Metodi Non-Parametrici basati su Kernel

Corso di Apprendimento Automatico
Laurea Magistrale in Informatica

Nicola Fanizzi

Dipartimento di Informatica
Università degli Studi di Bari

24 novembre 2009

Obiettivi

- introdurre i metodi non parametrici
- passare in rassegna le idee principali su cui si basano gli algoritmi di apprendimento mediante **kernel**
- presentare alcuni algoritmi basati su queste idee
- esempi di dati ed applicazioni che si possono trattare
  - stringhe, insiemi, vettori, alberi
  - classificazione
  - testo
- estensione alle rappresentazioni complesse

Metodi Non-Parametrici

limitazione dei metodi parametrici:
la funzione di densità potrebbe non corrispondere a quella da cui i dati sono stati generati ($\longrightarrow$ cattiva performance)

I metodi non-parametrici

- prescindono della distribuzione
  - nessuna assunzione sulla distribuzione dei dati
  - non-parametricità non significa assenza di parametri bensì che il numero e la natura dei parametri non è prefissato ed è flessibile
- servono a studiare popolazioni i cui elementi possono essere legati da misure di similarità ovvero da relazioni non necessariamente numeriche
  - come le relazioni d'ordine
    (es. le stelle assegnate ad un film)
- $-$ assunzioni = $+$ semplicità, $+$ versatilità, $+$ robustezza

Istogrammi I

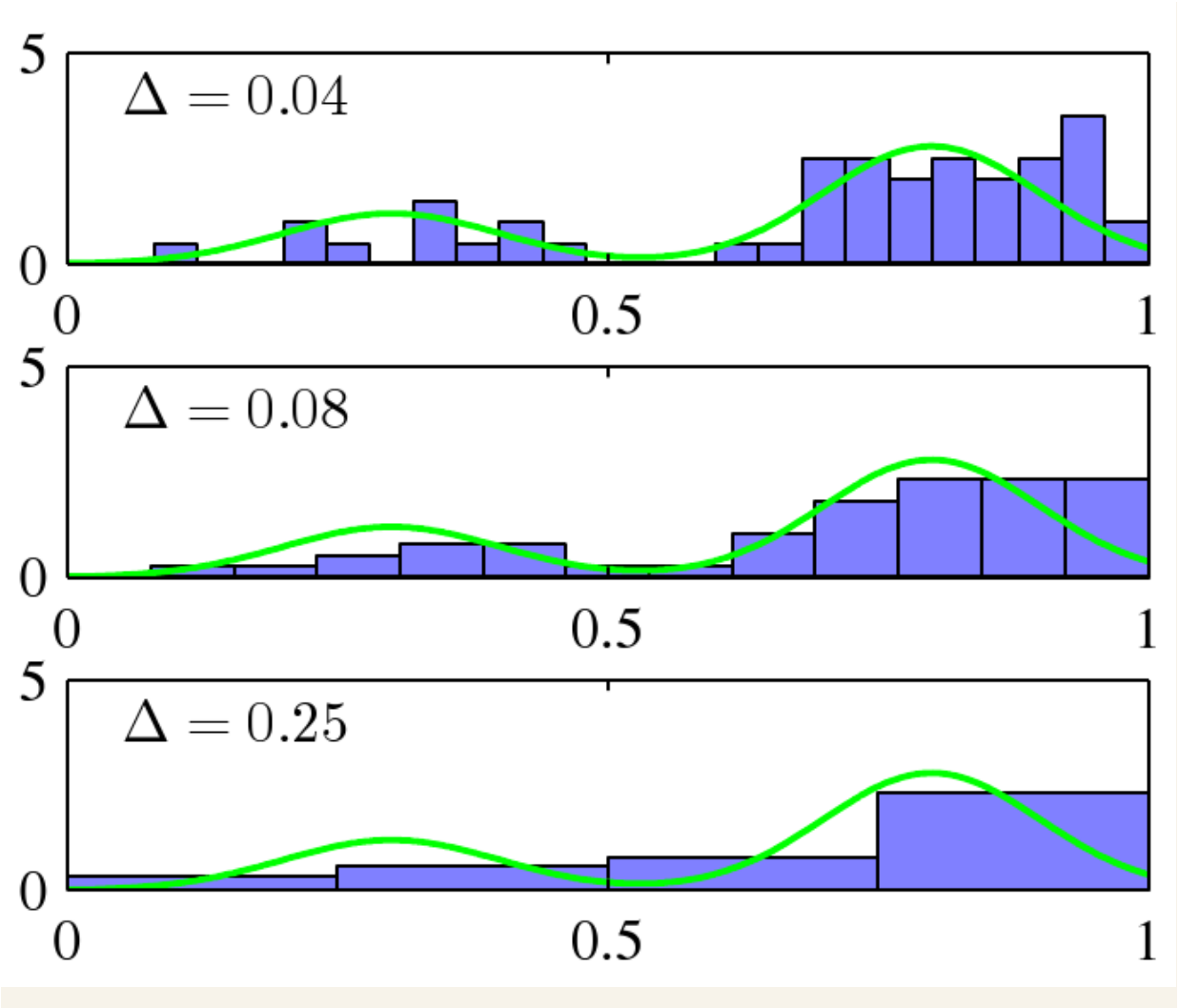
nei metodi basati su istogrammi:

- si partizionano uno o più assi in intervalli (*bins*) di dimensione Δ_i
- si conta il numero n_i di osservazioni x che ricadono nell'intervallo i-esimo.
- si normalizza dividendo per il numero di osservazioni n e per l'ampiezza dell'intervallo Δ_i in modo da ottenere i valori della probabilità per ogni intervallo

$$p_i = \frac{n_i}{n\Delta_i}$$

- In genere si scelgono i intervalli di pari ampiezza $\Delta_i = \Delta$

Istogrammi II



Variazione della stima al variare dell'ampiezza dell'intervallo (in verde la funzione di densità teorica)

Istogrammi III

Osservazioni

- una volta calcolato l'istogramma si può tralasciare il dataset
- approccio incrementale applicabile in caso di dati in sequenza nel tempo
- utile ad avere un'idea della distribuzione in caso di 2 o 3 dimensioni (non in generale)
- discontinuità della densità stimata
- non scala con l'incremento del numero delle dimensioni
  - dividendo ogni variabile in uno spazio D-dimensionale in M bin, totale M^D bin (*curse of dimensionality*)
  - in uno spazio a moltissime dimensioni, la mole di dati necessaria per una stima significativa sarebbe proibitiva

Località I

Proprietà di località rispetto ad una nozione di distanza: per stimare la densità in un punto si considerano i punti situati nelle vicinanze

Nel caso degli istogrammi:
la località dipende dal parametro di regolarizzazione (*smoothing*) naturale rappresentato dall'*ampiezza* dell'intervallo Δ: minore ampiezza = maggiore regolarità (minore scalettatura)

Obiettivo: altri metodi per scalare meglio rispetto alle dimensioni

Località II

Sia $f(x_0)$ il valore della funzione di densità da stimare (spazio euclideo D-dimensionale)

Per la località, consideriamo una piccola regione $\mathcal{R}$ attorno a x_0
La massa di probabilità associata alla regione è data da

$$P = \int_{\mathcal{R}} f(x)dx$$

Consideriamo un dataset di n osservazioni tratte da $f(x)$. Ogni punto ha probabilità P di ricadere in $\mathcal{R}$, quindi il totale dei k punti che ricadono in $\mathcal{R}$ sarà distribuito secondo una binomiale

Per cui la frazione media di punti nella regione è $E[k/n] = P$, con varianza: $var[k/n] = P(1 - P)/n$

Località III

Per grandi valori di n la distribuzione ha un picco attorno alla media per cui:

$$k \simeq nP$$

Se assumiamo che la regione sia sufficientemente piccola $f(x_0)$ è pressochè constante in $\mathcal{R}$, quindi:

$$P \simeq f(x_0)V$$

dove V è il volume di $\mathcal{R}$
Per cui una stima della densità sarà:

$$\hat{f}(x_0) \simeq \frac{k}{nV}$$

Località IV

Osservazione: è difficile avere $\mathcal{R}$ piccola per una densità constante ma anche grande per contenere k punti che determinino un picco della binomiale

$$f(x_0) = \frac{k}{nV}$$

può essere sfruttata in due modi:

1. fissato k si determina V dai dati $\longrightarrow$ k-NN
2. fissato V si determina k dai dati $\longrightarrow$ metodi con kernel

Si può mostrare che i due stimatori convergono asintoticamente ($n \to \infty$) verso la vera densità se V si restringe e k decresce rispetto a n

Stima della densità I

Dato il campione di punti i.i.d. $x_1, x_2, \ldots, x_n$ di una variabile aleatoria, l'approssimazione della densità f della variabile tramite kernel è il seguente:

$$\hat{f}_h(x) = \frac{1}{nh} \sum K\left(\frac{x - x_i}{h}\right)$$

dove K è un kernel ed h è un parametro di *smoothing* detto ampiezza di banda (*bandwidth*)

$$K(x, x_0) = D\left(\frac{x - x_i}{h}\right)$$

Epanichnikov Kernel

dove $D(t) = \begin{cases} \frac{3}{4}(1-t)^2 & \text{se } |t| \leq 1 \\ 0 & \text{altrimenti} \end{cases}$

Stima della densità II

Lo stimatore converge velocemente

- invece di raccogliere le osservazioni in barre come negli istogrammi, lo stimatore colloca piccole ”gobbe” (*bump*) determinate da K, in corrispondenza di ogni osservazione
- lo stimatore somma i bump risultando più *smooth* rispetto, ad es., ad un istogramma
- una maggiore densità di punti comporta valori di più alti della stima

