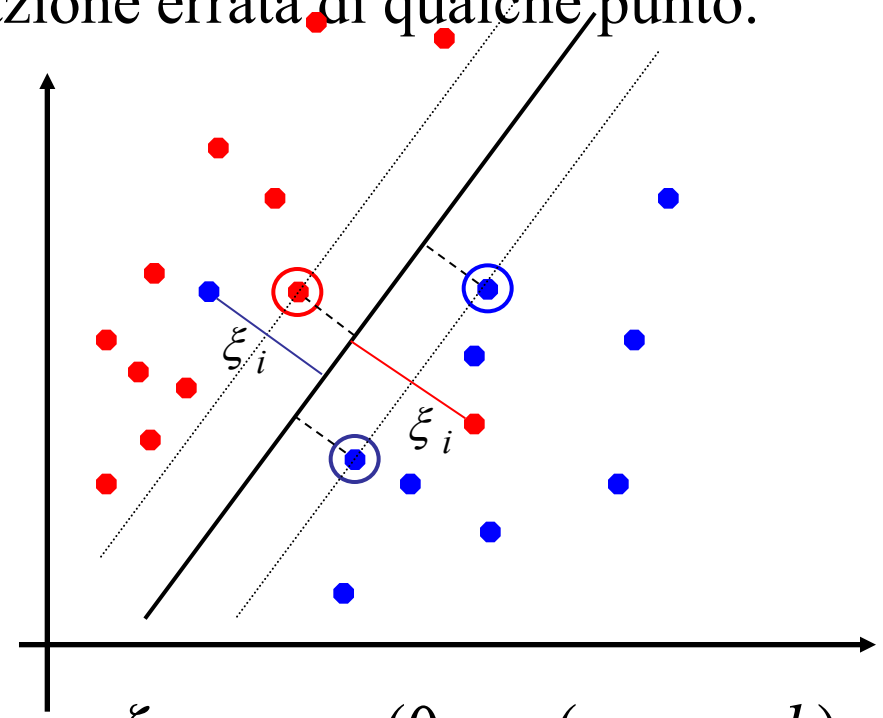
$$\min\left(\frac{1}{2}\sum(\alpha_i y_i x_i)\sum(\alpha_i y_i x_i) - \sum(\alpha_i y_i x_i)\sum(\alpha_i y_i x_i) - b\cdot 0 + \sum a_i\right) =$$

4) Ottie duale el variabil (formul

$$\max\left(\sum a_i - \frac{1}{2}\sum_{i,j=1..n}\alpha_i\alpha_j y_i y_j x_i x_j\right)$$

Margini "Soft"

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



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

Soft Margin Classification

- Problema lineare:

  Trova **w** e b t.c.
  $\mathbf{\Phi(w)} = \mathbf{1/2}\ \mathbf{w}^T\mathbf{w}$ 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:

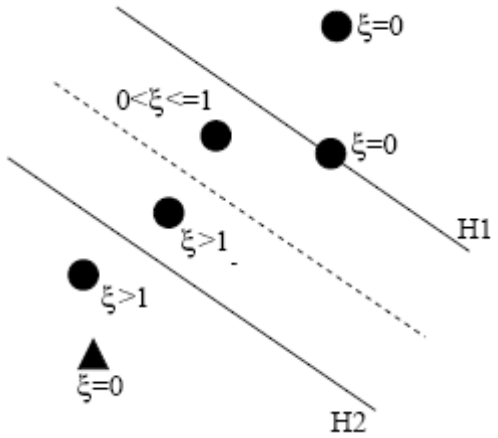
  Trova **w** e b t.c.
  $\mathbf{\Phi(w)} = \mathbf{1/2}\ \mathbf{w}^T\mathbf{w} + C\sum \xi_i$ è minimizzata e per ogni $\{(\mathbf{x_i}, y_i)\}$
  $y_i\ (\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ e $\xi_i \geq 0$ per ogni i

- Il parametro C controlla l'overfitting.

Soluzione per dati NLS

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

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



I valori di ξ si interpretano:

$\xi_i = 0$	classificazione corretta
$0 < \xi_i < 1$	classificazione corretta ma fuori H_i
$\xi_i = 1$	errore di classificazione

Sommario di SVM lineare

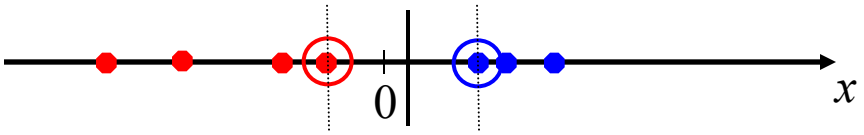
- Il classificatore (funzione obiettivo) è un iperpiano di separazione.
- I "punti" (esempi) più importanti sono i vettori di support ("sostengono" l'iperpiano, mantenedolo in equilibrio)
- Algoritmi di ottimizzazione quadratica identificano quali punti rappresentano il "supporto".
- Nella formulazione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

Trova $\alpha_1 \ldots \alpha_N$ t.c.
$\mathbf{Q}(\alpha) = \sum \alpha_i - \mathbf{1/2} \sum\sum \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}$ é massimizzata e
(1) $\sum \alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ per ogni α_i

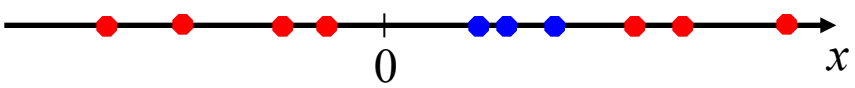
$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b$$

Non-linear SVMs

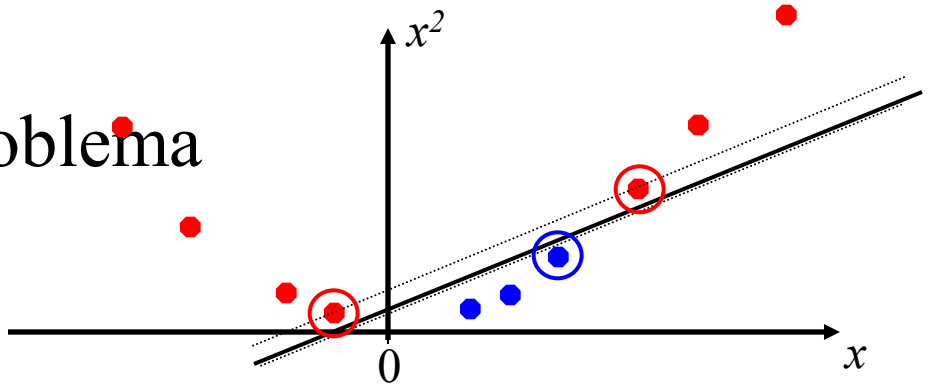
- Se i dataset sono separabili le cose funzionano:



- Altrimenti?

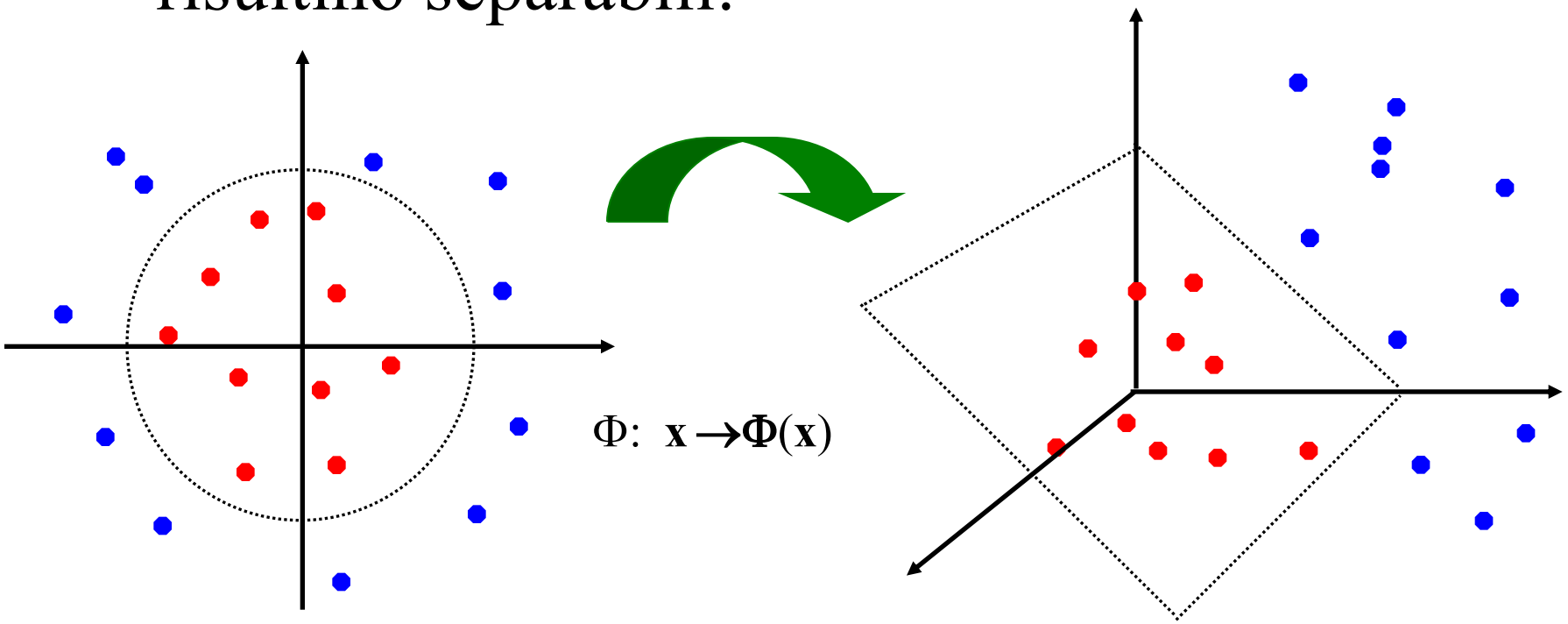


- Si può proiettare il problema in uno spazio di dimensioni maggiori:



Non-linear SVMs: Feature spaces

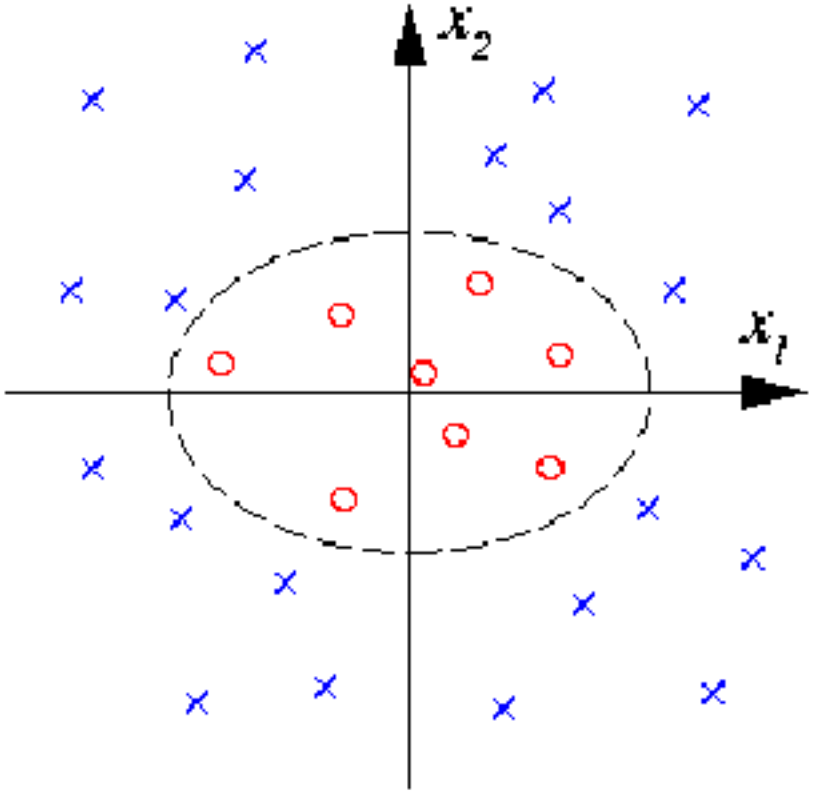
- Proiettare in uno spazio nel quale i dati risultino separabili:

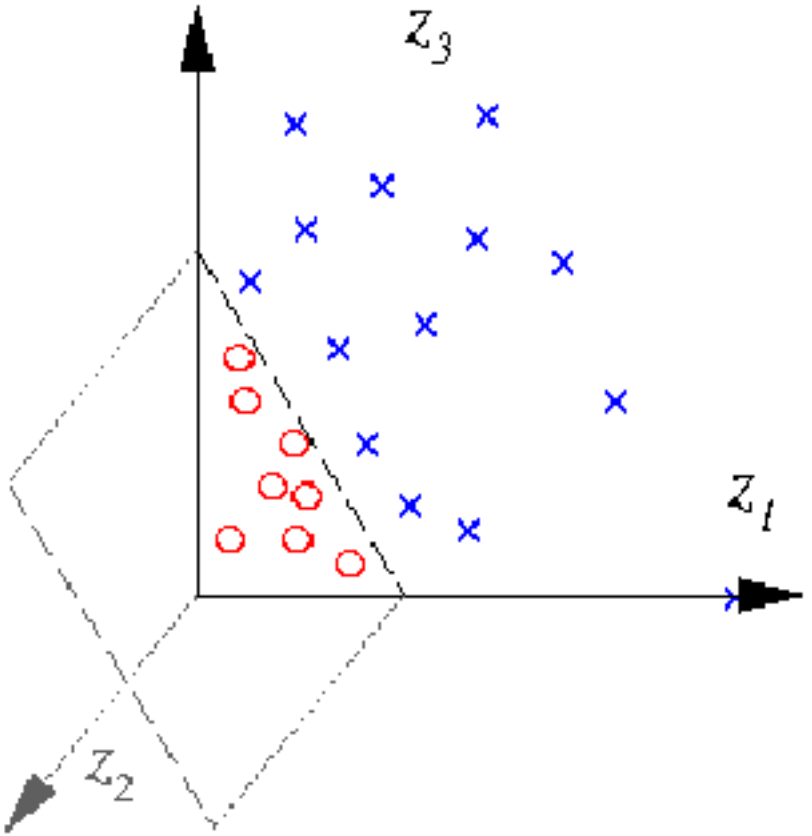


Esempio di funzione Φ

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

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





Funzioni Kernel

- Una funzione *kernel* è una funzione 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 Φ: $\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$ il prodotto scalare diventa:

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

dove x' e y' indicano trasformazioni non lineari

Funzioni kernel

- Esempio: vettori a 2 dim. $\mathbf{x}=[x_1 \; x_2]$;
  Sia $K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j})^2$,
  Dobbiamo mostrare che $K(\mathbf{x_i},\mathbf{x_j})= \mathbf{\Phi}(\mathbf{x_i})^{\mathbf{T}} \mathbf{\Phi}(\mathbf{x_j})$:

  $K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j})^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

  $= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^{\mathbf{T}} [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}] = \mathbf{\Phi}(\mathbf{x_i})^{\mathbf{T}}\mathbf{\Phi}(\mathbf{x_j})$, dove

  $$\mathbf{\Phi}(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$$

Quali funzioni sono Kernels?

- Per alcune funzioni $K(\mathbf{x}_i,\mathbf{x}_j)$ verificare che $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{\Phi}(\mathbf{x}_i)^{\mathrm{T}}\mathbf{\Phi}(\mathbf{x}_j)$ è complesso.
- Teorema di Mercer:

Ogni funzione semi-positiva definita simmetrica è un kernel

Examples of Kernel Functions

- Lineare: $K(\mathbf{x_i},\mathbf{x_j})= \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j}$
- Polinomiale potenza di p:

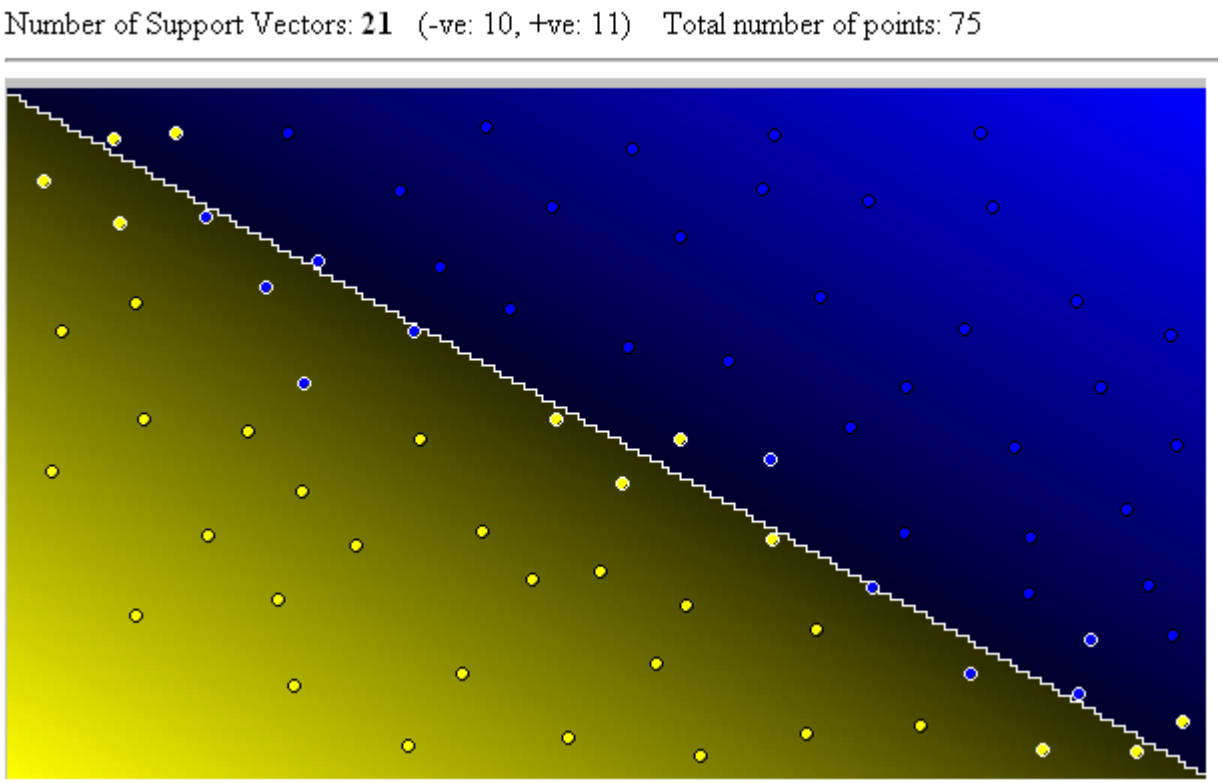
  $K(\mathbf{x_i},\mathbf{x_j})= (1+ \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j})^p$
- Gaussiana (*radial-basis function network*):

  $K(\mathbf{x_i},\mathbf{x_j})= e^{-\frac{\|\mathbf{x_i}-\mathbf{x_j}\|^2}{2\sigma^2}}$
- Percettrone a due stadi:
- $K(\mathbf{x_i},\mathbf{x_j})= \tanh(\beta_0 \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j} + \beta_1)$

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)

