
Tecniche di riconoscimento statistico

Applicazioni alla lettura automatica di testi
(OCR)

Parte 8 – Support Vector Machines

Ennio Ottaviani

On AIR srl

ennio.ottaviani@onairweb.com

<http://www.onairweb.com/corsoPR>

A.A. 2010-2011



Introduzione

- Le Support Vector Machine (SVM) sono state sviluppate presso gli AT&T Bell Laboratories, a partire dai primi anni `90, da Vapnik ed altri
- Dato il contesto industriale, la ricerca sulle SVM ha finora avuto una decisa inclinazione applicativa



Introduzione

- Primi lavori relativi a:
 - OCR (*Optical Character Recognition*, dove in breve le SVM divennero competitive con i metodi migliori)
 - Riconoscimento di oggetti
 - Identificazione di volti in immagini
 - Classificazione di testi
 - Bioinformatica



Introduzione

- L'algoritmo *Support Vector*, alla base delle SVM,
 - ✦ ...è una generalizzazione dell'algoritmo *Generalized Portrait*, sviluppato in Russia nei primi anni '60
 - ✦ ...appartiene alla categoria degli algoritmi di *statistical learning theory*, la teoria di Vapnik–Chervonenkis (Vapnik, Chervonenkis 1974)
- Essenzialmente, la teoria VC caratterizza le proprietà degli algoritmi di apprendimento che permettono loro di “estendere la conoscenza” a dati nuovi, in base ad elementi appresi in precedenza



Introduzione

- Una SVM è un classificatore binario che apprende il confine fra esempi appartenenti a due diverse classi
- Funziona proiettando gli esempi in uno spazio multidimensionale e cercando un iperpiano di separazione in questo spazio
- L'iperpiano di separazione massimizza la sua distanza (il "margine") dagli esempi di training più vicini
- Proprietà generali delle SVM:
 - ✦ improbabile l'overfitting
 - ✦ capacità di gestire dati con molte feature
 - ✦ estrazione dell'informazione "più significativa" contenuta nel data set in input



Classificazione

- Nel caso della classificazione di dati appartenenti a due sole classi, il processo di apprendimento può essere formulato come segue:
 - dato un insieme di funzioni di soglia

$$\{f_\lambda(x): \lambda \in \Lambda\}, \quad f_\lambda: \mathbb{R}^N \rightarrow \{-1, +1\}$$

dove Λ è un insieme di parametri reali, e dato un insieme di esempi preclassificati $(x_1, y_1), \dots, (x_m, y_m)$, $x_i \in \mathbb{R}^N$, $y_i \in \{-1, +1\}$, presi da una distribuzione sconosciuta $P(x, y)$, si vuole calcolare una funzione f_λ^* che minimizzi l'errore teorico:

$$R(\lambda) = \int |f_\lambda(x) - y| P(x, y) dx dy$$



Errore teorico

- L'insieme di parametri reali Λ genera una macchina in grado di risolvere un particolare problema (ad esempio Λ può corrispondere ai pesi delle sinapsi di una rete neurale)
- Le funzioni f_λ sono le ipotesi, e l'insieme $\{f_\lambda(x): \lambda \in \Lambda\}$ è lo spazio H delle ipotesi
- L'errore teorico rappresenta una misura di quanto sia buona un'ipotesi nel predire la classe y_i di un punto x
- L'insieme delle funzioni può essere realizzato da un MLP classico con un numero opportuno di unità nascoste



Errore empirico

- La distribuzione di probabilità $P(x,y)$ non è nota, quindi non è possibile calcolare l'errore teorico $R(\lambda)$; tuttavia è disponibile un campione di $P(x,y)$: il training set
 - si può calcolare un'approssimazione di $R(\lambda)$, l'errore empirico $R_{emp}(\lambda)$:

$$R_{emp}(\lambda) = \frac{1}{l} \sum_{i=1}^l |f_{\lambda}(x_i) - y_i|$$

- La legge dei grandi numeri garantisce che l'errore empirico converge in probabilità all'errore teorico



Classificatore lineare su dati linearmente separabili

- Ipotesi: insieme di dati linearmente separabili; si vuole calcolare il miglior iperpiano separatore
- Un insieme di dati è linearmente separabile quando esiste una coppia (w, b) tale che

$$\begin{array}{ll} wx_i + b \geq 0 & \text{per } x_i \in C_1 \\ wx_i + b < 0 & \text{per } x_i \in C_2 \end{array}$$



Classificatore lineare su dati linearmente separabili

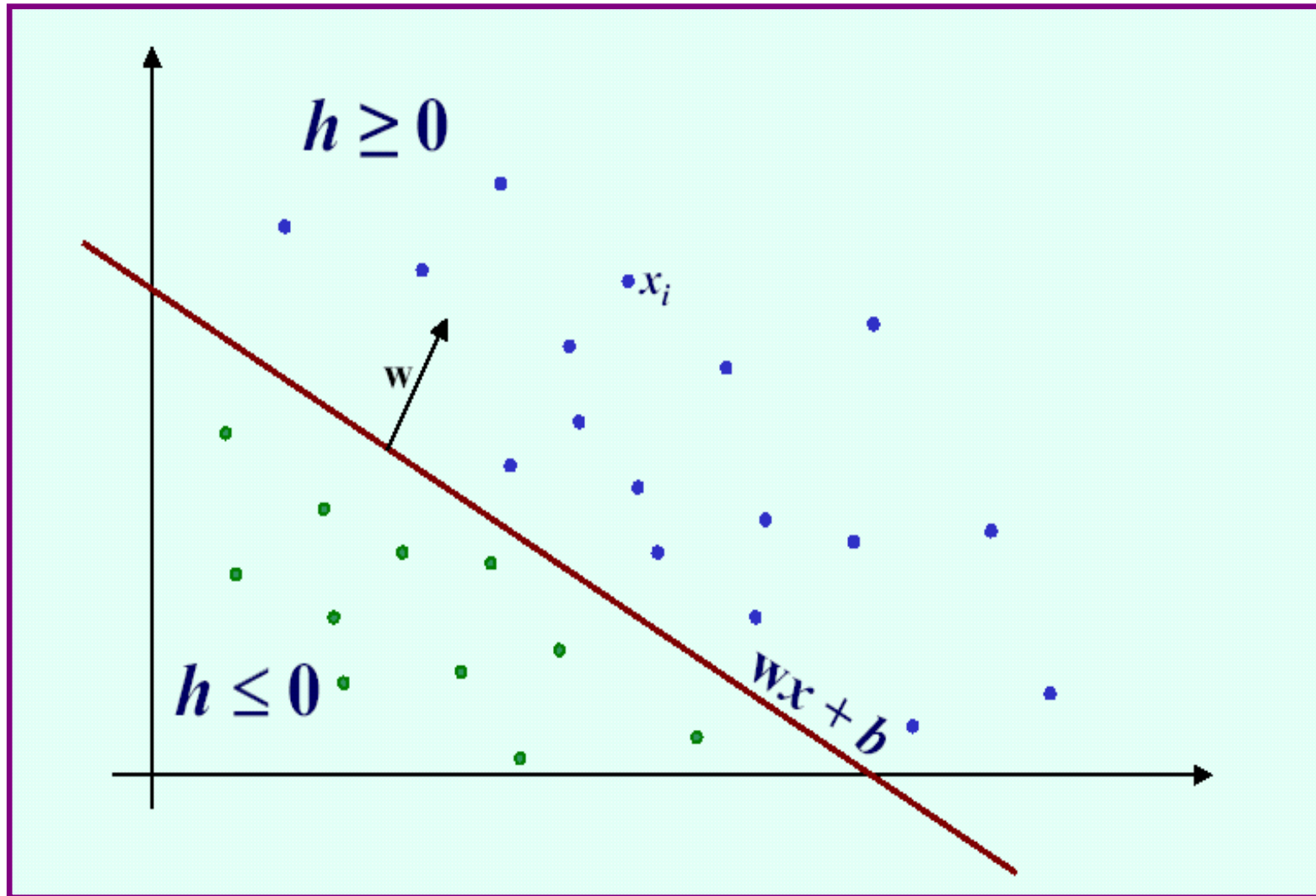
- Lo spazio delle ipotesi in questo caso è formato dall'insieme di funzioni

