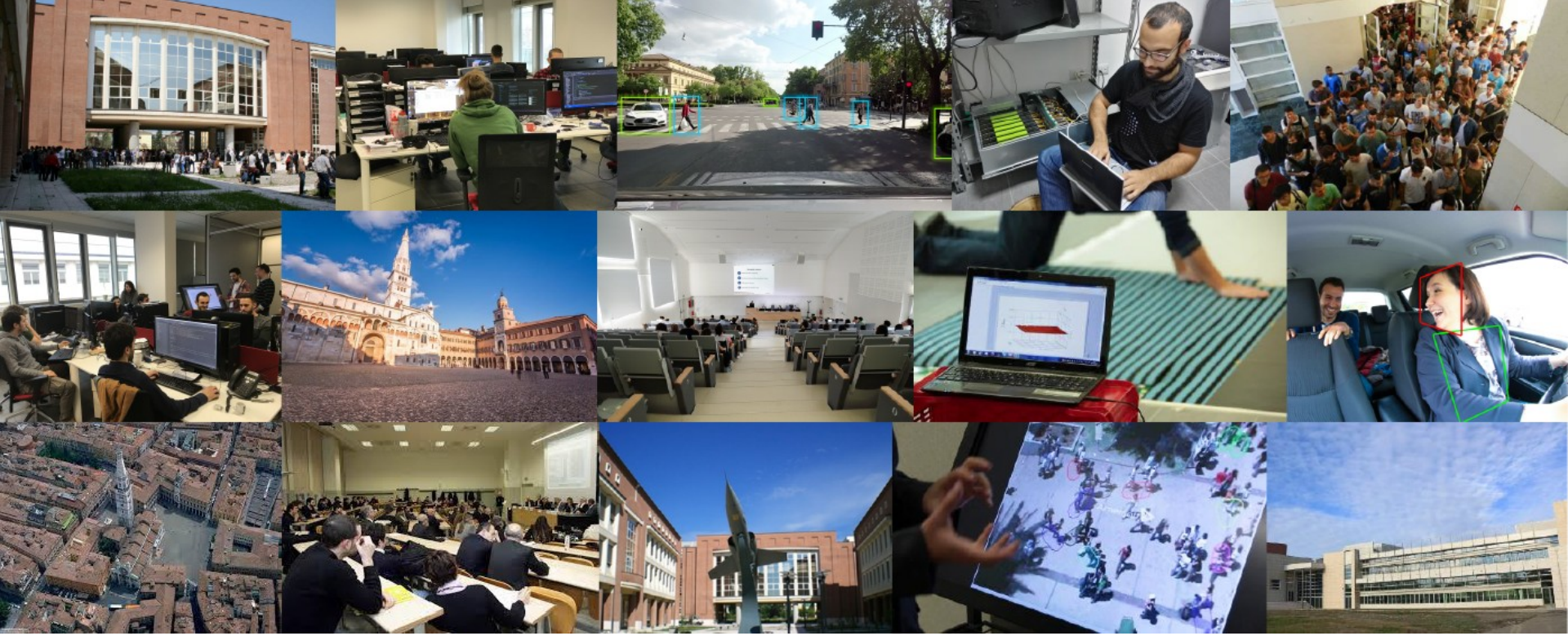


MACHINE LEARNING
- Support Vector Machine -

AImage Lab

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Notazione

x_j, j = 1, …, d	→	feature
$\mathbf{x} = [x_1, ..., x_d]$	→	feature vector
y	→	variabile target
$T = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$	→	dataset di training
$[w_0, w_1, \ldots, w_d] = [w_0, \mathbf{w}]$	→	vettore dei parametri
$[v_0, v_1, \ldots, v_d] = [v_0, \mathbf{v}]$	→	vettore dei parametri
$\mathbf{w}^T$	→	vettore **w** trasposto
$(\mathbf{x}^{(i)})^T$	→	vettore $\mathbf{x}^{(i)}$ trasposto
$X = R^d$	→	feature space
$W = R^{d+1}$	→	parameter space

Support Vector Machine

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Oggi parleremo di **Support Vector Machine** (**SVM**), che sono dei metodi discriminativi parametrici molto robusti rispetto al problema dell'overfitting (interpretabili anche come non-parametrici)

Le SVM sono dei classificatori ma possono essere estese per trattare task di regressione

La loro caratteristica è una definizione basata sugli aspetti geometrici dei loro bordi di decisione

Prima di trattare le SVM, però, è necessario approfondire l'interpretazione geometrica dei classificatori parametrici e dei loro decision boundary

Classificatori con score function lineare

Il classificatore (decision rule inclusa) della logistic regression può essere espresso come:

$C(\boldsymbol{x})$ = **If** $g(\boldsymbol{w}^T \boldsymbol{x} + w_0) > 0.5$ **then** y_1
else y_2

dove $[w_0, \boldsymbol{w}]$ sono i parametri, $g()$ è la logistic function e

$f_{[w0,\ \boldsymbol{w}]}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$

la chiameremo *score function* (in analogia con le score function della Softmax regression)
Dato $\boldsymbol{x}$, il suo «score» $f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$ è, appunto, un «punteggio» che $f()$ associa ad $\boldsymbol{x}$ (più alto è il punteggio, più «probabile» è l'appartenenza alla classe y_1 secondo $f()$). Lo score viene mappato in un valore di probabilità usando $g()$

La parte parametrica, ovvero la score function $f_{[w0,\ \boldsymbol{w}]}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$, è lineare

Esaminiamone le proprietà geometriche. Per semplicità, faremo a meno della funzione logistica $g()$ e ci concentreremo solo sulla parte parametrica lineare

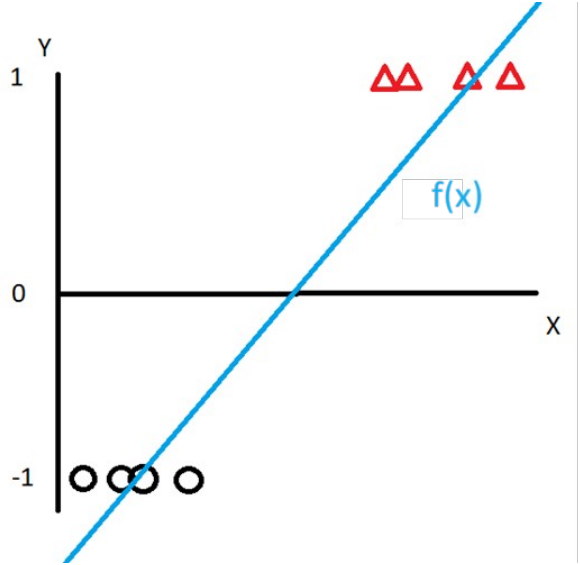
Classificatori Lineari

Consideriamo un task di classificazione binaria e tralasciamo il requisito di voler ottenere una stima di probabilità in output

Non siamo quindi più costretti a rappresentare la ground truth (*y*) con 0 o 1, perciò useremo etichette positive e negative per la variabile target: $y \in \{-1,1\}$ (cosa che semplifica la trattazione successiva)

Un *classificatore lineare* è basato su una decision rule e una *score function lineare* e può essere espresso come:

$C(\boldsymbol{x})$ = **If** $\boldsymbol{w}^T \boldsymbol{x} + w_0 > 0$ **then** y_1
else y_2



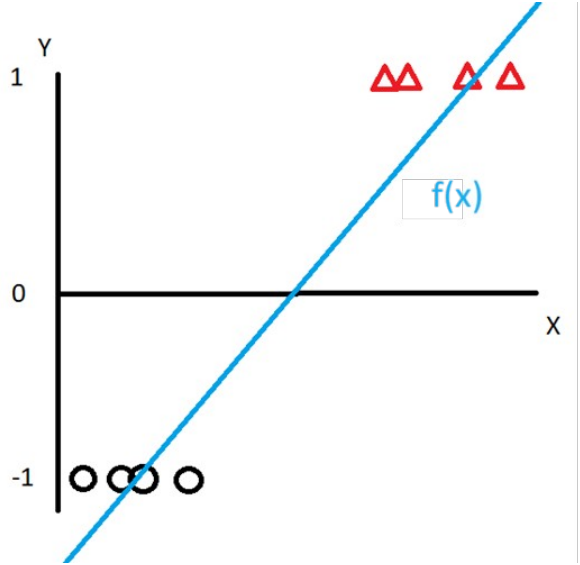
Classificatori Lineari

La parte parametrica di *C()*, ovvero:

$f_{[w0,\, w]}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$

(senza decision rule) è lineare, ovvero è un polinomio di primo grado e, quindi, geometricamente, corrisponde ad una funzione di predizione rappresentata da un (iper-)piano (lo dimostreremo formalmente tra un po'), esattamente come nel caso della linear regression: si tratta della stessa classe di funzioni!

Ma che forma ha il suo decision boundary?