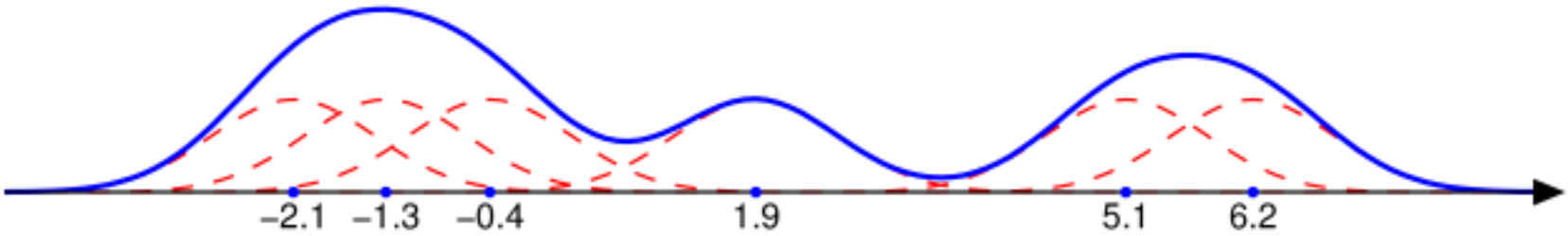


immagine tratta da Wikipedia: somma di sei gaussiane: $K\left(\frac{x-x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}$

Stima della densità III

Scelta di K

in generale:

$$K_{h_\lambda}(x) = D\left(\frac{\|x - x_i\|}{h_\lambda(x)}\right)$$

con:

- $\|\cdot\|$ una norma
- $h_\lambda(x_0)$ parametro (raggio del kernel)
- $D(\cdot)$ funzione positiva il cui valore (reale) decresce (o non cresce) al crescere della distanza tra x e x_0

k-NN Smoothing I

Per ogni x_0, si facrescere una ipersfera attorno fino a contenere i k vicini

Si stima il valore di $f(x_0)$ mediando i valori di tali punti

Formalmente:

$$h_k(x_0) = \left\| x_0 - x_{[k]} \right\|$$

con

$x_{[k]}$ k-esima istanza piú vicina a x_0

$$D(t) = \begin{cases} 1/2 & \text{se } |t| \leq 1 \\ 0 & \text{altrimenti} \end{cases}$$ Parzen window

ipercubo unitario attorno all'origine

Kernel Smoothing I

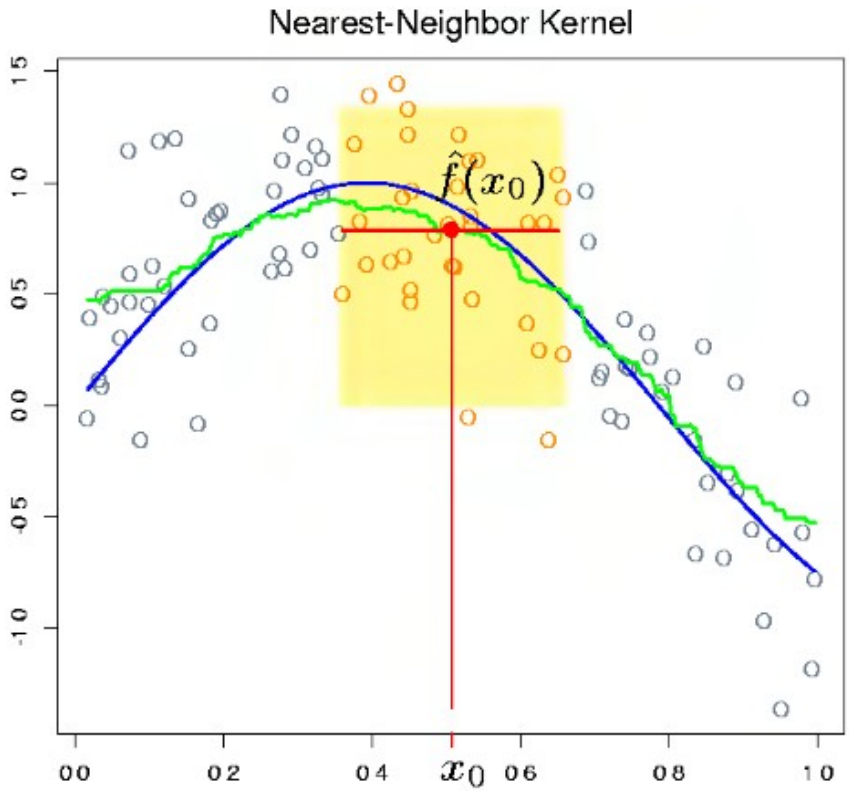
Sia $f(x) : \mathbb{R}^p \rightarrow \mathbb{R}$ una funzione continua
$\forall x_0 \in \mathbb{R}^p$, media pesata di Nadaraya-Watson:

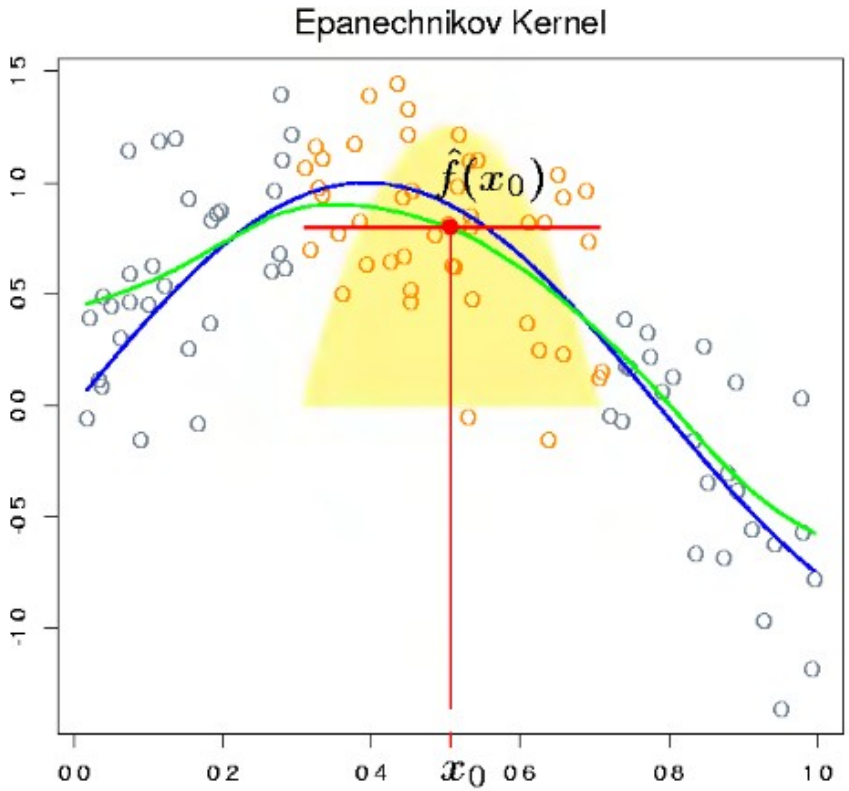
$$\hat{f}(x_0) = \frac{\sum_{i=1}^{n} K_{h_\lambda}(x_0, x_i) y_i}{\sum_{i=1}^{n} K_{h_\lambda}(x_0, x_i)}$$

con:

- n numero di esempi
- y_i etichetta di x_i

Kernel Smoothing II





Apprendimento con Kernel Methods (KM)

- KM: famiglia di algoritmi per l'analisi di pattern
  Tra i più noti: Support Vector Machine
- scopo molto generale
  dato un insieme di dati (in qualunque forma),
  ad es.
  - vettori di attributi
  - sequenze di proteine
  - rappresentazioni di geni
  - documenti testuali ed ipertestuali
  - ...

  trovare pattern (relazioni) di interesse:
  - classificazione, clustering, ranking,
  - correlazione, regressione, componenti principali, etc., ...

  escludendo quelli instabili (= overfitting)

Informalmente...

- La classe dei KM definisce implicitamente la classe dei possibili pattern, introducendo una nozione di similarità tra dati
- Esempi: similarità tra documenti:
  - basata sulla lunghezza
  - basata sull'argomento
  - basata sulla lingua
  - ...
- la scelta del criterio di similarità determina la scelta delle caratteristiche (features) rilevanti

Notazione

Dati

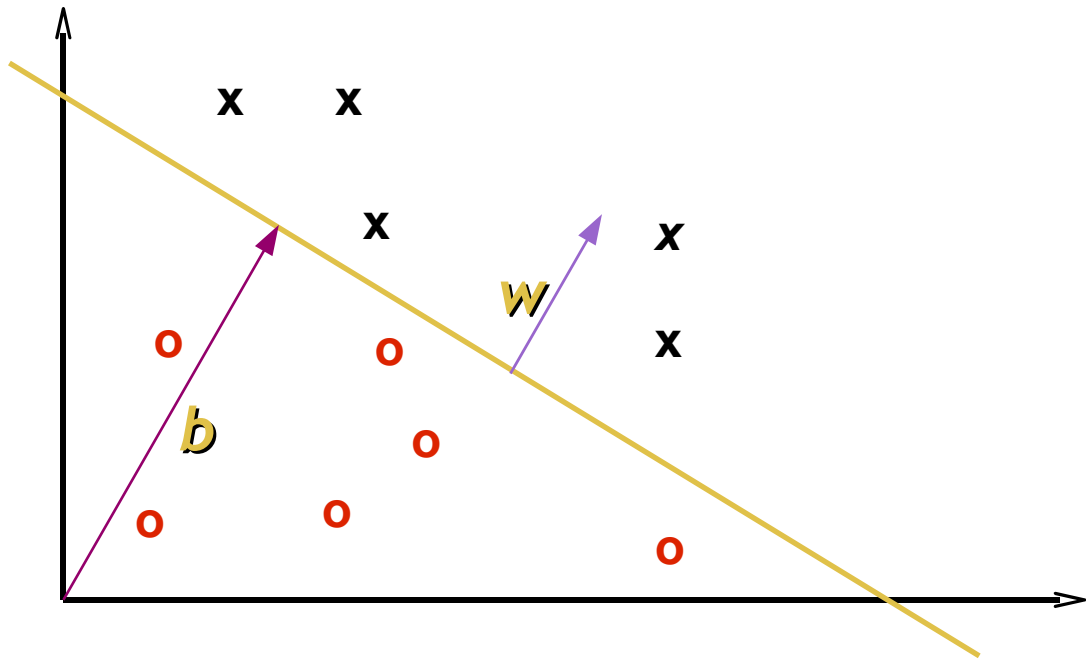
- un insieme X (input set)
- un insieme Y (output set), ad es. $Y = \{-1, +1\}$
- un sottoinsieme finito $S \subseteq (X \times Y)$
  (di solito di distribuzione sconosciuta)
  con elementi $(x_i, y_i) \in S \subseteq (X \times Y)$