$$h = f_{w,b} = \text{sign}(w \cdot x + b)$$

(*sign*, discriminatore binario: +/−, 0/1, vero/falso, etc.)

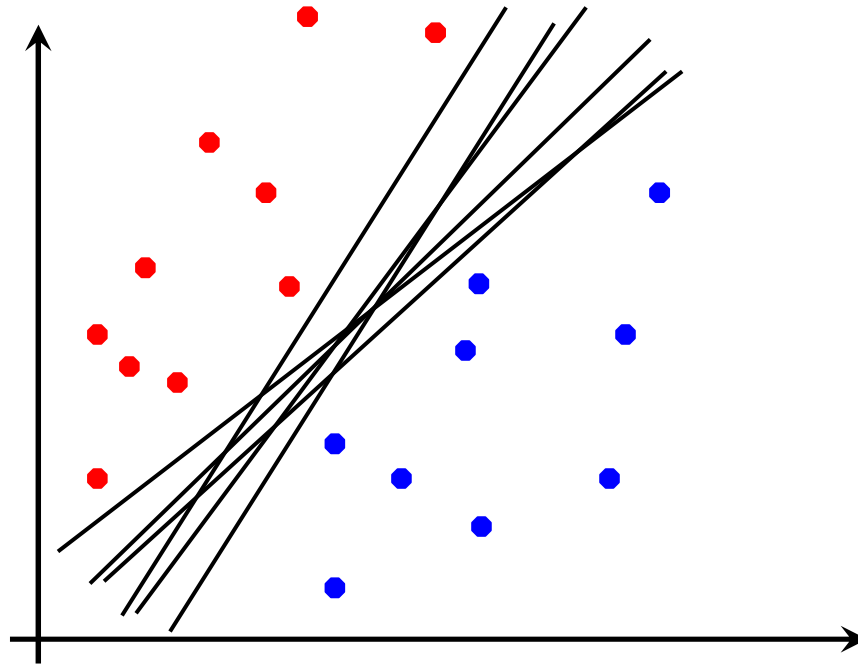


Separazione lineare (es. perceptron)



Separatori lineari

- Quale separatore è ottimo?



Iperpiani di separazione

- La distanza tra un punto x e l'iperpiano associato alla coppia (w,b) è:

$$d_w(x) = \frac{|w \cdot x + b|}{\|w\|}$$

- Se l'iperpiano $wx+b=0$ separa il training set D , si definisce *margin* la quantità $\rho(w,D)$

$$\rho(w, D) = \min_{i=1}^l d_w(x_i)$$



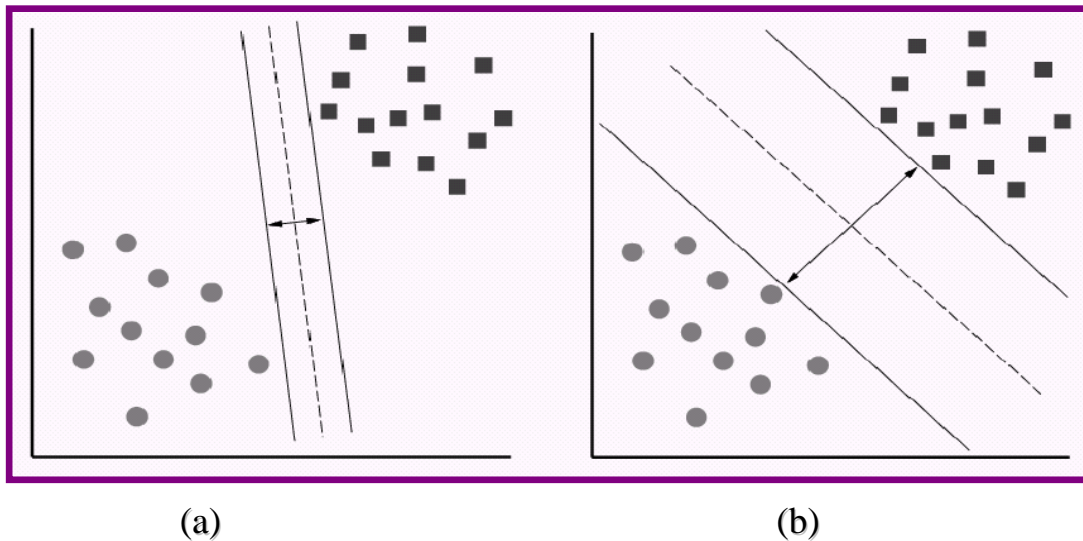
Iperpiani di separazione

- Come scegliere la parametrizzazione?
 - ✦ Un iperpiano separatore è parametrizzato dai coefficienti w e b , ma la scelta non è unica: moltiplicando i parametri per una costante positiva si ottiene lo stesso iperpiano
 - ⇒ Occorre fissare $\|w\|$
 - ✦ Prima soluzione: si fissa $\|w\|=1$, nel qual caso $d_w(x)=|w \cdot x + b|$ e $\rho(w, D) = \min_i y_i (w \cdot x_i + b)$
 - ✦ Altrimenti... si fissa $\|w\|$ in modo tale che $\rho(w, D) = 1/\|w\|$, cioè si impone $\min_i y_i (w \cdot x_i + b) = 1$
 - ⇒ è una parametrizzazione *data-dependent*



Iperpiani di separazione

- Se i dati sono linearmente separabili, lo scopo della SVM è quello di trovare, tra tutti gli iperpiani che classificano correttamente il training set, l'iperpiano che ha norma minima ($\|w\|$ minima), cioè margine massimo rispetto ai punti del training set



Le classi dei cerchi e dei quadrati sono separate dal piano tratteggiato, con un margine piccolo (a), o grande (b)

Nel caso (b) ci si aspetta un minor rischio di overfitting, ovvero una maggiore capacità di generalizzazione



