

Introduzione alle Support Vector Machines per la classificazione

Manuele Bicego

Dipartimento di Informatica

Università di Verona

bicego@sci.univr.it

29 agosto 2001

Sommario

Le Support Vector Machines rappresentano un tool di classificazione e regressione molto utilizzato negli ultimi anni. La loro nascita, da un punto di vista teorico, risale alla fine degli anni 70 (Vapnik 79), ma solo nell'ultima decade sono state massicciamente utilizzate in una svariata serie di applicazioni. Una lista non esaustiva comprende il riconoscimento di caratteri scritti a mano, il riconoscimento di oggetti, l'identificazione dello speaker in problemi di speech, i problemi di face detection e face recognition etc. I risultati ottenuti sono sempre stati altamente competitivi, se confrontati con le tecniche allo stato dell'arte. In questo lavoro vengono riassunte le teorie dell'Empirical Risk Minimization e della Structural Risk Minimization, in quanto basi per le support vector machines. Vengono poi presentate le support vector machines lineari e non lineari, mettendone in luce pregi e difetti.

1 Il problema della Classificazione

Dato un insieme di osservazioni etichettate $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ derivanti da una distribuzione $P(\mathbf{X}, Y)$, fissata ma sconosciuta, l'obiettivo della classificazione è quello di inferire la relazione che esiste tra \mathbf{X} e Y sulla base dell'insieme \mathcal{D} , detto *training set*. Occorre, in altre parole, essere in grado di predire, per ogni $\mathbf{x} \in \mathbf{X}$, la corrispondente etichetta y . Per fare questo, tipicamente, si cerca di trovare una funzione $f(\mathbf{x})$ che minimizzi una certa funzione di costo.

Uno dei principali obiettivi della Classificazione è quello di ottenere un classificatore in grado di "generalizzare" bene: per generalizzazione si intende la capacità di una macchina di riconoscere e classificare correttamente anche esempi non presenti nel training set. L'obiettivo è quindi quello di inferire dal training set il processo che ha generato tali punti, i.e. la $P(\mathbf{x}, y)$.

Per far questo occorre trovare il giusto equilibrio tra la capacità della macchina e lo smoothness dell'approssimazione:

- *capacità della macchina*: è definita come l'abilità di una funzione di imparare correttamente ("memorizzare") un qualsiasi training set, (i.e. l'abilità di non commettere nessun errore su un qualsiasi training set di dimensione data): questa caratteristica è legata alla *VC-dimension*;
- *smoothness dell'approssimazione*: non seguire troppo fedelmente i punti da approssimare, ma cercare di evincere l'andamento generale.

Ad esempio, una macchina con troppa capacità è come un botanico che afferma: "Questo non è un albero, perché ha un numero di foglie diverso da quello di tutti gli alberi che ho visto finora". Una macchina con troppa poca capacità, invece afferma: "Questo è un albero perché è verde!".

I teorici della Pattern Recognition (Vapnik, Györfi, Devroye, Burges, per citarne alcuni) lavorano da anni alla ricerca di regole che mettano in relazione la capacità di una macchina con le sue prestazioni, determinando limiti di convergenza asintotici o meno. Nel seguito del lavoro verranno introdotte due teorie: l'*Empirical Risk Minimization*, semplice e adatta ad introdurre alcuni concetti chiave, quali la *VC-dimension*, e la *Structural Risk Minimization*, nel cui contesto possono essere inserite le Support Vector Machines.

2 Empirical Risk Minimization

Siano date ℓ osservazioni etichettate $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, dove $\mathbf{x}_i \in \mathbb{R}^N$, $y_i \in \{-1, 1\}$, $i = 1 \dots \ell$ provenienti da una $P(\mathbf{X}, Y)$ fissata ma sconosciuta (i.e. \mathbf{x}_i sono i.i.d.). L'obiettivo risulta essere quello di inferire la relazione tra \mathbf{X} e Y dall'insieme \mathcal{D} ; questo significa essere in grado, per ogni $\mathbf{x} \in \mathbf{X}$, di predire il valore y , effettuando il minimo numero di errori. In altre parole questo si traduce nella determinazione di una funzione $f(\mathbf{x})$ che massimizzi una qualche funzione di costo (o di rischio).

Definizione 1 Dato un certo problema, il *Rischio Atteso (Expected Risk)* associato ad una funzione $f(\mathbf{x})$ è calcolato come:

$$I[f] = \int_{\mathbf{X} \times Y} V(y - f(\mathbf{x})) dP(\mathbf{X}, Y)$$

o, se esiste la funzione densità di probabilità $p(\mathbf{x}, y)$, equivalentemente

$$I[f] = \int_{\mathbf{X} \times Y} V(y - f(\mathbf{x})) p(\mathbf{x}, y) d\mathbf{x} dy$$

V è detta *loss function*, e può essere rappresentata da una qualsiasi funzione monotona crescente.

Esempi:
$$\begin{aligned} V(\bullet) &= |\bullet| \\ V(\bullet) &= (\bullet)^2 \end{aligned} \quad \longleftarrow \quad \text{in questo lavoro}$$

Il Rischio Atteso può essere riscritto come

$$\begin{aligned} I[f] &= \int_{\mathbf{X}} (f_0(\mathbf{x}) - f(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x} + \\ &+ \int_{\mathbf{X} \times Y} (y - f_0(\mathbf{x}))^2 p(\mathbf{x}, y) d\mathbf{x} dy \end{aligned}$$

dove

$$f_0(\mathbf{x}) = \int_Y yp(y|\mathbf{x})dy$$

rappresenta il rischio di Bayes. La funzione $f_0(\mathbf{x})$ rappresenta l'ottimo teorico, i.e. quello che minimizza il rischio atteso: è calcolabile solo se è nota la probabilità a posteriori $p(y|\mathbf{x})$, cosa che raramente avviene.