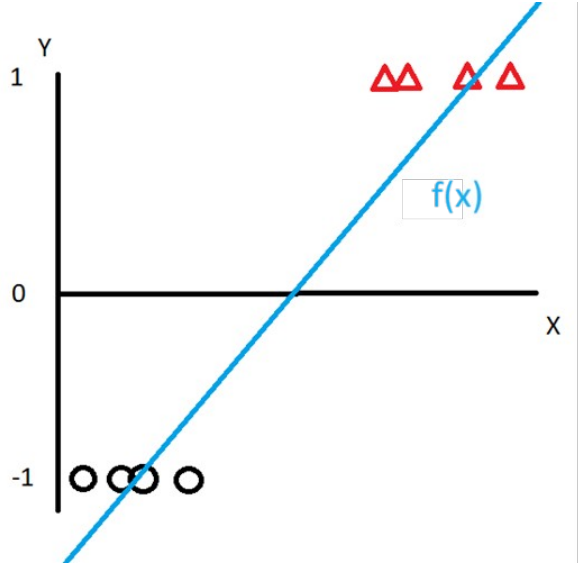
Classificatori Lineari

Il decision boundary (*H*) corrisponde ai punti in cui la score function assume valore 0, perché questi sono i punti del feature space in cui *C()* passa dal predire una classe al predire l'altra (sono i bordi delle decision region) :

$H = \{\boldsymbol{x} \mid f_{[w0,\ \boldsymbol{wj}]}(\boldsymbol{x}) = 0\}$

Se il feature space è composto da una sola feature, come nel grafico accanto, *H* è un punto

Nel caso generale di *d* feature, *H* è una (iper-)superficie in R^d



Classificatori Lineari: iperpiano di decisione

$H = \{\boldsymbol{x} \mid f_{[w0,\ \boldsymbol{w}]}(\boldsymbol{x}) = 0\}$

Per la precisione, H è un iperpiano. Dimostrazione:

Se $\boldsymbol{x}_1, \boldsymbol{x}_2 \in H$, allora:

$\boldsymbol{w}^T \boldsymbol{x}_1 + w_0 = \boldsymbol{w}^T \boldsymbol{x}_2 + w_0 = 0$

quindi:

$\boldsymbol{w}^T (\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$

Ricordandoci che $cos\ \theta = \boldsymbol{x}^T\boldsymbol{y} / ||\boldsymbol{x}||\ ||\boldsymbol{y}||$, ne deduciamo che l'angolo tra $\boldsymbol{w}$ e il vettore $(\boldsymbol{x}_1 - \boldsymbol{x}_2)$ è 90^0

Ne segue che $\boldsymbol{w}$ è ortogonale *ad ogni* vettore $(\boldsymbol{x}_1 - \boldsymbol{x}_2)$ giacente su H, quindi H è una superficie piana

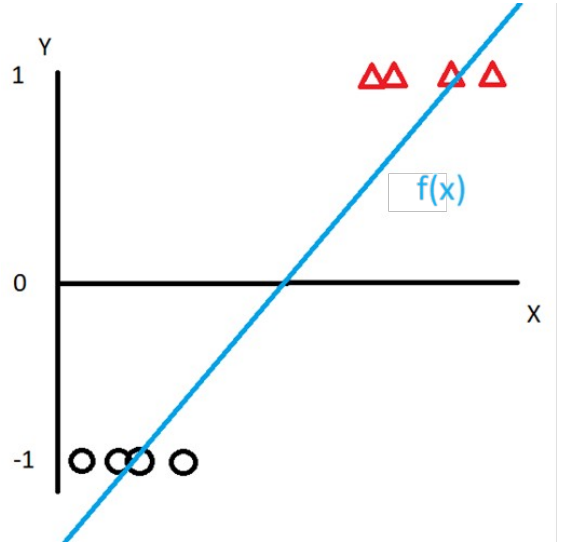
Classificatori Lineari: iperpiani corrispondenti

Un ragionamento analogo si può applicare all'equazione $f_{[w0,\ \mathbf{w}]}(\mathbf{x}) = y$, dimostrando che la superficie definita da $F = \{(\mathbf{x},y) \in R^{d+1} \mid f_{[w0,\ \mathbf{w}]}(\mathbf{x}) = y\} = \{(\mathbf{x},y) \in R^{d+1} \mid f_{[w0,\ \mathbf{w}]}(\mathbf{x}) - y = 0\}$ è un'iper-piano in R^{d+1}, dove R^{d+1} è lo spazio comprendente sia le feature ($\mathbf{x}$) che la variabile target (y)

Quando $f(\mathbf{x})$ interseca lo spazio delle feature alle coordinate $y = 0$, si ottiene H, che è anch'esso un iper-piano (in R^d)

Esempi:

- Se $d=1$, $f(\mathbf{x})$ è una retta in R^2 e H è un punto in R
- Se $d=2$, $f(\mathbf{x})$ è un piano in R^3 e H è una retta in R^2
- Se $d=3$, $f(\mathbf{x})$ è un iper-piano in R^4 e H è un piano in R^3

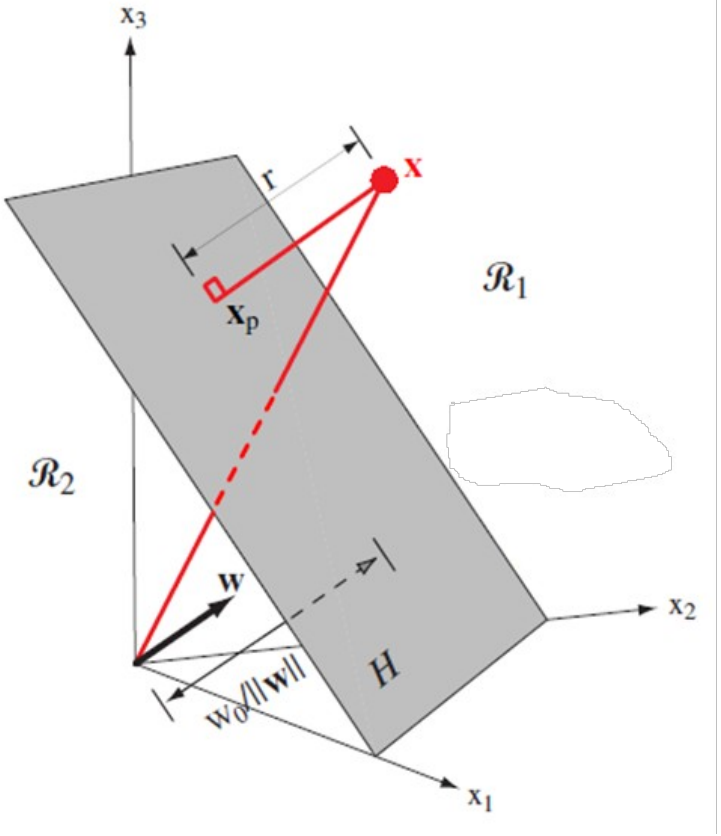


Classificatori Lineari: distanza dall'iperpiano di decisione

Si può dimostrare inoltre che la distanza *r* di un generico punto ***x*** da *H* è data da:

$r = |f(\mathbf{x})| / ||\mathbf{w}|| = |\mathbf{w}^T \mathbf{x} + w_0| / ||\mathbf{w}||$

dove $||\mathbf{w}||$ *è la norma* L_2 *di* ***w***



Classi linearmente separabili

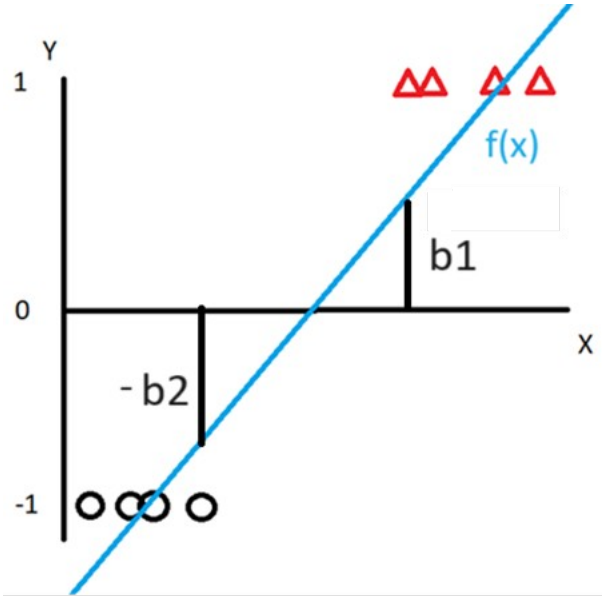
Due classi di punti si dicono «linearmente separabili» quando esiste un iper-piano *H* che li separi o, equivalentemente, quando esiste un classificatore lineare che separi completamente le due classi

Matematicamente, ciò è dato dall'esistenza di una score function $f_{[w0,\ \mathbf{w}]}(\mathbf{x})$ e di due valori reali positivi b_1 e b_2 tali che:

Per ogni $(\mathbf{x}^{(i)}, y^{(i)}) \in T$:

- $f_{[w0,\ \mathbf{w}]}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}^{(i)} + w_0 >= b_1$ iff $y^{(i)} = +1$
- $f_{[w0,\ \mathbf{w}]}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}^{(i)} + w_0 <= -b_2$ iff $y^{(i)} = -1$