Massimo margine

- L'iperpiano ottimo è quello che massimizza il margine, cioè la distanza tra se stesso e i punti più vicini dell'insieme di dati
- Per costruire l'iperpiano ottimo, occorre classificare correttamente i punti del training set nelle due classi $y_i \in \{-1, 1\}$ minimizzando la norma di w
- Il problema può essere formulato come segue:

Minimizzare

$$\Phi(w) = \frac{1}{2} \|w\|^2$$

con w, b soggetti al vincolo

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



Lagrangiana

- Questo problema si può risolvere con la tecnica dei moltiplicatori di Lagrange in cui si introduce un moltiplicatore per ogni vincolo
- La lagrangiana del problema è:

$$L(\mathbf{w}, b, \Lambda) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^l \lambda_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$$

in cui $\Lambda = (\lambda_1, \dots, \lambda_l)$ è il vettore dei moltiplicatori di Lagrange



Lagrangiana

- La lagrangiana deve essere minimizzata rispetto a w e b e contemporaneamente massimizzata rispetto a $\Lambda \geq 0$: si cerca un punto di sella
- Differenziando rispetto a w e b si ottiene...

$$\frac{\partial L(\mathbf{w}, b, \Lambda)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^l \lambda_i y_i \mathbf{x}_i = 0$$
$$\frac{\partial L(\mathbf{w}, b, \Lambda)}{\partial b} = \sum_{i=1}^l \lambda_i y_i = 0$$
$$\mathbf{w}^* = \sum_{i=1}^l \lambda_i^* y_i \mathbf{x}_i$$



Lagrangiana

- w^* può essere scritto come una combinazione lineare dei vettori x_i del training set:

$$w^* = \sum_i \lambda_i^* y_i x_i \Rightarrow f(x) = w^* x + b = \sum_i \lambda_i^* y_i x_i \cdot x + b$$

- I dati appaiono solo all'interno di prodotti scalari
- Sostituendo il valore calcolato w^* nella lagrangiana si può riformulare il problema in una forma *duale* più semplice
- Nella formulazione duale, la funzione da ottimizzare è una funzione quadratica nei λ_i



Lagrangiana

massimizzare

$$F(\Lambda) = \sum_{i=1}^l \lambda_i - \frac{1}{2} \|\mathbf{w}^*\|^2 = \sum_{i=1}^l \lambda_i - \frac{1}{2} \sum_{i,j=1}^l \lambda_i \lambda_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

soggetta ai vincoli

$$\lambda_i \geq 0 \quad i = 1 \dots l$$

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



Lagrangiana


Nel nuovo problema i vincoli originali sono sostituiti da vincoli sui moltiplicatori ed i vettori del training set appaiono solo sotto forma di prodotti scalari

La complessità del problema duale non dipende dalla dimensione degli ingressi, ma dalla cardinalità del training set



Support vectors

- La soluzione del nuovo problema fornisce i moltiplicatori di Lagrange
- L'iperpiano ottimo si ricava...
 - ...dall'espressione di w^* , dopo la sostituzione dei moltiplicatori calcolati
 - ...mentre il valore di b^* deriva dall'applicazione delle condizioni $\lambda_i^*(y_i(w^*x + b^*) - 1) = 0$


$$b^* = y_j - w^* \cdot x_j$$

- ✦ Il classificatore è dato da

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^l y_i \lambda_i^* (\mathbf{x} \cdot \mathbf{x}_i) + b^* \right)$$

per ogni vettore x



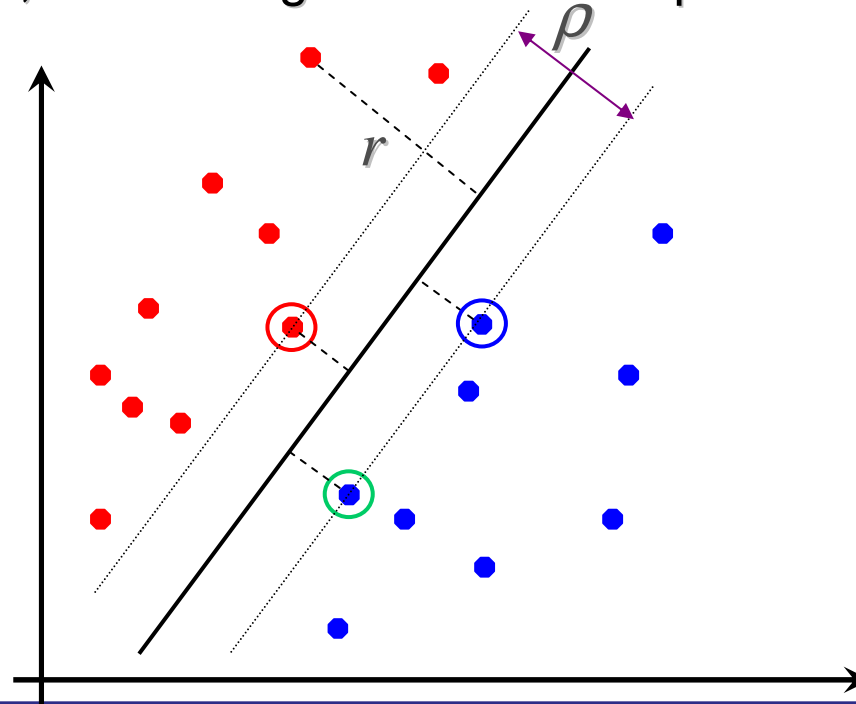
Support vectors

- Nella soluzione, tutti i punti \mathbf{x}_i per cui il corrispondente moltiplicatore λ_i è strettamente maggiore di zero vengono detti *support vector* e si trovano su uno dei due iperpiani H_1 , H_2 equidistanti dall'iperpiano ottimo e ad esso paralleli
- Tutti gli altri punti del training set hanno il corrispondente λ_i uguale a zero e non influenzano il classificatore
- I support vector sono i punti critici del training set e sono i più vicini all'iperpiano di separazione; se tutti gli altri punti venissero rimossi o spostati senza oltrepassare i piani H_1 e H_2 e l'algoritmo di apprendimento venisse ripetuto, darebbe esattamente lo stesso risultato



Da notare...

- L'iperpiano ottimo di separazione è determinato solo in base ai support vector che, in generale, sono in numero esiguo rispetto alla cardinalità l del training set
- Tutta l'informazione contenuta nel training set è concentrata nei soli vettori di supporto, che sono gli unici dati sui quali si effettua l'apprendimento

