Support Vector Machines

Corso di Modellizzazione Statistica, prof. M. Bilancia

Michele Di Nanni, mat. 7291871

Introduzione

L'algoritmo **Support Vector Machines** è uno degli algoritmi più utili, efficienti ed importanti, appartenenti alla categoria degli algoritmi di *machine learning* supervisionati. L'ambito applicativo più frequente è quello dell' elaborazione del linguaggio naturale (*NLP, natural language processing*), del riconoscimento vocale, delle immagini e della *computer vision*. Esso è utilizzato sia per scopi di classificazione, che di regressione. L'idea alla base è quella della presenza di una non separabilità dei dati in modo lineare, col conseguente obbiettivo di costruire un separatore, ovvero un iperpiano di separazione **ottimale**, che ci permetta di dividere al meglio i dati presenti nel *dataset* in classi. Si cerca per prima cosa un iperpiano di separabilità lineare e qualora ve ne fossero più di uno, si andrebbe a cercare quello contenente il *margine* più alto, al fine di migliorare l'accuratezza del modello. Tuttavia, se tale iperpiano non esiste l'**SVM** utilizza una **mappatura non lineare** per trasformare i dati di *training* in uno spazio di dimensionalità maggiore(spazio delle variabili), utilizzando il *kernel trick*, in modo da ricondurci al caso di separabilità lineare.

1.1 Iperpiano di separazione ottimale e separabilità lineare

Ciò che vogliamo trovare nelle **SVM** è quell' iperpiano, che nel caso di due classi sarà proprio una retta, che meglio classifica e, quindi, separa i nostri dati. Teoricamente, potremmo avere un numero infinito di rette (e quindi anche di iperpiani) che separino le istanze dei dati di training. L'obbiettivo è proprio trovare quella retta(iperpiano) che sia **ottimale**, generando il più piccolo errore di classificazione su dati di test. Ciò che vorremmo, pertanto, è che i nostri dati siano il più lontano possibile dalla retta(iperpiano), pur restando nella parte corretta, cioè quella di appartenenza a quella specifica classe.

Partendo dal caso in cui abbiamo *due classi* con etichette -1 e $+1$, consideriamo il campione $\chi = \{\mathbf{x}^t, \mathbf{r}^t\}$ dove:

1. $\mathbf{x}^t$ è l'insieme dei dati di training
2. $\mathbf{r}^t$ sono le etichette ottenute, ovvero gli output

$$r^t = \begin{cases} +1, & \text{se } \mathbf{x}^t \in C_1 \\ -1, & \text{se } \mathbf{x}^t \in C_2 \end{cases}$$

L'obbiettivo è quello di trovare $\mathbf{w}$ e w_0 tale che:

$$\mathbf{w}^T\mathbf{x}^t + w_0 \geq +1 \quad per \quad r^t = +1$$

$$\mathbf{w}^T\mathbf{x}^t + w_0 \leq -1 \quad per \quad r^t = -1$$

In altre parole, si vuole ricercare quella combinazione lineare per cui, se la etichetta di classe è $+1$, allora la combinazione dovrà essere maggiore o al più uguale a $+1$; viceversa, se l'etichetta è -1, allora occorrerà trovare quella combinazione lineare che sia minore o al più uguale a -1. Potremmo comunque pensare di "fondere" queste ultime disequazioni, nella seguente:

$$r^t(\mathbf{w}^T\mathbf{x}^t + w_0) \geq +1$$

cioe, se abbiamo una risposta pari a $+1$ ci riferiremo ad una regione, altrimenti ci riferiremo all'altra. Il vettore $\mathbf{w}$ è il cosiddetto *vettore dei pesi* che avrà norma unitaria, cioè $||\mathbf{w}|| = 1$.

Diamo adesso delle definizioni, utili per comprendere al meglio il concetto di *iperpiano di separazione ottimale*:

Def. **Vettori di supporto**

I vettori di supporto sono gli esempi di training **più vicini** all'iperpiano. Questi punti dipendono dal dataset che analizziamo e, pertanto, qualora fossero rimossi o modificati, la posizione dell'iperpiano di divisione verrebbe alterata. A tal punto, possiamo dire che costituiscono *gli elementi critici* del dataset.

Def. **Margine**

Il margine è la definito come la distanza *minima* fra l'iperpiano di separazione e i vettori di supporto. È fondamentale chiarire che a metà di questa distanza viene tracciato l'iperpiano, o la retta nel caso in cui abbiamo due classi. La dimensione del margine massimo è:

$$\frac{1}{||\mathbf{w}||} + \frac{1}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

Def. **Iperpiano di separazione ottimale**

L' iperpiano di separazione ottimale è quell'iperpiano che massimizza il margine.

Pertanto, l'equazione $\mathbf{w}^T\mathbf{x} = 0$ definirà il limite di decisione, noto anche col termine **decision boundary**; mentre $\mathbf{w}^T\mathbf{x} = -1$ definisce l' *iperpiano negativo*, ovvero la regione "negativa" e l'equazione $\mathbf{w}^T\mathbf{x} = +1$ definisce l'*iperpiano positivo*, ovvero la regione "positiva". Osserviamo queste considerazioni appena fatte all'interno dell'immagine seguente:

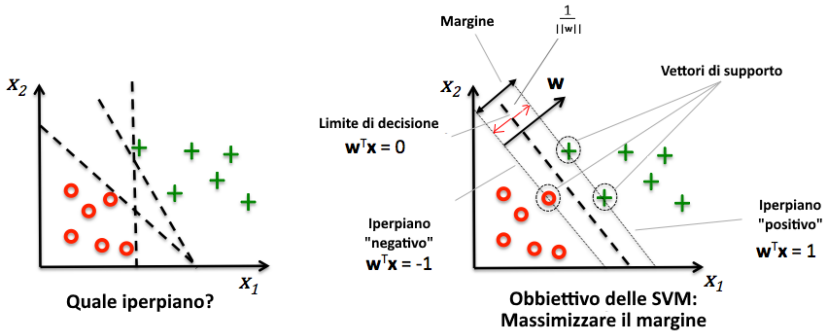


Figure 1: Support Vector Machines: idee di base

Come possiamo evincere dalla *Fig.*1, il concetto alla base delle **SVM** è, quindi, quello di trovare l'iperpiano ottimale che crei il margine più grande fra le istanze di training appartenenti alla classe -1 ed alla classe $+1$.

Il problema può essere ricondotto ad un problema di *ottimizzazione*, in cui vogliamo trovare:

$$\max_{\mathbf{w},w_0,\ ||\mathbf{w}||=1} \rho \qquad t.c. \quad r^t(\mathbf{w}^T\mathbf{x}^t + w_0) \geq \rho, \ \ \forall t$$

dove ρ indica il valore da massimizzare, cioè il margine. Tuttavia, per ottenere una rappresentazione più efficiente del problema, possiamo pensare di minimizzare $||\mathbf{w}||$, quindi il problema diventa:

$$\min_{\mathbf{w},w_0} \frac{1}{2}||\mathbf{w}||^2 \qquad t.c. \quad r^t(\mathbf{w}^T\mathbf{x}^t + w_0) \geq +1, \ \ \forall t$$

Nota: Abbiamo posto $\frac{1}{2}$ nella definizione del valore da minimizzare e la norma al quadrato, in quanto per poter calcolare tale minimo occorrerà il calcolo della derivata ed in tal modo, derivando appunto, quella frazione scomparirà perchè sarà moltiplicata per l'esponente della norma.