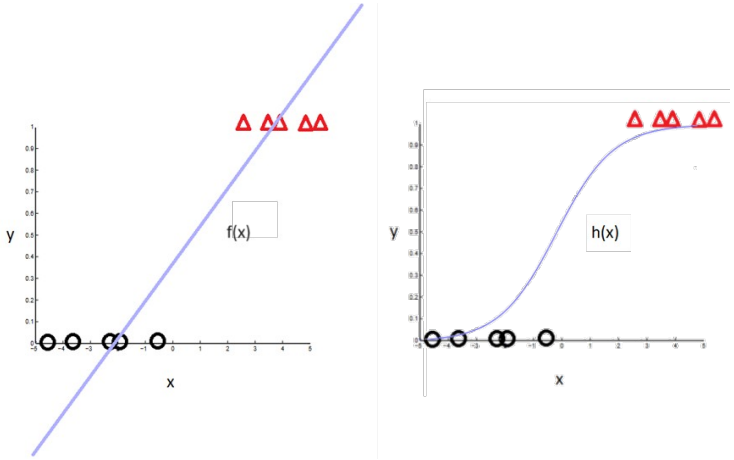
Se il bordo di decisione di $f_{[w0,\ \mathbf{w}]}(\mathbf{x})$ è a metà strada tra i punti delle due classi ad esso più vicini, allora $b_1 = b_2 = b$



Decision boundary della Logistic Regression

Nel caso della logistic regression la score function *f()* viene combinata con la logistic function *g()*, e la funzione ipotesi finale $h(\boldsymbol{x}) = g(f(\boldsymbol{x}))$ non è più lineare (nè lo è la supeficie corrispondente a $\{(\boldsymbol{x},y) \in R^{d+1} \mid h(\boldsymbol{x}) = y\}$)

Tuttavia, è possibile dimostrare che il decison boundary della logistic regression continua ad essere un iper-piano



Classificatori Lineari: ricerca dei parametri

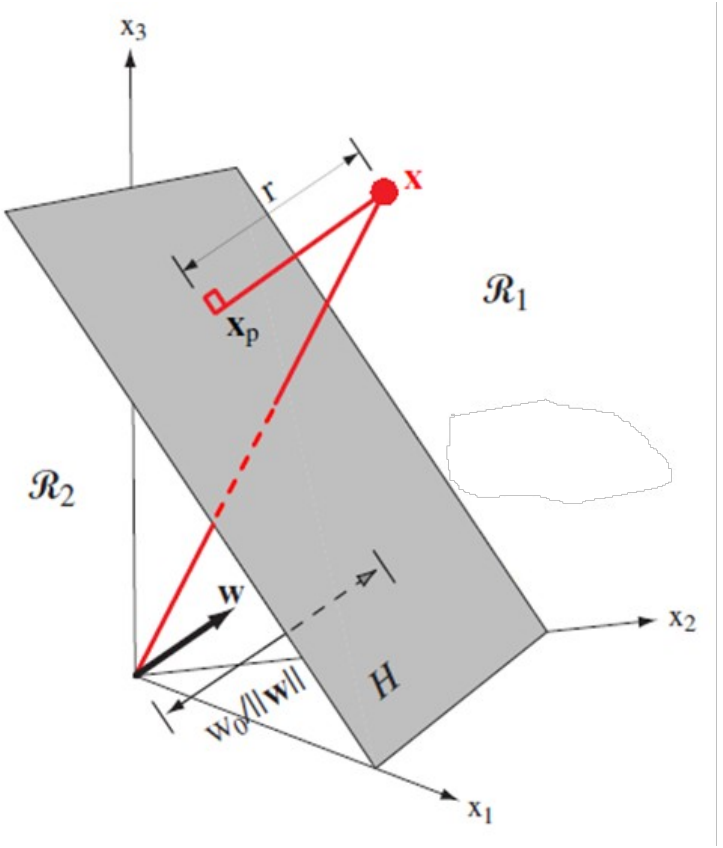
Come trovare [w_0, $\boldsymbol{w}$]?

Abbiamo visto che, nel caso della logistic regression, *g()* ha la funzione di convertire gli score in valori di probabilità

Ciò permette di definire una (log-)likelihood function, massimizzando la quale troviamo [w_0, $\boldsymbol{w}$]

Nel caso delle SVM, invece, la ricerca del «miglior» valore per [w_0, $\boldsymbol{w}$] è effettuata diversamente e si basa sulla distanza *r* dei punti del training set rispetto ad un generico *H*, dove *H* è il decision boundary della funzione ipotesi (dettagli tra un po'…)

Ora possiamo introdurre le SVM



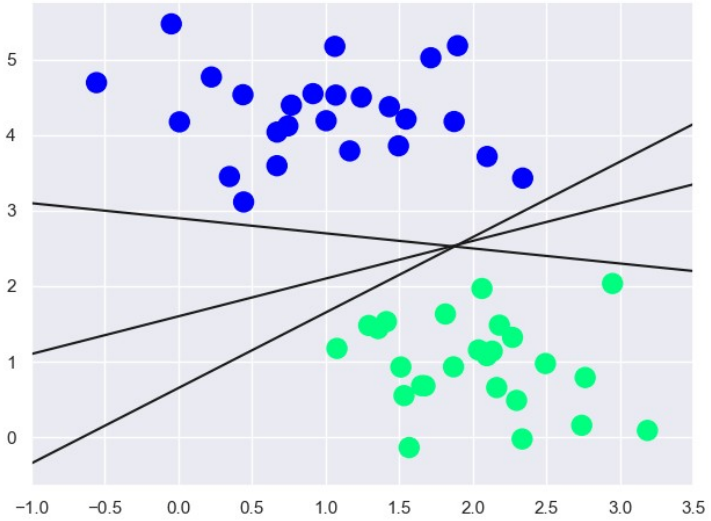
Support Vector Machine

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Le **Support Vector Machine** (**SVM**) sono dei classificatori *binari lineari* che cercano l'iperpiano separatore tra due classi con distanza massima tra gli esempi più vicini delle due classi

Intuitivamente, l'iperpiano con distanza massima dagli esempi di training delle due classi è quello che probabilmente generalizza meglio a inference time

Le SVM possono essere estese per problemi non binari e trattare classi non linearmente separabili (e anche task di regressione)



SVM: trovo i parametri impostando un problema geometrico

Nel caso delle SVM la ricerca di [w_0, $\boldsymbol{w}$] viene impostata come un problema geometrico

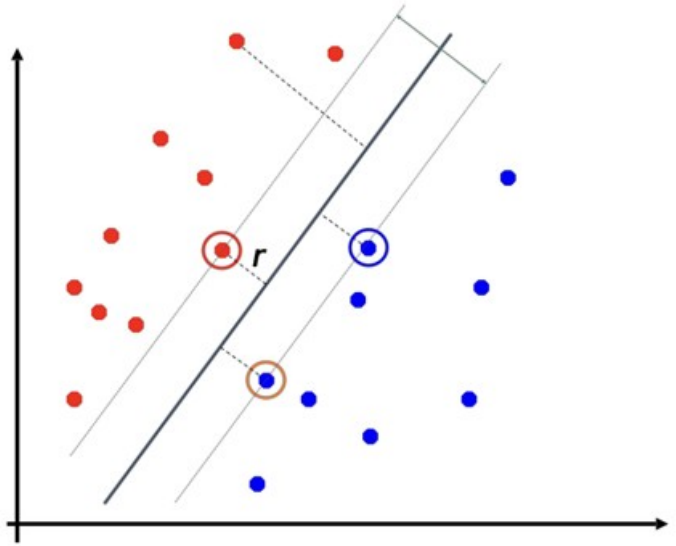
L'obiettivo è trovare l'iperpiano *H* che meglio separi le due classi

Il «miglior» *H* possibile è definito come quello che massimizza il «margine», ovvero la distanza dagli esempi $\boldsymbol{x}_s$ ($s = 1, \ldots k$) più vicini ad *H*

Tali esempi «più vicini ad *H*» sono chiamati *vettori di supporto* (cerchiati nella figura a lato)

N.B.: Di solito $k < n$ ($n = |T|$)

Nelle SVM assumo che *H* sia situato a metà strada tra i support vector delle due classi



SVM: problema geometrico

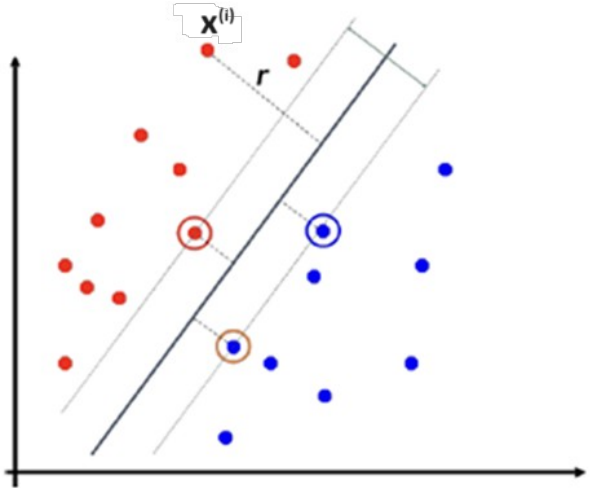
Dato che le SVM non modellano delle distribuzioni di probabilità, posso usare valori diversi da 0 e 1 per la ground truth di *y*. Assumeremo che $y \in \{-1,1\}$