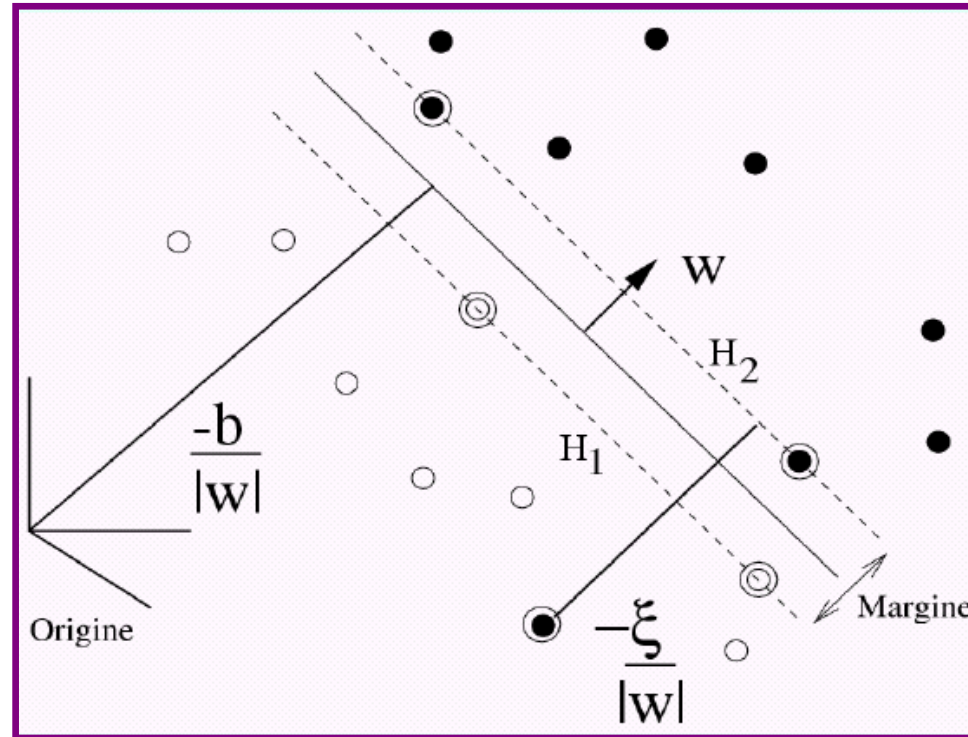


Classificatore lineare su dati non linearmente separabili

- Per come è stato definito, il classificatore a supporto vettoriale lineare non è in grado gestire casi in cui le classi non siano linearmente separabili
- Come risolvere il problema?
 - Rilassare i vincoli di corretta classificazione, tollerando un certo numero di errori
 - Realizzare altre superfici di separazione oltre l'iperpiano



Classificatore lineare su dati non linearmente separabili



- Piano separatore per un insieme di punti non linearmente separabili; il piano ha distanza $-\frac{b}{\|w\|}$ dall'origine e viene determinato dai support vector (i punti cerchiati)
- Il punto in posizione anomala dista $-\frac{\xi}{\|w\|}$ dalla sua classe



Dati non linearmente separabili

- Generalizzazione dell'iperpiano ottimo
 - ✦ Nella fase di ottimizzazione si rilassa il vincolo di esatta classificazione dei campioni del training set (*soft margin*), introducendo delle variabili *slack* ξ_i
 - ✦ I vincoli diventano:
$$w \cdot x_i + b \geq +1 - \xi_i \text{ per } y_i = 1$$
$$w \cdot x_i + b \geq -1 + \xi_i \text{ per } y_i = -1$$
con $\xi_i \geq 0$, per ogni i



Dati non linearmente separabili

- In base al valore assunto dalla corrispondente variabile di slack, i punti del training set saranno:
 - disposti al di là degli iperpiani H_1 e H_2 e correttamente classificati ($\xi_i=0$)
 - posti tra gli iperpiani H_1 e H_2 e correttamente classificati ($0<\xi_i<1$)
 - erroneamente classificati ($\xi_i>1$)



Dati non linearmente separabili

- Il problema può essere formulato come

Minimizzare

$$\Phi(\mathbf{w}, \Xi) = \frac{1}{2} \|\mathbf{w}\|^2 + C \left(\sum_{i=1}^l \xi_i \right)^k$$

con w , b e Ξ vincolati da:

$$y_i (w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad i = 1, \dots, l$$

dove C e k sono parametri che devono essere determinati a priori:
ad un alto valore di C corrisponde un'alta penalità dovuta agli errori



Dati non linearmente separabili

- Le variabili di *slack* si introducono per rilassare il vincolo di separabilità
 - $\xi_i \geq 0 \Rightarrow x_i$ ha margine $< \|w\|^{-1}$
- In pratica, l'algoritmo SVM cerca di minimizzare $\|w\|$ e di separare i punti dati commettendo il numero minimo di errori possibile
- La soluzione al problema di ottimizzazione si trova in modo analogo al caso linearmente separabile



Dati non linearmente separabili

- La lagrangiana del problema è:

$$L(\mathbf{w}, b, \Lambda, \Xi, \Gamma) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^l \lambda_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^l \gamma_i \xi_i + C \left(\sum_{i=1}^l \xi_i \right)^k$$

in cui i moltiplicatori $\Lambda = (\lambda_1, \dots, \lambda_l)$, $\Gamma = (\gamma_1, \dots, \gamma_l)$ sono associati ai vincoli

- L deve essere minimizzata rispetto a w , b e $\Xi = [\xi_1, \dots, \xi_l]$ e massimizzata rispetto a $\Lambda \geq 0$ e $\Gamma \geq 0$



Dati non linearmente separabili

- Supponendo $k=1$ per semplificare i calcoli, si arriva ad una riformulazione duale del problema simile ancora al caso linearmente separabile

$$\mathbf{max} \quad F(\Lambda) = \sum_{i=1}^l \lambda_i - \sum_{i,j=1}^l \lambda_i \lambda_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

soggetta ai vincoli: $0 \leq \lambda_i \leq C, \quad i = 1, \dots, l.$



Dati non linearmente separabili

- La soluzione del problema è identica a quella del caso separabile tranne per il vincolo sui moltiplicatori, che adesso sono limitati superiormente da C
- Il classificatore è:

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^l y_i \lambda_i^* (\mathbf{x} \cdot \mathbf{x}_i) + b^* \right)$$

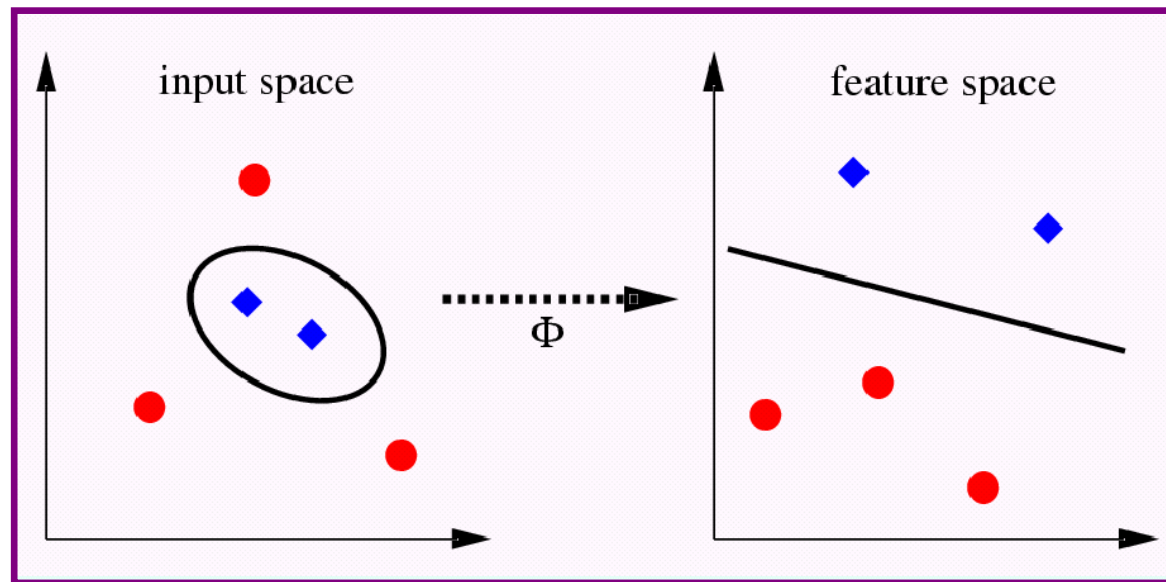


Classificatore non lineare

- Alternativamente, nel caso di dati non linearmente separabili, si introduce un mapping $\Phi(x)$ verso uno spazio di dimensione “molto più grande”, in cui gli esempi che compongono l’insieme degli ingressi siano linearmente separabili
 - ⇒ Invece di aumentare la complessità del classificatore (che resta un iperpiano) si aumenta la dimensione dello spazio delle feature