Classification examples



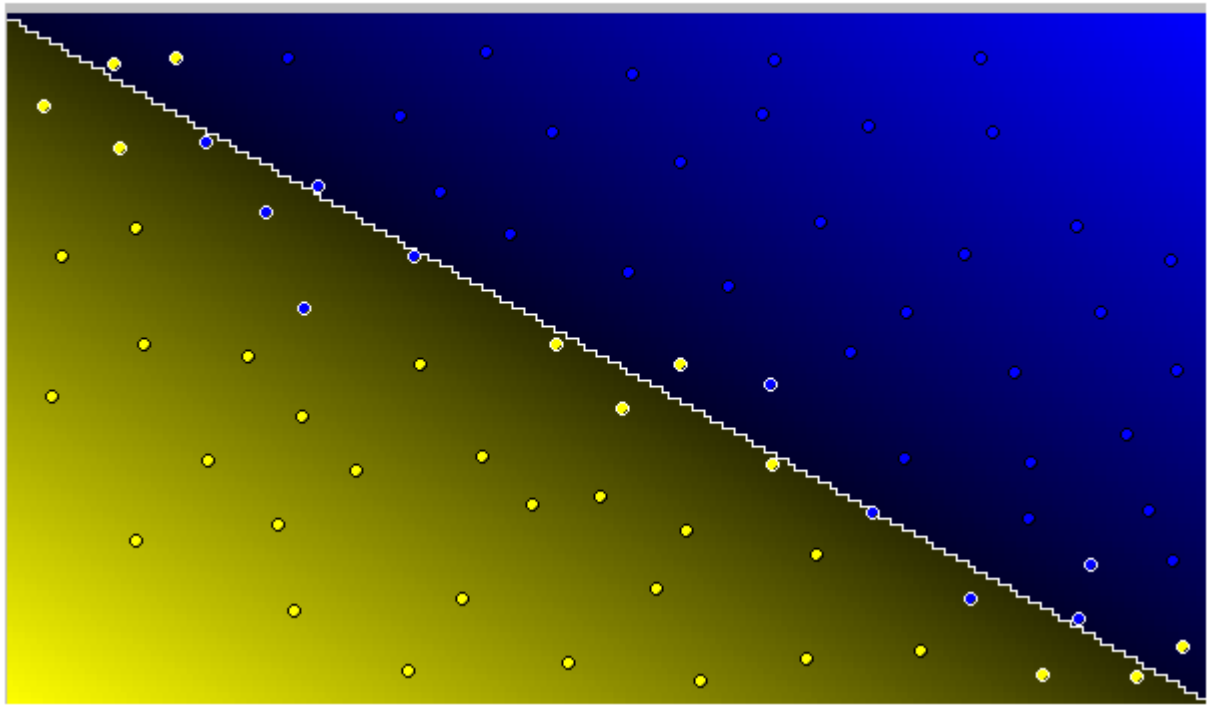
Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples



Linear kernel

Polynomial, deg 1

Polynomial, deg 2

Polynomial, deg 3

Polynomial, deg 4

Polynomial deg 3, cost 100

Polynomial deg 3, cost 1000

Polynomial deg 3, cost 10000

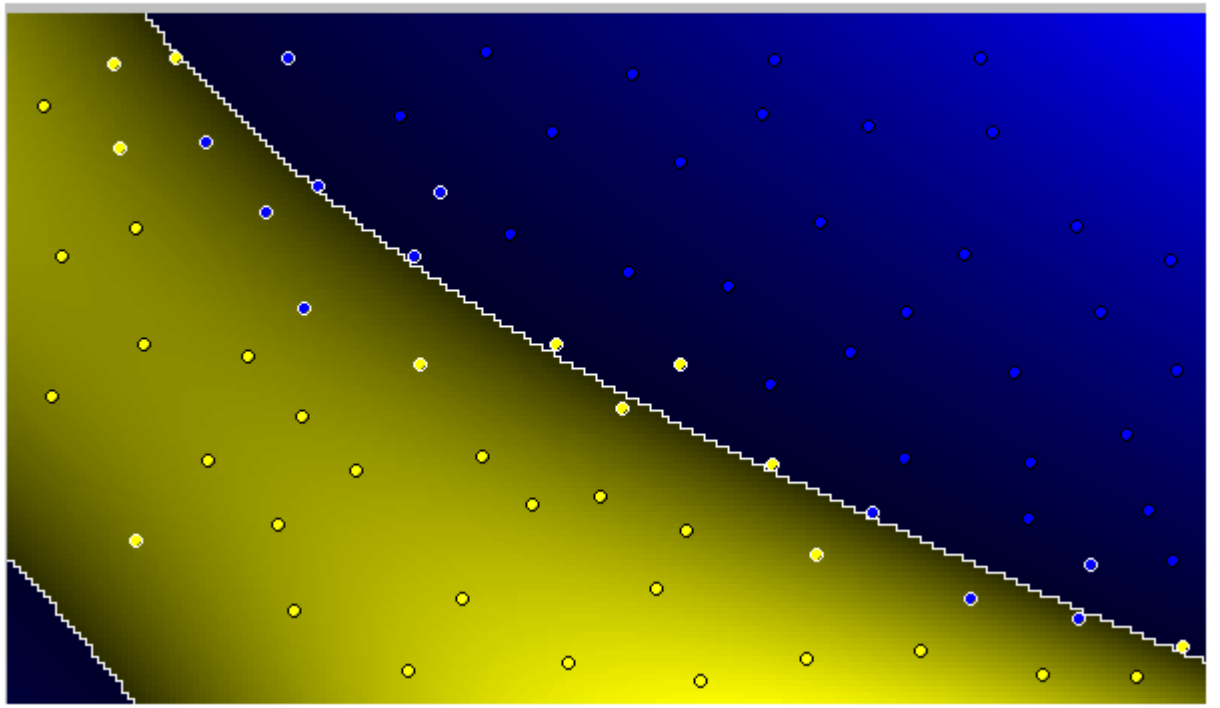
Radial basis function, sigma 0.5

Radial basis function, sigma 1

Radial basis function, sigma 2

Radial basis function, sigma 7

Classification examples



Linear kernel

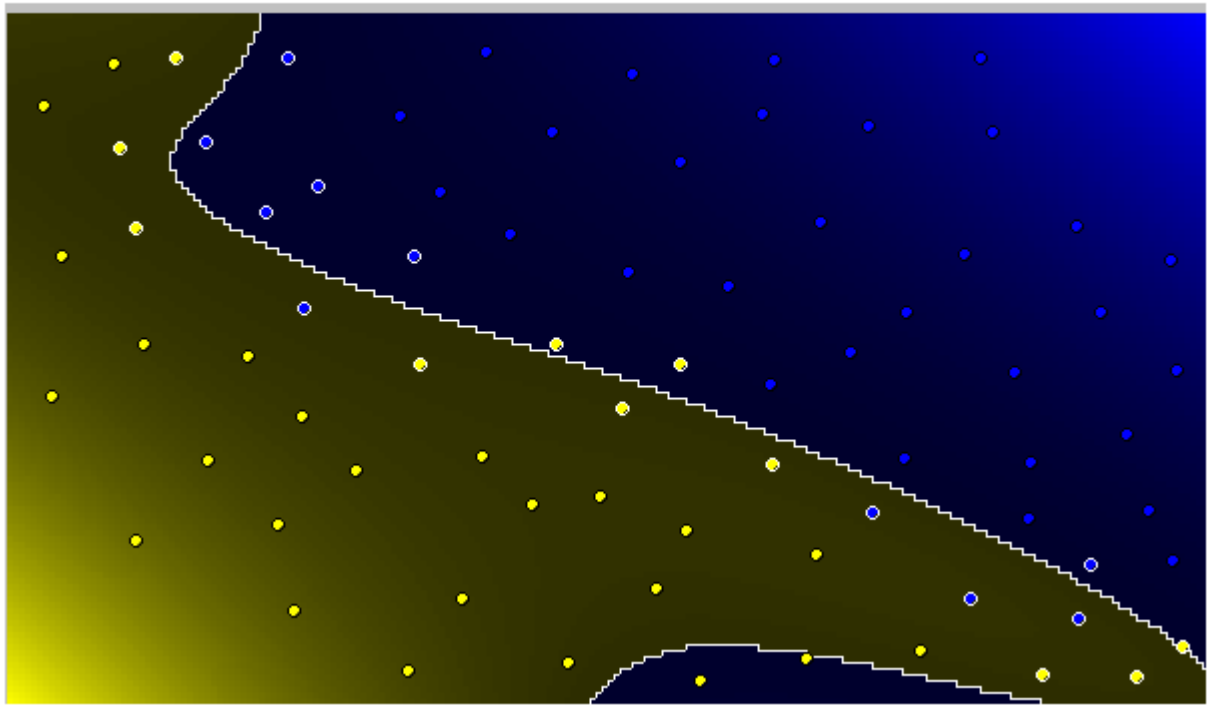
Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples

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



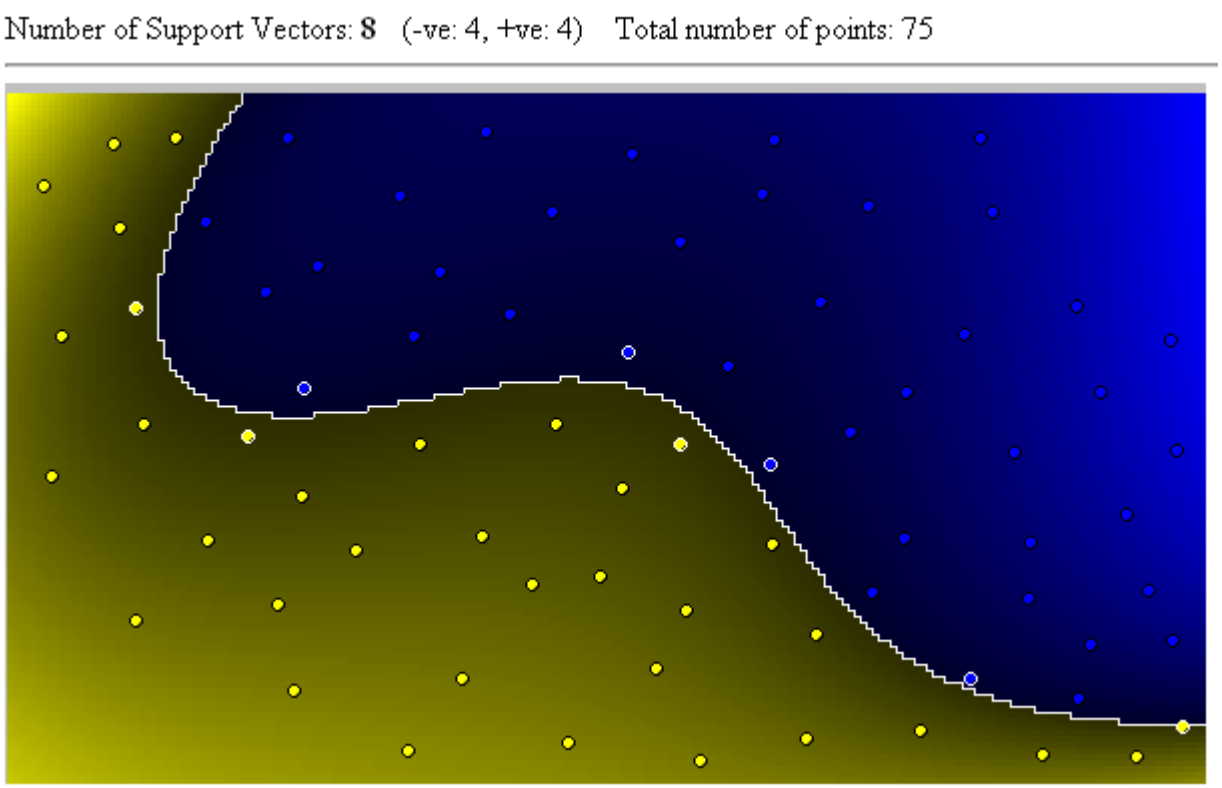
Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples



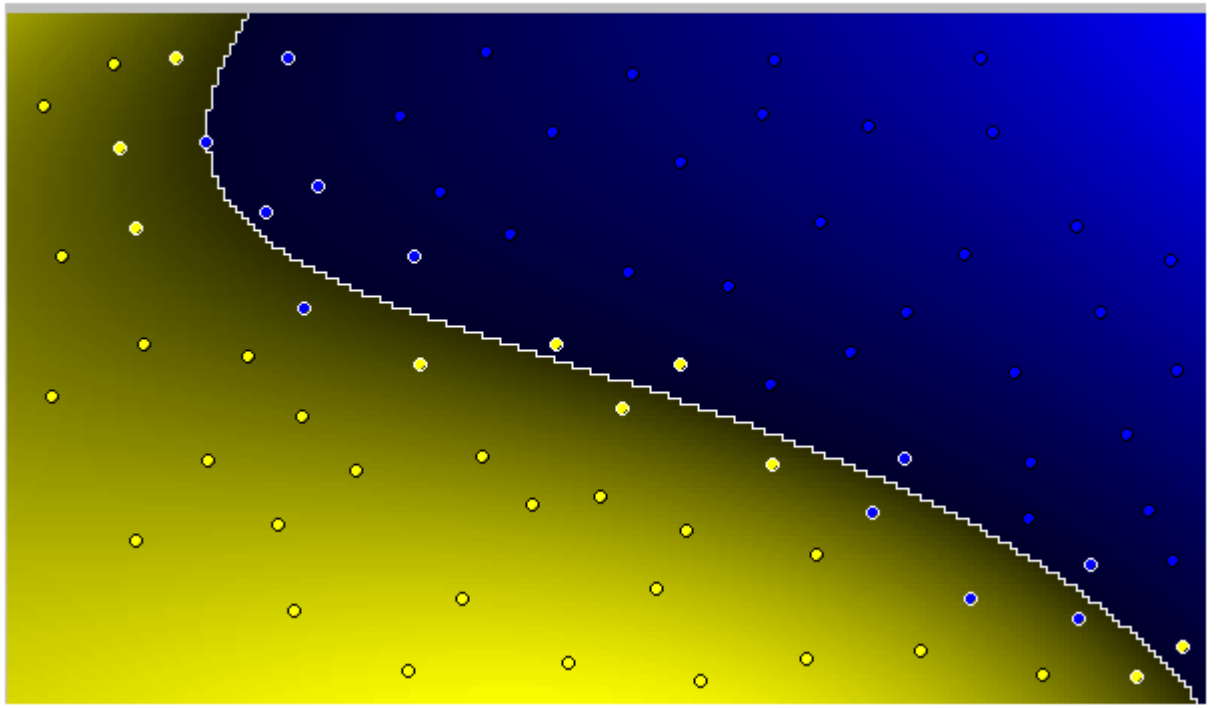
Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples



Linear kernel

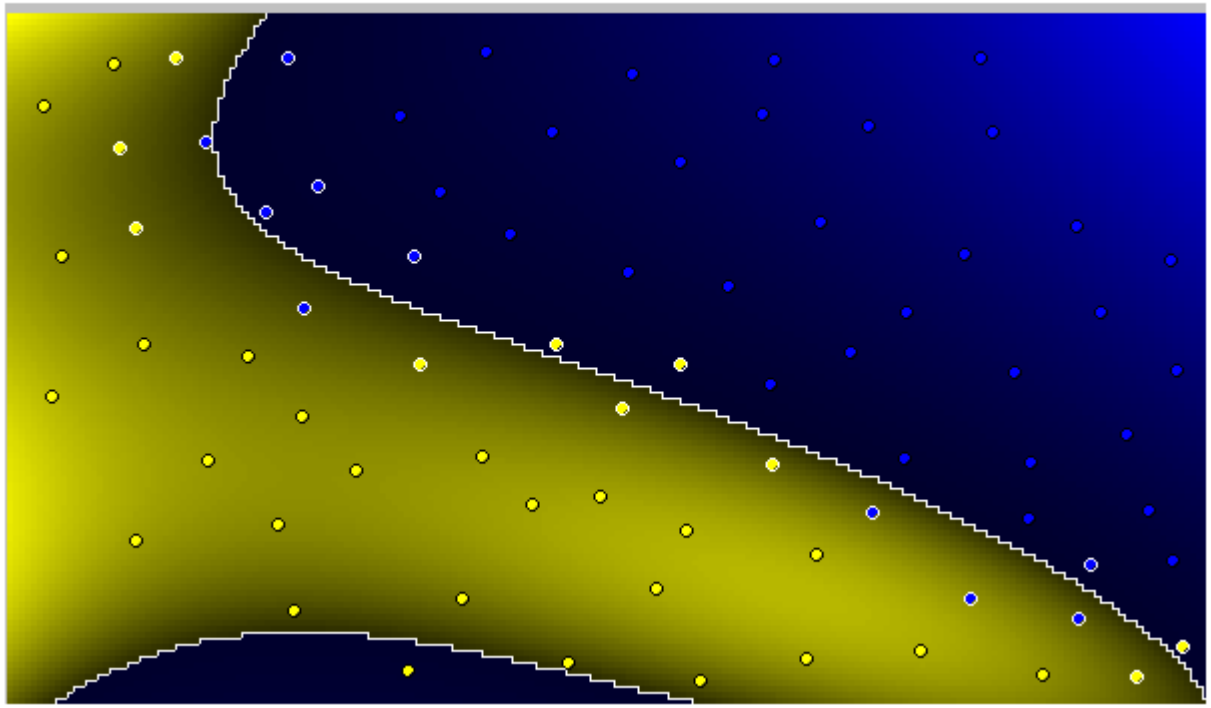
Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples

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



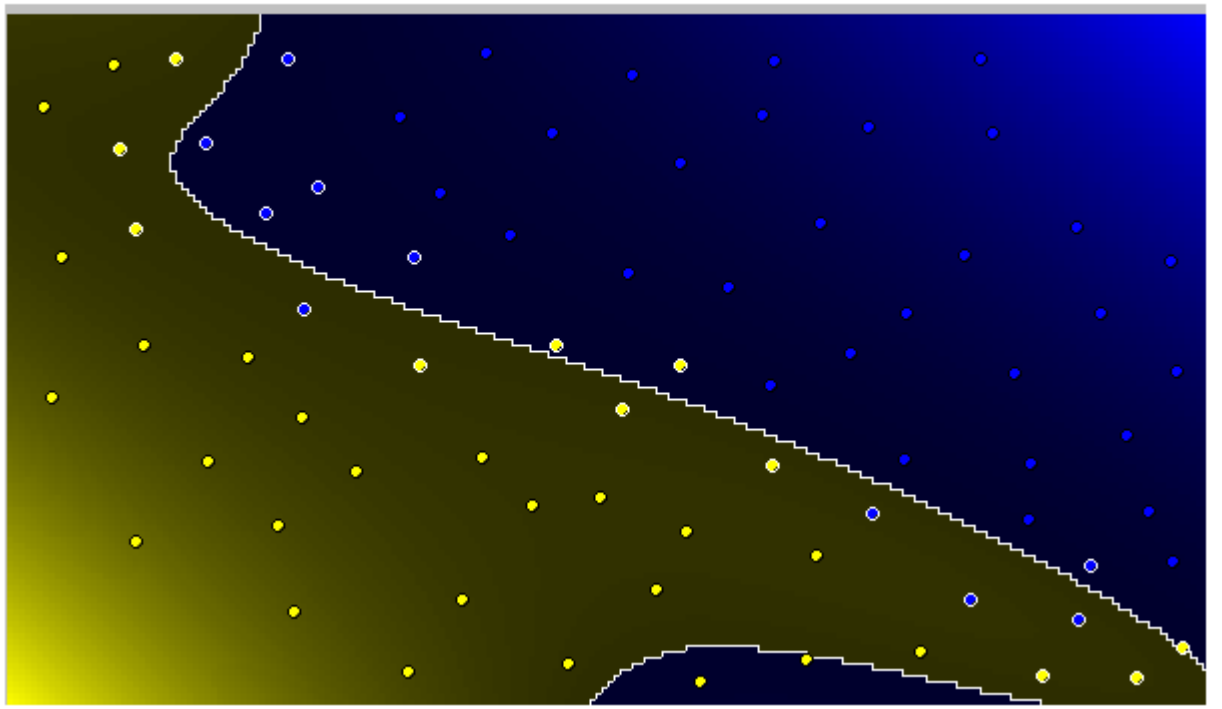
Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples



Linear kernel

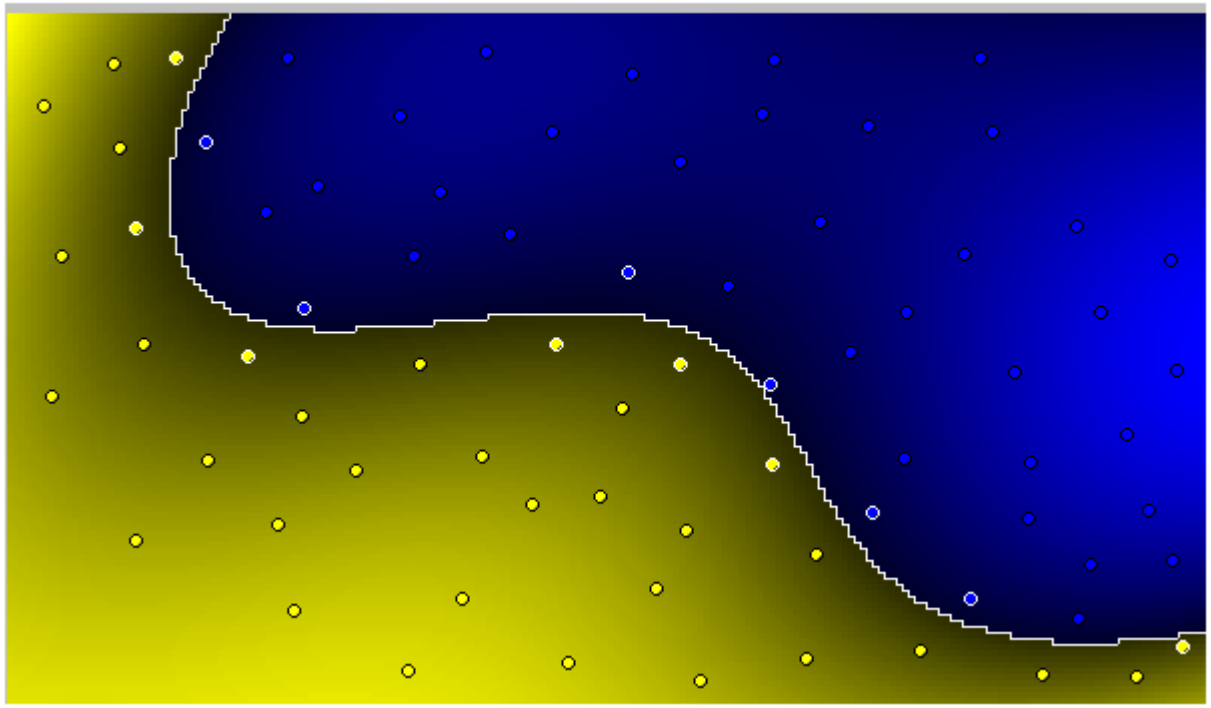
Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples

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



Linear kernel

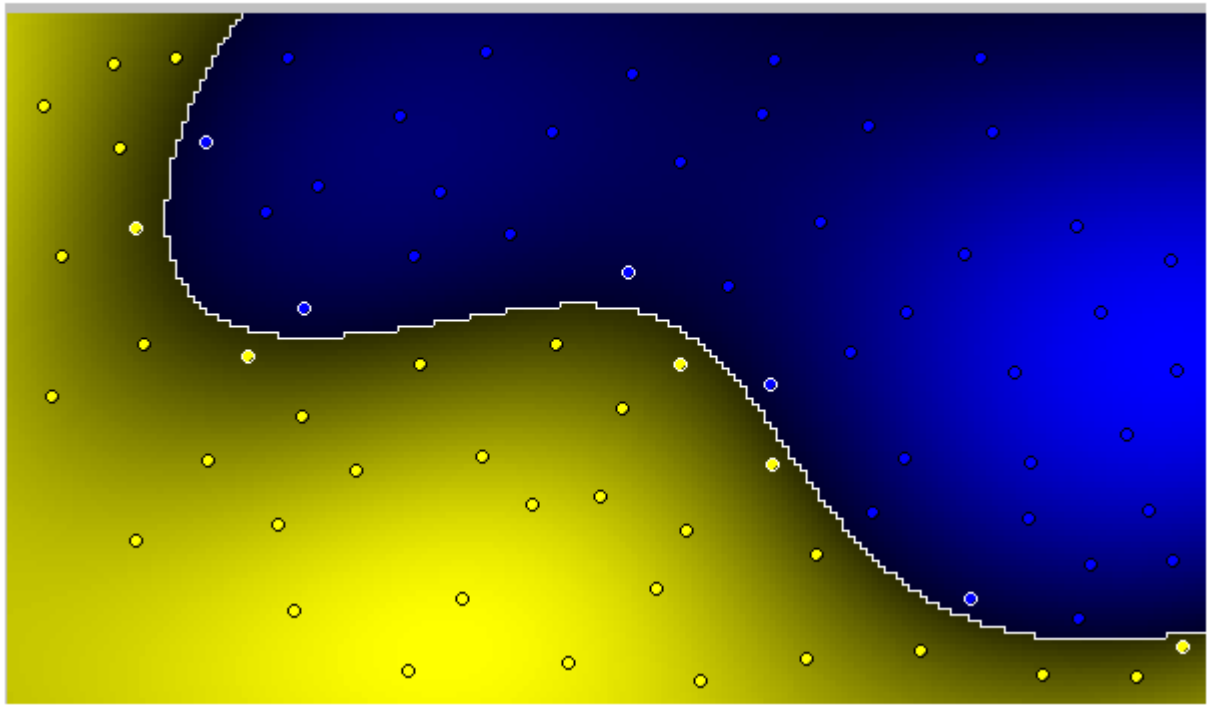
Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples

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



Linear kernel

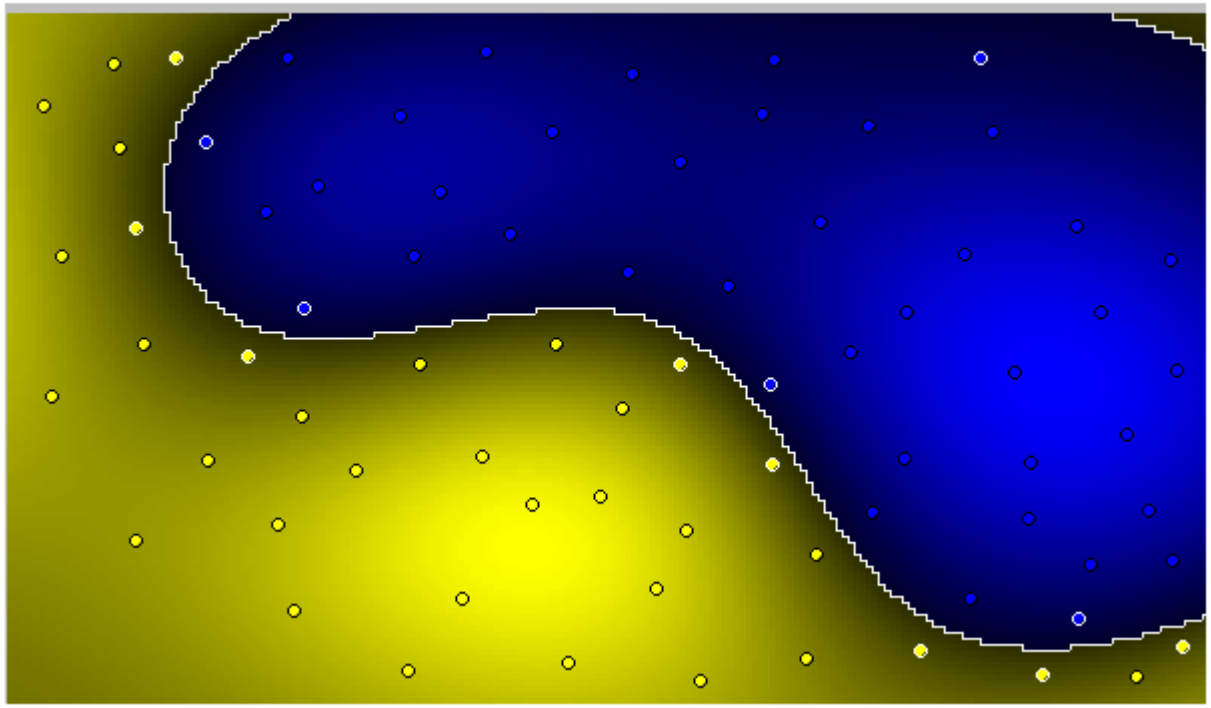
Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples

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



Linear kernel

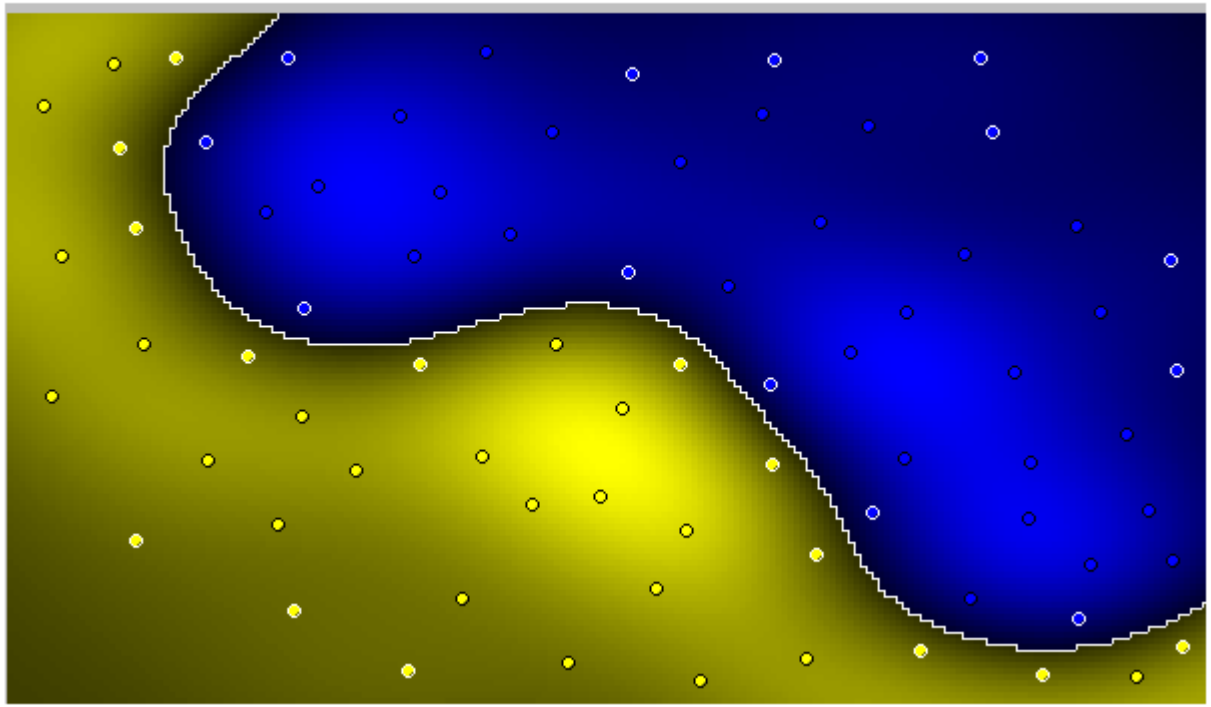
Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