Se *H* esiste, ovvero le due classi sono *lineramente separabili*, allora posso assumere che esita un classificatore lineare $C(\boldsymbol{x})$ basato su una score function che assegni valori positivi/negativi ad ogni $(\boldsymbol{x}^{(i)}, y^{(i)}) \in T$ a seconda del valore di $y^{(i)}$:

Per ogni $(\boldsymbol{x}^{(i)}, y^{(i)}) \in T$:

- $\boldsymbol{v}^T \boldsymbol{x}^{(i)} + v_0 >= b$ iff $y^{(i)} = +1$
- $\boldsymbol{v}^T \boldsymbol{x}^{(i)} + v_0 <= -b$ iff $y^{(i)} = -1$

Dove *b* è un valore positivo qualsiasi e $[v_0, \boldsymbol{v}]$ sono i parametri che devo cercare



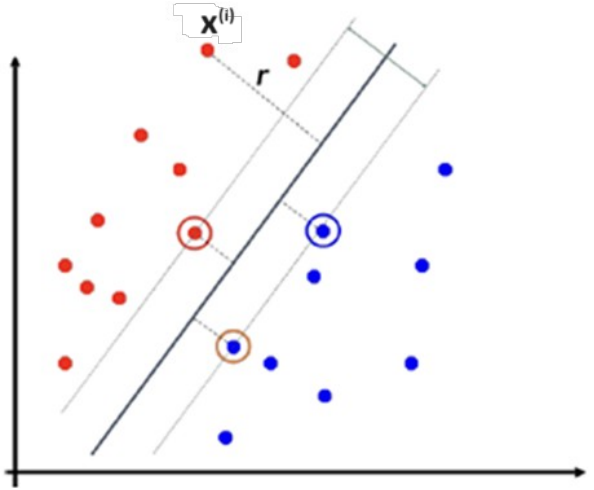
SVM: problema geometrico

Tuttavia, se uso: $\boldsymbol{w}^T = \boldsymbol{v}^T / b$ e $w_0 = v_0 / b$ posso semplificare i calcoli ed assumere:

- $\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0 >= 1$ iff $y^{(i)} = +1$
- $\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0 <= -1$ iff $y^{(i)} = -1$

Ovvero, per ogni $(\boldsymbol{x}^{(i)}, y^{(i)}) \in T$:

$y^{(i)} (\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0) >= 1$

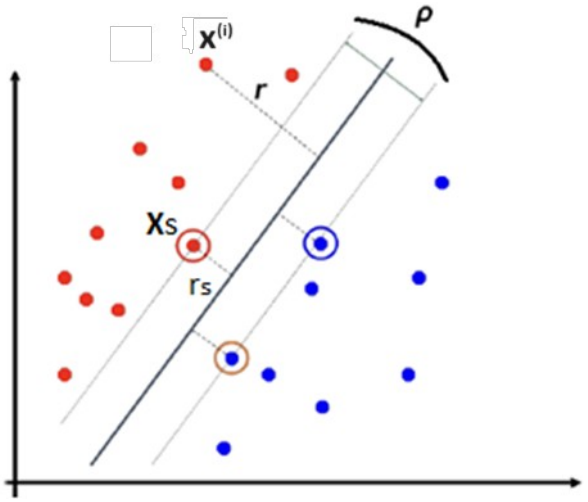


SVM: margine

Per i support vector $\boldsymbol{x}_s$ la diseguaglianza diventa un'eguaglianza (ricordiamoci che i support vector di ambo le classi sono a distanza uguale da *H* per definizione e quindi hanno lo stesso score):

$y_s (\boldsymbol{w}^T \boldsymbol{x}_s + w_0) = 1$

Per cui abbiamo che $f(\boldsymbol{x}_s) = \boldsymbol{w}^T \boldsymbol{x}_s + w_0 = 1 / y_s = y_s$
($y_s = 1$ o $y_s = -1$)



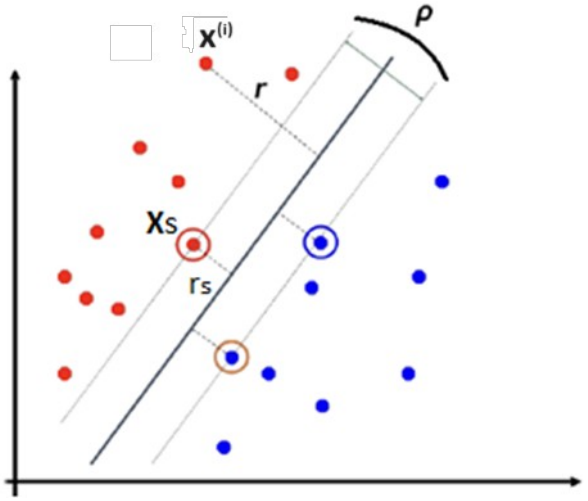
SVM: margine

Ne segue che la distanza (r_s) di $\boldsymbol{x}_s$ da *H* sarà:

$$r_s = \frac{|\boldsymbol{w}^T \boldsymbol{x}_s + w_0|}{||\boldsymbol{w}||} = \frac{1}{||\boldsymbol{w}||}$$

Il *margine* è definito come due volte tale distanza (v. figura) :

$$\rho = 2r_s = \frac{2}{||\boldsymbol{w}||}$$

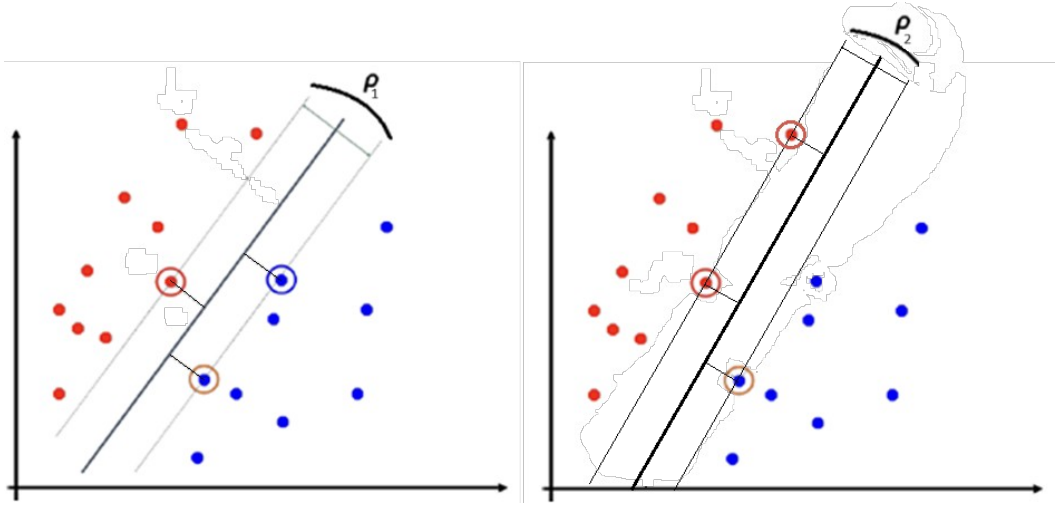


SVM: margine

$$\rho = 2r_s = \frac{2}{||\boldsymbol{w}||}$$

L'obiettivo è trovare la score function definita da [w_0, $\boldsymbol{w}$] il cui corrispondente margine sia massimo

Attenzione: cambiando [w_0, $\boldsymbol{w}$] cambia l'iper-piano e, quindi, anche il sottoinsieme di punti di T che costituiscono i support vector, come mostra la figura qui sotto



SVM: funzione obiettivo

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Usando la definizione di margine è possibile impostare la seguente funzione obiettivo:

$$[w_0^*, \boldsymbol{w}^*] = \arg \max_{[w_0, \boldsymbol{w}] \in W} \frac{2}{||\boldsymbol{w}||} \quad s.t.$$

$$y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0) \geq 1 \quad \forall (\boldsymbol{x}^{(i)}, y^{(i)}) \in T$$

⇦ Vincoli

o, equivalentemente:

$$[w_0^*, \boldsymbol{w}^*] = \arg \min_{[w_0, \boldsymbol{w}] \in W} \frac{1}{2}||\boldsymbol{w}||^2 \quad s.t.$$

$$y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0) \geq 1 \quad \forall (\boldsymbol{x}^{(i)}, y^{(i)}) \in T$$

⇦ Vincoli

SVM: primal solution

$$[w_0^*, \boldsymbol{w}^*] = \arg \min_{[w_0, \boldsymbol{w}] \in W} \frac{1}{2} ||\boldsymbol{w}||^2 \quad s.t.$$

$$y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0) \geq 1 \quad \forall (\boldsymbol{x}^{(i)}, y^{(i)}) \in T$$

⇦ Vincoli

Il problema di ottimizzazione così impostato è detto *soluzione primale*

La funzione obiettivo, quadratica rispetto a [w_0, **w**], è convessa, quindi ammette un unico minimo globale (potrebbe essere risolta in closed form solution)

Tuttavia, la presenza di *vincoli* ($\{y^{(i)} (\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0) >= 1\}_{i=1,\dots,n}$) rende l'ottimizzazione più difficile