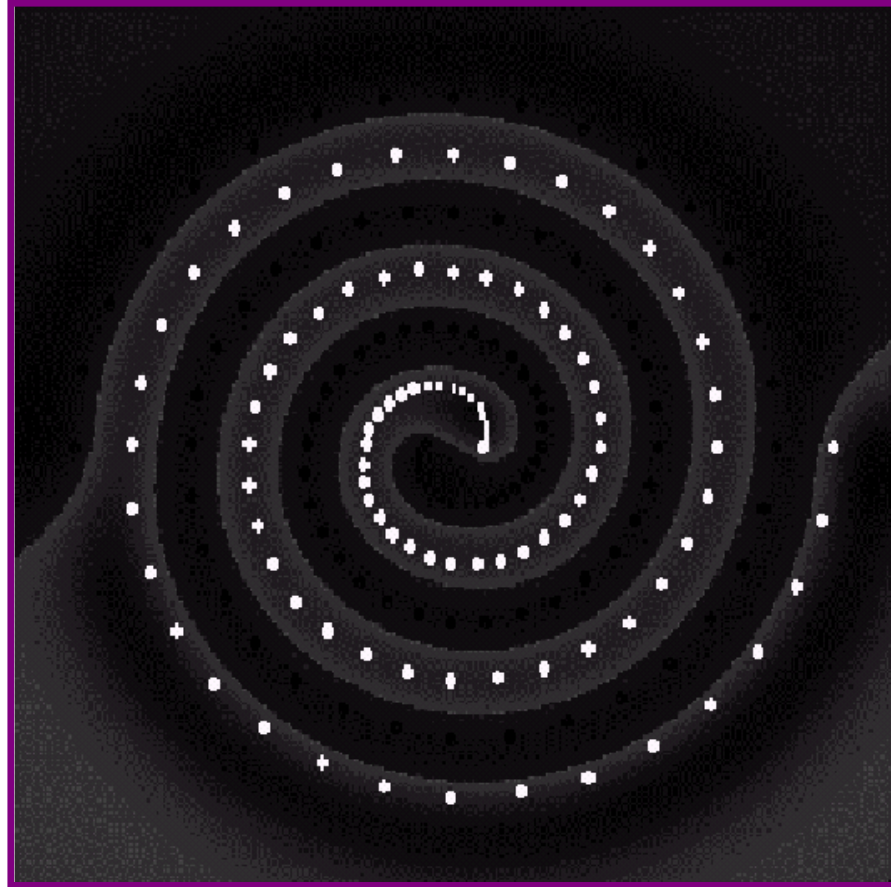
Classificatore non lineare



- Le due classi rappresentate dai cerchi e dai quadrati nello spazio di input non sono linearmente separabili, ma attraverso la funzione Φ i punti vengono mappati in uno spazio in cui divengono linearmente separabili