L'obiettivo del learning diventa ora trovare la funzione f_{min} tale che

$$f_{min}(\mathbf{x}) = \arg \min_{f \in \mathcal{H}} I[f]$$

\mathcal{H} è chiamato *Spazio delle Ipotesi*. Gli elementi di \mathcal{H} sono funzioni della forma $f = f(\mathbf{x}, \alpha)$, dove α rappresenta un vettore di parametri. Una volta fissato α la funzione è determinata. Possiamo rappresentare in modo esplicito questa dipendenza di f da α , riscrivendo il rischio atteso come:

$$I[\alpha] = \int_{\mathbf{x} \times Y} (y - f(\mathbf{x}, \alpha))^2 p(\mathbf{x}, y) d\mathbf{x} dy$$

L'obiettivo del learning diventa quindi trovare

$$\alpha_{min} = \arg \min_{\alpha \in \Lambda} I[\alpha]$$

Il problema maggiore consiste nel fatto che la funzione densità di probabilità $p(\mathbf{x}, y)$ è sconosciuta, e quindi $I[\alpha]$ è sconosciuto. Una soluzione consiste nello stimare il rischio atteso con il *rischio empirico*, definito come:

$$I_{emp}[\alpha] = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - f(\mathbf{x}_i))^2$$

Definizione 2 *Il principio Empirical Risk Minimization (ERM) risolve il problema dell'apprendimento da un insieme di esempi \mathcal{D} scegliendo la funzione f_{ℓ} che minimizza il rischio empirico, i.e.*

$$f_{\ell} = \arg \min_{f \in \mathcal{H}} I_{emp}[f]$$

o, esprimendo il tutto in funzione dei parametri α , trovando i parametri α_{ℓ} t.c.

$$\alpha_{\ell} = \arg \min_{\alpha \in \Lambda} I_{emp}[\alpha]$$

2.1 Consistenza del principio ERM

Per la legge dei Grandi Numeri vale che, per ogni funzione f

$$\lim_{\ell \rightarrow \infty} I_{emp}[f] = I[f]$$

Quindi, avendo a disposizione un numero grande di esempi, la stima del rischio atteso risulta essere corretta. Non è comunque detto che sia anche

$$\lim_{\ell \rightarrow \infty} \min_{f \in \mathcal{H}} I_{emp}[f] = \min_{f \in \mathcal{H}} I[f]$$

cioè non è ovvio che il minimo del rischio empirico corrisponda al minimo del rischio atteso: questa è una condizione più forte. Questo però risulta essere vero, scegliendo lo spazio delle ipotesi \mathcal{H} in modo che non sia “troppo grande” (vedremo meglio il concetto tra un attimo).

Definizione 3 Il principio ERM è consistente rispetto ad una famiglia di funzioni $\mathcal{H} = \{f(\mathbf{x}, \alpha)\}$ se valgono le seguenti due condizioni:

$$\begin{aligned} 1) \quad & \lim_{\ell \rightarrow \infty} I_{emp}[\alpha_\ell] \stackrel{P}{=} \inf_{\alpha \in \Lambda} I[\alpha] \\ 2) \quad & \lim_{\ell \rightarrow \infty} I[\alpha_\ell] \stackrel{P}{=} \inf_{\alpha \in \Lambda} I[\alpha] \end{aligned}$$

dove $\stackrel{P}{=}$ rappresenta la convergenza in probabilità¹

La prima condizione richiede che il minimo del rischio atteso sia uguale in probabilità al minimo del rischio empirico. La seconda richiede invece che α_ℓ (che minimizza il rischio empirico) e α_{min} che minimizza il rischio atteso producano lo stesso valore di minimo nel rischio atteso. Questa condizione risulta essere una condizione più leggera rispetto a quella di imporre che

$$\lim_{\ell \rightarrow \infty} \alpha_\ell = \alpha_{min}$$

La consistenza del principio ERM è garantita da questo teorema

Teorema 1 Condizione necessaria e sufficiente affinché la stima ERM, effettuata in uno spazio delle ipotesi $\mathcal{H} = \{f(\alpha, \mathbf{x})\}$, sia consistente è che la VC-dimension di questo spazio \mathcal{H} sia finita.

2.2 La VC-dimension

Definizione 4 Dato un insieme di funzioni $\mathcal{H} = \{f(\alpha, \mathbf{x})\}$, la Vapnik Chervonenkis (VC) dimension è definita come il numero h tale che:

- Esiste almeno un insieme di h punti che possono essere etichettati in qualsiasi maniera da funzioni di \mathcal{H} : in altre parole qualsiasi sottoinsieme di questi punti può essere correttamente classificato da una $f(\alpha, \mathbf{x})$; detto in un altro modo ancora, per ogni scelta delle etichette di questi h punti, può essere trovata una funzione in \mathcal{H} che assegni correttamente tali etichette ai punti²;
- Non esiste nessun insieme di $h+1$ punti che possono essere etichettati in qualsiasi modo.

Un'osservazione: la prima condizione afferma che la condizione deve valere per *almeno* un insieme di cardinalità h . Se la VC-dimension è h , generalmente la condizione non vale per *tutti* gli insiemi di cardinalità h .

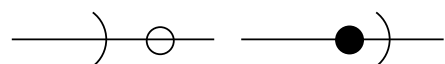
Esempi. Consideriamo esempi nello spazio monodimensionale: i pallini pieni rappresentano il sottoinsieme da caratterizzare. Per ogni insieme di punti, vengono considerate tutte le combinazioni, i.e. tutti i possibili sottoinsiemi.

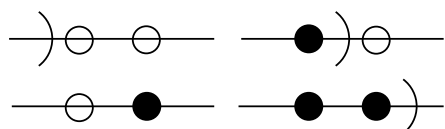
¹Una sequenza di funzioni $f_n(x)$, $x \in X$, converge in probabilità a f , si scrive $f_n \stackrel{P}{\rightarrow} f$, se:

$$\lim_{n \rightarrow \infty} P\{x : |f_n(x) - f(x)| > \delta\} = 0, \quad \forall \delta > 0$$

²In questo caso si dice che la famiglia \mathcal{H} fa uno *shattering* dell'insieme