Tipicamente la soluzione viene trovata utilizzando tecniche di ottimizzazione quadratica (detta anche *quadratic programming*)

SVM: dual solution

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Una formulazione alternativa è data introducendo i cosiddetti “moltiplicatori di Lagrange” ($\alpha_i >= 0$), uno per ogni esempio in *T*, i quali permettono di includere i vincoli direttamente nella funzione obiettivo, ottenendo la *soluzione duale*:

$$\boldsymbol{\alpha}^* = [\alpha_1^*, ..., \alpha_n^*] = \arg\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (\boldsymbol{x}^{(i)})^T \boldsymbol{x}^{(j)} \quad s.t.$$

$$\alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

Nella soluzione duale ho ancora dei vincoli, ma sono più semplici da risolvere

Inoltre, la soluzione duale ci permette di individuare i vettori di supporto, che sono gli $\boldsymbol{x}^{(i)}$ associati ad $\alpha_i > 0$

SVM: confronto tra soluzioni primale e duale

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Primale: $$[w_0^*, \boldsymbol{w}^*] = \arg \min_{[w_0, \boldsymbol{w}] \in W} \frac{1}{2} ||\boldsymbol{w}||^2 \quad s.t.$$

$$y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0) \geq 1 \quad \forall (\boldsymbol{x}^{(i)}, y^{(i)}) \in T$$

Duale: $$[\alpha_1^*, ..., \alpha_n^*] = \arg \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (\boldsymbol{x}^{(i)})^T \boldsymbol{x}^{(j)} \quad s.t.$$

$$\alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

- Entambe le soluzioni possono essere ottimizzate usando tecniche di ottimizzazione quadratica
- I metodi più recenti, tuttavia, utilizzano delle varianti del Gradient Descent/Ascent
- La soluzione primale consiste nel risolvere un problema di ottimizzazione in *d+1* variabili
- La soluzione duale consiste nel risolvere un problema di ottimizzazione in *n* variabili
- N.B.: nella soluzione duale i parametri della funzione ipotesi (score function) sono scomparsi…

SVM: modello predittivo (inference time)

A inference time, il classificatore finale sarà:

C(***x***) = **If** *f*(***x***) > *0* **then** y_1
else y_2

dove, usando la soluzione primale, avremo: $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$

che è un modello parametrico lineare

Usando invece la soluzione duale avremo : $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y^{(i)} (\boldsymbol{x}^{(i)})^T \boldsymbol{x} + w_0$

dove è possibile calcolare la sommatoria limitandosi ai soli $\alpha_i > 0$, ovvero usando solo i support vector

w_0, nel caso duale, può essere ricavato utilizzando un qualsiasi support vector $\boldsymbol{x}_s$ e sfruttando il fatto che $f(\boldsymbol{x}_s) = y_s$:

$$y_s = f(\boldsymbol{x}_s) = \sum_{i=1}^{n} \alpha_i y^{(i)} (\boldsymbol{x}^{(i)})^T \boldsymbol{x}_s + w_0$$

$$w_0 = y_s - \sum^{n} \alpha_i y_i (\boldsymbol{x}^{(i)})^T \boldsymbol{x}_s$$

SVM: modello predittivo (inference time)

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

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y^{(i)} (\boldsymbol{x}^{(i)})^T \boldsymbol{x} + w_0 = \sum_{i=1}^{n} \alpha_i y^{(i)} (\boldsymbol{x}^{(i)})^T \boldsymbol{x} + y_s - \sum_{i=1}^{n} \alpha_i y_i (\boldsymbol{x}^{(i)})^T \boldsymbol{x}_s$$

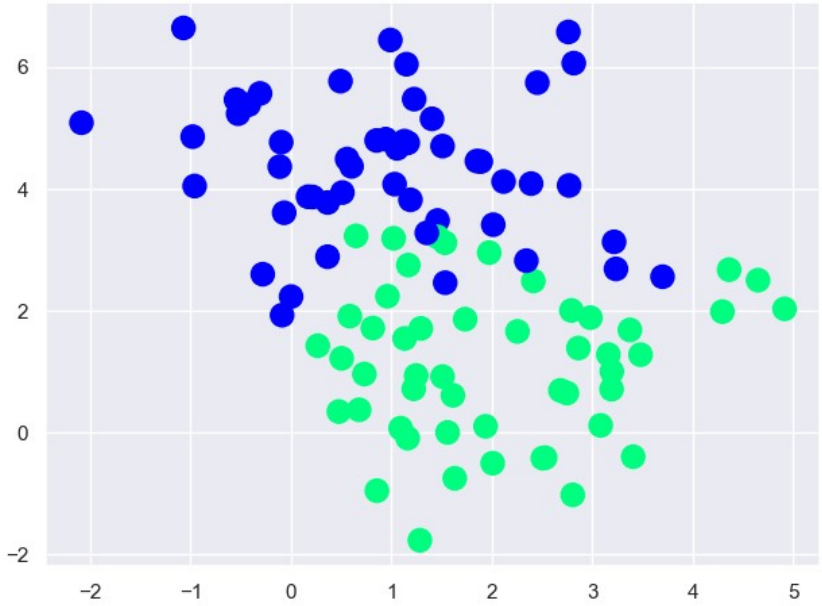
Le due soluzioni, primale e duale, sono equivalenti (e sono entrambi modelli parametrici lineari), per cui il vettore dei parametri ***w*** è tale che:

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \boldsymbol{x}^{(i)}$$

Classi parzialmente sovrapposte

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Quando le due classi sono parzialmente sovrapposte, come nella figura a lato, un iper-piano separatore non esiste



Classi parzialmente sovrapposte

Lasciamo allora che la SVM commetta un certo numero di «errori», violando i vincoli per alcuni samples (ma facendole pagare una penalità per ogni violazione):

$$w_0^*, \boldsymbol{w}^*, \xi_1^*, ..., \xi_n^* = \arg \min_{[w_0, \boldsymbol{w}] \in W, \xi_i \geq 0} \frac{1}{2}||\boldsymbol{w}||^2 + C \sum_{i=1}^{n} \xi_i \quad s.t.$$

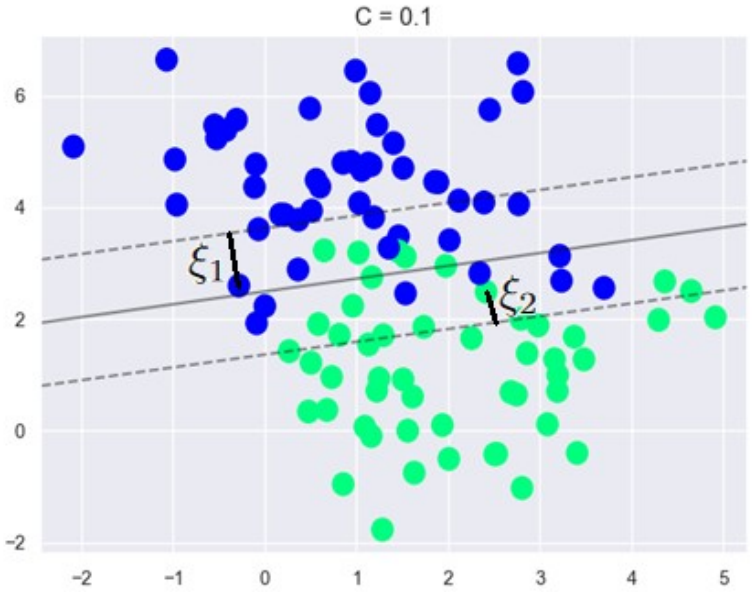
$$y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0) \geq 1 - \xi_i \quad \forall (\boldsymbol{x}^{(i)}, y^{(i)}) \in T$$

Confronto con la soluzione primale senza penalità:

$$[w_0^*, \boldsymbol{w}^*] = \arg \min_{[w_0, \boldsymbol{w}] \in W} \frac{1}{2}||\boldsymbol{w}||^2 \quad s.t.$$

$$y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0) \geq 1 \quad \forall (\boldsymbol{x}^{(i)}, y^{(i)}) \in T$$

Le *n* variabili $\xi_1, ..., \xi_n$ sono chiamate «slack variables»



Classi parzialmente sovrapposte

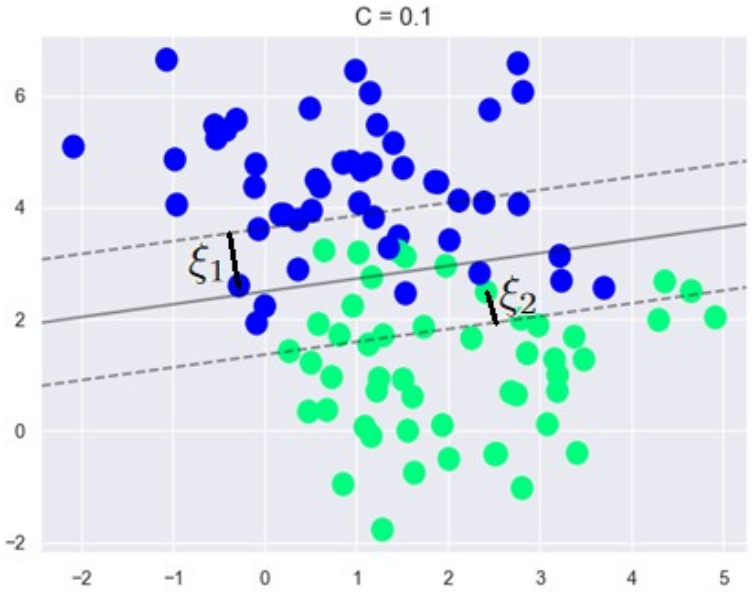
AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

$$w_0^*, \boldsymbol{w}^*, \xi_1^*, ..., \xi_n^* = \arg \min_{[w_0, \boldsymbol{w}] \in W, \xi_i \geq 0} \frac{1}{2}||\boldsymbol{w}||^2 + C \sum_{i=1}^{n} \xi_i \quad s.t.$$

$$y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + w_0) \geq 1 - \xi_i \quad \forall (\boldsymbol{x}^{(i)}, y^{(i)}) \in T$$