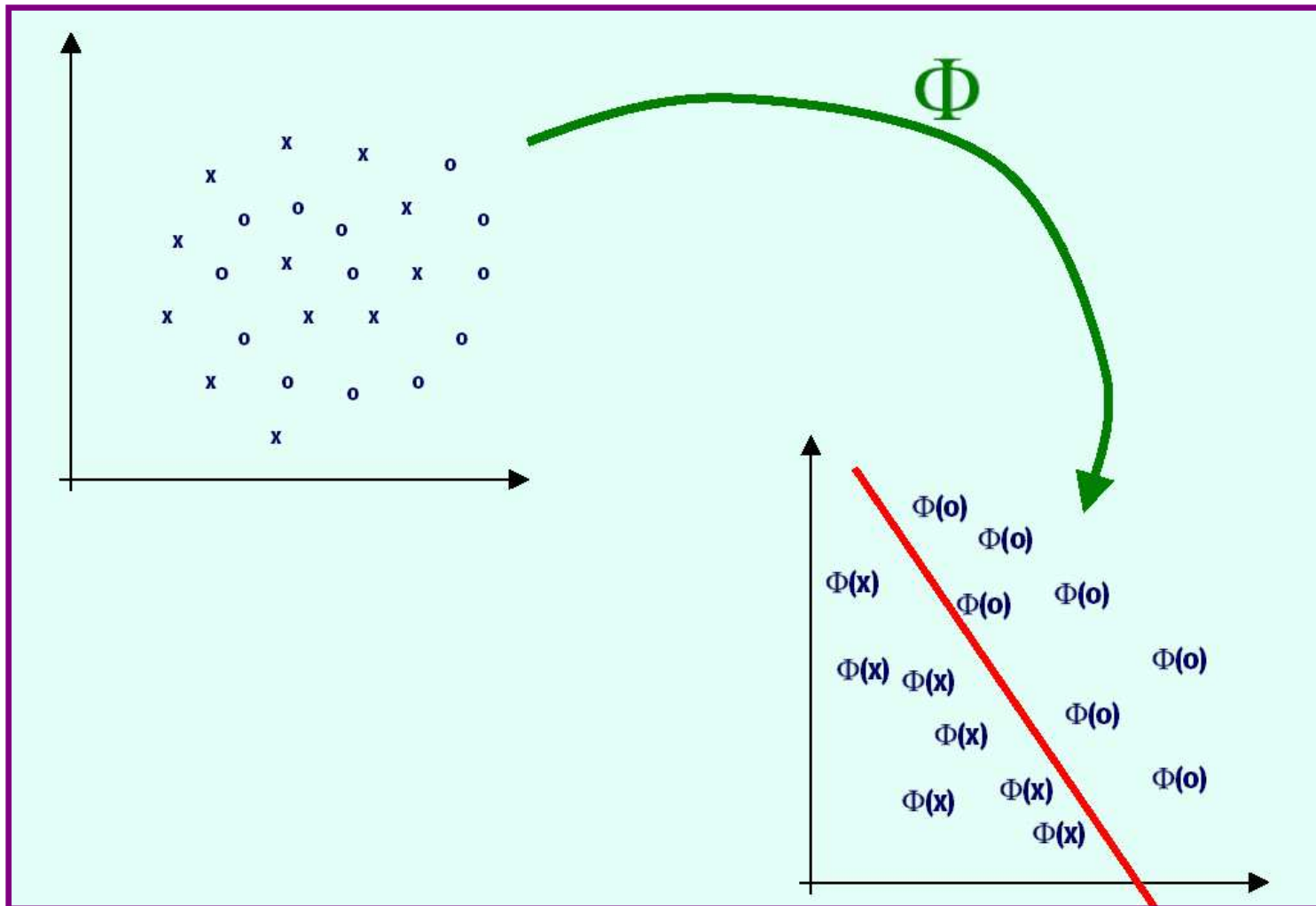
Esempio: le due spirali



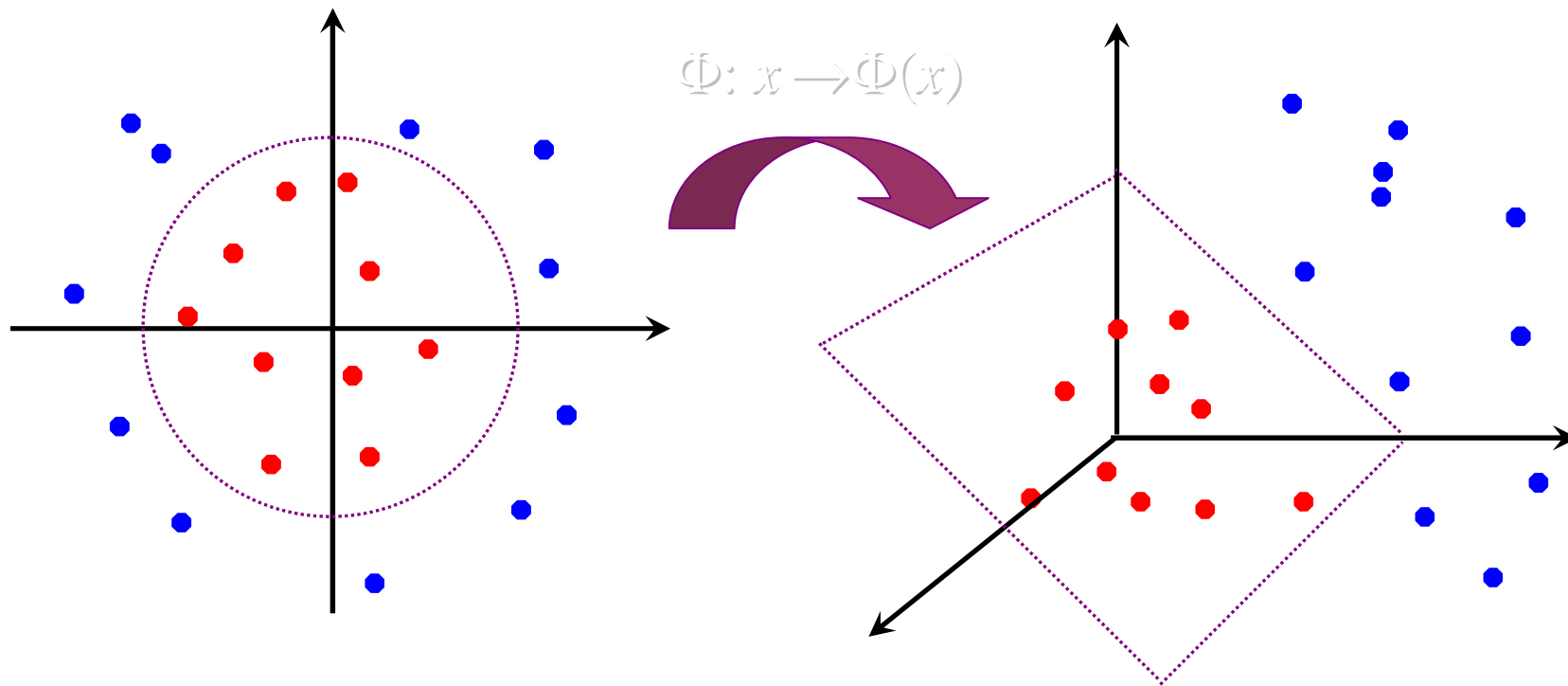
- ◆ Esiste un iperpiano nello spazio delle feature per cui i dati sono linearmente separabili



Esempio: le due spirali



Un'altra situazione possibile...



Funzioni kernel

- Supponiamo di mappare i dati iniziali non linearmente separabili in uno spazio di dimensione superiore in cui essi siano linearmente separabili, usando una funzione di mapping $\Phi: \mathcal{X}^d \rightarrow H$
- Uno spazio di dimensione maggiore causa però seri problemi di calcolo, perché l'algoritmo di apprendimento deve lavorare con vettori di grandi dimensioni (con molte componenti)
- Tuttavia, in questa situazione, l'algoritmo di apprendimento dipende dai dati solamente tramite il prodotto delle loro immagini attraverso Φ in H , cioè tramite funzioni della forma $\Phi(x_i) \cdot \Phi(x_j)$



Kernel

- Per ovviare al problema dell'aumento della dimensione dei vettori di feature si introduce quindi una *funzione kernel* che restituisce il prodotto delle immagini, attraverso Φ , dei suoi due argomenti:

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

⇒ è possibile evitare di eseguire il prodotto esplicito tra le immagini dei vettori, è sufficiente conoscerne la “forma funzionale”

- Pertanto è possibile sostituire K all'interno dell'algoritmo e ignorare la forma esplicita di Φ



Quali funzioni sono kernel ?

- Per alcune funzioni $K(x_i, x_j)$ verificare che $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$ è complesso
- Teorema di Mercer
Ogni funzione simmetrica semi-definita positiva è un kernel
- L'impiego di funzioni kernel di fatto “nasconde” il mapping nello spazio multi-dimensionale



Matrice kernel

- È una rappresentazione matriciale della funzione kernel (argomenti: gli m vettori di apprendimento):

K =	K(1,1)	K(2,1)	K(3,1)		K(m,1)
	K(1,2)	K(2,2)	K(3,2)		K(m,2)
	K(1,m)	K(2,m)	K(3,m)		K(m,m)

- ◆ Contiene tutte le informazioni necessarie per l'apprendimento
- ◆ Riassume informazioni sui dati e sul kernel
- ◆ È simmetrica e semi-definita positiva