Trovare una funzione $y = h(x)$ che sia coerente con i dati
h si adatta (fits) ai dati evitando l'overfitting

h può variare a seconda degli obiettivi scelti

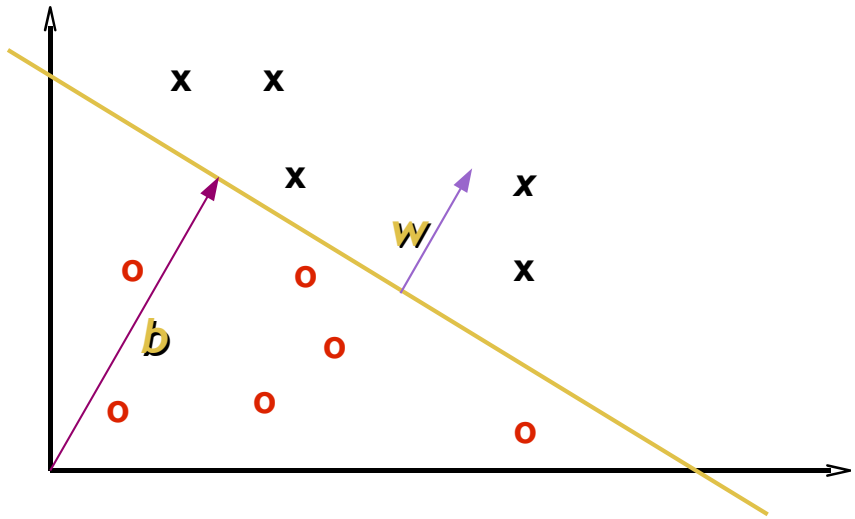
Metodi di Discriminazione Lineare I

- Prodotto interno tra vettori $\langle x, z \rangle = \sum_{i=1}^{n} x_i z_i$
- Iperpiano $\langle x, w \rangle + b = 0$

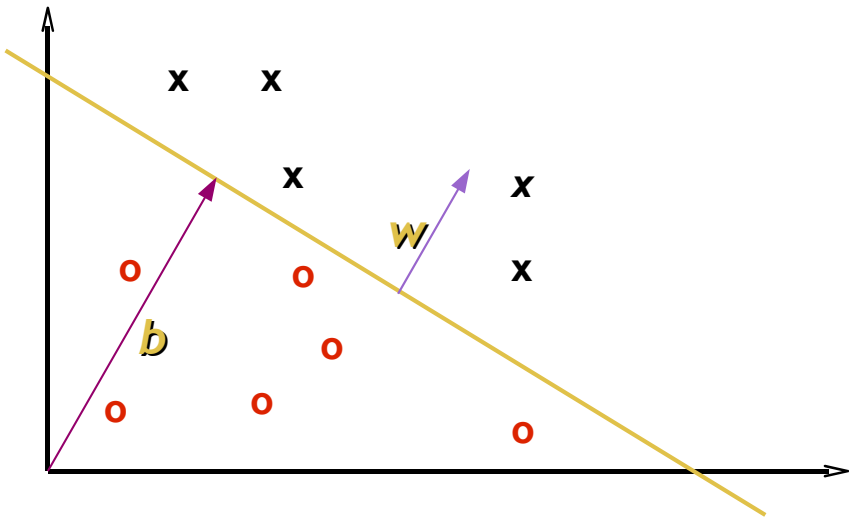


Metodi di Discriminazione Lineare II

- Dati $\{x_i\}$ in uno spazio vettoriale X, diviso in 2 classi $\{-1, +1\}$
- Trovare una separazione lineare ossia un iperpiano $\langle w, x \rangle + b = 0$
- Metodi **lineari** come il percettrone [Rosenblatt,57]



Percettrone



- Separazione lineare dello spazio di input
  $f(x) = \langle w, x \rangle + b$
- *Ipotesi*
  $h(x) = \text{sign}(f(x))$
- *Algoritmo*
  regola di aggiornamento:
  se $y_i \langle w_k, x_i \rangle \leq 0$
  allora

  $$w_{k+1} \leftarrow w_k + \eta y_i x_i$$
  $$k \leftarrow k + 1$$

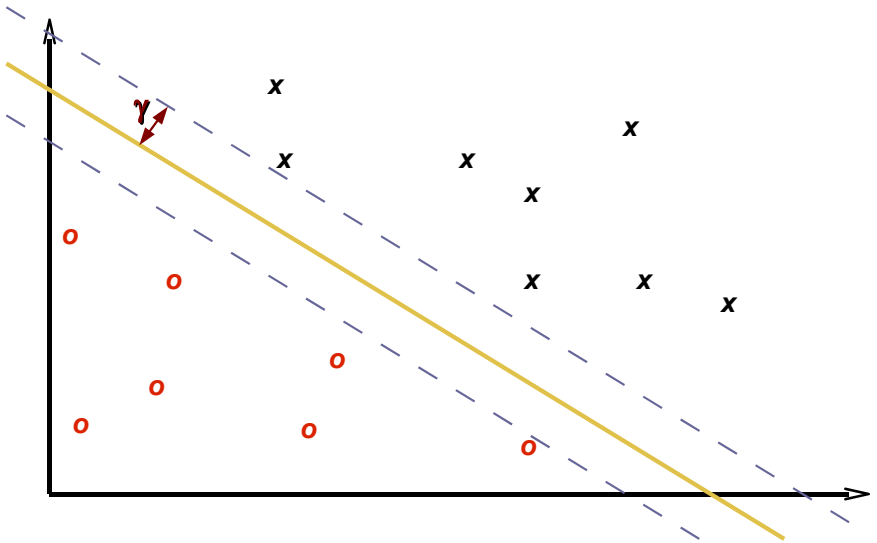
Osservazioni

- soluzione: combinazione lineare dei punti (di training):

$$w = \sum \alpha_i y_i x_i$$

  con $\alpha_i \geq 0$
- sono impiegati solo punti informativi (mistake driven)
- ogni coefficiente nella combinazione è non-negativo; riflette la ”difficoltà” del punto corrispondente

Mistake bound: $M \leq (R/\gamma)^2$



Rappresentazione Duale

Riscrivendo[1] la funzione di decisione:

$$f(x) = \langle w, x \rangle + b = \sum \alpha_i y_i \langle x_i, x \rangle + b$$

dove $w = \sum \alpha_i y_i x_i$

- Servono solo i prodotti interni tra i punti (non le loro coordinate) $\langle x_j, x_i \rangle$
- Riscrivendo la regola di aggiornamento equivalente:
  se $y_i \sum_k \alpha_k y_k \langle x_k, x_i \rangle + b \leq 0$
  allora $\alpha_i \leftarrow \alpha_i + \eta$
- Si passa
  da dim. parametri w = dim. dello spazio dei dati $|X|$
  a dim. parametri α = dim. del campione $|S|$

[1] Nota. x_i, w sono vettori, α_i, y_i sono scalari

Limiti e soluzioni

Limiti

1. I classificatori lineari semplici non possono trattare
   - dati non linearmente separabili
   - dati *rumorosi*
2. Questa formulazione del problema tratta solo dati in forma vettoriale

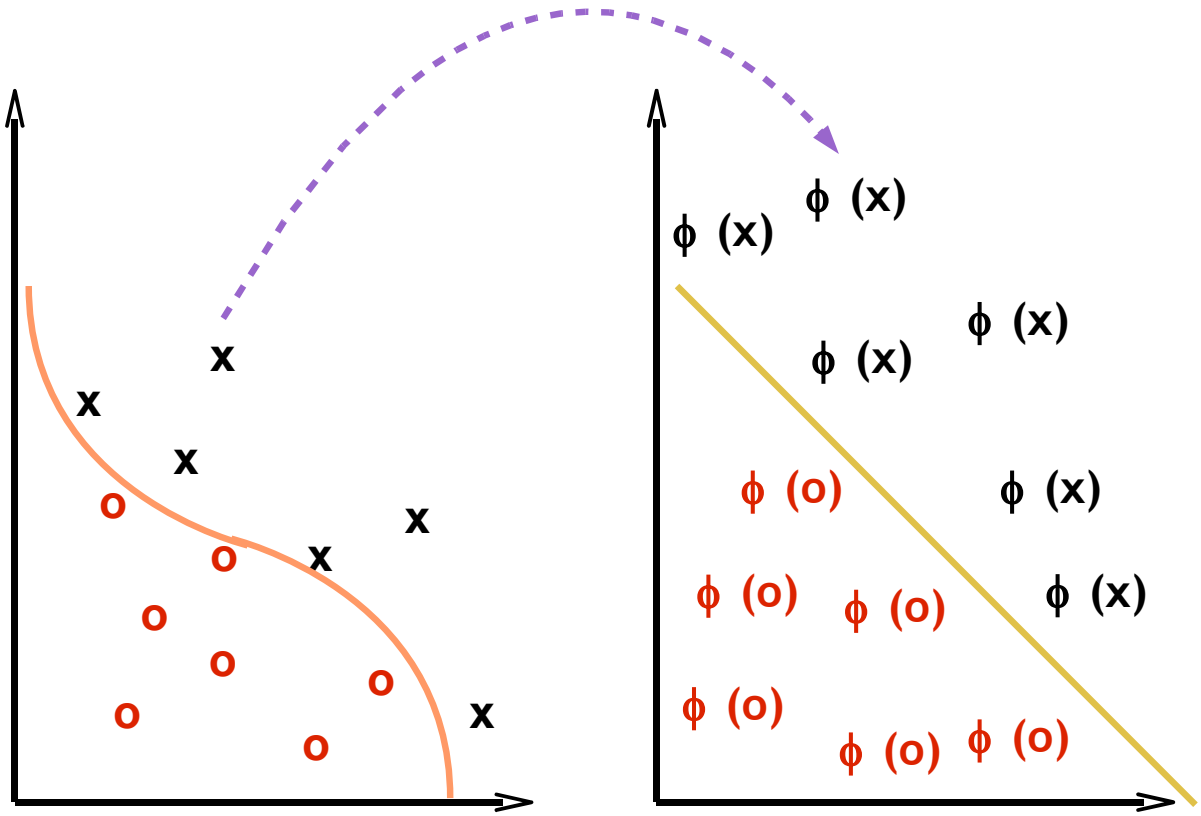
Possibili Soluzioni

1. Rete di classificatori lineari semplici:
   es. una rete neurale
   Problemi: minimi locali, proliferazione di parametri, necessità di euristiche, ...
2. Mappare i dati in uno spazio più ricco che includa caratteristiche non lineari e usare un classificatore lineare