Anche questa è una forma di *regolarizzazione* che serve ad alleviare l'overfitting, diminuendo la dipendenza dell'iper-piano dall'esatta collocazione spaziale dei sample di frontiera tra le due classi

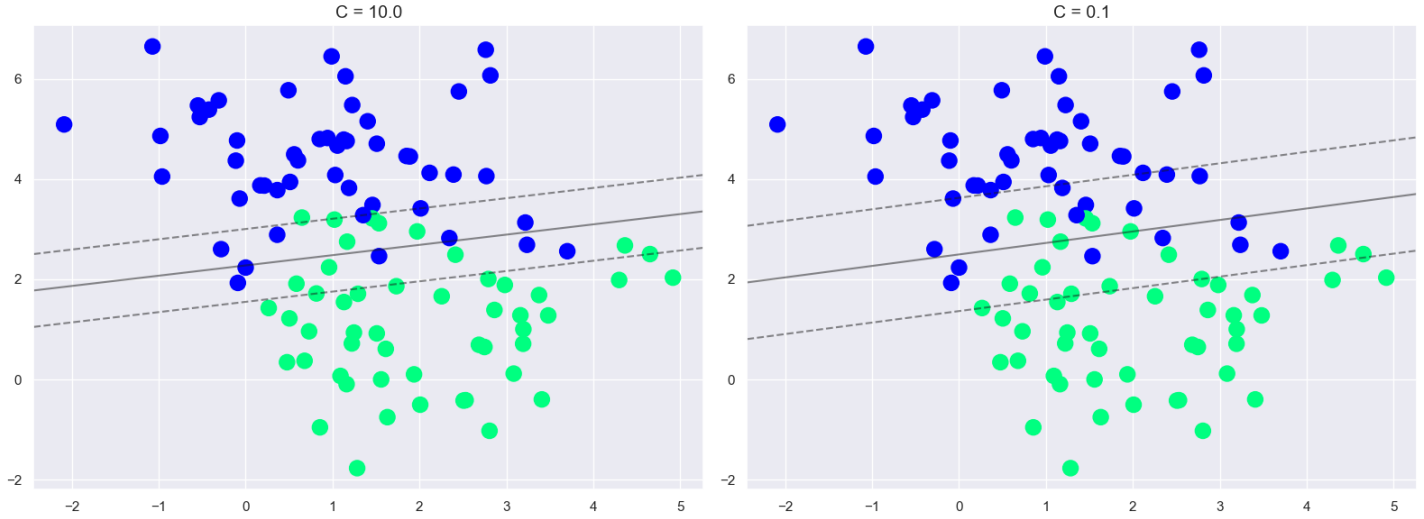
Usando la soluzione duale si può seguire un ragionamento analogo



Classi parzialmente sovrapposte

L'iper-parametro *C* determina il trade-off tra la massimizzazione del margine e la minimizzazione del numero di sample che violano i vincoli:

- Con *C* grande, ci si concentra di più sull'evitare errori nella separazione lineare delle due classi, al costo di avere un margine più piccolo (v. figura sotto a sinistra)
- Con *C* piccolo, si dà meno importanza agli errori di separazione e ci si concentra di più sul massimizzare il margine rispetto ai samples linearmente separabili (cioè quelli che non corrispondono ad «errori»)



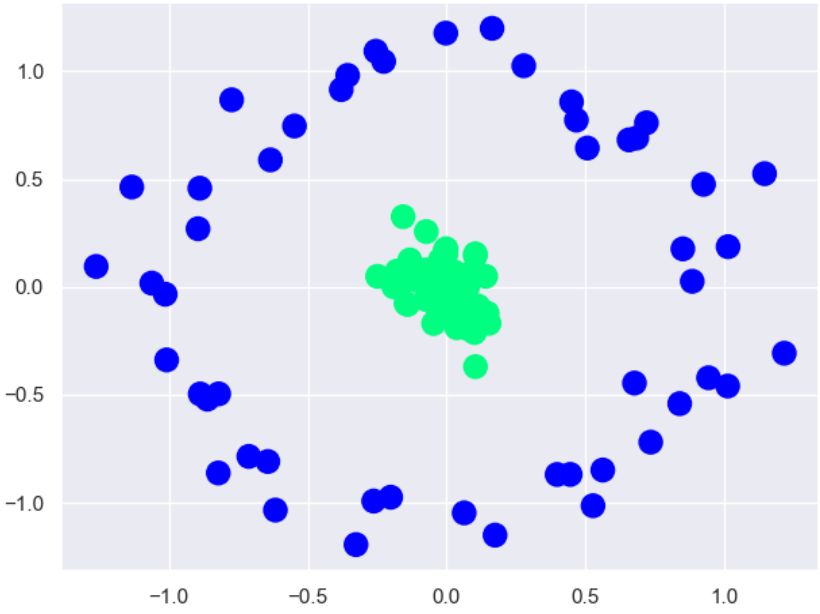
Non-linear Support Vector Machine

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Se le due classi, anzichè essere solo parzialmente sovrapposte, sono distribuite in modo tale da non poter essere separate linearmente (e.g., v. figura accanto), allora il problema non può essere risolto con una semplice regolarizzazione

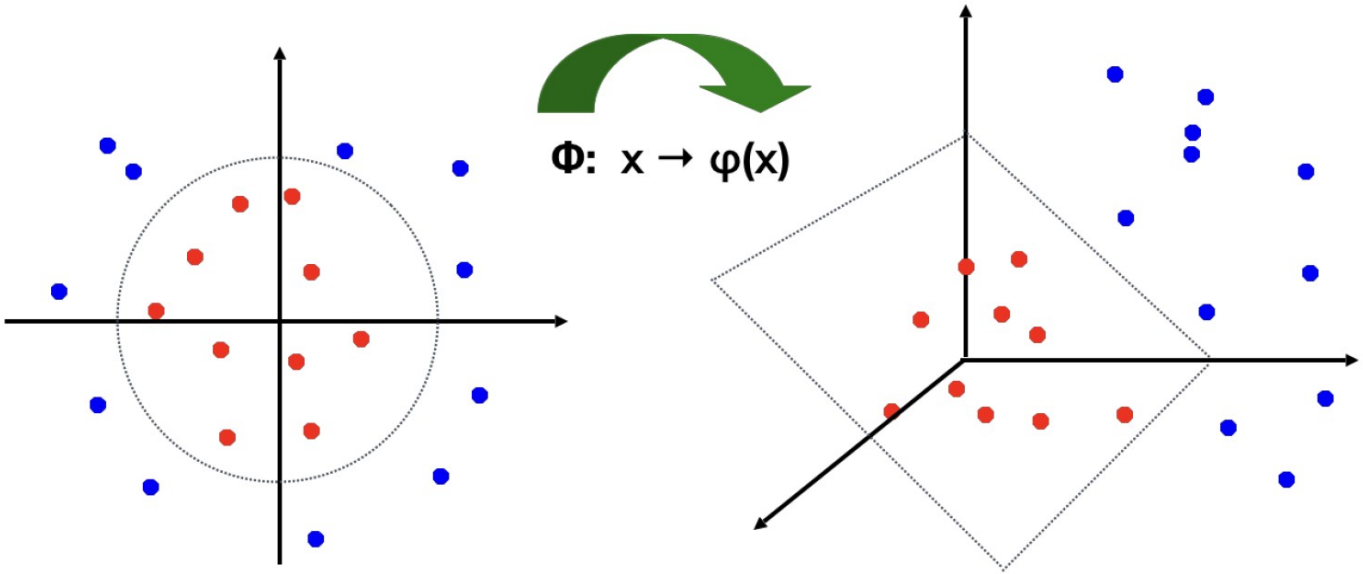
Come faccio a capire se le classi sono lineramente separabili, solo parzialmente sovrapposte o non sono linearmente separabili?