Dunque, il problema rientra nel contesto dei problemi di ottimizzazione quadratici, che possono essere risolti direttamente andando alla ricerca di $\mathbf{w}$ e w_0. Il parametro di complessità è d, ovvero il numero di *features* presenti nel dataset. Per poter trovare il miglior iperpiano, possiamo convertire il problema facendo in modo che il parametro di complessità sia N, ovvero il numero di *istanze di training*.

1.1.1 Risoluzione del problema di minimizzazione

Il problema può essere risolto con l'ausilio del metodo dei moltiplicatori di Lagrange, il quale è un metodo analitico che ci permette di trovare massimi e minimi vincolati.

Per questo motivo, poniamo:

$$L_p \quad = \quad \frac{1}{2}||\mathbf{w}||^2 - \sum_{t=1}^{N} \alpha^t [r^t(\mathbf{w}^T\mathbf{x}^t + w_0) - 1]$$

$$= \frac{1}{2}||\mathbf{w}||^2 - \sum_{t=1}^{N} \alpha^t r^t(\mathbf{w}^T\mathbf{x}^t + w_0) + \sum_t \alpha^t \qquad (1.1)$$

A questo punto dovremmo minimizzare $\mathbf{w}$ e w_0 e massimizzare $\alpha^t \geq 0$: il *punto di sella* ci fornisce la soluzione. Questo è un problema di ottimizzazione, per questo motivo possiamo equivalentemente risolvere il problema *duale*, usando le condizioni di *Karush-Kuhn-Tucker*: massimizzare L_p rispetto ad α^t, soggetto ai vincoli che il gradiente di L_p rispetto a $\mathbf{w}$ e w_0 sia nullo e che $\alpha^t \geq 0$. Infatti, la teoria dell'ottimizzazione ci dice che un problema possiede una forma duale più semplice da risolvere, se la funzione di costo e i vincoli sono strettamente convessi: è il nostro caso.

Svolgendo i prodotti e riscrivendo la norma otteniamo:

$$= \frac{1}{2}(\mathbf{w}^T\mathbf{w}) - \mathbf{w}^T \sum_t \alpha^t r^t \mathbf{x}^t - w_0 \sum_t \alpha^t r^t + \sum_t \alpha^t \qquad (*)$$

Per poter procedere, occorre sia calcolare le derivate rispetto a $\mathbf{w}$ e w_0, ponendole uguali a zero, sia considerare che $\alpha^t \geq 0$. Ovvero:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_t \alpha^t r^t \mathbf{x}^t \qquad (i)$$

$$\frac{\partial L_p}{\partial w_0} = 0 \quad \Rightarrow \quad \sum_t \alpha^t r^t = 0 \qquad (ii)$$

Da (i) e da (ii) abbiamo ottenuto due risultati fondamentali; procediamo quindi a fare le opportune sostituzioni all'equazione contrassegnata con $(*)$, otteniamo la forma duale:

$$L_d = -\frac{1}{2}(\mathbf{w}^T\mathbf{w}) + \sum_t \alpha^t$$

$$= -\frac{1}{2}\sum_t\sum_s \alpha^t\alpha^s r^t r^s (\mathbf{x}^t)^T\mathbf{x}^s + \sum_t \alpha^t$$

che vogliamo massimizzare rispetto ad α^t, coi vincoli che $\sum_t \alpha^t r^t = 0$ e che $\alpha^t \geq 0$,$\forall t$. La dimensione dipende da N, ovvero dalla dimensione del campione, e non da d, rispettivamente la *dimensionalità* dell'input. La soluzione deve soddisfare le condizioni di Karush-Kuhn-Tucker (vedi appendice), che includono (1), (2), $\alpha^t \geq 0$ e:

$$\alpha^t[r^t(\mathbf{w}^T\mathbf{x}^t + w_0) - 1] = 0 \quad \forall t$$

Possiamo osservare che:

- Se $\alpha^t = 0$, allora $\mathbf{x}^t$ non è sul confine del margine, tuttavia
- Se $\alpha^t > 0$, allora $r^t(\mathbf{w}^T\mathbf{x}^t + w_0) = 1$, ovvero, $\mathbf{x}^t$ è sul confine del margine

Per questo motivo, i vettori $\mathbf{x}^t$ tali che $\alpha^t > 0$ sono proprio i **vettori di supporto**. Il discriminante, cioè la retta(o l'iperpiano) trovato è chiamato *macchina a vettore di supporto*(nota con l'acronimo **SVM**).

Durante la fase di *testing*, andremo a calcolare $g(x) = \mathbf{w}^T\mathbf{x} + w_0$ e sceglieremo in base al segno di $g(x)$: se $g(x) > 0$ allora sceglieremo la classe $C1$, altrimenti la classe $C2$, cioe:

$$sign(g(x)) = sign(\mathbf{w}^T\mathbf{x} + w_0) = \begin{cases} +1 & \rightarrow C_1 \\ -1 & \rightarrow \mathbf{C}_2 \end{cases}$$

Riassumiamo queste considerazioni fatte finora, nel caso in cui ci troviamo ad operare con dataset di due classi, dove abbiamo la separabilità lineare, nella $Fig.2$:

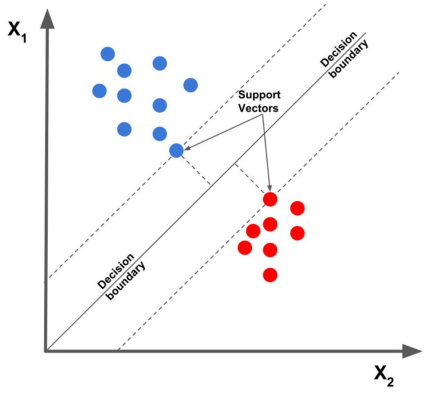


Figure 2: Problema a due classi con separabilità lineare

1.2 Non separabilità lineare

Finora abbiamo considerato il caso in cui vi fosse una separabilità lineare fra i dati, tuttavia, nella maggior parte delle applicazioni reali questo non accade e quindi abbiamo bisogno di trovare una alternativa che ci permetta di ricondurci al caso lineare.

Se le due classi non sono, dunque, *linearmente separabili*, andremo alla ricerca di un certo iperpiano che presenti l'errore minimo di errata classificazione. In questo caso, possiamo ammettere che alcuni vincoli enunciati precedentemente siano violati e abbiamo necessità di definire le cosiddette "variabili deboli", che denoteremo da ora in poi come variabili **slack**, $\xi = (\xi^1, \xi^2, ..., \xi^t)$ *t.c.* $\xi^t \geq 0$, che consentono la classificazione errata di qualche punto e che codificano la deviazione del margine. Pertanto, il vincolo diventa:

$$\mathbf{w}^T\mathbf{x}^t + w_0 \geq 1 - \xi^t \quad se \quad r^t = +1, \quad \forall t$$
$$\mathbf{w}^T\mathbf{x}^t + w_0 \leq -1 + \xi^t \quad se \quad r^t = -1 \quad \forall t$$

Che possono essere riassunti in:

$$r^t(\mathbf{w}^T\mathbf{x}^t + w_0) \geq 1 - \xi^t \quad \forall t$$

Ci troviamo dinnanzi a due differenti tipi di deviazione del margine: una istanza può trovarsi sul lato sbagliato ed essere non classificata correttamente; oppure, può essere nel lato corretto ma trovarsi sul margine, cioè, non sufficientemente lontana dall'iperpiano. Pertanto, se $\xi^t = 0$ allora avremo corretta classificazione; invece se $0 < \xi^t < 1$, l'istanza $\mathbf{x}^t$ è classificata correttamente ma nella zona del margine; se $\xi^t > 1$, l'istanza $\mathbf{x}^t$ non è classificata correttamente. Osserviamo queste considerazioni nella *Fig.*3