Lavorare in uno spazio di feature I

Mappare i dati in un nuovo spazio di feature F

$$x \longrightarrow \phi(x)$$



Lavorare in uno spazio di feature II

Problemi

Lavorare in uno spazio ad alta dimensionalità risolve il problema anche per funzioni molto complesse

ma sussistono

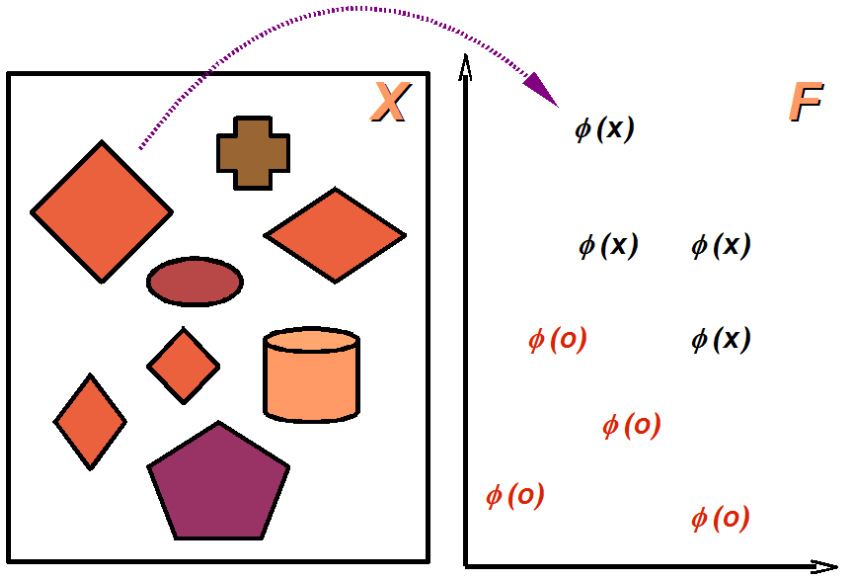
1. problemi di efficienza computazionale lavorando con grandi vettori
2. problemi di (teoria della) generalizzazione curse of dimensionality

Kernel e mapping implicito I

I Kernel Methods lavorano:

- mappando (implicitamente) i dati su un diverso spazio vettoriale *F* (feature space o embedding space)
- cercando relazioni (lineari) in tale spazio

$$x \rightarrow \phi(x)$$



Kernel e mapping implicito II

Osservazioni

1. Se la corrispondenza viene scelta accuratamente, le relazioni potranno essere semplici e facilmente deducibili
2. La geometria dei dati nello spazio F (posizioni relative) è contenuta nei prodotti interni (rappresentazione duale):

$$f(x) = \sum \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$

3. Si può lavorare su tale spazio con un prodotto interno definito sui suoi punti
4. Prodotto spesso calcolabile in modo molto efficiente anche se la dim. di F e anche ϕ possono restare ignote

Kernel I

$$K(x, z) = \langle \phi(x), \phi(z) \rangle$$

- I kernel sono funzioni che calcolano prodotti interni tra le immagini di punti in un qualche spazio
- Sostituendo i kernel ai prodotti negli algoritmi lineari si ottengono rappresentazioni flessibili
- La scelta di K equivale alla scelta di ϕ, della ossia la mappatura per l'embedding
- Anche per spazi di grandi dimensioni, i kernel sono spesso calcolabili in modo efficiente

Kernel II

- Possono lavorare in modo naturale con tipi di dati generici non vettoriali
- Esistono
  - kernel per sequenze (basati sullo string-matching o su HMM [Haussler])
  - kernel per alberi, grafi, struttura generiche
  - kernel semantici per il testo, ecc.
  - kernel basati su modelli generativi

Teorema di Mercer I

Dato un campione S dei dati
si può definire una matrice di kernel

$$K = \begin{pmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \dots & \langle x_1, x_m \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \dots & \langle x_2, x_m \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_m, x_1 \rangle & \langle x_m, x_2 \rangle & \dots & \langle x_m, x_m \rangle \end{pmatrix}$$

nota anche come Gram matrix

Teorema di Mercer II

- La matrice di un kernel è simmetrica e (semi-)definita positiva ($\forall v: \; v^t M v \geq 0$)
- Ogni matrice simmetrica definita positiva può essere considerata come matrice di kernel (prodotto interno in un certo spazio)

Teorema [Mercer]

Ogni funzione K (semi-)definita positiva simmetrica è un kernel ossia, esiste una ϕ tale che si possa scrivere:

$$K(x, z) = \langle \phi(x), \phi(z) \rangle$$

richiamo: K definita positiva sse

$$\forall f \in \mathcal{L}_2 : \int K(x, z) f(x) f(z) dx dz \geq 0$$

Teorema di Mercer III

- Espansione dei kernel di Mercer negli autovalori (*eigenvalues*)

$$K(x, z) = \sum_i \lambda_i \phi_i(x) \phi_i(z)$$

  con $\lambda_i > 0$ or $v^t K v > 0 \qquad \forall v \neq \mathbf{0}$

- Si dice anche che le autofunzioni (*eigenfunction*) $\{\phi_i\}_{i=1}^n$ si comportano come *feature*

Esempi di kernel I

- Polinomiale:

$$K(x, z) = (\langle x, z \rangle + \theta)^d$$

con $d, \theta \in \mathbb{R}$

- Gaussiano:

$$K(x, z) = \exp(-\|x - z\|^2 / c)$$

con $c \in \mathbb{R}$

- Sigmoidale:

$$K(x, z) = \tanh(\kappa \langle x, z \rangle + \theta)$$

con $\kappa, \theta \in \mathbb{R}$

Esempi di kernel II

- Multiquadratico inverso:

$$K(x, z) = 1/\sqrt{||x - z||^2 + c^2}$$

con $c \in \mathbb{R}_+$

- Normalizzazione: $x \mapsto \phi(x) \mapsto \frac{\phi(x)}{||\phi(x)||}$

$$K(x, z) = \left\langle \frac{\phi(x)}{||\phi(x)||}, \frac{\phi(z)}{||\phi(z)||} \right\rangle = \frac{K(x, z)}{\sqrt{K(x, x)K(z, z)}}$$

Esempi di kernel III

- Un kernel può essere considerato un'oracolo che indovini la similarità degli input sotto la distribuzione $q(f)$:

$$\begin{aligned} K_q(x, z) &= \int_q f(x) f(z) q(f) df \\ &= P_q(f(x) = f(z)) - P_q(f(x) \neq f(z)) \end{aligned}$$

- Dati due sottoinsiemi $A, B \subseteq \mathcal{U}$:

$$K(A, B) = 2^{|A \cap B|}$$

Esempi di kernel IV

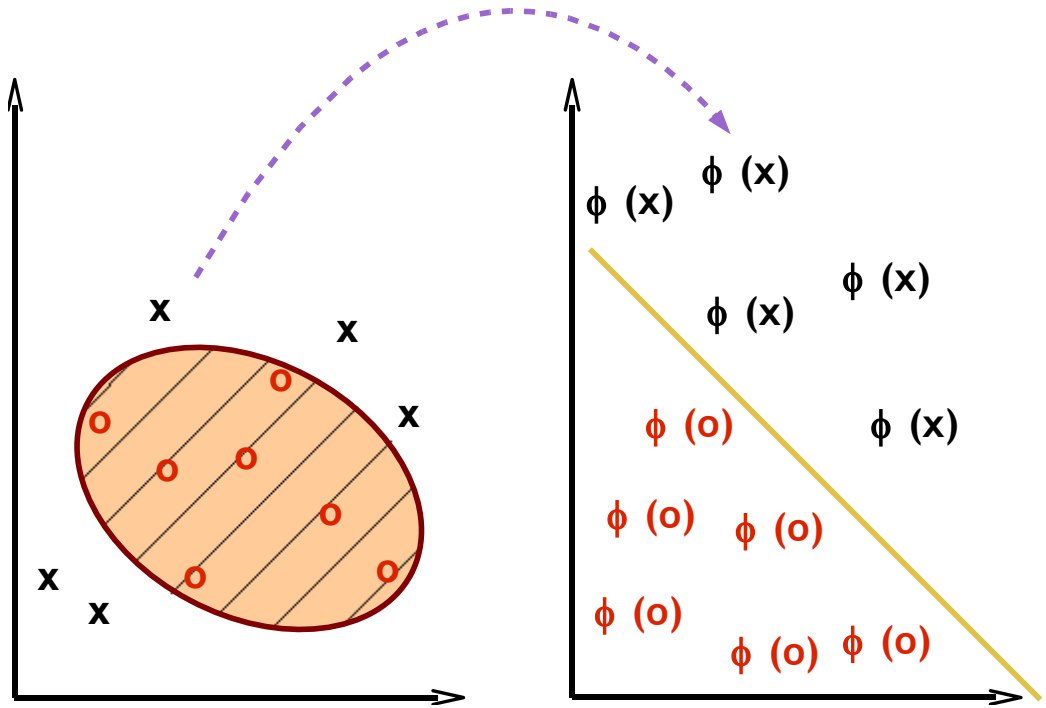
Kernel Polinomiale $K(x, z) = (\langle x, z \rangle + \theta)^d$
Consideriamo lo spazio $X = \mathbb{R}^2$
con $d = 2$ e $\theta = 0$