1. La famiglia di funzioni \mathcal{H} è rappresentata dagli intervalli $(-\infty, a)$

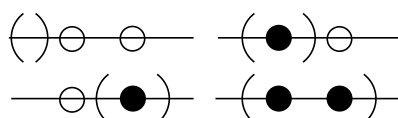
 1 punto: OK

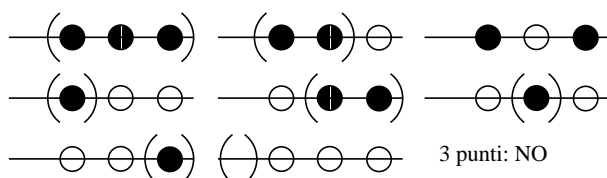
 2 punti: NO

La VC-dimension è 1;

2. La famiglia di funzioni \mathcal{H} è rappresentata dagli intervalli chiusi (a, b)

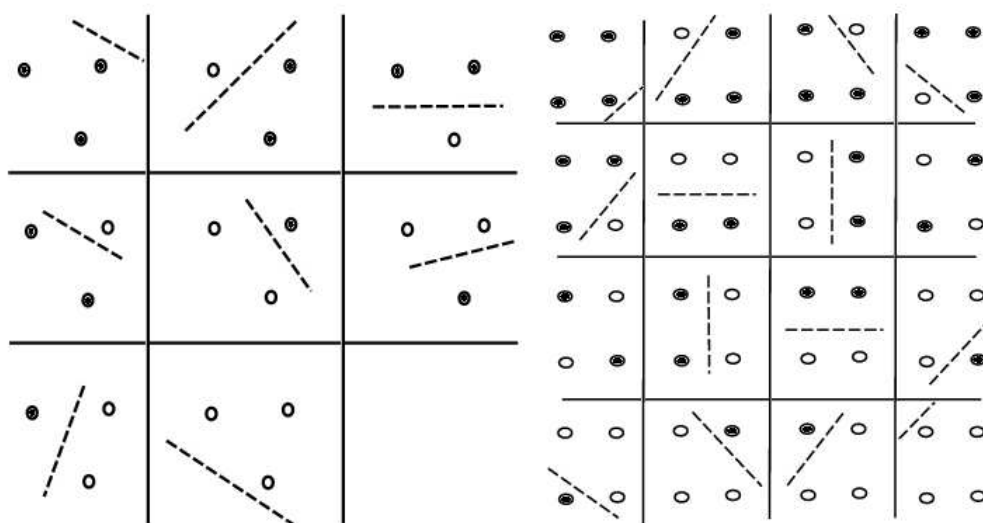
 1 punto: OK

 2 punti: OK

 3 punti: NO

La VC-dimension è 2;

3. Nello spazio bidimensionale, consideriamo la famiglia dei semipiani;



La VC-dimension è 3. Più generalmente, la VC-dimension dell'insieme degli iperpiani orientati in \mathbb{R}^n è $n + 1$.

La VC-dimension misura la ricchezza e la flessibilità di una famiglia di funzioni, e può essere considerata come una misura della capacità di una classe di classificatori. Maggiore è la VC-dimension, maggiore sarà l'abilità della macchina nel classificare correttamente un training set di dimensione fissata.

3 Structural Risk Minimization

Vapnik ha dimostrato che vale la seguente disuguaglianza:

Teorema 2 (VC bound) *Per ogni $\alpha \in \Lambda$, con probabilità $1 - \delta$ vale che:*

$$I[\alpha] \leq I_{emp}[\alpha] + \sqrt{\frac{h \left(\ln \frac{2\ell}{h} + 1 \right) - \ln \left(\frac{\delta}{4} \right)}{\ell}} \quad (1)$$

dove ℓ è il numero di osservazioni, mentre h è la VC-dimension

Osservazioni:

1. Il bound (1) non dipende da $P(\mathbf{X}, Y)$;
2. Non è un limite asintotico, risulta essere sempre calcolabile se la VC-dimension h è finita;
3. Il bound della differenza tra il rischio atteso e il rischio empirico diminuisce al crescere del numero di campioni ℓ , ma cresce al crescere della VC-dimension;
4. A parità di rischio empirico I_{emp} , scegliendo una funzione da una classe che ha una VC-dimension minore ottengo un bound minore.

Nota Importante: Un bound minore non implica necessariamente delle prestazioni migliori.

L'idea della *Structural Risk Minimization* è proprio quella di trovare la funzione che minimizza la (1): occorre quindi considerare la VC-dimension come variabile del problema della minimizzazione. Definiamo il rischio garantito come:

$$\text{Guaranteed Risk} = I_{emp}[\alpha] + \sqrt{\frac{h \left(\ln \frac{2\ell}{h} + 1 \right) - \ln \left(\frac{\delta}{4} \right)}{\ell}} \quad (2)$$

Il secondo addendo viene anche definito confidenza del bound.

Definizione 5 *Siano $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_h, \dots$ una serie di spazi di ipotesi con VC-dimension pari a $1, 2, \dots, h$, tali che:*

$$\mathcal{S}_1 \subset \mathcal{S}_2 \subset \dots \subset \mathcal{S}_h \subset \dots$$

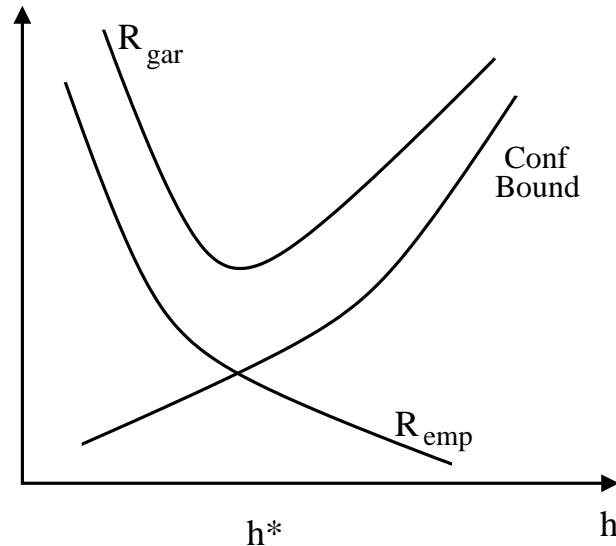
Il principio *Structural Risk Minimization (SRM)* risolve il problema dell'apprendimento da un insieme di esempi \mathcal{D} scegliendo la funzione che minimizza il rischio garantito, i.e.