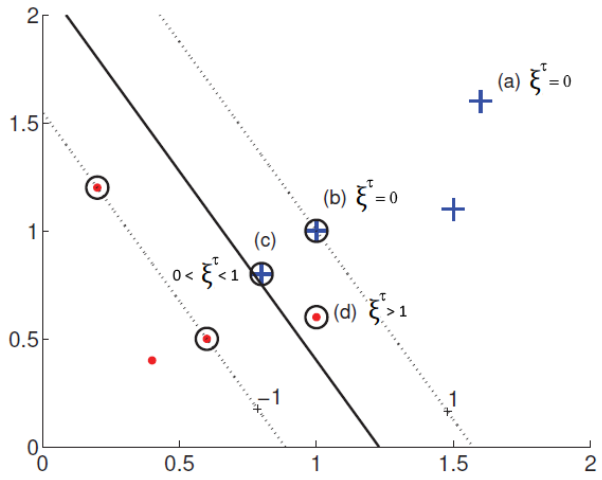


Figure 3: Non linearità

Possiamo notare di come l'istanza (a) sia classificata *correttamente*, per questo motivo $\xi^t = 0$, ovvero $r^t g(\mathbf{x}^t) > 1$, e dunque molto lontana dal margine. L'istanza (b) si trova nella zona corretta, quindi $\xi^t = 0$, ma è sul margine, mentre l'istanza (c) si trova nel lato corretto ma è all'interno del margine, dunque non sufficientemente lontana $(0 < \xi^t < 1)$. Infine, l'istanza (d) è classificata in modo erroneo, pertanto $\xi^t > 1$. Tutti i casi tranne (a) sono vettori di supporto.

A questo punto definiamo il *numero di classificazioni errate* come $\#\{\xi^t \geq 1\}$ e il ***soft error*** come la somma delle variabili slack su dati di training, cioè $\sum_t \xi^t$.

Occorre modificare la funzione di costo, in modo da penalizzare le variabili slack che non sono a 0, introducendo una costante positiva C che misura il *trade-off* tra la massimizzazione del margine e la

minimizzazione dell'errore. È anche noto col termine di "fattore di penalità". Inoltre ciò che penalizziamo non sono solo i punti mal classificati(tali che $\xi^t \geq 1$), ma anche quelli presenti nel margine($0 < \xi^t < 1$) al fine di ottenere una migliore generalizzazione. Quindi avremo che:

$$L_p = \frac{1}{2}||\mathbf{w}||^2 + C\sum_t \xi^t \quad t.c. \quad \xi^t \geq 0,\ r^t(\mathbf{w}^T\mathbf{x}^t + w_0) \geq 1 - \xi^t \quad \forall t$$

L'obbiettivo è calcolare il minimo di L_p con gli opportuni vincoli, ovvero occorre calcolare:

$$\min L_p = \min_{\mathbf{w},w_0} \frac{1}{2}||\mathbf{w}||^2 + C\sum_t \xi^t \quad t.c. \quad \xi^t \geq 0,\ r^t(\mathbf{w}^T\mathbf{x}^t + w_0) \geq 1 - \xi^t \ \ \forall t$$

1.2.1 Risoluzione del problema di minimizzazione

A questo punto, occorre risolvere il problema di minimo, allo stesso modo di come è stato operato nella sezione 1.1.1: considerando i vincoli suddetti, la funzione Langragiana dell' equazione (1.1) , otterremo:

$$L_p = \frac{1}{2}||\mathbf{w}||^2 + c\sum_t \xi^t - \sum_t \alpha^t[r^t(\mathbf{w}^T\mathbf{x}^t + w_0) - 1] + \xi^t] - \sum_t \mu^t\xi^t \qquad (*)$$

dove, μ^t sono i nuovi parametri di Lagrange che ci permettono di garantire la positività a ξ^t. Il nostro obbiettivo è minimizzare $\mathbf{w}, w_0$ e ξ^t(quest'ultimo poichè più vicino a zero è tale valore, più piccolo sarà l'errore di classificazione). Calcoliamo le derivate, le poniamo a zero e otteniamo:

$$\frac{\partial L_p}{\partial \mathbf{w}} \Rightarrow \mathbf{w} = \sum_t \alpha^t r^t \mathbf{x}^t \qquad (i)$$

$$\frac{\partial L_p}{\partial w_0} \Rightarrow \mathbf{0} = \sum_t \alpha^t r^t \qquad (ii)$$

$$\frac{\partial L_p}{\partial \xi^t} \Rightarrow \mathbf{0} = C - \alpha^t - \mu^t \quad \Rightarrow \quad \alpha^t = C - \mu^t \qquad (iii)$$

Avendo ottenuto questi risultati, possiamo procedere a sostituirli nell'equazione denotata con $(*)$. Otterremo la seguente equazione che vogliamo massimizzare rispetto ad α^t:

$$L_d = \sum_t \alpha^t - \frac{1}{2}\sum_t\sum_s \alpha^t\alpha^s r^t r^s (\mathbf{x}^t)^T\mathbf{x}^s \quad t.c. \ \sum_t \alpha^t r^t = 0,\ 0 \leq \alpha^t \leq C,\ \forall t$$

In aggiunta alle condizioni $(i), (ii), (iii)$, le condizioni di Karush-Kuhn-Tucker includono i tre vincoli seguenti:

$$\alpha^t[r^t(\mathbf{w}^T\mathbf{x}^t + w_0) - (1 - \xi^t)] = 0$$
$$\mu^t\xi^t = 0$$
$$r^t(\mathbf{w}^T\mathbf{x}^t + w_0) - (1 - \xi^t) \geq 0$$

La soluzione per $\mathbf{w}$ è data da:

$$\hat{\mathbf{w}} = \sum_t \hat{\alpha}^t r^t \mathbf{x}^t$$

Le istanze per cui il valore di $\alpha^t = 0$ sono quelle istanze che si trovano nella zona corretta e che sono caratterizzate dal fatto che $\xi^t = 0$. Invece, le istanze per cui il valore di $\alpha^t > 0$ sono le istanze che definiscono i *vettori di supporto*: in particolare tra questi troviamo quelli che risiedono sul bordo del margine($\xi^t = 0$), per cui $0 < \alpha^t < C$, i quali possono essere usati per calcolare

w_0. Infine, le istanze per cui $\alpha^t = C$ e $\xi^t > 0$ sono quelle che non sono abbastanza lontane dal margine.

Otteniamo le soluzioni $\hat{\mathbf{w}}$ e $\hat{w_0}$ e la funzione di decisione possiamo riscriverla come segue:

$$g(x) = sign[\hat{\mathbf{w}}^T \mathbf{x} + \hat{w_0}]$$

Quindi per poter discriminare andremo a guardare il segno di questa combinazione lineare.