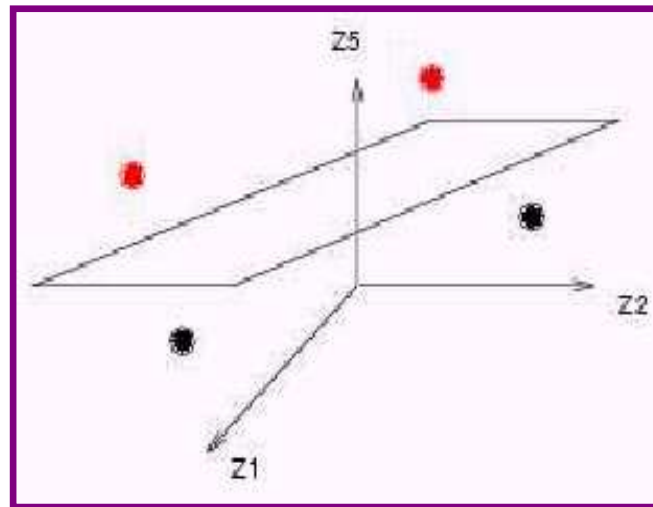
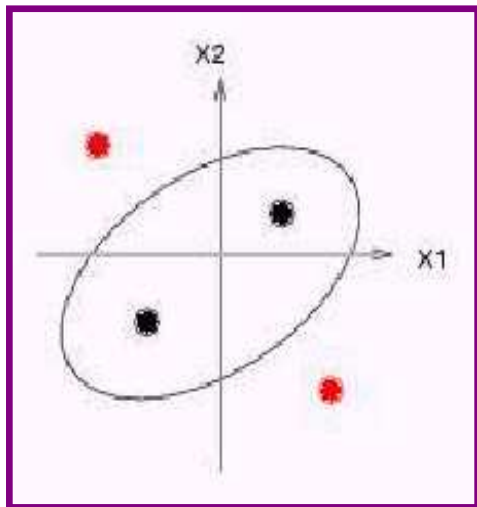
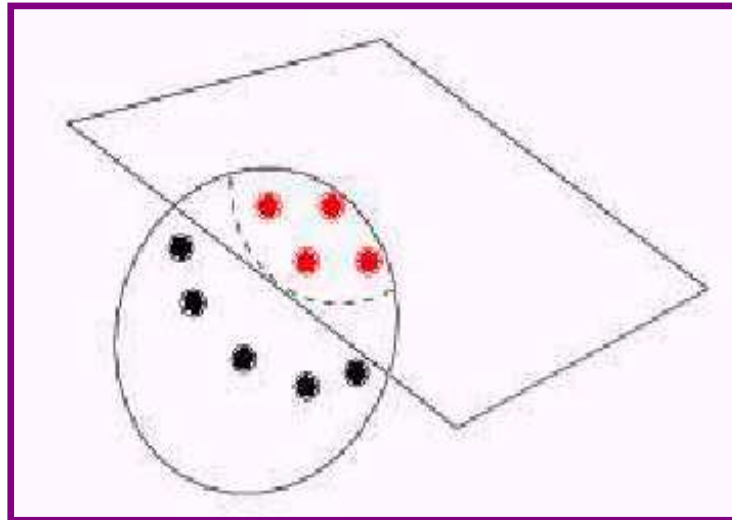
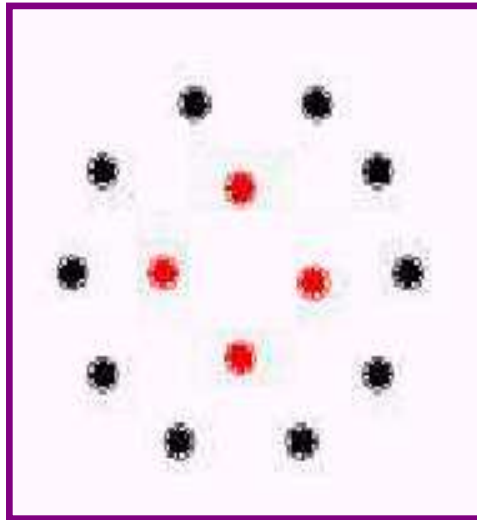


Ancora sulla lineare separabilità...

- Sostituendo $\Phi(x_i) \cdot \Phi(x_j)$ con $K(x_i, x_j)$ nell'algoritmo, si genera una Support Vector Machine che "lavora" in H e fornisce il risultato nella stessa quantità di tempo che impiegherebbe se lavorasse con i dati originali non trasformati
- Tutte le considerazioni fatte nel caso linearmente separabile restano valide, poiché si sta costruendo un classificatore lineare in uno spazio differente
- In pratica, l'estensione a superfici di decisione complesse avviene in maniera semplice: mappando la variabile in input x in uno spazio di dimensione maggiore e lavorando poi con una classificazione lineare nel nuovo spazio



Feature expansion



Classificatore non lineare

- Un punto x viene mappato in un vettore di “feature” tramite la funzione $\Phi: x \rightarrow \Phi(x) = (a_1\Phi_1(x), a_2\Phi_2(x), \dots)$ dove gli a_i sono numeri reali e le Φ_i sono funzioni reali
- Quindi, si applica lo stesso algoritmo del caso linearmente separabile sostituendo la variabile x con un nuovo vettore di feature $\Phi(x)$



Classificatore non lineare

- La funzione di decisione con il mapping diventa quindi:

$$f(\mathbf{x}) = \text{sign}\left(\Phi(\mathbf{x}) \cdot \mathbf{w}^* + b^*\right) = \text{sign}\left(\sum_{i=1}^l y_i \lambda_i^* \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i) + b^*\right)$$

Sostituendo i prodotti scalari $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ con una funzione kernel:

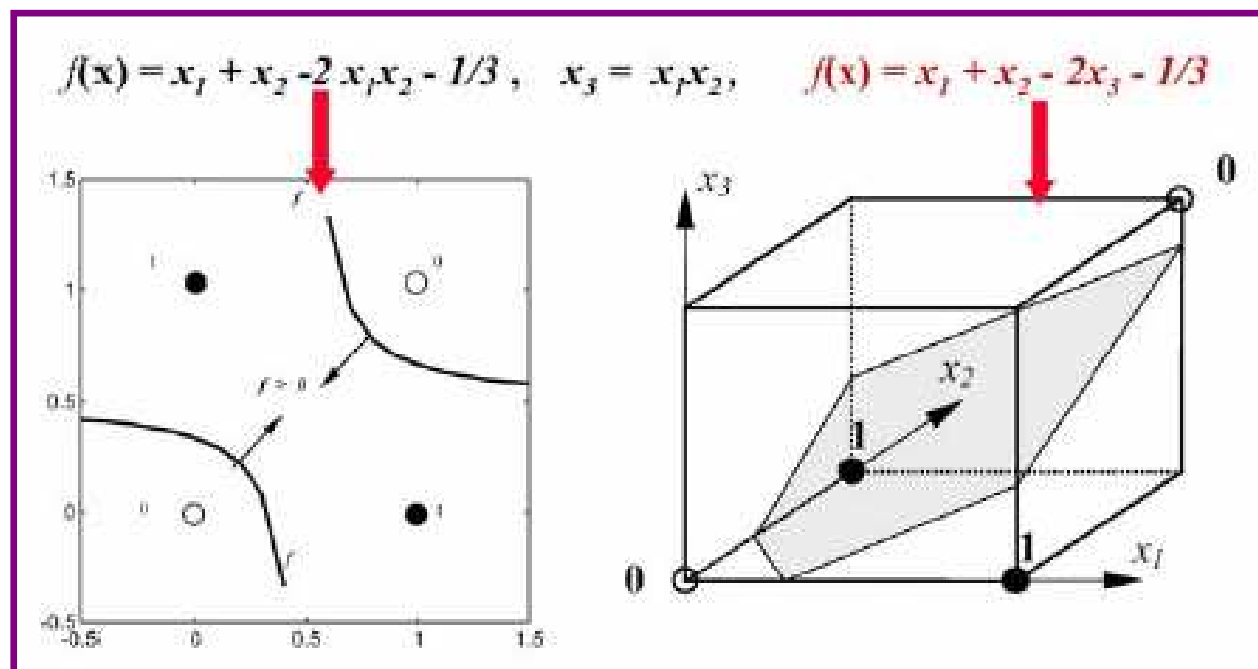
$$K(\mathbf{x}, \mathbf{y}) \equiv \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) = \sum_{n=1}^{\infty} a_n^2 \Phi_n(\mathbf{x}) \cdot \Phi_n(\mathbf{y})$$

si ottiene

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^l y_i \lambda_i^* K(\mathbf{x}, \mathbf{x}_i) + b^*\right)$$