$$\min_{\mathcal{S}_h} \left[I_{emp}[\alpha] + \sqrt{\frac{h \left(\ln \frac{2\ell}{h} + 1 \right) - \ln \left(\frac{\delta}{4} \right)}{\ell}} \right]$$

Esempi:

- $\mathcal{S}_h \equiv \{ \text{Multi Layer Perceptron network con } h \text{ neuroni nello strato nascosto} \}$
- $\mathcal{S}_h \equiv \{ \text{Polinomi di grado } h \}$

L'idea che sta alla base della SRM è quindi quella di “controllare” la capacità della famiglia di classificatori, gestendo la VC-dimension, in modo da migliorare l'accuratezza di generalizzazione. In pratica, si parte utilizzando una macchina con capacità bassa, che produrrà una stima grossolana dei dati, con un corrispondente rischio empirico I_{emp} elevato. Aumentando lentamente la capacità, si avrà che diminuirà il rischio empirico, ma aumenterà il secondo termine della disuguaglianza (1): la diminuzione del rischio empirico costa un certo fattore, in termini di rischio garantito. Continuando questo processo di crescita della capacità, si arriverà ad un livello in cui il miglioramento del rischio empirico introdotto sarà minore del costo pagato. A questo punto ci si ferma. L'idea è rappresentata in figura: “ R_{emp} ” è il rischio empirico, “Conf bound” è la confidenza del bound, mentre “ R_{gar} ” rappresenta il rischio garantito, ovvero la somma delle due curve. h^* rappresenta il modello migliore.



Il problema maggiore è che questo è comunque un procedimento molto lungo. Le support vector machines possono essere considerate come un'approssimazione di questo metodo, come vedremo nel seguito.

4 Support Vector Machines lineari

Cominciamo dal caso più semplice: le Support Vector Machines producono una superficie di separazione che è un piano.

4.1 Dati linearmente separabili

Definizione 6 Sia dato un insieme di osservazioni etichettate $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, $i = 1 \dots \ell$, $y_i \in \{-1, 1\}$, $\mathbf{x}_i \in \mathbb{R}^d$. \mathcal{D} è linearmente separabile se, per qualche $\mathbf{w} \in \mathbb{R}^d$ e $b \in \mathbb{R}$, vale che

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \text{for } y_i = +1 \quad (3)$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{for } y_i = -1 \quad (4)$$

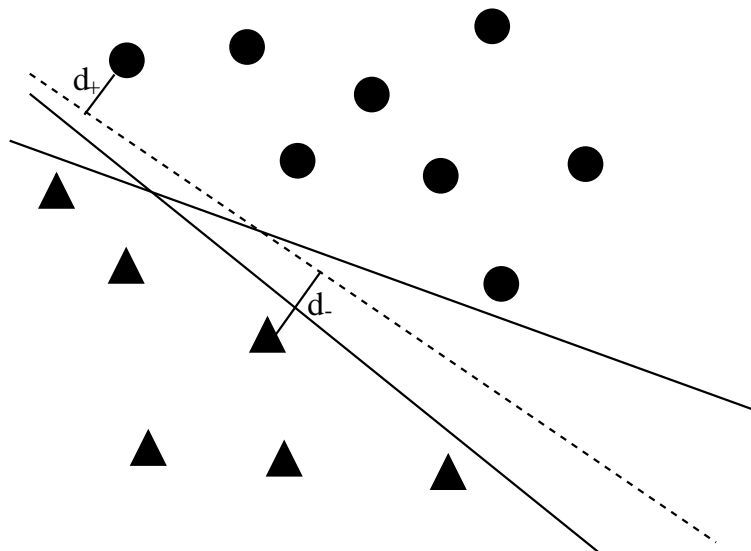
o, equivalentemente,

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \text{for } i = 1, 2, \dots, \ell \quad (5)$$

L'iperpiano $\mathbf{w} \cdot \mathbf{x} + b = 0$ è chiamato piano separatore:

- \mathbf{w} è la normale all'iperpiano;
- $\frac{|b|}{\|\mathbf{w}\|}$ è la distanza ortogonale del piano dall'origine;
- $\|\mathbf{w}\|$ rappresenta la norma euclidea di \mathbf{w} .

Definiamo d_+ (d_-) come la minima distanza del piano separatore dal più vicino punto positivo (negativo). Definiamo il *margin* di un piano separatore come $d_+ + d_-$. Nel caso di punti linearmente separabili, esistono molti piani separatori (linee piene nella figura sottostante): tutti questi piani classificano correttamente il training set; da un punto di vista della capacità di generalizzazione essi sono però diversi. Le Support Vector Machines cercano un iperpiano che non sia solo corretto sul training set, ma che riesca anche a generalizzare bene. Il criterio scelto consiste nel trovare l'iperpiano che massimizza il margine (linea tratteggiata in figura).

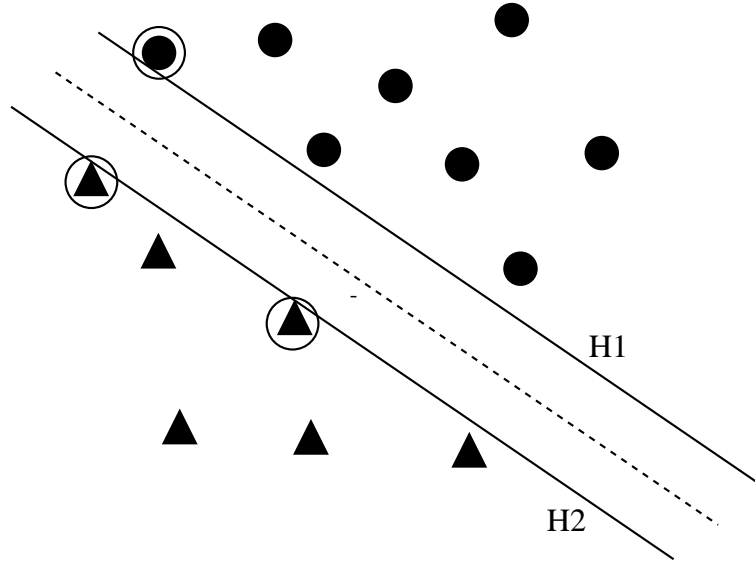


Consideriamo i punti per cui in (3) vale l'uguaglianza. Questi punti stanno sull'iperpiano $H_1 : \mathbf{x}_i \cdot \mathbf{w} + b = +1$, con normale \mathbf{w} e distanza dall'origine $\frac{|1-b|}{\|\mathbf{w}\|}$. In modo analogo, i punti per cui in (4) vale l'uguaglianza stanno nell'iperpiano $H_2 : \mathbf{x}_i \cdot \mathbf{w} + b = -1$, che dista dall'origine $\frac{|-1-b|}{\|\mathbf{w}\|}$. In questo caso $d_+ = d_- = \frac{1}{\|\mathbf{w}\|}$, e quindi il margine è $\frac{2}{\|\mathbf{w}\|}$. Quindi si può massimizzare il margine minimizzando $\|\mathbf{w}\|^2$, soggetto al vincolo (5).