1.2.2 Il parametro C e la $\nu - SVM$

Il parametro C è il parametro di costo, che può essere regolato utilizzando una *cross-validation*. Esso definisce il *trade-off* fra la massimizzazione del margine e la minimizzazione dell'errore: se questo è troppo grande, avremo una grossa penalizzazione per punti che sono non separabili, e potremmo, perciò, immagazzinare molti vettori di supporto ed ottenere *overfitting*; tuttavia, se questo è settato ad un valore molto piccolo, avremmo soluzioni troppo semplici, col il rischio di incorrere in *undefitting*. Possiamo vedere questi risultati nella *Fig.*4 seguente:

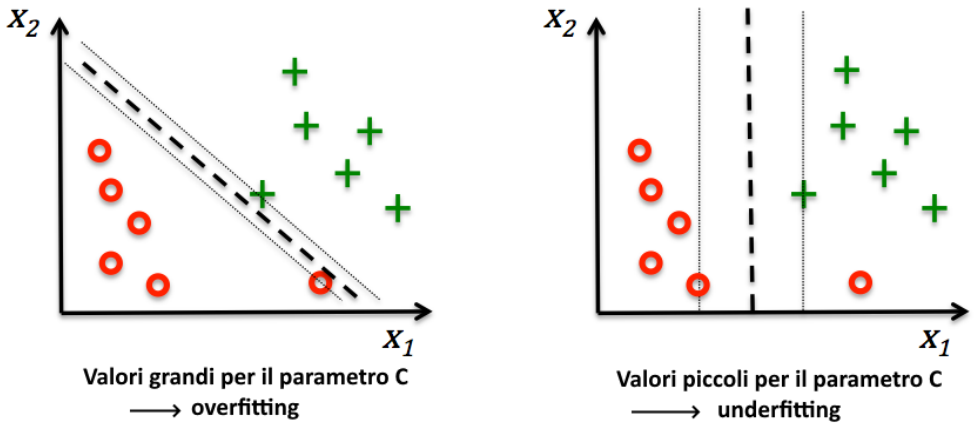


Figure 4: Regolazione del parametro C

Tuttavia, esiste anche una ulteriore formulazione equivalente dell' iperpiano di miglior separazione nel caso non lineare (*soft margin hyperplane*), chiamata $\nu - \mathbf{SVM}$. Il concetto di base è quello di utilizzare il parametro ν piuttosto che il parametro C, in quanto il primo varia tra il range $[0, 1]$, mentre il secondo varia tra $[0, +\infty)$, quindi potrebbe essere più difficile da stimare e da regolare.

1.2.3 *Hinge Loss*

La *Hinge Loss* è la funzione che ci permette di effettuare una misura dell' errore: è perciò, una funzione di perdita molto robusta rispetto alle altre, in quanto penalizza sia quelle istanze che si trovano nel margine, sia quelle mal classificate, il tutto per ottenere una efficienza maggiore dell'algoritmo.

Per poter definire la *Hinge Loss*, dobbiamo definire l'errore atteso di test:

$$E_N[P(error)] \leq \frac{E_N[\#vettori\ di\ supporto]}{N}$$

dove $E_N[\cdot]$ rappresenta la previsione sul training set di dimensione N: notiamo che tale rapporto di errore non dipende dalla *dimensionalità* dell'input.

Sapendo che un errore è prodotto se l'istanza si trova nel lato sbagliato oppure se l'istanza si trova nel margine, ovvero quando $0 < \xi^t < 1$, possiamo definire la *Hinge Loss*, cioè la funzione di penalizzazione dell'errore nelle SVM

Def. **Hinge Loss**

Se sappiamo che $y^t = \mathbf{w}^t\mathbf{x}^t + w_0$ è l'output ottenuto, mentre r^t è l'output atteso, possiamo definire la *Hinge Loss* come:

$$L_{hinge}(y^t, r^t) = \begin{cases} 0 & \text{se } y^t r^t \geq 1 \\ 1 - y^t r^t & \text{altrimenti} \end{cases}$$

Possiamo confrontare questa funzione di perdita con l'errore quadratico, con la *cross-entropy* utilizzata nella discriminazione logistica, e con la *0/1 loss*: quest'ultima assumerà banalmente valore 0 se la risposta ottenuta e quella predetta coincidono, 1 in tutti gli altri casi. Nell'immagine seguente possiamo notare il confronto della *hinge loss* rispetto ad altre funzioni utilizzate in contesti differenti(come ad esempio l'errore quadratico utilizzato molto spesso nella regressione). L'errore quadratico è dato da $(1 - y^t)^2$. La *cross-entropy* è data da $\log(\frac{1}{1+\exp(-y^t)})$.

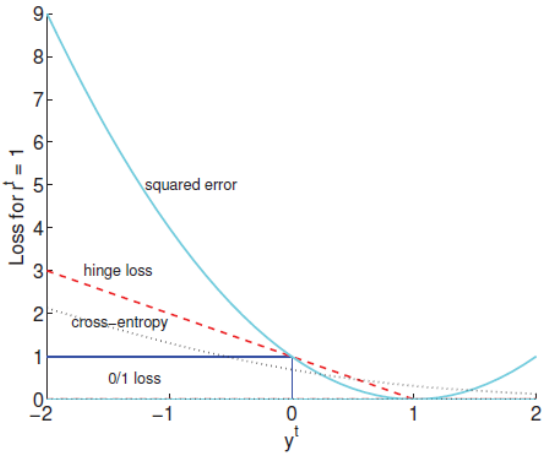
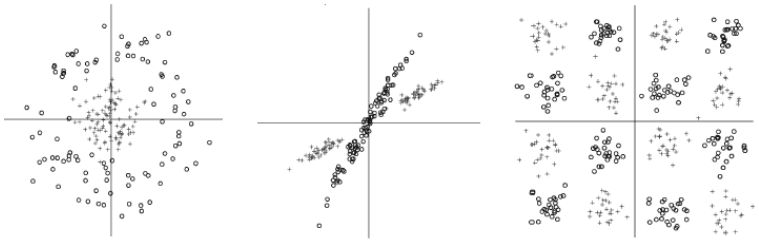


Figure 5: Confronti fra *hinge loss* ed altre funzioni di perdita($r^t = 1$)

1.3 Il Kernel Trick



Ci sono molti casi in cui un iperpiano di separazione rappresenta una soluzione troppo semplice. Pertanto, potremmo mappare le istanze in un nuovo spazio effettuando una trasformazione in uno spazio di **dimensione maggiore**, in modo da poterle separare più facilmente e quindi di ricondurci al caso lineare. Tale trasformazione è effettuata usando *funzioni di base*.