Classification examples

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



Linear kernel

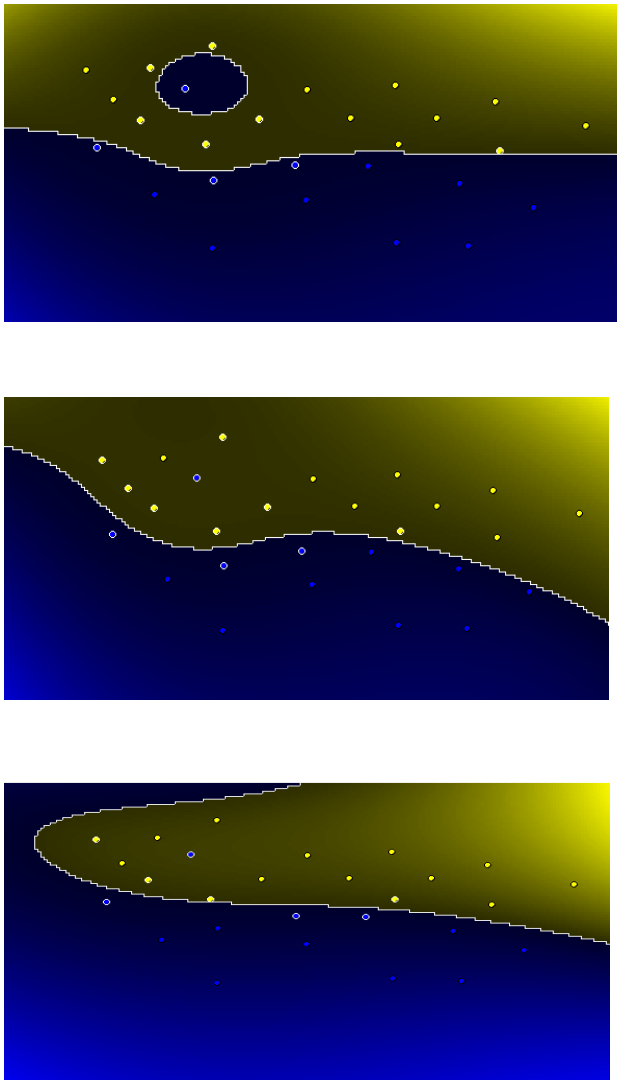
Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

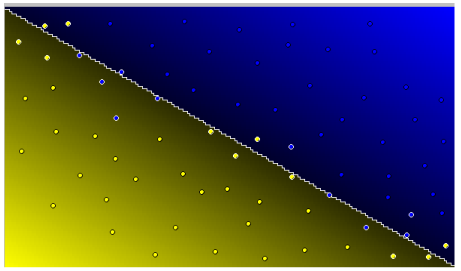
Comments

- Kernel selection and parameter tuning are critical
- Cost C has a huge impact on the generalization ability
- Lowering degree or sigma can avoid overfitting
- Number of support vectors is a measure of generalization performance



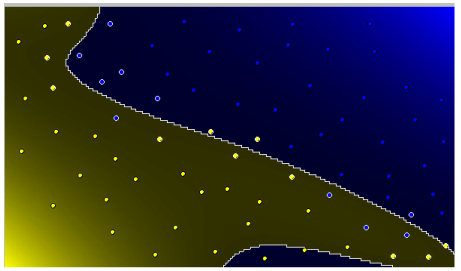
Kernel selection

Linear kernel



- 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

Polynomial

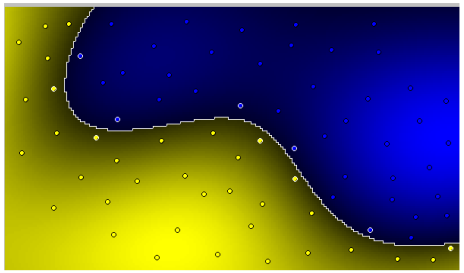


- Has numerical difficulties approaching 0 or infinity
- A good choice for well known and well conditioned tasks
- One additional parameter (degree p)

Kernel selection

Radial basis functions

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



Sigmoid

- 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

Choosing the right kernel is still an art!

Cookbook approach

1. Conduct simple scaling on the data
2. Consider RBF kernel
3. Use cross-validation to find the best parameter C and σ
4. Use the best C and σ 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

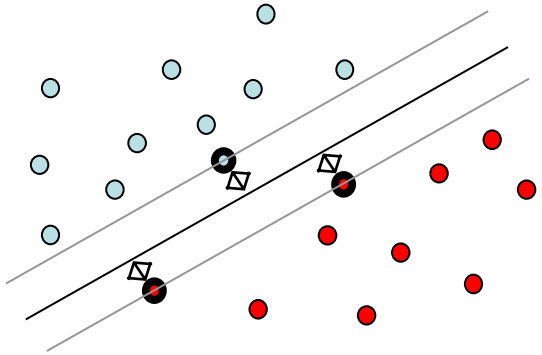
→Solution: compare features on random subsets of the entire dataset to contain computational cost

- Search ranges for C and σ are tricky to choose

→Solution: literature suggests using exponentially growing values like $C = 2^{[-5..15]}$ and $\sigma = 2^{[-15..5]}$

Summary

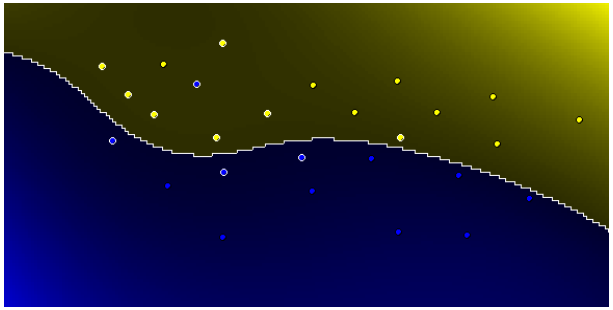
- Support Vector Machines, two key ideas
  - Margin maximization
  - Kernel trick



- Training
  - a quadratic optimization problem
  - always convergent to the optimal solution
  - solvable in polynomial time

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

- Free parameters
  - The cost C
  - The kernel type
  - The kernel parameters



Advantages

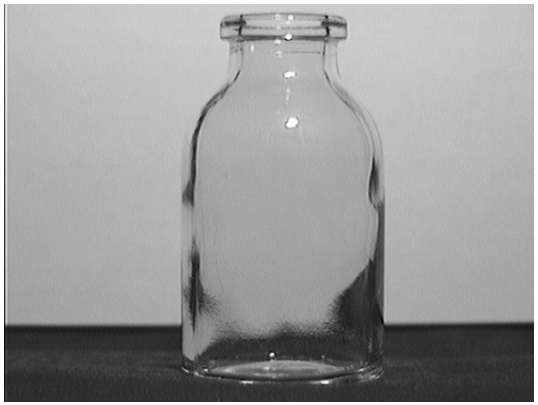
- The SVM theory is an elegant and highly principled
- SVMs have a simple geometric interpretation
- The use of kernels provides a efficient solution to non-linear classification and dispels the ”curse of dimensionality”
- Convergence to the solution is guaranteed
- Support vectors give a compact representation of the entire dataset; their number is a measure of the generalization performance

Disadvantages

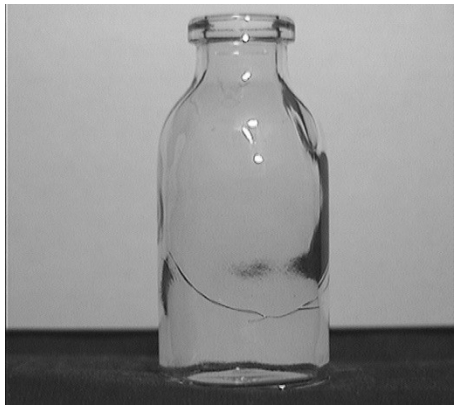
- Kernel and parameter choice is crucial
- Training can sometimes be tricky and time consuming
- Training is not incremental; the whole dataset must be processed for every new addition
- There are no optimized extension to multi-class problems: a problem with N classes requires N classifiers

A concrete example:
Vial Defects Classification

Identify mint bottles from
those damaged or defected



good



crack

bubble

Challenges

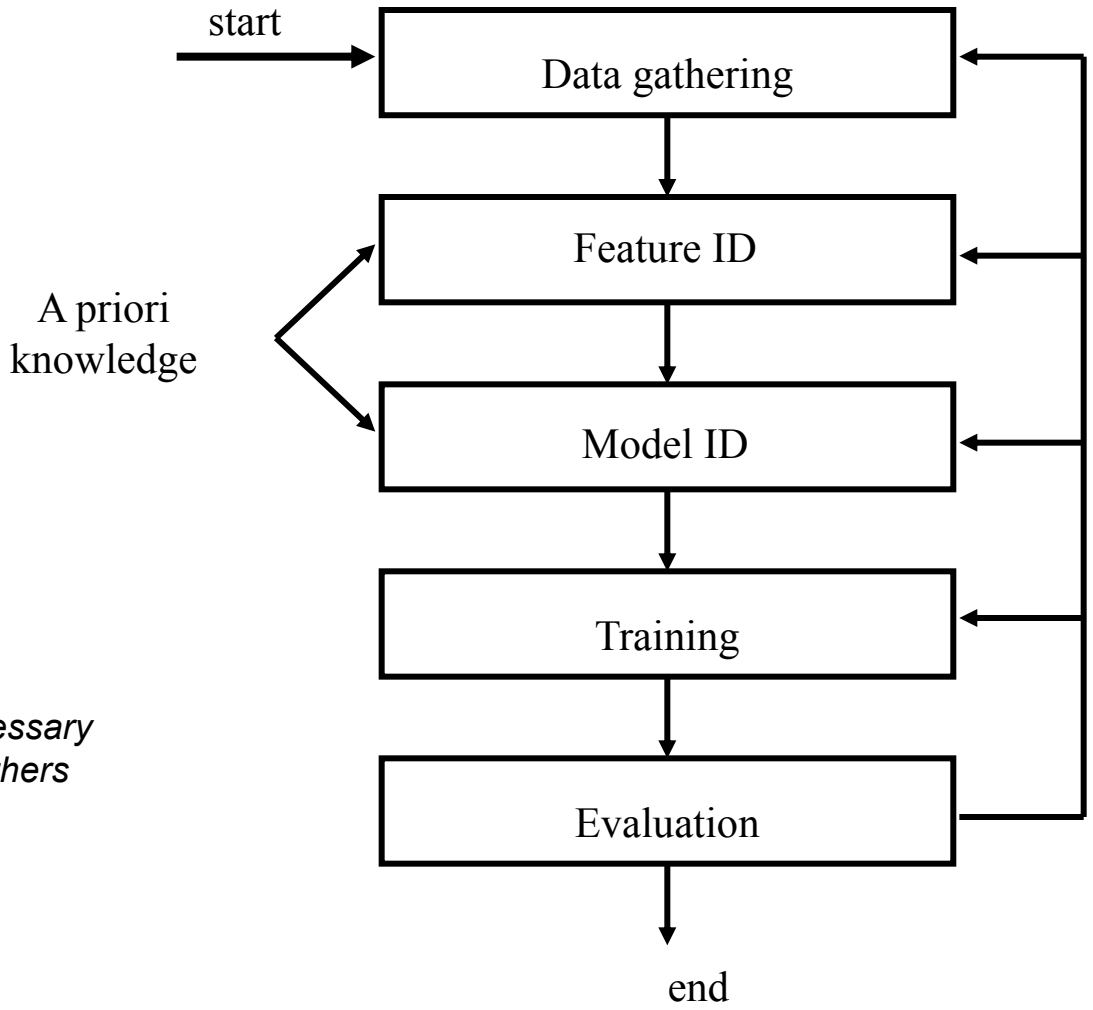
- Molded glass do not consistently deflect light
- Various types of defects must be considered
- Defect can be subtle
- Sterilized environment imposes additional constraints on sensors and illumination placement (vials are for antibiotics)