Dati:
$x = (x_1, x_2)$ e
$z = (z_1, z_2)$

$$\begin{aligned} K(x,z) = \langle x, z \rangle^2 &= (x_1 z_1 + x_2 z_2)^2 = \\ &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 = \\ &= x_1^2 z_1^2 + x_2^2 z_2^2 + x_1 x_2 \sqrt{2} \cdot z_1 z_2 \sqrt{2} = \\ &= \langle (x_1^2, x_2^2, x_1 x_2 \sqrt{2}), (z_1^2, z_2^2, z_1 z_2 \sqrt{2}) \rangle = \\ &= \langle \phi(x), \phi(z) \rangle \end{aligned}$$

Esempi di kernel V

Metodo lineare + kernel polinomiale per apprendere, in modo efficiente, separazioni in origine non lineari

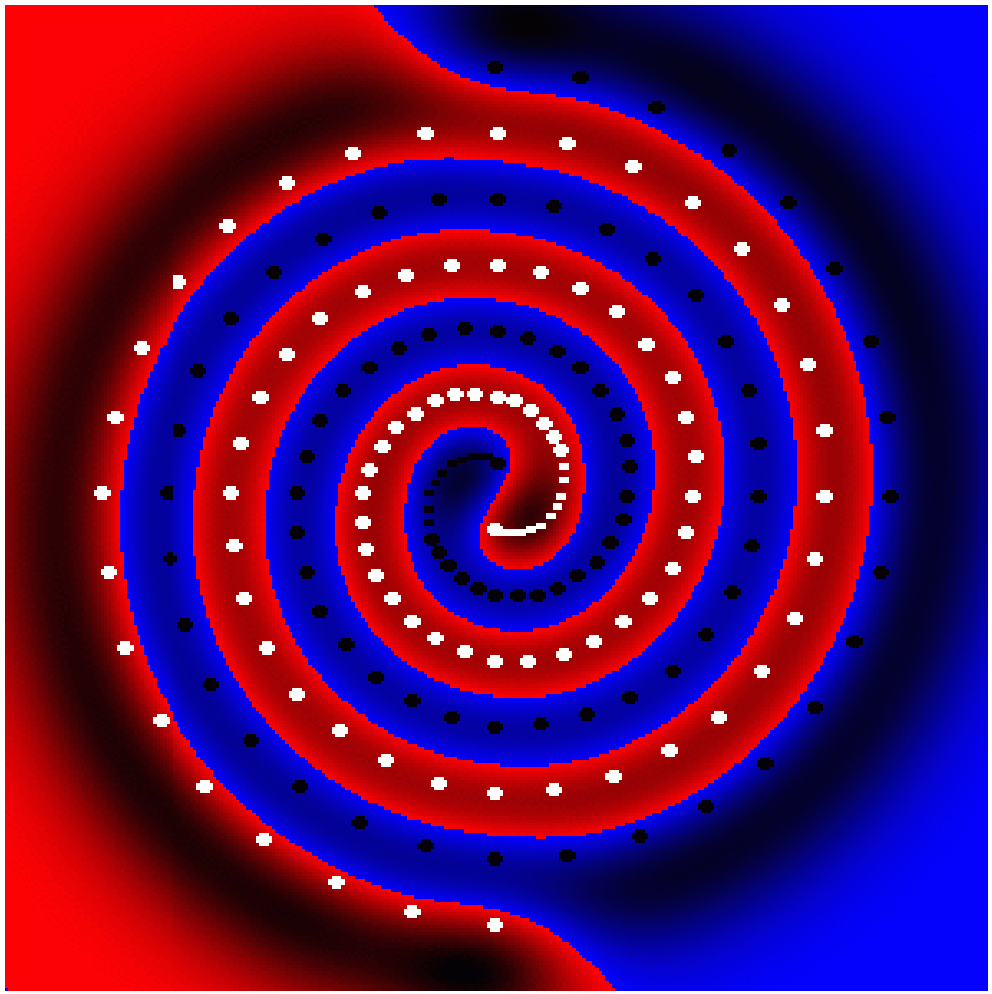


$$x = (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2) = \phi(x)$$

Esempi di kernel VI

Kernel Gaussiano

Iperpiano in uno spazio individuato da un kernel gaussiano:



Costruire nuovi kernel I

La classe dei kernel è chiusa rispetto a varie operazioni:

- dati i kernel K e K' per un certo spazio X
  - cK è un kernel $\forall c \in \mathbb{R}$
  - $K \pm K'$ è un kernel
  - quindi anche $aK \pm bK'$ è un kernel $\forall (a, b) \in \mathbb{R}^2$
    si dice anche che la classe costituisce un
    cono chiuso convesso
- **prodotto**:
  $K = K_1 \cdot K_2$ è un kernel
  $K(x, z) = f(x)f(z)$ è un kernel
- $K(x, z) = K'(\phi(x), \phi(z)) \qquad \forall \phi : X \mapsto \mathbb{R}^N$

Costruire nuovi kernel II

- $K(x, z) = x^t Bz$ $\quad \forall B$ semi-definite positive
- **zero-estensione**:
  se K è un kernel per X allora lo è anche per ogni $S \supseteq X$,
  ponendo: $K(x, z) = 0$ se $x \in S \setminus X$ o $z \in S \setminus X$
- inoltre la classe è chiusa rispetto alla somma diretta $\oplus$ e al prodotto tensoriale $\otimes$

Costruire nuovi kernel III

- **convoluzione** [Haussler,99]
  Dato un insieme di kernel
  $\{K_d : X_d \times X_d \mapsto \mathbb{R} \mid d = 1, \ldots, D\}$,
  siano $x, z \in X$ punti complessi che raggruppano elementi da questi spazi, scomponibili mediante la relazione
  $R \subseteq X_1 \times \cdots \times X_D \times X$
  (i.e. valgono $R(x_1, \ldots, x_D, x)$ e $R(z_1, \ldots, z_D, z)$):

$$K(x, z) = \sum_{\substack{\vec{x} \in R^{-1}(x) \\ \vec{z} \in R^{-1}(z)}} \prod_{d=1}^{D} K_d(x_d, z_d)$$

  dove $R^{-1}(x)$ restituisce il vettore delle d componenti di x

Scelta del kernel

Kernel con matrice pressochè diagonale
(i.e. punti ortogonali tra loro)

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

manca di strutturazione $\Rightarrow$ inutile

- Problema che può discendere dal mapping su uno spazio con troppe feature irrilevanti
- Per scegliere un buon kernel occorre conoscenza di fondo sulla funzione (concetto) obiettivo

Modularità I

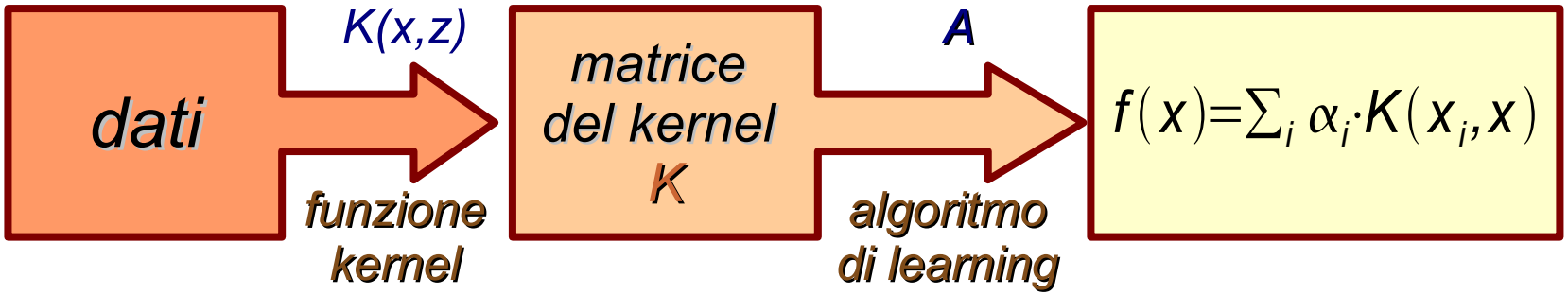
- Esistono sia algoritmi sia kernel più sofisticati di quello lineare o polinomiale

$$f(x) = \sum_i \underbrace{\alpha_i}_{\textit{Learning}} \cdot \underbrace{K(x_i, x)}_{\textit{Kernel}}$$

- **Idea**: un modulo di apprendimento general-purpose dei coefficienti e un kernel specifico per il problema (rappresentazione)
- Ideale da un punto di vista ingegneristico

Apprendimento

1. si estrae dal dataset la matrice del kernel
2. si analizzano i dati sulla base dell'informazione contenuta nella matrice



Problema della Generalizzazione

- Sarebbe facile, dati N esempi di training mappare gli stessi su uno spazio di (almeno) N dimensioni
- Curse of dimensionality:
  facile cadere nell'overfitting in spazi a più dimensioni
  - le regolarità trovate nel training set sono spesso solo accidentali
- il problema è ancora mal posto:
  tra tanti possibili iperpiani trovare i migliori
  - occorrono quindi principi su cui basare la scelta
  - Bayes, MDL, Statistical Learning Theory / PAC, ...