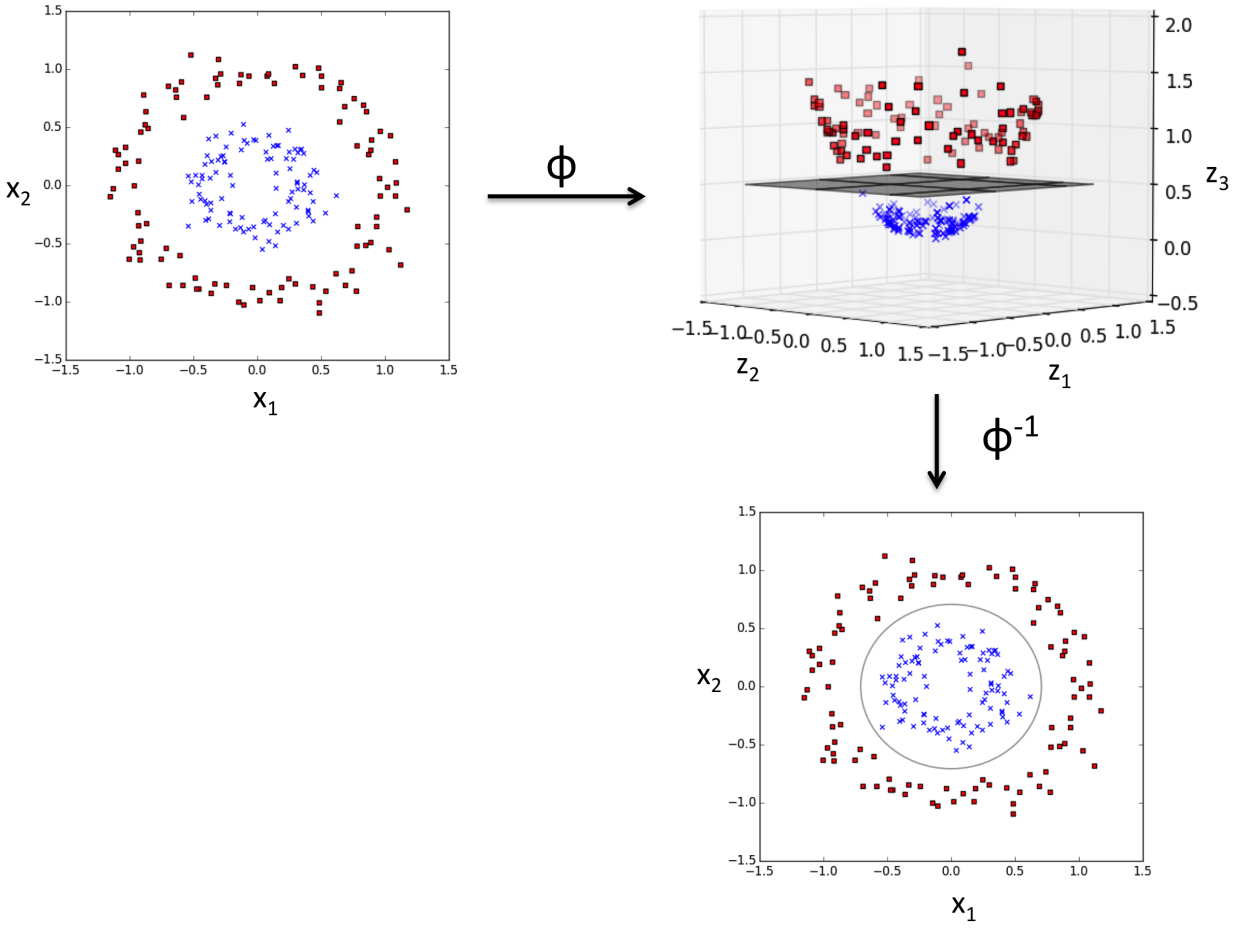


Figure 6: Utilizzo di una funzione di base per ottenere separabilità lineare

La nuova dimensione è calcolata attraverso le funzioni di base:

$$\mathbf{z} = \boldsymbol{\phi}(\mathbf{x}) \; dove \; z_j = \phi_j(\mathbf{x}), \; j = 1, ..., k$$

In questo modo possiamo fare un mapping dallo spazio d dimensionale $\mathbf{x}$ ad uno spazio k-dimensionale $\mathbf{z}$ per questo possiamo definire il discriminante come:

$$g(\mathbf{z}) = \mathbf{w}^T\mathbf{z} + w_0$$

$$g(\mathbf{x}) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + w_0$$

Di solito, $k \gg d$, cioè k è molto più grande di d e k solitamente è anche più grande di N, ovvero il numero di istanze.

Il problema è il medesimo visto nella sezione precedente, con le dovute modifiche ai vincoli. Pertanto avremo che:

$$L_p = \frac{1}{2}||\mathbf{w}||^2 + C\sum_t \xi^t \quad t.c. \quad r^t(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}^t) + w_0) \geq 1 - \xi^t$$

La funzione Langragiana sarà la seguente:

$$L_p = \frac{1}{2}||\mathbf{w}||^2 + C\sum_t \xi^t - \sum_t \alpha^t[r^t(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}^t) + w_0) - 1 + \xi^t] - \sum_t \mu^t\xi^t$$

la quale appare essere la medesima vista nel caso del *soft margin*, con la dovuta sostituzione di $\mathbf{x}^t$ con $\boldsymbol{\phi}(\mathbf{x}^t)$.

Calcolando le derivate rispetto ai parametri e ponendole a 0, otterremo:

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} = \sum_t \alpha^t r^t \boldsymbol{\phi}(\mathbf{x}^t)$$

$$\frac{\partial L_p}{\partial \xi^t} = \quad 0 \quad = \quad C - \alpha^t - \mu^t$$

$$\frac{\partial L_p}{\partial w_0} \quad = \sum_t a^t r^t \quad = \quad 0$$

La forma *duale* ottenuta sarà:

$$L_d = \sum_t \alpha^t - \frac{1}{2}\sum_t\sum_s \alpha^t\alpha^s r^t r^s \boldsymbol{\phi}(\mathbf{x}^t)^T\boldsymbol{\phi}(\mathbf{x}^s) \qquad t.c. \quad \sum_t \alpha^t r^t = 0 \;\; e \;\; 0 \leq \alpha^t \leq C, \; \forall t$$

L'idea delle $K - SVM$, cioè delle macchine a vettori di supporto basate su kernel, è quella di rimpiazzare il prodotto fra $\boldsymbol{\phi}(\mathbf{x}^t)^T\boldsymbol{\phi}(\mathbf{x}^s)$ con una *funzione kernel* $K(\mathbf{x}^t, \mathbf{x}^s)$: anzichè mappare due istanze $\mathbf{x}^t$ e $\mathbf{x}^s$ nello spazio z-dimensionale e poi calcolare il prodotto , applichiamo direttamente la funzione kernel nello spazio originale. Pertanto sostituendo $K(\mathbf{x}^t, \mathbf{x}^s)$ al posto del prodotto delle funzioni di base $\boldsymbol{\phi}(\mathbf{x}^t)^T\boldsymbol{\phi}(\mathbf{x}^s)$, otterremo che:

$$L_d = \sum_t \alpha^t - \frac{1}{2}\sum_t\sum_s \alpha^t\alpha^s r^t r^s K(\mathbf{x}^t, \mathbf{x}^s)$$

Inoltre, sapendo che il discriminante è $g(\mathbf{x}) = \mathbf{w}^t\boldsymbol{\phi}(\mathbf{x}) + w_0$, possiamo riscriverlo come:

$$g(\mathbf{x}) \quad = \quad \sum_t \alpha^t r^t \boldsymbol{\phi}(\mathbf{x}^t)^T\boldsymbol{\phi}(\mathbf{x}) + w_0 \quad = \quad \sum_t \alpha^t r^t K(\mathbf{x}^t, \mathbf{x}) + w_0$$

La soluzione al problema è data da:

$$\hat{g}(\mathbf{x}) = \sum_t \hat{\alpha}_i r^t K(\mathbf{x}^t, \mathbf{x}) + \hat{w_0}$$

Possiamo notare di come possiamo applicare la funzione *kernel* direttamente nello spazio originale, con una conseguente miglior semplicità di utilizzo, rispetto al calcolo di $\boldsymbol{\phi}(\mathbf{x}^t)$ e di $\boldsymbol{\phi}(\mathbf{x})$ e del prodotto scalare fra le due funzioni di base.

La matrice contenente le funzioni *kernel* è chiamata *"Matrice di Gram"* ed è una matrice **simmetrica semi-definita positiva**.

Richiamo *Matrice semi-definita positiva*

Sia A una matrice quadrata, $A \in \mathbb{R}^{n \times n}$, simmetrica, $A^T = A$, allora diremo che essa è simmetrica e semi-definita positiva se il suo spettro contiene *autovalori* positivi o al più uguali a zero, ovvero:

$$\sigma(A) = \{\lambda \in \mathbb{R} \mid p(\lambda) = 0 \; ove \; \lambda_i \geq 0\}$$

dove $p(\lambda)$ denota il *polinomio caratteristico*.