Approach

- Construction of a specific classifier for each type of defect:
  - Cracks
  - Bubbles
  - Splinters
  - others…

- Different views to be considered
  - Front
  - Shoulder
  - Bottom
  - Top

The usual classification pipeline



Each phase is not necessary separated from the others

Data gathering

- No samples of defects were readily available
- Defected samples were relatively rare
- Collection and acquisition of samples were a slow, manual, error-prone, labor-intensive task
- The contractor had no internal policy on vial rejection (a 0.2 mm bubble is a defect?)

→ All those factors made the acquisition phase the longer and most painful of all.

Do not underrate it!

Data gathering phases

Phase	Samples per class	Defect classes	Views for vial	Acquisition method	Duration (weeks)
1	~30	3	1	Manual	2
2	~100	4	1	Semi-automatic	2
3	~400	4	~24	Automatic	4

How much data is “sufficient” and “representative”?

Feature selection

- The following features has been selected for this first batch of experiments:
  - raw image
  - gradient image
  - gabor transform
  - wavelet transform
  - Fourier transform
  - Harris operator
  - sift descriptors
  - spin images
  - histograms
  - laplacian and LoG
  - morphological operators

- Initial selection guided from domain knowledge
- Secondary selection based on early tests

The model ID and modus operandi

- Support Vector Machines
- Construction of a specific classifier for each type of defects to recognize
- Validation with a *leave-k-out* tests.

Results from phase 1 (June ’05)

- Using ~40 manually acquired samples per class
- Defect centered in the vial and generally well visible
- Using normalized raw gray levels as features

bubble

hit

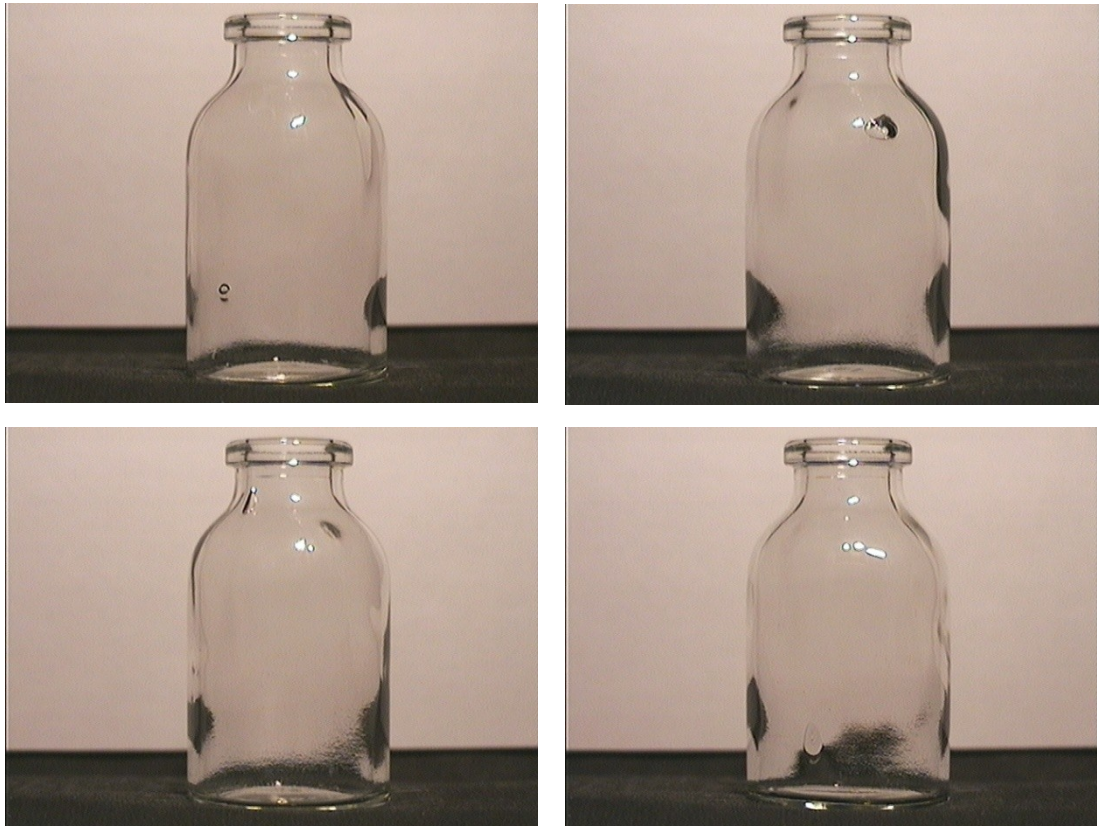
(big) hit

crease

Results from phase 1: bubbles

- 60 good samples
- 30 bad samples

- Leave-1-out: 84.44 %

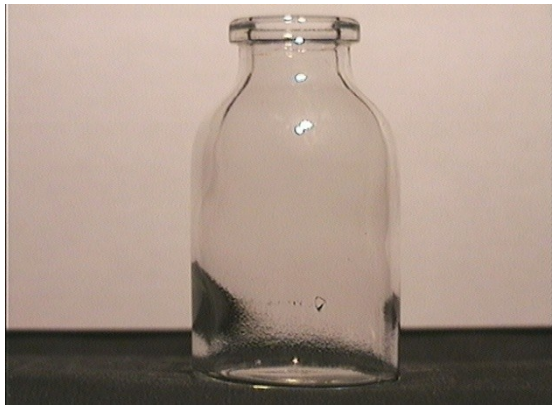


Results from phase 1: hits

- 60 good samples
- 70 bad samples

- Leave-1-out: 96.92 %



Results from phase 1: hits (bigger)

- 60 good samples
- 40 bad samples

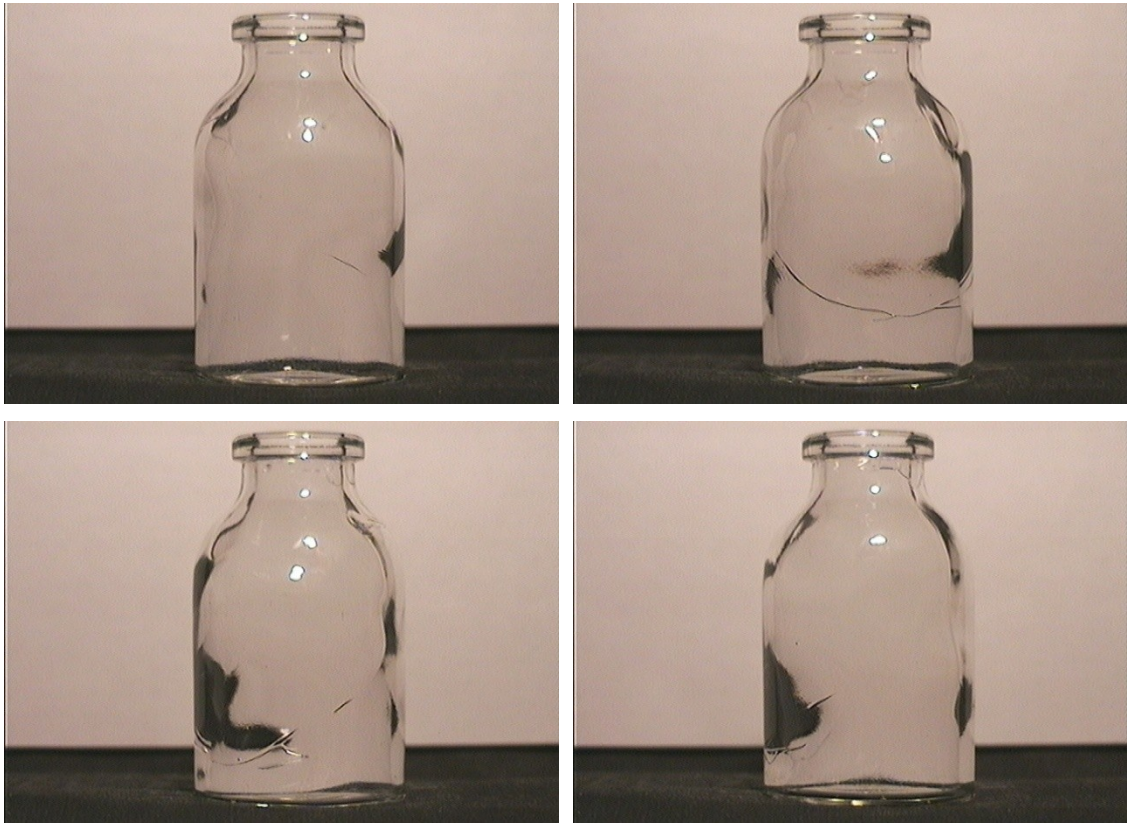
- Leave-1-out:
  89.00 %




Results from phase 1: creases

- 60 good samples
- 65 bad samples

- Leave-1-out:
  96.00 %



Results from phase 1

- Using ~40 manually acquired samples per class for training
  - Defect centered in the vial and generally well visible
    - Using normalized raw gray levels as features