Osservazioni:

- H_1 e H_2 sono paralleli;
- nessun punto del training set è tra di loro
- per avere il massimo margine occorre trovare la coppia di H_1, H_2 che massimizza $\|\mathbf{w}\|^2$, soggetti al vincolo (5).

I punti per cui nella (5) vale l'uguaglianza sono detti *support vectors*, e sono cerchiati in figura. Essi sono gli unici punti del training set utilizzati per la determinazione dell'iperpiano ottimale, e la loro rimozione provoca cambiamenti nel risultato.



Matematicamente occorre risolvere il seguente problema **P1**

$$\min \frac{1}{2} \mathbf{w} \cdot \mathbf{w} \quad \text{con il vincolo } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \text{for } i = 1, 2, \dots, \ell$$

Applicando la tecnica dei moltiplicatori di Lagrange, la soluzione del problema **P1** è equivalente alla determinazione del punto di sella della funzione

$$L = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^{\ell} \alpha_i \{y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\}$$

dove gli α_i sono i moltiplicatori di Lagrange. Devo quindi minimizzare rispetto a \mathbf{w} e b e massimizzare rispetto a α_i , i.e. devo trovare un punto di sella della funzione L . Consideriamo la funzione duale

$$\Theta(\alpha) = \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$$

Ora, nel punto di sella avremo che:

$$\begin{aligned}\frac{\partial L}{\partial b} &= \sum_{i=1}^{\ell} y_i \alpha_i = 0 \\ \frac{\partial L}{\partial \bar{\mathbf{w}}} &= \mathbf{w} - \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i = 0\end{aligned}$$

Sostituendo queste uguaglianze nella funzione duale $\Theta(\alpha) = L(\bar{\mathbf{w}}, \bar{b})$ ottengo il problema duale **P2**, che rappresenta quello che poi viene risolto al calcolatore:

$$\max -\frac{1}{2} \alpha^T B \alpha + \sum_{i=1}^{\ell} \alpha_i \quad \text{con i vincoli} \quad \sum_{i=1}^{\ell} y_i \alpha_i = 0 \text{ e } \alpha_i \geq 0 \quad (6)$$

dove B è una matrice $\ell \times \ell$ definita come

$$B_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

La (6) può essere anche riscritta come:

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad \text{con i vincoli} \quad \sum_{i=1}^{\ell} y_i \alpha_i = 0 \text{ e } \alpha \geq 0$$

Una volta trovato il minimo $\bar{\alpha}$, l'iperpiano ottimale viene determinato con

$$\bar{\mathbf{w}} = \sum_{i=1}^{\ell} \bar{\alpha}_i y_i \mathbf{x}_i$$

\bar{b} può essere determinato risolvendo le ℓ condizioni di Kuhn-Tucker

$$\bar{\alpha}_i (y_i (\bar{\mathbf{w}} \cdot \mathbf{x}_i + \bar{b}) - 1) = 0$$

ed effettuando poi la media dei ℓ valori ottenuti, in modo da ridurre eventuali errori di approssimazione. Possiamo notare come $\bar{\mathbf{w}}$ sia ottenuto come combinazione lineare dei vettori del training set \mathbf{x}_i per cui risulta $\bar{\alpha}_i \neq 0$. Questi vettori sono i support vectors, e usualmente sono molto pochi.

L'addestramento delle support vector machines è quindi un problema di QP (Quadratic Programming). In caso di dataset grandi (ad esempio $> 1.000.000$), risulta essere improponibile l'uso di tecniche numeriche standard per la risoluzione del problema. Sono stati proposti alcuni algoritmi per risolvere il problema: due esempi sono il cosiddetto metodo del "chunking" (Vapnik e Cortes 1995) e il Sequential Minimal Optimization (SMO - Platt 1999).

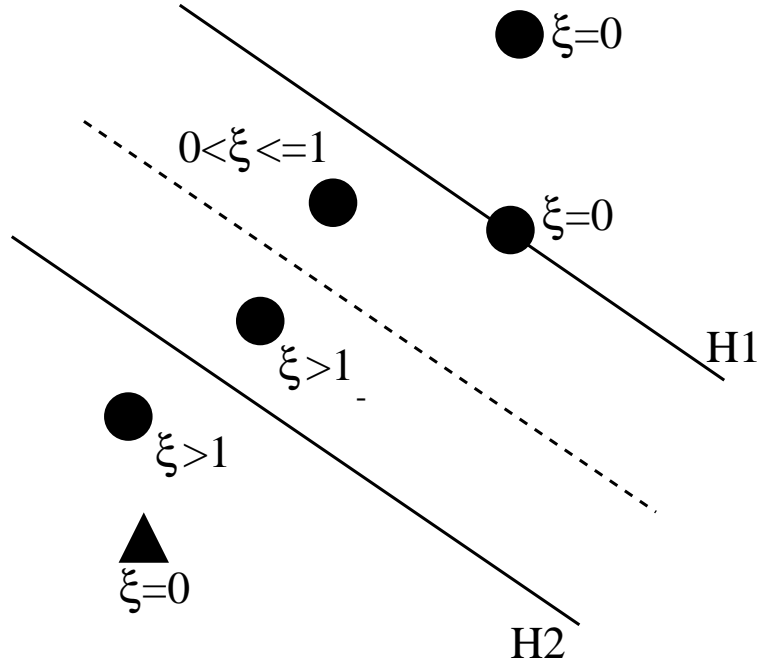
Una volta terminato il processo di addestramento, e ottenuti $\bar{\mathbf{w}}$ e \bar{b} , il processo di classificazione di un nuovo punto \mathbf{x} avviene molto semplicemente, calcolando come classe di appartenenza $\text{sgn}(\bar{\mathbf{w}} \cdot \mathbf{x} + \bar{b})$.