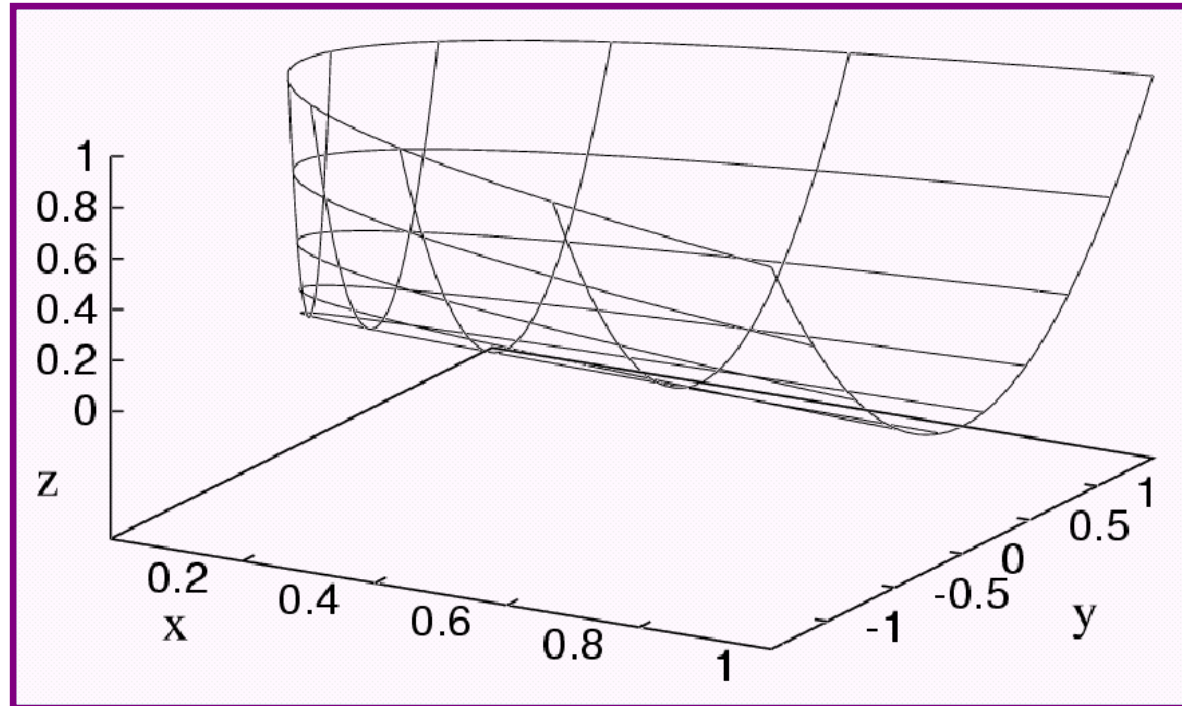
Esempio



- L'XOR diventa linearmente separabile per $\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$, con $\Phi(x_1, x_2) = (x_1, x_2, x_1 \times x_2)$



Esempio



- L'immagine di Φ può esistere in uno spazio anche di dimensione infinita, ma è solo una superficie, anche se molto “contorta”, la cui dimensione intrinseca, il numero di parametri necessari per specificare un punto, è la stessa dello spazio dei vettori x



Tipi di kernel

- La funzione kernel va scelta accuratamente per il particolare tipo di problema: è sempre possibile trasformare l'input in uno spazio di dimensione maggiore del numero di punti del training set e produrre un classificatore perfetto...
- ...Tuttavia, questi generalizzerebbe malissimo su dati nuovi, a causa dell'overfitting



Tipi di kernel

- Tipi di kernel comunemente usati sono:

Lineare $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$

Polinomiale $K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^d$

Radial Basis function $K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$

Gaussian Radial Basis function $K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{(\mathbf{x} - \mathbf{y})^2}{2\sigma^2}\right)$

Multi-Layer Perceptron $K(\mathbf{x}, \mathbf{y}) = \tanh(b(\mathbf{x} \cdot \mathbf{y}) - c)$



Modularità

- Una SVM è composta da due moduli:
 - ✦ Un modulo generale di apprendimento
 - ✦ Una funzione kernel specifica per il problema da affrontare
- Qualsiasi SVM può operare con qualsiasi kernel
- Il problema di apprendere un compito difficile si sposta nel problema di scegliere opportunamente una funzione kernel (di nuovo un problema difficile...) per trasformarlo in un compito facile (perché linearmente separabile)



Come scegliere un classificatore SVM ?

- Per impiegare una SVM è allora necessario definire:
 - ✧ tipo di kernel da impiegare
 - ✧ parametri del particolare kernel
 - ✧ valore di C
- Non esistono criteri teorici per queste scelte; tipicamente occorre una verifica su un insieme di validazione



Classificazione multiclasse

- Le SVM si applicano solo a problemi di classificazione binaria (cioè sono capaci di predire il valore di una variabile booleana)
- Come fare per problemi multiclasse ad N classi?
 - Si addestrano N support vector machine
 - ✦ SVM 1 apprende “Output==1” vs “Output != 1”
 - ✦ ...
 - ✦ SVM N apprende “Output== N ” vs “Output != N ”
- Per predire l’output relativo ad un nuovo input, si sceglie quella SVM che mappa il pattern nel semipiano positivo, con massimo margine rispetto all’iperpiano separatore



Conclusioni

- Il perceptron è lo strumento più semplice per costruire un classificatore lineare
 - ✦ Problemi: uso inefficiente dei dati, overfitting, mancanza di espressività
- Le SVM risolvono questi problemi, mediante l'utilizzo di margini e di tecniche di *feature expansion*
 - ✦ Per realizzare l'espansione dello spazio di input con un carico computazionale accettabile si ricorre al trucco della definizione dei kernel
 - ✦ La presenza del kernel evita calcoli onerosi di prodotti scalari fra vettori ad alta dimensionalità



Conclusioni

- Le SVM sono diventate popolari perché capaci di apprendere dataset “difficili”, con buone prestazioni in generalizzazione
- Le SVM sono attualmente fra i migliori classificatori in una varietà di problemi (es. classificazione di testi e genomica)
- Il tuning dei parametri nelle SVM è un’*arte*: la selezione di uno specifico kernel e dei relativi parametri viene eseguita in modo empirico (*trial-and-error*)