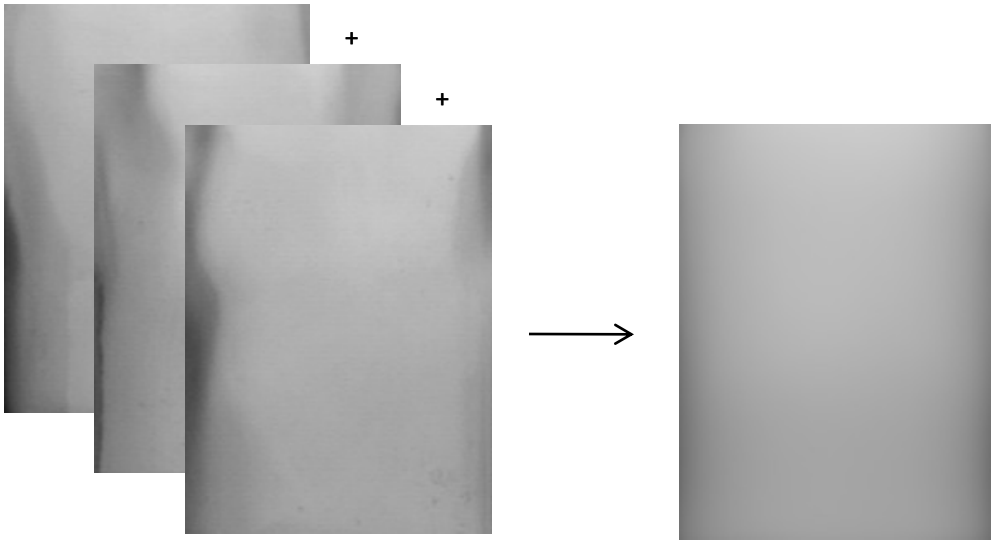
Bubbles	Hits	Big hits	Creases
84.44%	96.92%	89.00%	96.00%



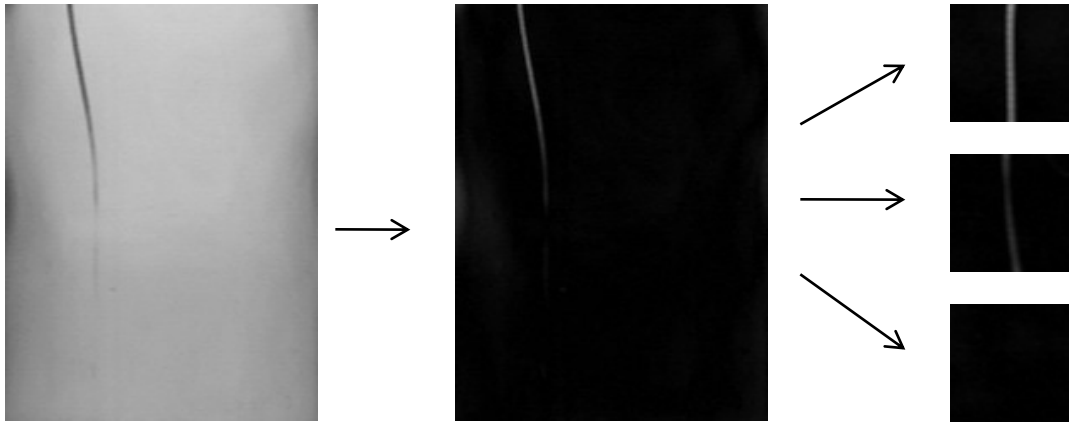

Results from phase 2

- Focus on the two most critical defects
  - Cracks
  - Bubbles
- ~100 samples for training, acquired semi-automatically
- Better illumination
- Now working on sub-windows

Feature extraction



Background modeling



Background subtraction

Sub-windows extraction

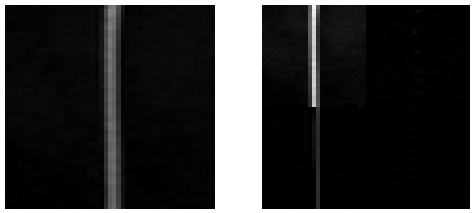
(manual task)

Results from phase 2: cracks

Gray level	98.54
Sobel	96.70
Harris	94.64
Log	97.69
Wavelet: Haar	98.76
Wavelet: Db2	98.22
Fourier	98.58
Dct	97.69

Cracks: best case analysis

- 98.76% with the Haar wavelet
- Matching 98.76



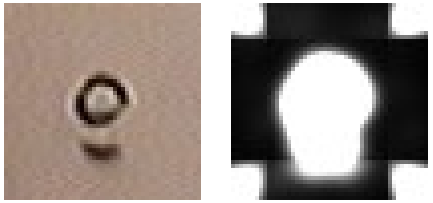
- Positives 232 (41.21) *correctly classified non-defects*
- Negatives 324 (57.55) *correctly classified defects*
- False positives 4 (0.71) *misclassified defects*
- False negatives 3 (0.53) *misclassified non-defects*

Results from phase 2: bubbles

Gray level	94.56
Sobel	96.60
Harris	98.64
Log	90.48
Wavelet: Haar	95.24
Wavelet: Db2	94.92
Fourier	83.67
Dct	80.27

Bubbles: best case analysis

- 98.64% with the Harris operator
- Matching 98.64



- Positives 102 (69.39) *correctly classified non-defects*
- Negatives 42 (28.57) *correctly classified defects*
- False positives 1 (0.68) *misclassified defects*
- False negatives 2 (1.36) *misclassified non-defects*

Results from phase 2 (Sept. ’05)

- ~100 samples
- Sub-win
- Background s

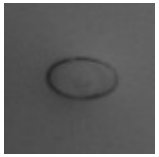
Cracks	Bubbles
98.76%	98.64%
Haar	Harris

Results from phase 3 (Feb. ’06)

- ~400 samples per class for training, acquired automatically
- Features used
  - Histograms (ldg, gradient)
  - Morphological operators
- Working on sub-windows
- Additional (shoulder) view

Big bubbles: best case analysis

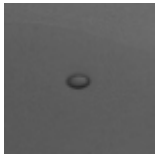
- 95.76% with morphological operator on sub-windows
- Matching 95.76%



• Positives	253 (48.75)	*correctly classified non-defects*
• Negatives	244 (47.01)	*correctly classified defects*
• False positives	17 (3.28)	*misclassified defects*
• False negatives	5 (0.96)	*misclassified non-defects*

Small bubbles: best case analysis

- 91.07% with morphological operator on sub-windows
- Matching 91.07%



- Positives 215 (47.99) *correctly classified non-defects*
- Negatives 193 (43.08) *correctly classified defects*
- False positives 9 (2.01) *misclassified defects*
- False negatives 31 (6.92) *misclassified non-defects*

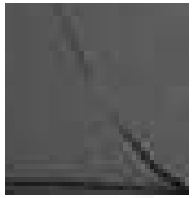
Cracks, front: best case analysis

- 91.91% with the gradient histogram
- Matching 91.91%

• Positives	286 (52.57)	*correctly classified non-defects*
• Negatives	214 (39.34)	*correctly classified defects*
• False positives	14 (2.57)	*misclassified defects*
• False negatives	30 (5.51)	*misclassified non-defects*

Cracks, shoulder: best case analysis

- 99.75% with morphological operator on sub-windows
- Matching 99.75%



- Positives 197 (49.75) *correctly classified non-defects*
- Negatives 198 (50.00) *correctly classified defects*
- False positives 1 (0.25) *misclassified defects*
- False negatives 0 (0.00) *misclassified non-defects*

Results from phase 3

- ~400 samples per class for training, acquired automatically

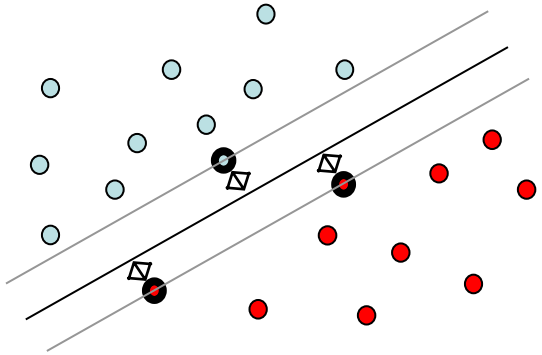
Cracks		Bubbles	
Front	Shoulder	Big	Small
91.91%	99.75%	95.76%	91.07%
GradHist	Hist	Morph	Morph

Comments

- Obtained results were satisfactory and in line with the customer requirements
- The acquisition phase is crucial (garbage in, garbage out)
- Support vector number was not monitored

Summary

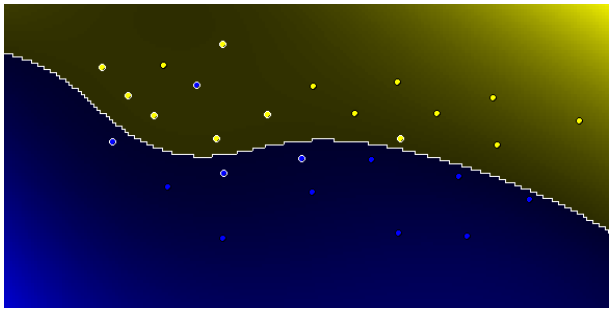
- Support Vector Machines, two key ideas
  - Margin maximization
  - Kernel trick



- Training
  - a quadratic optimization problem
  - always convergent to the optimal solution
  - solvable in polynomial time

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

- Free parameters
  - The cost C
  - The kernel type
  - The kernel parameters



References

An introduction to Support Vector Machines
(and other kernel-based learning methods)
N. Cristianini and J. Shawe-Taylor
Cambridge University Press (2000)

Support Vector and Kernel Machines
Tutorial presented at ICML, Nello Cristianini

A tutorial on support vector machines for pattern recognition
C. Burges, Data Mining and Knowledge Discovery, Vol 2(2), June 1998

http://www.kernel-machines.org/
http://www.support-vector.net/
http://www.google.com/