4.2 Dati non linearmente separabili

L'algoritmo della sezione precedente, se applicato a dati non linearmente separabili, non trova una soluzione accettabile, in quanto la funzione obiettivo (la lagrangiana del problema duale) può crescere all'infinito. Per evitare questo si è quindi pensato di rendere meno forti i vincoli (3) e (4), ma solo quando necessario. Per far questo, in tali vincoli vengono introdotte delle variabili dette "slack" (to slack = allentare), che permettono ai dati di superare i confini del margine (ricordo che il margine è lo spazio compreso tra i due iperpiani H_1 e H_2).

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \quad (7)$$

Quando $\xi_i = 0$, il punto \mathbf{x}_i si trova o su uno dei due iperpiani H_1, H_2 oppure fuori dal margine, esattamente come nel caso dei dati separabili. Se $0 < \xi_i \leq 1$ allora il punto \mathbf{x}_i si trova dentro il margine, ma ancora nella parte corretta del piano classificatore. Se infine $\xi_i > 1$, allora il punto è oltre il piano classificatore, e rappresenta un errore di classificazione nel training set. Graficamente, la situazione è la seguente.



L'idea è quindi quella di permettere queste situazioni, cercando però di limitarle il più possibile, inserendo quindi un termine costo, che dipende dai ξ_i , nella funzione obiettivo:

$$\min \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \left(\sum_{i=1}^{\ell} \xi_i \right)^k \quad \text{con il vincolo } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

C rappresenta un parametro scelto dall'utente, che determina la sensibilità della macchina agli errori: utilizzare un C grande significa penalizzare fortemente gli errori. Scegliendo il parametro $k = 1$, cosa che normalmente avviene, abbiamo che né gli ξ_i , né i corrispondenti coefficienti di Lagrange, appaiono nella formulazione duale del problema, che diventa:

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad \text{con i vincoli } \sum_{i=1}^{\ell} y_i \alpha_i = 0 \text{ e } 0 \leq \alpha_i \leq C$$

L'iperpiano separatore ottimale viene calcolato con la stessa formula del caso di dati linearmente separabili.

$$\mathbf{w} = \sum_{i=1}^{\ell} \bar{\alpha}_i y_i \mathbf{x}_i$$

Esiste un'interpretazione geometrica per gli α_i : se $\alpha_i = 0$, allora il punto è classificato correttamente; se $0 < \alpha_i < C$, allora il punto si trova all'interno del margine, compreso tra l'iperpiano H_1 e l'iperpiano classificatore, ma dalla parte giusta (i.e. viene classificato correttamente). Infine se $\alpha_i = C$ allora si tratta di un errore, che viene pesato con C indipendentemente dalla sua distanza dall'iperpiano classificatore. Anche in questo caso, i vettori per cui $\alpha_i \neq 0$ rappresentano i support vectors: possiamo osservare che i punti del training set che sono classificati in modo errato rappresentano anch'essi dei support vectors.

5 Support Vector Machines non lineari

Superfici di decisione lineari sono a volte troppo semplicistiche: come generalizzare i risultati ottenuti nella sezione precedente al caso di superfici di separazione non lineari? L'idea che sta alla base delle SVM non lineari è questa: invece di costruire un classificatore non lineare nello spazio del problema, si potrebbe cercare di trasformare i dati in uno spazio in cui un classificatore lineare sia "sufficiente" per discriminare correttamente. In altre parole, si possono estrarre delle features dai dati, e costruire un classificatore lineare nello spazio delle features invece che nello spazio originale.

Più precisamente l'idea è quella di mappare i dati in un altro spazio euclideo \mathcal{H} (di dimensione qualsiasi, anche enorme), utilizzando una funzione chiamata Φ .

$$\begin{aligned} \Phi : \mathbb{R}^d &\rightarrow \mathcal{H} \\ \mathbf{x} &\mapsto \mathbf{z}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_N(\mathbf{x})) \end{aligned}$$

dove ϕ_i sono funzioni fissate. A questo punto viene determinato un classificatore lineare nello spazio \mathcal{H} .

Esempio:

$$\begin{aligned} \mathbf{x} &= (x, y, z) \\ \phi_1(\mathbf{x}) &= x \\ \phi_2(\mathbf{x}) &= y \\ \phi_3(\mathbf{x}) &= z \\ \phi_4(\mathbf{x}) &= x^2 \\ \phi_5(\mathbf{x}) &= y^2 \\ \phi_6(\mathbf{x}) &= z^2 \\ \phi_7(\mathbf{x}) &= xy \\ \phi_8(\mathbf{x}) &= xz \\ \phi_9(\mathbf{x}) &= yz \end{aligned}$$

$$\mathbf{z}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_9(\mathbf{x}))$$

Un classificatore lineare nello spazio $\langle \mathbf{z}(\mathbf{x}) \rangle$ corrisponde ad un classificatore polinomiale nello spazio $\langle \mathbf{x} \rangle$ originale.

Sorge un problema: se lo spazio \mathcal{H} è di dimensionalità enorme (può essere infinita), diventa veramente difficile lavorare con la funzione Φ . Inoltre può essere molto difficile calcolare questo mapping Φ . Questi problemi sono risolti dalle SVM in modo molto efficace, partendo dalla seguente osservazione: i dati del training set, nelle formule dell'algoritmo di training, appaiono sempre nella forma di prodotti scalari $\mathbf{x}_i \cdot \mathbf{x}_j$. Trasformando i dati \mathbf{x} in $\Phi(\mathbf{x})$, il processo di training continuerà a dipendere dal prodotto scalare dei dati trasformati $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Ora, se esistesse una funzione “kernel” K tale per cui

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

allora si potrebbe utilizzare direttamente K nell'algoritmo di training, senza esplicitamente trattare con il mapping Φ .