1.3.1 Le differenti tipologie di Kernel

Esistono differenti tipologie di *kernel* utilizzabili, note in letteratura:

1. *Kernel polinomiale di grado q*

   Il kernel polinomiale di grado q è definito come:

   $$K(\mathbf{x}^t, \mathbf{x}) = (\mathbf{x}^T \mathbf{x}^t + 1)^q$$

   dove il parametro q è definito dall'utente.

   Esempio 1. Applicazione del kernel polinomiale di grado q, con vettori a *dimensionalità* 2

   Siano:

   $$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

   $$K(x, z) = (\mathbf{x}^T \mathbf{z} + 1)^2 = (x_1 z_1 + x_2 z_2)^2$$

   $$= x_1^2 z_1^2 + x_2^2 z_2^2 + 1 + 2x_1 x_2 z_1 z_2 + 2x_1 z_1 + 2x_2 z_2$$

Considerando che :

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

$$\boldsymbol{\phi}(\mathbf{z}) = [1, \sqrt{2}z_1, \sqrt{2}z_2, \sqrt{2}z_1z_2, z_1^2, z_2^2]$$

Possiamo concludere che:

$$\boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{z}) = K(\mathbf{x}, \mathbf{z})$$

2. *Funzioni di base radiale*

   L'utilizzo di funzioni di base radiale ci permette di ottenere una alta efficienza dal punto di vista della separabilità. Una funzione di base radiale è una funzione a variabili reali il cui valore dipende solo dalla distanza dall'origine. È definita come segue:

$$K(\mathbf{x}^t, \mathbf{x}) = \exp\left[-\frac{||\mathbf{x}^t - \mathbf{x}||^2}{2s^2}\right]$$

3. *Funzioni sigmoidali*

   Una funzione sigmoidale è definita da:

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

Riassumiamo il tutto nella figura seguente:

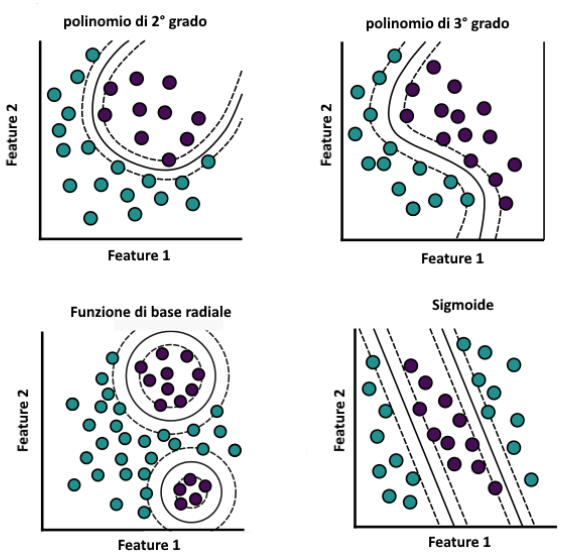


Figure 7: Differenze fra le varie funzioni *kernel*

1.3.2 Definizione di Kernel per il specifico dominio applicativo

È possibile definire, talvolta, kernel specifici per un peculiare contesto applicativo:

Information Retrieval

Supponiamo di avere due documenti D_1 e D_2 costituiti da un insieme di parole, meglio noto col termine *bag of words*, in cui un documento può essere visto come un vettore M-dimensionale contenente per ogni elemento la frequenza del termine i-esimo, misurata attraverso la *term frequency-inverse document frequency* (*tf-idf*).

Possiamo definire la funzione di base $\boldsymbol{\phi}(D_1)$ come il vettore binario contenente M-elementi, in cui ogni elemento avrà valore 1 se la parola i-esima appare nel documento D_1 e 0 altrimenti; pertanto, il prodotto fra $\boldsymbol{\phi}(D_1)^T\boldsymbol{\phi}(D_2)$ sarà il conteggio delle parole **condivise** dai due documenti.

Possiamo pensare di definire e implementare $K(D_1, D_2)$ come il numero di parole condivise: in questo modo non dovrò fare la binarizzazione del documento e fare il prodotto fra le due funzioni di base, ovvero, ad esempio:

$$D_1 = (2, 1, 0, 3, 0) \;\rightarrow\; \phi(D_1) = (1, 1, 0, 1, 0)$$

$$D_2 = (8, 0, 2, 1, 0) \;\rightarrow\; \phi(D_1) = (1, 0, 1, 1, 0)$$

$$\phi(D_1)^T\phi(D_2) = 2$$

che corrisponde al numero di parole in comune fra i due documenti.

Con il kernel trick $K(D_1, D_2)$ non avrò bisogno di generare la rappresentazione *bag of words* in modo esplicito, proprio perchè il kernel definito corrisponde al numero di parole presenti in entrambi i documenti.

Il kernel spettrale nel contesto della *bioinformatica*

Nell'ambito della bioinformatica potremmo avere la necessità di calcolare la similarità fra oggetti, sequenze di geni in questo caso.

In particolare, il kernel spettrale ci permette di contare le sottosequenze comuni di lunghezza k fra due stringhe aventi lunghezza M. Consideriamo due stringhe di geni x^t e x^s e definiamo le *k-mers*, ovvero le stringhe di lunghezza k ottenibili dalle due stringhe di geni.

L'obbiettivo è contare le *k-mers* in x^t e x^s ed effettuare la somma dei prodotti dei conteggi appena calcolati. Vediamo questo nell'esempio seguente, in cui $k = 2$:

Esempio 2. Calcolo del kernel spettrale

Siano x^t = AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
x^s = TACCTAATTATGAAATTAAATTTCAGTGTGCTGATGGAAACGGAGAAGTC

due sequenze di geni.

3-mer	AAA	AAC	...	CCA	CCC	...	TTT
# *in* x^t	2	4	...	1	0	...	3
# *in* x^s	3	1	...	0	0	...	1

Sulle righe possiamo notare il conteggio della sottosequenza nella particolare stringa genomica.

Il kernel effettuerà la somma dei prodotti degli elementi sulle colonne:

$$K(x^t, x^s) = 2 \cdot 3 + 4 \cdot 1 + \ ... \ + 1 \cdot 0 + 0 \cdot 0 \ + 3 \cdot 1$$

Altre tipologie di kernels che possono essere definite sono: il *kernel di Fisher* e il *kernel di diffusione*; tuttavia è possibile costruire nuovi kernels effettuando una combinazione di kernels più semplici. Ad esempio se $K_1(\mathbf{x}, \mathbf{y})$ e $K_2(\mathbf{x}, \mathbf{y})$ sono kernels validi e $c \in \mathbb{R}$ è una costante, possiamo dire che anche i seguenti kernels sono kernels validi:

$$K(\mathbf{x}, \mathbf{y}) = \begin{cases} cK_1(\mathbf{x}, \mathbf{y}) \\ K_1(\mathbf{x}, \mathbf{y}) + K_2(\mathbf{x}, \mathbf{y}) \\ K_1(\mathbf{x}, \mathbf{y}) \cdot K_2(\mathbf{x}, \mathbf{y}) \end{cases}$$

1.4 Il caso multiclasse

In alcuni casi potremmo incorrere in datasets che contengono un numero di classi maggiore di 2 ($K > 2$). Una modalità con cui è possibile affrontare tale problema è quella di definire K problemi a due classi, ognuno che separi una classe da tutte le altre e che apprenda K SVM $g_i(\mathbf{x})$ per $i = 1, ..., K$. Per cui quello che viene fatto è costruire K classificatori SVM a due classi per separare una classe da tutte le altre.