Statistical Learning Theory (SLT)

SLT Testi: [Vapnik,95] [Vapnik,98]

- Limitare il rischio di overfitting
- Ricorrendo alla SLT si possono dare limiti inferiori e superiori proporzionali alla dimensione VC (Vapnik-Chervonenkis)
  - misura del più grande sottoinsieme di X separabile (*shattered*) dall'ipotesi h
- $VC \propto m$ (dim. spazio F)
  $VC =$ n. di dimensioni di $X + 1 \gg m$

$$\varepsilon = \tilde{O}(VC/m)$$

Come scegliere l'iperpiano?

Errore e Funzioni di Rischio

Migliore funzione $h(\cdot)$
Obiettivo: minimizzare l'errore atteso (rischio)

$$R(h) = \int l(h(x), y) dP(x, y)$$

dove l denota un'opportuna loss function

Ad es. $l(h(x), y) = \Theta(-yh(x))$ (0/1 *loss function*)
$\Theta(z) = \begin{cases} 0 & z < 0 \\ 1 & \text{altrimenti} \end{cases}$

Sfortunatamente
distribuzione P ignota $\Rightarrow$ R non può essere minimizzato direttamente

Rischio Empirico

Minimizzare l'errore empirico (sul training set)

$$R_{\text{emp}}(h) = \frac{1}{n} \sum_i l(h(x_i), y_i)$$

Rappresenta una stima di R:
per la *legge dei grandi numeri*

$$\lim_{n \to \infty} R_{\text{emp}}(h) = R(h)$$

Come fare per piccoli campioni di dati ?

Problema del sovradattamento

- Controllare la complessità di h:
  - preferire funzioni semplici (es. lineari) (rasoio di Occam)
  - termini di regolarizzazione per limitare la complessità delle funzioni che l'algoritmo può scegliere
  - comporta problemi di selezioni del modello (model selection): come trovare la complessità ottimale della funzione ?
- Un criterio è basato sulla dim. VC della classe di funzioni da cui scegliere h

Dimensione VC I

La VC dimension misura ricchezza e flessibilità di una famiglia di ipotesi:

Def. [VC dimension]

Data una famiglia di funzioni $H = \{h_j\}$,
la *VC dimension* è il numero d tale che:

- Esiste almeno un insieme di d esempi che possono essere classificati in qualsiasi maniera da funzioni di H;
  ossia, per ogni possibile classificazione dei d esempi, esiste una funzione in H che assegni la classe corretta
- Non esiste nessun insieme di $d + 1$ esempi che possono essere classificati in qualsiasi modo

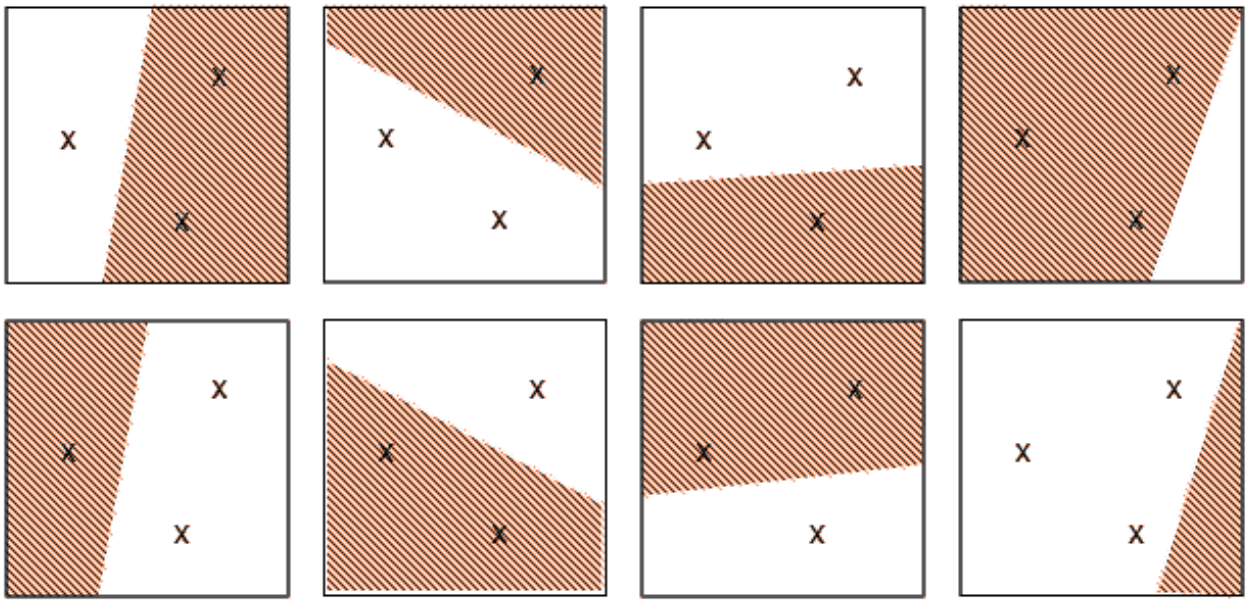
Dimensione VC II

Esempio: semi-spazi in $\mathbb{R}^2$:

$h(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2$

- si possono separare fino a 3 punti (non sulla stessa retta)
- non si possono separare 4 punti

La VC dimension è allora 3 (pari anche al num. di parametri)



Limitare la Complessità I

Teorema [Vapnik]

Data una funzione h scelta in una classe H

$$R(h) \leq R_{\text{emp}}(h) + \sqrt{\frac{d(\ln(2n/d) + 1) - \ln(\delta/4))}{n}} \qquad (1)$$

con probabilita $1 - \delta$ per $n > d$

- R_{emp} definito dalla loss function
- d VC-dimension della classe H
- δ parametro di confidenza

Limitare la Complessità II

Osservazioni

- Il limite non dipende dalla distribuzione degli esempi
- Il limite non è asintotico ma calcolabile per ogni d finita
- La differenza tra i rischi
  - si riduce al crescere di n
  - cresce al crescere di d
- A parità di rischio empirico,
  si ottiene un limite inferiore scegliendo una ipotesi da una famiglia con VC-dimension minore

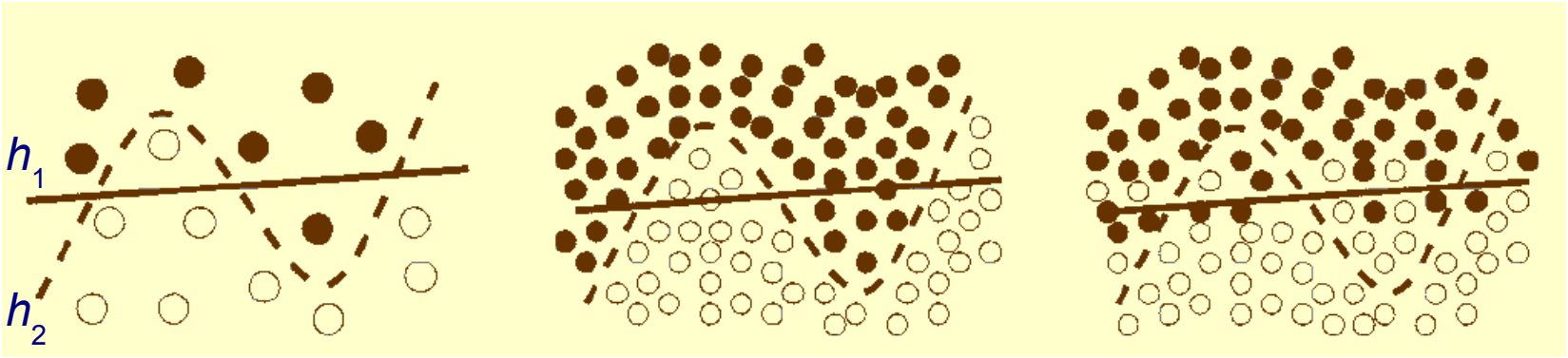
Limitare la Complessità III

Tornando alla relazione (1) del teorema, occorre (contemporaneamente):

- tenere basso $R_{emp} \approx 0$
- controllare la complessità della funzione h della famiglia H
  In pratica, VC-dimension:
  - ignota
  - infinita

Limitare la Complessità IV

Due ipotesi: una più semplice h_1 (linea continua) e una più complessa h_2 (linea tratteggiata)



Piccolo campione di dati

1. h_1 e h_2 potrebbero essere corrette entrambe, ma h_2 ha meno errore di training

Campione più grande:

2. se fosse corretta h_2, h_1 sarebbe in sottoadattamento (*underfit*)
3. se fosse corretta h_1, h_2 sarebbe in sovradattamento (*overfit*)

Limitare la Complessità V

in generale:

- iperpiani separatori in $\mathbb{R}^n$ hanno dimensione VC pari a $n+1$
- quindi iperpiani per spazi di feature di grandi dimensioni avranno una grande VC dimension

La complessità può essere limitata mediante iperpiani che massimizzino i margini (con bassa VC-dimension)

Support Vector Machine (SVM)

Si miminizza il rischio scegliendo iperpiani che massimizzano i margini nello spazio delle feature

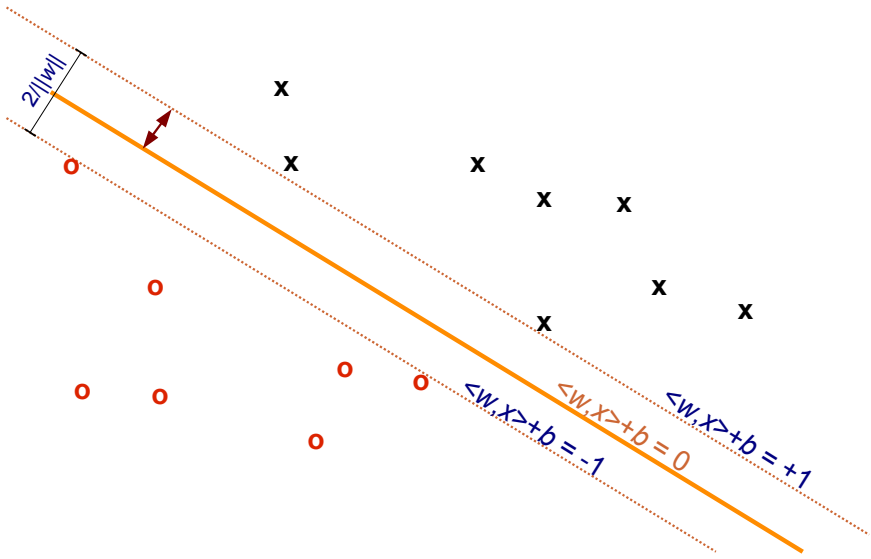
Le SVM controllano la capacità aumentando il margine, e non riducendo il numero di gradi di libertà