Esempio. Supponiamo che i dati appartengano allo spazio \mathbb{R}^2 , i.e. $\mathbf{x} = (x, y)$. Un esempio di kernel è allora

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$$

a cui corrisponde una mappa Φ nello spazio $\mathcal{H} = \mathbb{R}^3$

$$\Phi(\mathbf{x}) = \begin{pmatrix} x^2 \\ \sqrt{2}xy \\ y^2 \end{pmatrix}$$

Possiamo notare come questa Φ non è l'unica, infatti, sempre con $\mathcal{H} = \mathbb{R}^3$

$$\Phi(\mathbf{x}) = \begin{pmatrix} (x^2 - y^2) \\ 2xy \\ (x^2 + y^2) \end{pmatrix}$$

oppure con $\mathcal{H} = \mathbb{R}^4$

$$\Phi(\mathbf{x}) = \begin{pmatrix} x^2 \\ xy \\ xy \\ y^2 \end{pmatrix}$$

Dato un kernel K , lo spazio \mathcal{H} corrispondente di dimensione minima è detto *spazio minimale*.

È importante notare che, rimpiazzando $\mathbf{x}_i \cdot \mathbf{x}_j$ con $K(\mathbf{x}_i, \mathbf{x}_j)$ nell'algoritmo di training, l'algoritmo determinerà una support vector machine che vive in uno spazio di dimensione grande a piacere; inoltre questo risultato verrà ottenuto approssimativamente nello stesso tempo necessario per creare una SVM per i dati nello spazio originale.

Perché esista questo dualismo mapping (Φ) - kernel (K) deve essere rispettata la seguente condizione:

Teorema 3 *Esiste un mapping Φ e un kernel K tali per cui vale*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \sum_k \Phi(\mathbf{x}_i)_k \Phi(\mathbf{x}_j)_k$$

se e solo se vale la condizione di Mercer, ovvero per ogni $g(\mathbf{x})$ tale per cui

$$\int g(\mathbf{x})^2 d\mathbf{x} \quad \text{è finito}$$

allora vale che

$$\int K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$$

Se vale la condizione di Mercer allora K viene detto “Kernel positivo”.

Esempi di Kernel:

- Kernel Polinomiali di grado p : $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$
- Kernel Gaussian Radial Basis Function di varianza σ^2 : $K(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}}$
- $K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x} \cdot \mathbf{y} - \delta)$

6 VC-dimension e generalizzazione delle SVM

La *VC-dimension* di una SVM può essere di entità molto elevata, come enunciato nel seguente teorema:

Teorema 4 Sia $K(\mathbf{x}_i, \mathbf{x}_j)$ un kernel positivo a cui corrisponde uno spazio minimale \mathcal{H} . Allora la *VC-dimension* della corrispondente SVM è $\dim(\mathcal{H}) + 1$.

Intuitivamente lo si poteva capire anche dall'esempio (3) della sezione 2.2: la *VC-dimension* di un iperpiano separatore in \Re^n è $n + 1$. In questo caso n dipende dal tipo di kernel, e rappresenta la dimensionalità dello spazio di arrivo del mapping Φ indotto dal kernel K .

Esempio. Dato il kernel in \Re^{D_L}

$$K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^p$$

allora la *VC-dimension* della corrispondente SVM vale

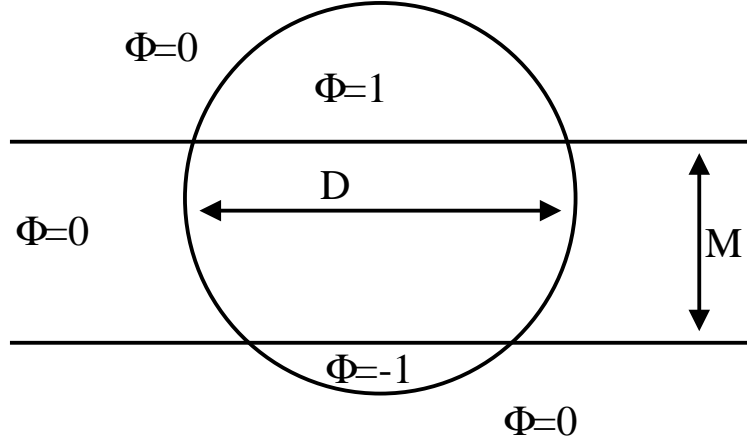
$$\binom{D_L + p - 1}{p} + 1$$

Si può osservare come questa quantità cresca molto velocemente.

Nonostante la VC-dimension di una Support Vector Machine possa essere elevata, questo tipo di classificatore presenta spesso un'ottima capacità di generalizzazione. Non è ancora stata trovata una solida teoria che giustifichi questo comportamento, almeno a conoscenza dell'autore. Tuttavia ci sono delle giustificazioni non formali che definiscono le SVM come un'approssimativa implementazione del principio della *Structural Risk Minimization*. Vediamo di seguito un esempio di una famiglia di classificatori, detti *Gap Tolerant Classifier*,

che sono simili alle SVM, e a cui è possibile applicare il principio SRM: questo esempio ci permetterà di capire perché è importante massimizzare il margine.

Un *Gap Tolerant Classifier* $\phi \in \Phi$, in \mathbb{R}^d , è determinato dalla posizione e dal diametro di una ipersfera in \mathbb{R}^d e da due iperpiani paralleli, ancora in \mathbb{R}^d . I punti compresi tra i due iperpiani sono detti *Margin set*. Il classificatore funziona come segue: i punti che stanno all'interno della sfera, ma non nel *Margin set*, vengono etichettati con $\{\pm 1\}$, a seconda che siano sopra o sotto il *Margin set*. Gli altri punti non vengono considerati dal classificatore, si suppone che siano classificati correttamente (i.e. non danno nessun contributo nel calcolo del rischio). D rappresenta il diametro della sfera, mentre M rappresenta il margine.