Uso il validation set (lo vedremo meglio nella prossima lezione)



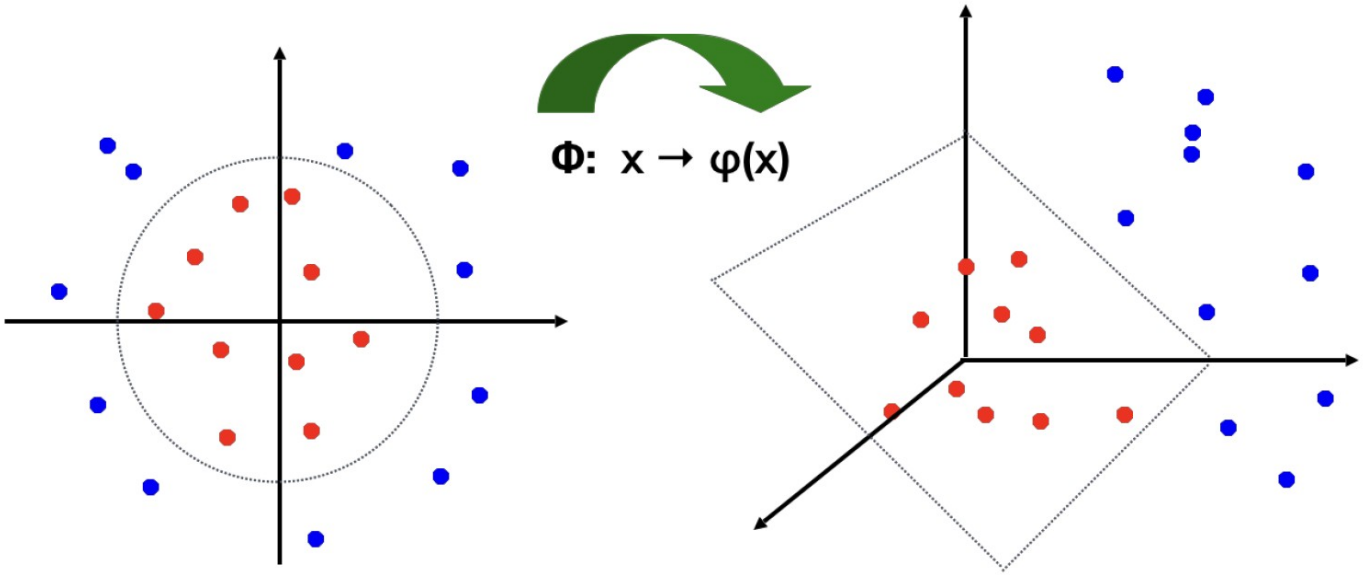
Non-linear Support Vector Machine

Nel caso le classi non siano linearmente separabili possiamo applicare una trasformazione non lineare $\phi(\boldsymbol{x})$ per mappare i feature vector $\boldsymbol{x} \in X = R^d$ in uno spazio con una dimensione più grande ($\phi(\boldsymbol{x}) = \boldsymbol{x}' \in X' = R^D$, $D > d$), dove i sample saranno (sperabilmente) linearmente separabili



Non-linear Support Vector Machine

Benchè, in teoria, una tale trasformazione non lineare ϕ() esiste sempre, in pratica trovarla (automaticamente) non è semplice
È però possibile aggirare questo problema ed evitare di dover calcolare ϕ() esplicitamente usando il cosiddetto «kernel trick» (ora vedremo come…)



Non-linear SVM

Ricordiamoci che, usando la soluzione duale, si ha che,

- in fase di training:

$$[\alpha_1^*, ..., \alpha_n^*] = \arg\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (\boldsymbol{x}^{(i)})^T \boldsymbol{x}^{(j)} \quad s.t.$$

$$\alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

- e a inference time:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y^{(i)} (\boldsymbol{x}^{(i)})^T \boldsymbol{x} + w_0$$

Notiamo che i vettori delle feature ($\boldsymbol{x}$, $\boldsymbol{x}^{(i)}$, $\boldsymbol{x}^{(j)} \in R^d$) *compaiono solo in prodotti scalari*

Non-linear SVM: kernel trick

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Questo ci permette di evitare di trasformare direttamente ***x***, ovvero di dover trovare un nuovo feature space *X'*, e di limitarci a definire il solo prodotto scalare in uno spazio delle feature «ipotetico» *X'*

In altre parole, invece di definire esplicitamente ϕ(), ci basta definire una funzione che calcola il prodotto scalare di due feature vector in uno spazio a dimensioni maggiori di *d,* senza che questo feature space nuovo sia effettivamente definito

Tale funzione è detta «funzione kernel». Dati due feature vector ***x***, ***z*** $\in X$, il kernel restituisce un valore scalare che rappresenta il loro prodotto scalare in un feature space a dimensioni maggiori di *X*:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{z}) \in \mathbb{R}$$

In pratica, *K()* stima una qualche misura di «similarità» (non lineare) tra ***x*** e ***z*** (tra un po' vedremo degli esempi di kernel)

Non-linear SVM con kernel trick

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Con la soluzione duale si ha, in fase di training:

$$[\alpha_1^*, ..., \alpha_n^*] = \arg\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \phi(\boldsymbol{x}^{(i)}) \phi(\boldsymbol{x}^{(j)})$$

che, usando il kernel, diventa:

$$[\alpha_1^*, ..., \alpha_n^*] = \arg\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$$

Non-linear SVM con kernel trick

Mentre, ad inference time, ho:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}) + w_0$$

che, usando il kernel, diventa:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + w_0$$

dove, come nel caso senza kernel, se $\boldsymbol{x}_s$ è un support vector, sfruttando il fatto che $f(\boldsymbol{x}_s) = y_s$, posso calcolare:

$$w_0 = y_s - \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}^{(i)}, \boldsymbol{x}_s)$$

Quindi non ho bisogno di definire $\phi(\boldsymbol{x})$: sia in fase di training che di inferenza, mi basta definire $K(\boldsymbol{x}, \boldsymbol{z})$. Questo modo di procedere è chiamato “kernel trick”

Non-linear Support Vector Machine

I kernel più usati sono:

- **Kernel lineare.** Prodotto scalare standard: $K(\boldsymbol{x},\boldsymbol{z}) = \boldsymbol{x}^T\boldsymbol{z}$ (i.e., nessun kernel e nessuna deformazione di R^d)

- **Kernel polinomiale**. Permette di di «curvare» R^d:

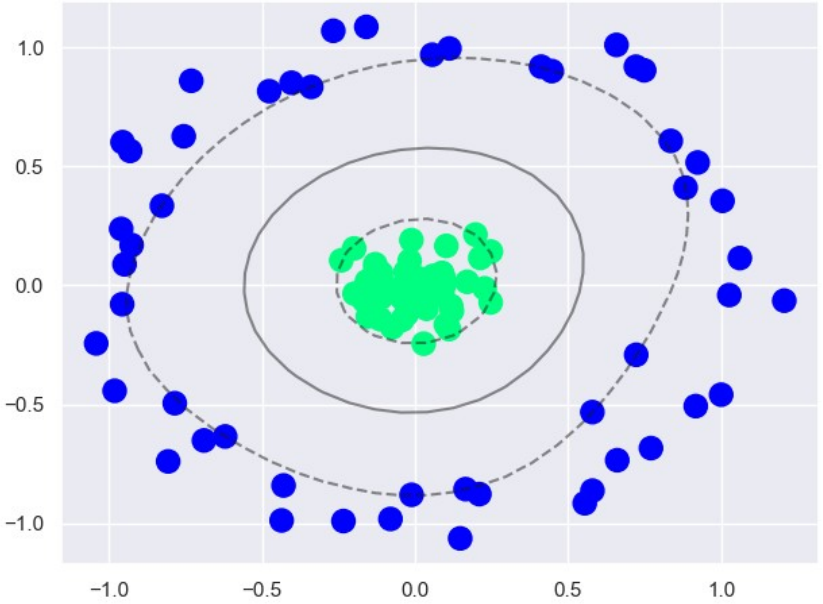
$$K(\boldsymbol{x},\boldsymbol{z}) = (\boldsymbol{x}^T\boldsymbol{z} + a)^m$$

dove *a* ed *m* sono iper-parametri

- **Radial Basis Function (RBF) o Gaussian kerne'** Crea deformazioni complesse di R^d:

$$K(\boldsymbol{x},\boldsymbol{z}) = \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{z}||^2}{2\sigma^2}\right)$$

E' il kernel più usato e σ è il suo iper-parametro



Non-linear SVM: modello non parametrico

Se uso i kernel, però, non posso usare la soluzione primale, in quanto, non avendo esplicitamente definito lo spazio delle feature $X' = R^D$, non ho neanche lo spazio dei parametri ($W' = R^{D+1}$) e il vettore dei parametri $\boldsymbol{w}$ non è definito

Similmente, a inference time, non posso calcolare direttamente il vettore dei parametri: non avendo definito ϕ(), non posso *sommare* i vettori trasformati e quindi non posso calcolare:

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\boldsymbol{x}_i)$$

Non-linear SVM: modello non parametrico

Se uso i kernel, quindi, devo necessariamente memorizzare i support vector, perchè, a inference time, dovrò calcolare:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + w_0 \qquad w_0 = y_s - \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}^{(i)}, \boldsymbol{x}_s)$$

Questa cosa ha due implicazioni

La prima è di natura pragmatica: finito il training, devo memorizzarmi tutti i support vector e, a inference time, applicare il kernel ad ognuno di essi

Considerando che, di solito, il numero dei support vector non è piccolissimo, ciò comporta un certo svantaggio computazionale in spazio e tempo