- Ben fondate su basi statistiche (SLT)
- Popolari tra gli addetti ai lavori
  - introdotte a COLT'92 [Boser, Guyon & Vapnik,92]
- Rappresentano lo *stato dell'arte* in molte applicazioni

Margini I

Criterio massimizzazione del margine (nello spazio F)

- margine: distanza dell'iperpiano dal punto più vicino (nella separazione)
- iperpiano individuato in base ad un sottoinsieme di esempi detti support vector



Margini II

Nell'ipotesi che un iperpiano $\langle w, x \rangle + b$ di separazione esista

Siano $x_+ \in P^+$ e $x_- \in P^-$ punto più vicino a x_+
con $P^+ : \langle w, x_i \rangle + b = +1$ e $P^- : \langle w, x_i \rangle + b = -1$.
Quindi: $\langle w, x_+ \rangle + b = +1$ e $\langle w, x_- \rangle + b = -1$

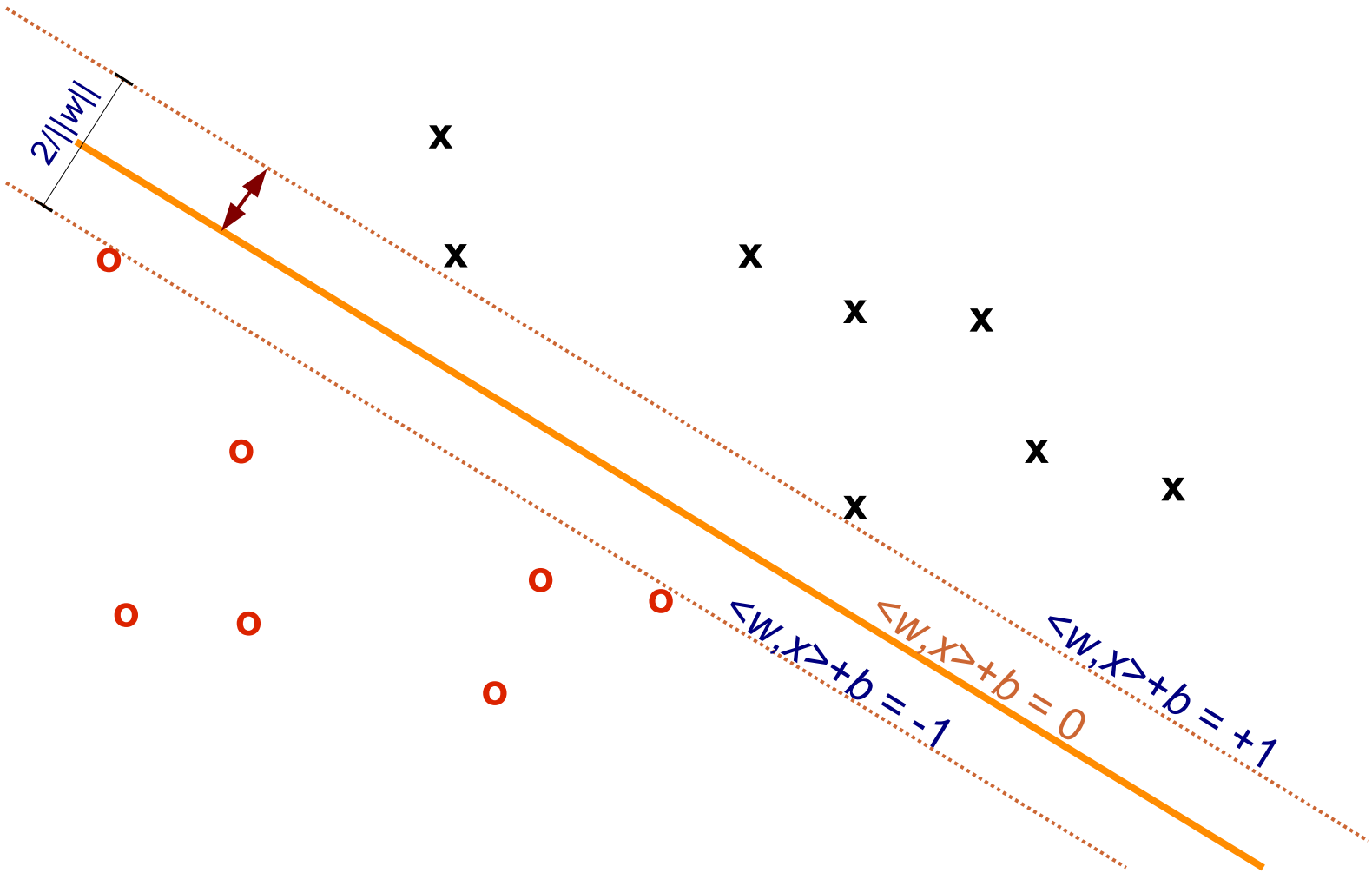
Da cui, sottranendo m. a m. e normalizzando w,
il margine risulta:

$$\left\langle \frac{w}{\|w\|}, (x_+ - x_-) \right\rangle = \frac{2}{\|w\|} = 2C$$

distanza tra i due punti (sulla perpendicolare)

Massimizzare il margine $2C$ equivale a minimizzare $||w||^2$ o anche $||w||^2/2$ (per convenienza di calcolo)

Margini III



Margini e VC-dimension

[Vapnik,95]

$$d \leq \Lambda^2 R^2 \text{ e } \|w\|_2 \leq \Lambda$$

dove R = raggio della ipersfera più piccola *attorno* ai punti

Occorre:

1. limitare il margine si controlla la dimensione VC
2. si possono usare funzioni lineari mediante i kernel

Problema di apprendimento per le SVM

Problema

trovare un iperpiano che massimizzi i margini

- usando un iperipiano come classificatore per X
  condizioni di assenza di errore:
  $y_i(\langle w, x_i \rangle + b) \geq 1, \qquad i = 1, \ldots, n$
- usando un classificatore non lineare per F:
  $y_i(\langle w, \phi(x_i) \rangle + b) \geq 1, \qquad i = 1, \ldots, n$
- Si devono apprendere $w \in F$ e b in modo da minimizzare il rischio: rischio empirico + complessità
  - $R_{\text{emp}} \approx 0$
  - controllare la complessità attraverso la dim. VC, cfr. eq. precedenti: $d \leq \|w\|^2 R^2$

Programmazione Quadratica

Problema risolubile mediante tecniche di

Programmazione Quadratica (QP) convessa

la cui formulazione consente di trattare anche il *rumore*

Problema
trovare un iperpiano tale che

$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$\text{sub } y_i(\langle w, \phi(x_i)\rangle + b) \geq 1, \qquad i = 1, \ldots, n$$

Problemi in QP

Primale

- $\frac{1}{2}\langle w, w\rangle - \sum \alpha_i [y_i(\langle w, x_i\rangle + 2) - 1]$
  sotto le condizioni:
  - $\alpha_i \geq 0$

utilizzando il metodo dei moltiplicatori di Lagrange si può riscrivere il problema nella forma

Duale

- $W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j\rangle$
  sotto le condizioni:
  - $\alpha_i \geq 0$
  - $\sum_{i=1}^{n} \alpha_i y_i = 0$

Proprietà della soluzione

- Problema di ottimizzazione convesso con un'unica soluzione α^*
- α^* gode di una delle proprietà di ottimalità di Karush-Kuhn-Tucker (KKT):
  coefficienti α_i^* tutti nulli
  tranne quelli relativi ai *vettori di supporto*
  (i più vicini al separatore)
- Il duale w^* risulta una combinazione lineare di tali vettori:

$$w^* = \sum_i \alpha_i^* y_i x_i$$

Funzione di decisione con SVM

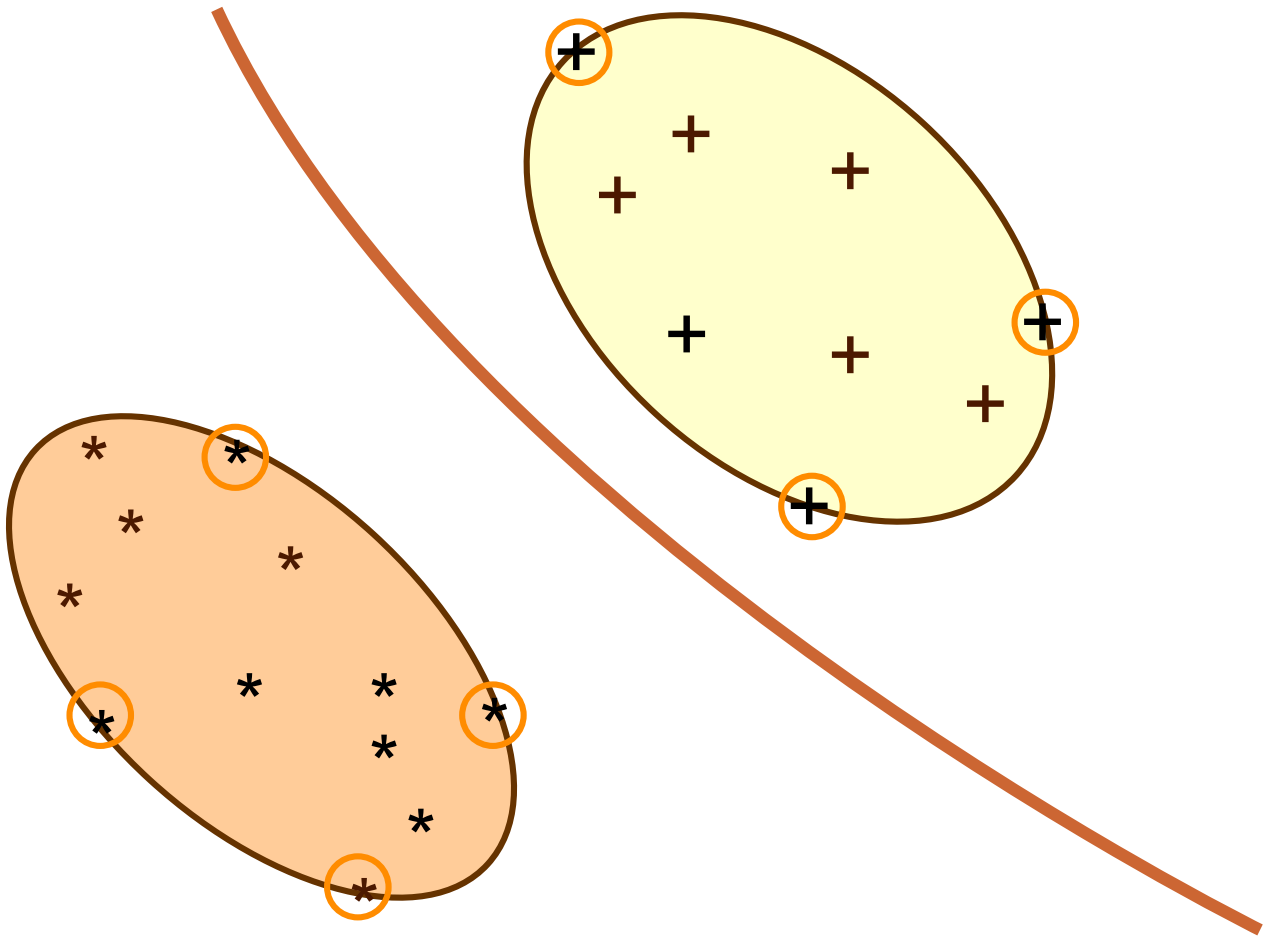
Passando ai kernel
il termine $\langle x_i, x_j \rangle$ diventa $\langle \phi(x_i), \phi(x_j) \rangle$ e quindi $K(x_i, x_j)$

Il che porta alla funzione di decisione:

$$\begin{aligned} h(x) &= \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i \langle \phi(x_i), \phi(x) \rangle + b\right) \\ &= \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + b\right) \end{aligned}$$

Separazione non Lineare

Passando ai kernel si possono apprendere funzioni non lineari:



Hard e Soft-margin SVM

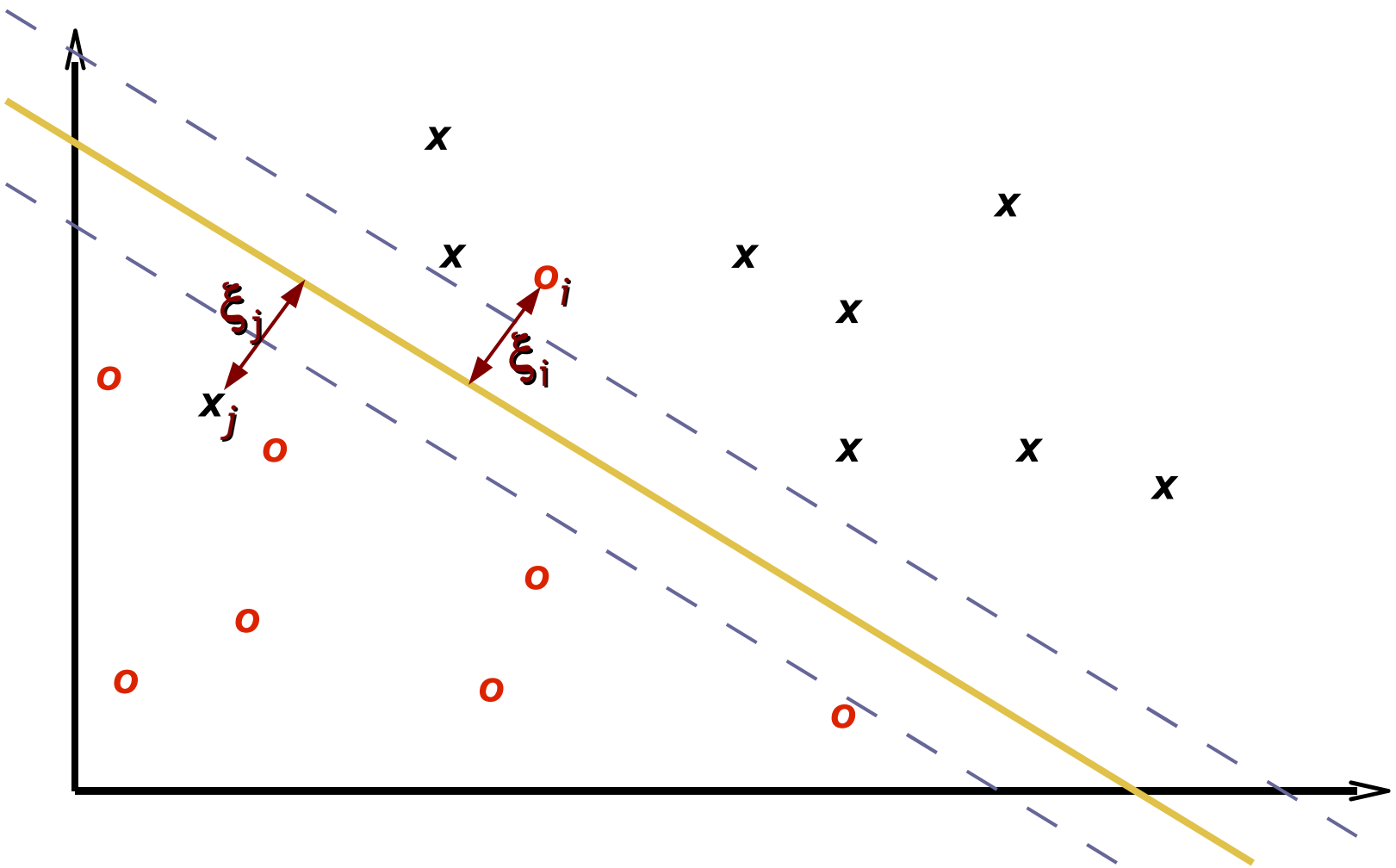
- hard-margin SVM
  modello consistente con gli esempi a disposizone
  può essere reso *instabile* all'arrivo di nuovi esempi
  $\longrightarrow$ serve prevedere la presenza di outlier/rumore

- soft-margin SVM
  utilizzando variabili slack, il problema diventa:

$$y_i(\langle w, \phi(x_i)\rangle + b) \geq 1 - \xi_i \qquad i = 1, \ldots, n$$

  consentendo errori di classificazione in fase di training

Rumore I



Rumore II

Oltre a controllare la complessità attraverso la dim. VC, occorre controllare il limite per $\sum_{i=1}^{n} \xi_i$ nell'errore empirico:

$$\min_{w,b,\xi} \left(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i \right)$$

C = termine di regolarizzazione per bilanciare le due fonti

Problema Duale

- $W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$ sotto le condizioni:
  - $0 \leq \alpha_i \leq C$ (box constraint)
  - $\sum_{i=1}^{n} \alpha_i y_i = 0$

Risolto nella medesima maniera

Osservazioni

$\alpha_i^* = 0$ es. classificato correttamente;

$0 < \alpha_i^* < C$ es. all'interno del margine, compreso tra l'iperpiano dei SV e l'iperpiano classificatore e dalla parte giusta;

$\alpha_i^* = C$ es. classificato erroneamente, pesato con C indipendentemente dalla sua distanza dall'iperpiano classificatore;

Calcolo della soglia b

Si utilizzano i casi delle x_i in cui la variabile slack $\xi_i = 0$.
Se la matrice è sparsa allora valgono le condizioni di ottimalità KKT:

$$0 \leq \alpha_i \leq C \Rightarrow y_i h(x_i) = 1$$

ossia per i support vector x_i risulta $i \in I = \{k \mid 0 < \alpha_k < C\}$, quindi:

$$y_i \left(b + \sum_{j=1}^{n} \alpha_j y_j k(x_i, x_j) \right) = 1$$

da cui, mediando:

$$b = \frac{1}{|I|} \sum_{i \in I} \left(y_i - \sum_{j=1}^{n} \alpha_j y_j k(x_i, x_j) \right)$$

Proprietà delle SVM

1. Convessità niente minimi locali:
   training = ottimizzazione convessa
2. Dualità: lavorano sul problema in forma duale cui applicare il kernel trick
3. Sparsità: basate su pochi vettori (di supporto)
4. Complessità polinomiale

SVM: un bilancio

Vantaggi

- risuluzione problemi non-lineari
- eliminando il problema della dimensionalità
- ottimizzazione globale
- intepretazione geometrica
- prestazioni che dipendono dal numero di SV

Svantaggi

- la scelta del kernel risulta decisiva
- training lento con grandi dataset
- non trattano dati discreti
- non incrementale
- difficile da generalizzare al caso multiclassi

Credits

- John Shawe-Taylor & Nello Cristianini
  Kernel Methods for Pattern Analysis,
  Cambridge University Press, 2004
- K.-R. Müller, S. Mika, F. Rätsch, K. Tsuda, B. Schölkopf
  An Introduction to Kernel-Based Learning Algorithms
  In: *IEEE Transactions on Neural Networks*, Vol. 12, No. 2, 2001
- Support Vector and Kernel Machines
  di Nello Cristianini (Tutorial ICML 2001)
- Altri testi, articoli, survey e tutorial:
  - `http://www.kernel-machines.org`
  - `http://www.support-vector.net`