Durante la fase di training, dunque, gli esempi di classe C_i saranno etichettati con $+1$, mentre esempi di classe C_k (per $k \neq i$) con -1.

Nella fase di testing calcoleremo tutte le $g_i(x)$ e sceglieremo il massimo.

Quest'approccio è solitamente quello più utilizzato.

Un'altro approccio consiste nella cosiddetta *"pairwise separation"* che consiste nell'andare a trovare un iperpiano di separazione per ogni coppia di classe, difatti ci sono classi che non sono separabili linearmente, ma lo sono a coppie. Ogni $g_{ij}(\mathbf{x})$ assumerà valore $+1$ per istanze di classe C_i e assumerà valore -1 per istanze di classe C_j.

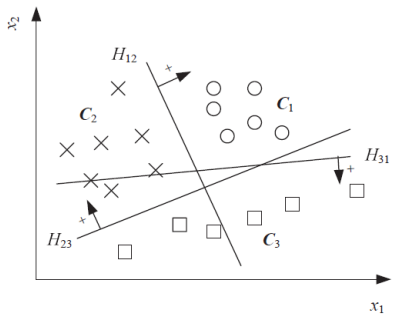


Figure 8: *Pairwise separation*

SVM monoclasse, outlier detection → Le SVM possono essere anche utilizzate per il ritrovamento di *outlier*, ovvero valori anomali: infatti, data una sfera di centro a e di raggio R, possiamo pensare di racchiudere quante più istanze possibili in tale sfera, andando alla ricerca del raggio più piccolo possibile che le racchiuda. Possiamo visualizzare questo aspetto nella $Fig.9$:

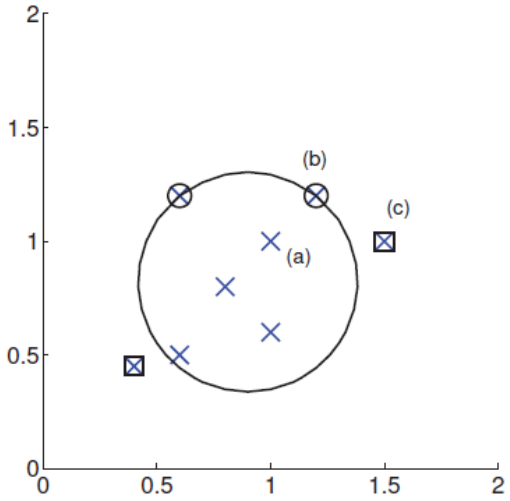


Figure 9: Ritrovamento di valori anomali (*outlier*)

1.5 Le SVM utilizzate nella regressione

Finora ci siamo focalizzati sul task di classificazione nelle SVM. Ora diamo dei brevi cenni al caso in cui il nostro problema sia di regressione. La risoluzione del problema è analoga a quella della classificazione, visto in dettaglio dal punto di vista analitico in precedenza. Il modello di regressione lineare presuppone l'esistenza di una relazione, appunto, lineare fra i dati:

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

Nella regressione utilizziamo l'errore quadratico per verificare la bontà del modello, cioè:

$$E(r^t, f(\mathbf{x}^t)) = [r^t - f(\mathbf{x}^t)^2]$$

che è la differenza degli tra il valore di output osservato ed il valore calcolato dal modello.

Nella regressione SVM è utilizzata la funzione $\epsilon - sensitive$ che ci permette di avere un effetto lineare e non quadratico, come si può evincere dalla $Fig.10$

Al fine di stimare il valore di $\mathbf{w}$, occorre calcolare:

$$min \frac{1}{2}||\mathbf{w}||^2 + C \sum_t (\xi_+^t + \xi_-^t)$$

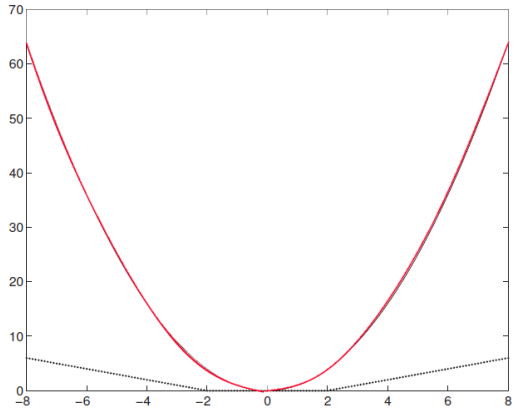


Figure 10: Errore quadratico(in rosso) vs. $\epsilon - sensitive$

con vincoli:

$$r^t - (\mathbf{w}^T\mathbf{x} + w_0) \leq \epsilon + \xi_+^t$$
$$(\mathbf{w}^T\mathbf{x} + w_0) - r^t \leq \epsilon + \xi_-^t$$
$$\xi_+^t, \xi_-^t \geq 0$$

Dove abbiamo una parte positiva ed una parte negativa delle variabili slack.

Il problema viene risolto allo stesso modo della classificazione, ovvero:

1. Calcolo della funzione Langragiana(L_p)
2. Calcolo delle derivate parziali della Langragiana(L_p), in questo caso rispetto ai parametri $\mathbf{w}$, w_0, ξ_+^t, ξ_-^t.
3. Calcolo della forma duale L_d, effettuando la sostituzione delle derivate calcolate al passo (3) alla L_p

Calcolando le derivate e la forma duale, otterremo i parametri α_-^t e α_+^t dai quali possiamo desumere che tutte le istanze con $\alpha_-^t = \alpha_+^t = 0$ sono quelle istanze che non troviamo sul margine, ma all'interno del margine, nel cosiddetto "tubo". I vettori di supporto sono caratterizzati da $\alpha_-^t > 0$ oppure da $\alpha_+^t > 0$ e possono essere di due tipologie differenti:

a) Possono trovarsi sul margine (in tal caso α_-^t o α_+^t saranno fra 0 e C) e useremo questi per calcolare w_0

b) Possono trovarsi al di fuori del "tubo" e sono caratterizzati dalla presenza di $\alpha_+^t = C$

Riepiloghiamo questi concetti in $Fig.11$

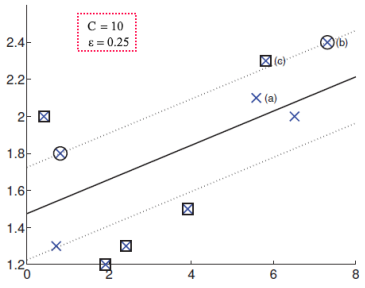


Figure 11: Regressione SVM

In questa figura possiamo notare di come l'istanza (a) sia all'interno del "tubo"($\alpha_+^t = 0$, $\alpha_-^t = 0$), l'istanza (b) sul margine (al limite) ($0 < \alpha_+^t < C$), l'istanza (c) al di fuori del "tubo" con un valore della variabile slack positivo($\xi_+^t > 0$). I vettori di supporto sono (b) e (c).