Per questa classe di classificatori, la VC-dimension h è legata al massimo diametro D_{max} e al minimo margine M_{min} dalla seguente disuguaglianza:

$$h \leq \min \left\{ \left\lceil \frac{D_{max}^2}{M_{min}^2} \right\rceil, d \right\} + 1 \quad (8)$$

Da qui si deduce che, a parità di diametro, più grande è il margine, minore sarà il bound sulla VC-dimension. Vediamo come viene implementata la *Structural Risk Minimization* su questa classe Φ di classificatori: consideriamo il sottoinsieme $\Phi_S \subset \Phi$ contenente tutti i classificatori che etichettano $\{\pm 1\}$ tutti i punti del training set; in altre parole con ogni classificatore di Φ_S tutti i punti del training set stanno dentro la sfera. Sia N_{min} il minimo numero di errori commettabili sul training set dai classificatori in Φ_S . L'addestramento procede nel seguente modo:

1. si trovano in Φ_S tutti i classificatori che commettono N_{min} errori nel training set;
2. all'interno di questo sottoinsieme si trova il classificatore ϕ che massimizza il margine: questo classificatore produce il minor bound per la VC-dimension, e quindi il minor valore del rischio garantito;
3. memorizzo il valore del rischio garantito (2), dove sostituisco alla VC-dimension il bound (8);
4. Considero ora il sottoinsieme di Φ_S contenente i classificatori che commettono $N_{min} + 1$ errori, e re-itero.

Alla fine delle iterazioni, considero il classificatore che presenta il minimo rischio garantito.

Il classificatore risultante è un tipo particolare di support vector machine che non considera come errori i punti che stanno fuori dalla sfera contenente il training set o all'interno del margin set. Questo classificatore risulta essere determinato applicando il principio della *Structural Risk Minimization*. Lungi dall'essere una dimostrazione formale, sembra essere tuttavia piuttosto ragionevole concludere che le SVM, che vengono addestrate con obiettivi simili, posseggano anch'esse una specie di controllo della capacità nella fase di training, e possano essere considerate come un'approssimativa implementazione del principio SRM. Il numero di errori commettabili sul training set (N_{min} nel caso dei *Gap Tolerant Classifier*) può essere approssimato con la scelta del parametro C , che determina il costo di una classificazione errata: nel caso di C grande, si avrà un classificatore che commetterà pochi errori, nel caso di C piccolo si avrà una curva più *smooth*.

Un ulteriore bound per l'errore di generalizzazione delle SVM è stato introdotto da Vapnik nel 1995, interessante in quanto, pur essendo semplicemente un bound, permette di dare un'idea quantitativa della capacità di generalizzazione di una SVM al termine della fase di training. Dato un training set di ℓ esempi, sia $P(error)$ il rischio calcolato su $\ell - 1$ esempi, e sia $E[P(error)]$ il valor medio di tale rischio, calcolato su tutte le ℓ possibili scelte di $\ell - 1$ elementi dal training set. $E[P(error)]$ rappresenta il cosiddetto errore *Leave One Out*. Sia infine $E[\#support\ vectors]$ il numero medio di support vectors determinato dagli addestramenti delle ℓ SVM. Allora vale che

$$E[P(error)] \leq \frac{E[\#support\ vectors]}{\ell}$$

Alla fine del training si può quindi avere un'idea della capacità di generalizzazione della macchina contando il numero di support vectors determinati.

Un'ultima considerazione: le disuguaglianze presentate sono solamente dei bound, e come tali rappresentano dei limiti superiori: in alcuni casi possono risultare triviali o privi di informazione. Ad esempio, se i support vectors sono il 50% dei vettori di training, il bound dice che la probabilità di errore è minore del 50%, considerazione di nessuna utilità!

7 Conclusioni

Riassumendo, le Support Vector Machines classificano utilizzando due idee: la massimizzazione del margine e il "trucco" del kernel. Il primo mira ad ottenere migliori prestazioni di generalizzazione, il secondo permette di determinare superfici di separazione altamente non lineari ad un costo computazionale pressoché equivalente a quello delle superfici lineari.

L'addestramento delle SVM equivale a risolvere un problema di minimizzazione quadratica con tante variabili quanti sono i dati nel training set. Il minimo calcolato è globale, e sotto alcune condizioni, anche unico. Per unico si intende che l'iperpiano separatore è determinato dallo stesso insieme di support vectors.

L'addestramento delle SVM comporta la scelta di alcuni parametri:

- la costante C che determina il costo dell'errore di classificazione errata;
- se il classificatore è non lineare, occorre determinare il kernel, che determina la forma della curva decisionale, e i suoi parametri di smoothing (varianza σ^2 , grado del polinomio p).

7.1 Vantaggi e svantaggi

L'utilizzo delle Support Vector Machines in problemi di classificazione presenta notevoli vantaggi, ma anche qualche svantaggio: vediamoli nel dettaglio.

Vantaggi:

- l'uso dei kernels permette di creare in modo non oneroso superfici di separazione altamente non lineari;
- l'uso dei kernels permette di eliminare decisamente il problema della “curse of dimensionality”; si può quindi lavorare con problemi di qualsivoglia dimensionalità;
- a differenza delle reti neurali, la soluzione trovata nella fase di training è globale;
- le prestazioni di generalizzazione possono essere caratterizzate alla fine del processo di training, contando il numero dei support vectors;
- ha un'interpretazione geometrica semplice e comprensibile;
- permette di rappresentare l'intero training set con un numero (tipicamente) ridotto di punti, i support vectors.

Svantaggi:

- la scelta del kernel è cruciale;
- la velocità dell'addestramento può rappresentare un problema, soprattutto per grossi datasets;
- non tratta con dati discreti;
- non è incrementale, i.e. se aggiungo un elemento al training set devo riaddestrare la macchina;
- la generalizzazione ad un problema di classificazione a più classi non è ottimizzata: al momento, con un problema a N classi, occorre costruire N SVM (classe 1 vs il resto, classe 2 vs il resto, ...).

Riferimenti bibliografici

- [1] C.J. Burges: “A tutorial on Support Vector Machines for Pattern Recognition” *Data Mining and Knowledge Discovery* 2(2) 121-167 (1998).
- [2] V. Vapnik: *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, (1995).
- [3] Lucent Technologies SVM demo applet:
<http://svm.research.bell-labs.com/SVT/SVMsvt.html>