Non-linear SVM: modello non parametrico

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + w_0$$

La seconda implicazione è di natura, per così dire, più “filosofica”, nel senso che le SVM con kernel non lineari non possono più essere considerate come metodi parametrici (senza kernel invece sono parametrici a tutti gli effetti)

Questo perchè, per implementare la formula di sopra, che esprime la funzione di score finale, non posso fare a meno di basarmi su un algoritmo che iteri per tutti i support vector, e questo loop non può essere espresso con una combinazione di funzioni analitiche note, quindi non rientra nella definizione di “metodo parametrico”

In altri termini, *f()* ora dipende dagli specifci support vector estratti dallo specifico *T* e corrispondente task che stiamo esaminando, e non c’è modo di esprimerli con un numero finito di parametri (il parameter vector) di una funzione analitica

Non-linear SVM: modello non parametrico

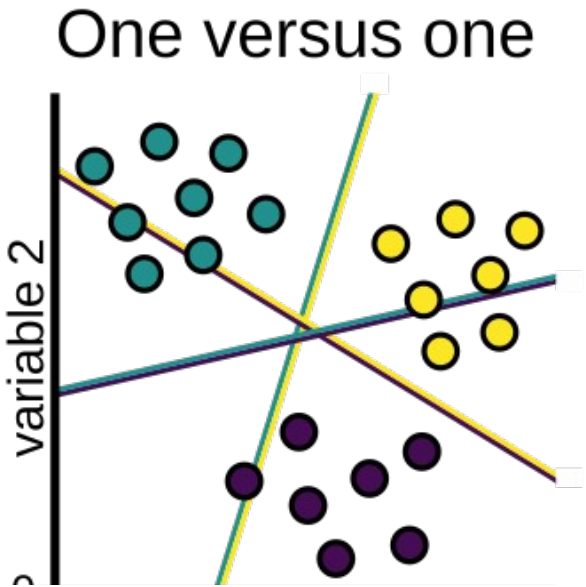
AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + w_0$$

In un certo senso, la soluzione duale con kernel non lineari può essere vista come una specie di k-Nearest Neighbors (k-NN), in cui:

- a inference time, il vettore di testing ***x*** viene confrontato solo con i *k* vettori di supporto (indipendentemente dal valore di ***x*** la scelta dei *k* samples per il confronto è sempre la stessa e sono sempre i *k* vettori di supporto),
- il kernel è usato come misura di *similarità* tra due vettori (maggiore è il valore $K(\boldsymbol{x}^{(i)}, \boldsymbol{x})$, maggiore è la loro similarità) al posto (dell'inverso) della distanza Euclidea,
- α_i è usato come peso fisso per determinare l'importanza relativa del vettore di supporto $\boldsymbol{x}^{(i)}$,
- i valori di ground truth y_i vengono pesati da $\alpha_i K(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ e la media pesata viene restituita in output come nel caso del k-NN usato per regressione

SVM: estensione a problemi multiclasse ($y \in \{1, \ldots, k\}$)



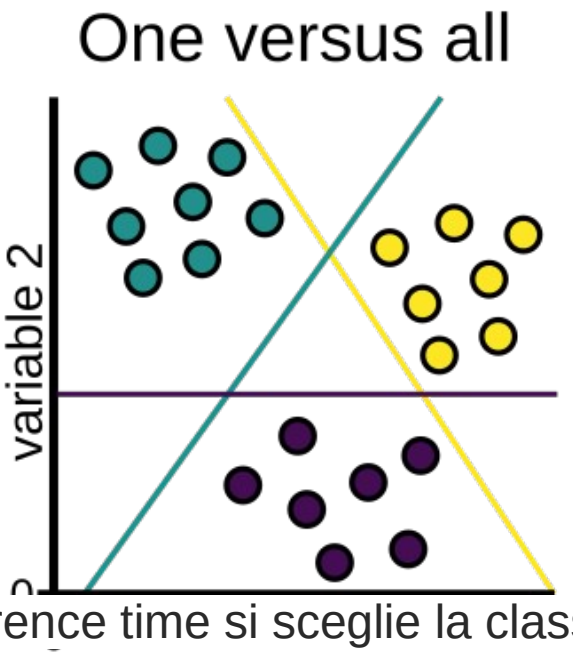
In **One-vs-One** (OvO) si addestrano *M* classificatori binari per discriminare ogni possibile paio di classi, per un totale di

$$M = \frac{k(k-1)}{2}$$

classificatori binari

A inference time si applica uno schema di voto (eventualmente pesato usando lo score): la classe che riceve più voti «vince»

SVM: estensione a problemi multiclasse ($y \in \{1, \ldots, k\}$)



In **One-vs-All** (OvA), o **One-vs-Rest** (OvR) si addestrano *M* classificatori binari per discriminare ciascuna delle *k* classi da tutte le altre *k-1* classi, per un totale di

classificatori binari $M = k$

A inference time si sceglie la classe corrispondente al classificatore più “confident”, ovvero quello che, per la “sua” classe positiva ha stimato lo score col valore maggiore rispetto agli altri classificatori

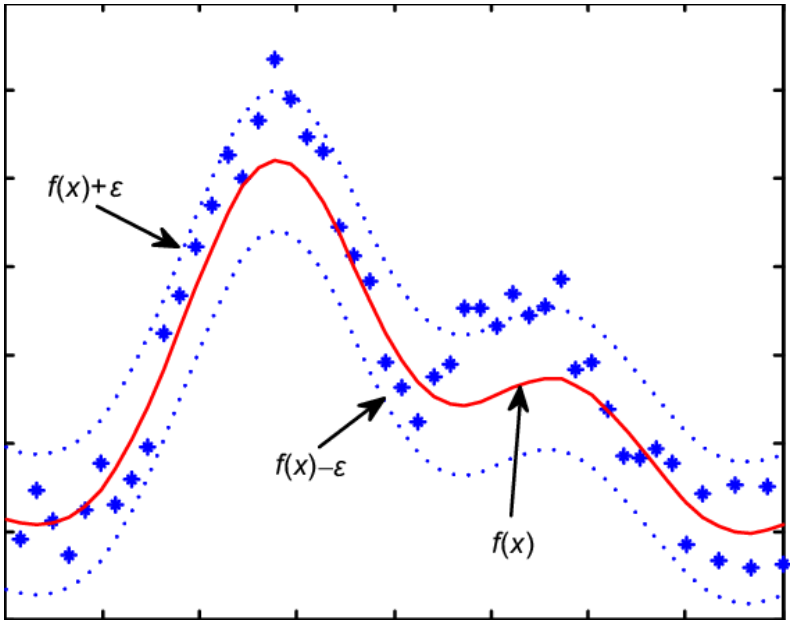
Support Vector Regression (SVR)

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Come sappiamo, in un task di regressione vogliamo trovare una funzione che approssimi la relazione tra le feature e la variabile target, dove quest'ultima assume valori numerici continui

Le SVM possono essere usate come metodo di regressione, mantenendo tutte le loro caratteristiche principali. In tal caso, si chiamano Support Vector Regression (SVR)

Nel caso della regressione i vincoli delle SVM vengono riformulati imponendo che, per ogni $(\boldsymbol{x}_i, y_i) \in T$, la differenza tra il valore di ground truth y_i e la predizione $f(\boldsymbol{x}_i)$ sia inferiore o uguale ad un margine massimo di errore definito dall'iper-parametro **ε**
Usando il kernel trick la funzione di regressione risultante può essere altamente non lineare (e.g., v. figura a fianco)



Scalamento dei dati

Anche con le SVM/SVR è utile scalare i dati prima di iniziare il training perché:

- L'ottimizzazione, sia per la soluzione primale che duale, avviene con algoritmi iterativi come il Gradient Descent
- Sono metodi definiti rispetto ad un «margine», che è una distanza (Euclidea), il cui calcolo è influenzato dalle unità di misura delle varie feature
- Anche il calcolo del kernel è influenzato dalle unità di misura delle varie feature

Support Vector Machine

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Vantaggi:

- Strumento molto potente, con una notevole resistenza all'overfitting (spesso maggiore ad altri metodi)
- Possono trattare problemi non lineari (se uso i kernel)

Limitazioni:

- Non c'è nessuna garanzia che si riesca a trovare (facendo delle prove col validation set…) il kernel e i relativi iper-parametri giusti che permettano di separare due classi non linearmente separabili

Svantaggi:

- Nel caso duale con kernel non-lineare bisogna memorizzare tutti i support vector, che potrebbero essere tanti con un dataset di training enorme
- Non esiste un'interpretazione probabilistica diretta per l'appartenenza di un sample a una classe (ma questa potrebbe essere ricavata come post-processing a partire dallo score)

Riferimenti

- https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf
- **A tu per tu col Machine Learning. L'incredibile viaggio di un developer nel favoloso mondo della Data Science**, Alessandro Cucci, The Dot Company, 2017 [cap.5]
- **Pattern Classification, second edition,** R. O. Duda, P. E. Hart, D. G. Stork, Wiley-Interscience, 2000
- **An introduction to statistical learning**, Gareth M. James, Daniela Witten, Trevor Hastie, Robert Tibshirani, New York: Springer, 2013 [cap. 4]