La soluzione del problema sarà data da:

$$\hat{f}(x) = \sum_t (\alpha_+^t - \alpha_-^t)(\mathbf{x}^t)^T \mathbf{x} + w_0$$

È possibile rimpiazzare nuovamente il prodotto con una funzione kernel $K(\mathbf{x}^t, \mathbf{x})$; i risultati ottenibili possiamo visualizzarli in *Fig.*12:

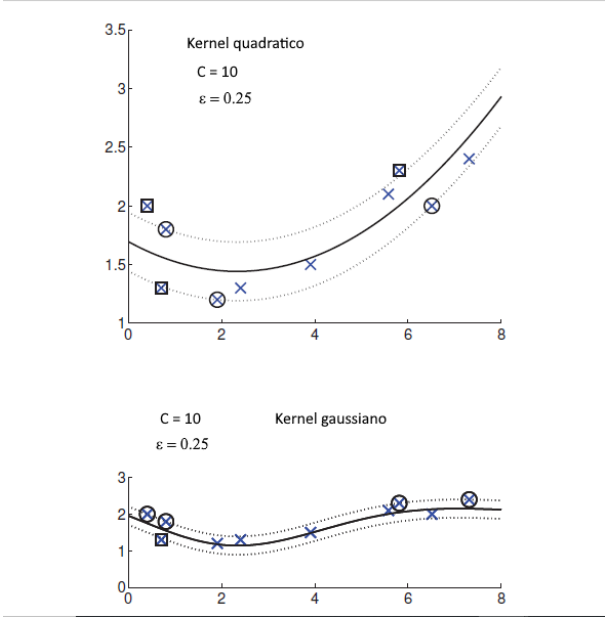


Figure 12: Kernel quadratico e gaussiano nella regressione

Nota: Le istanze cerchiate rappresentano i vettori di supporto sui margini, le istanze inquadrate sono i vettori di supporto *outliers*.

Anche in questo caso regressivo, allo stesso modo di quello classificativo, possiamo avere un tipo di regressione in cui l'iperparametro di complessità da settare è ν, quindi si parlerà di $\nu - regression$.

1.6 Riduzione della dimensionalità

Possiamo utilizzare i kernels per effettuare la riduzione della dimensionalità.

La PCA(*principal component analysis*) è un metodo *non supervisionato* che ci permette di ridurre la dimensionalità, ottenendo le cosiddette "componenti principali" tramite la risoluzione del sistema lineare $\mathbf{Y} = \mathbf{X} \cdot \mathbf{A_d}$, dove: $\mathbf{X}$ è la matrice dei dati centrati; $\mathbf{A_d}$ è la matrice contenente gli autovettori della matrice di covarianza (ottenuta sui dati centrati); mentre $\mathbf{Y}$ sono i cosiddetti "scores", ovvero le componenti principali ottenute.

In questo contesto, possiamo parlare di *Kernel PCA*, ovvero di una versione "kernelizzata" della PCA standard: difatti, tale tecnica utilizza gli autovettori e gli autovalori della matrice dei kernels, permettendoci una riduzione della dimensionalità lineare nello spazio $\phi(\mathbf{x})$.

Nella versione "kernelizzata", andremo a lavorare nello spazio $\boldsymbol{\phi}(\mathbf{x})$, piuttosto che nello spazio $\mathbf{x}$ originale. La procedura resta la medesima della PCA tradizionale, con la modifica della matrice da considerare: in questo caso dovremo lavorare con gli autovalori e autovettori della matrice $\boldsymbol{\phi}^T\boldsymbol{\phi}$ e quindi della matrice dei kernels K.

Nella *Fig.*13 seguente notiamo di come la *K-PCA* ci permetta di ottenere risultati migliori rispetto alla PCA standard:

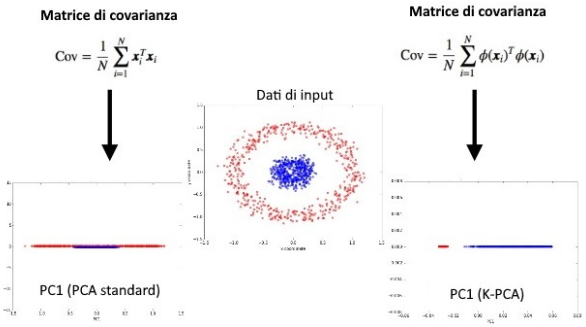


Figure 13: PCA tradizionale vs K-PCA

Riepilogo

Riassumiamo l'algoritmo SVM nei seguenti steps:

Fase di training

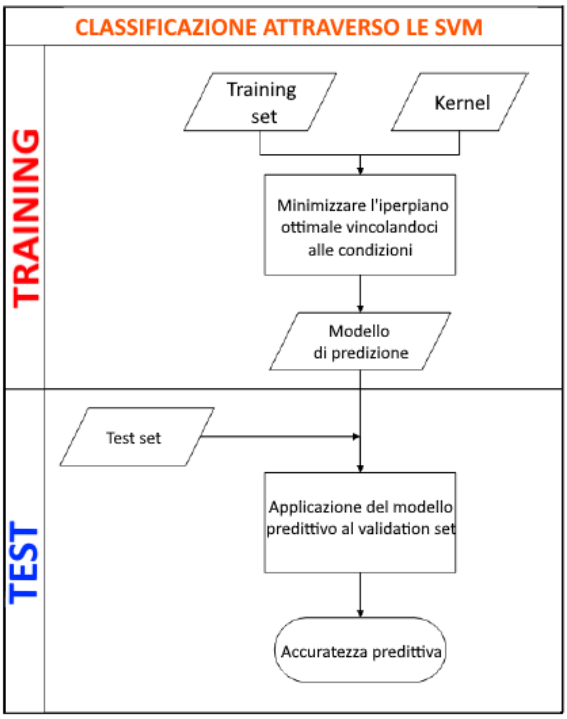
1. Input: Training set; funzione kernel
2. Procedura principale : minimizzare l'iperpiano ottimale

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_t \xi^t \qquad t.c. \qquad \xi^t \geq 0,\ r^t(\mathbf{w}^T\mathbf{x}^t + w_0) \geq 1 - \xi^t \ \ \forall t$$

3. Output: modello di predizione

Fase di testing

1. Input: test set, modello di predizione ottenuto nella fase di training
2. Procedura principale: applicazione del modello di predizione al test set
3. Output: accuratezza predittiva in percentuale


CLASSIFICAZIONE ATTRAVERSO LE SVM
TRAINING
Training set
Kernel
Minimizzare l'iperpiano ottimale vincolandoci alle condizioni
Modello di predizione
TEST
Test set
Applicazione del modello predittivo al validation set
Accuratezza predittiva

APPENDICE

Condizioni di Karush-Kuhn-Tucker

Le condizioni di Karush-Kuhn-Tucker sono usate per esprimere le variabili primali in funzione delle variabili duali; in questo modo la funzione Langragiana diventa esclusiva delle variabili duali e sarà massimizzata rispetto a queste variabili.

Dato il seguente problema di ottimizzazione:

$$\min_{x \in \mathbb{R}^l} f(x) \quad \textit{con vincoli } g_i(x) \leq 0 \;\; i = ...,n \;\; h_j(x) = 0 \qquad j = 1,...,m \quad 1 \leq m \leq p-1$$

Il teorema di Karush-Kuhn-Tucker afferma che:

Sia $\tilde{x}$ un punto di minimo locale per f per cui soddisfi le condizioni precedenti. Siano f e g_i, per i = 1, . . . , n funzioni differenziabili. Esiste allora un vettore $\tilde{\lambda} \geq 0$ dove $\tilde{\lambda} = (\tilde{\lambda}_1, \tilde{\lambda}_2, ..., \tilde{\lambda}_n)^T$, per cui:

$$\mathcal{L}(\tilde{x}, \tilde{\lambda}) = \nabla f(\tilde{x}) + \sum_{i=1}^{n} \tilde{\lambda}_i \nabla g_i(\tilde{x}) = 0 \qquad (\star)$$

$$\tilde{\lambda}_i g_i(\tilde{x}) = 0 \quad i = 1, ..., n$$

L'equazione $\star$ prende il nome di **condizione di Karush-Kuhn-Tucker**.