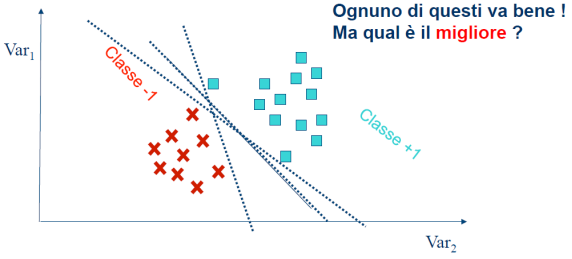
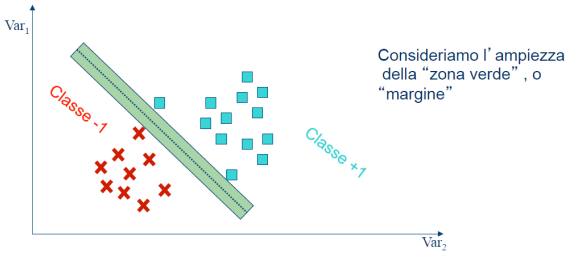
Lezione 7 17/11/2023

SVM (Support Vector Machine)

Se i punti sono separati, allora l'ipotesi è una qualsiasi di quelle che divide i punti correttamente. Dato che noi immaginiamo di aver punti diversi da provare dopo l'allenamento non tutte le linee delle ipotesi saranno uguali. Quindi qual è quella migliore?





L'idea è che l'ipotesi migliore sia quella che si piazza in mezzo lasciando il maggior margine possibile.

Nel margine non ci sono punti, ed è racchiuso tra 2 iperpiani.

Le SVM servono proprio a trovare il **miglior iperpiano separatore** per classificare un insieme di punti linearmente separabili.

- Supponiamo di avere un insieme di punti di training $S = \{(\mathbf{x}_1,y_1), (\mathbf{x}_2,y_2), \cdots, (\mathbf{x}_n,y_n)\}$
- Ad ogni $\mathbf{x}_i$ (vettore) è associata la rispettiva classe di appartenenza y_i $y_i \in \{-1, +1\}$
- I punti sono linearmente separabili $\langle \mathbf{w}, \mathbf{x}_i\rangle + b > 0$ se $y_i = +1$; $\langle \mathbf{w}, \mathbf{x}_i\rangle + b < 0$ se $y_i = -1$
- Ma questo lo possiamo scrivere in un solo vincolo $y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) > 0 \quad (i = 1, \cdots, n)$

L'addestramento è fatto su un insieme di punti etichettati.

La linea corrisponde a definire un vettore di pesi, uno spostamento (b) tali che il risultato del prodotto interno + b è positivo per le istanze positive e negativo per le istanze negative.

Qui sul set di istanze etichettate, SVM farà un calcolo unico, tutto di un colpo, senza dividere le istanze. Alla fine dell'apprendimento produrrò il w e il b.

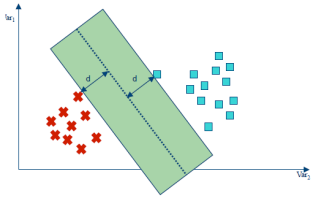
Alla fine si può scrivere un'unica espressione booleana.

Quest'espressione è vera se l'ipotesi etichetta bene le istanze. Infatti le ipotesi producono $< w, xi > +b$ mentre y() ritorna un valore > 0 se il target è 1 o < 0 se il target è -1 (ovvero la sua classe di appartenenza).

Se i punti sono separabili w può variare durante l'apprendimento ma deve essere sempre corretta.

Dire che voglio rendere il margine il più grande possibile, vuol dire che il > 0 di quest'ultima forma sia > di un numero più grande possibile. Quindi il valore calcolato deve essere il più grande possibile (forse sarà positivo per i valori sopra al piano e negativo per quelli sotto). Vogliamo trovare quindi il w che renda il margine il più grande possibile.

Sia d_+ (d_-) la distanza tra l' iperpiano separatore e il punto positivo (negativo) più vicino



Def: i **margini** di un iperpiano
- Margine funzionale
- Margine geometrico

Quindi o ragiono sul margine funzionale (ovvero la formula matematica precedente) o ragiono a livello geometrico (massimizzando l'area verde).

Prima formulazione della soluzione (attraverso) i vettori di supporto

- La soluzione al problema

$$\min \ \tau(\mathbf{w}) = \frac{1}{2} \| \mathbf{w} \|^2$$
$$s.t. \quad y_i(< w, x_i > +b) \geq 1 \quad (i = 1 \ldots n)$$

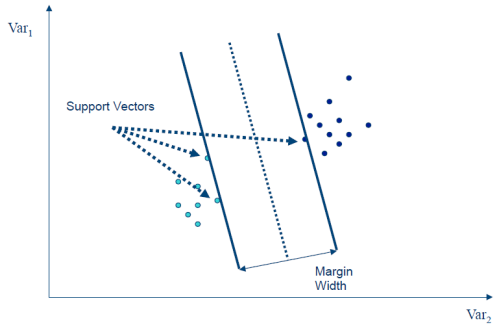
- può essere scritta

$$\mathbf{w} = \sum_{i \in Q} \alpha_i \mathbf{x}_i$$

  - Ovvero è scritta in termini di un sottoinsieme di esempi del training set (noto come vettori di supporto) che giacciono sul margine dell' iperpiano

Rovescio il problema in un problema di minimizzazione (geometrico).

Saltando la matematica, si ritrova l'ipotesi ottimale w, non sappiamo i calcoli dell'alpha. Questo sarà calcolato con l'uso dell'intero dataset (perché noi siamo all'esterno del calcolo quindi riceviamo il risultato e basta, anche se poi l'algoritmo legge un esempio alla volta).



Tutti i vettori che non toccano il margine hanno un alpha i = 0, mentre quelli che toccano il margine hanno alpha i diversi da zero.

I vettori di supporto sono quelli che calcolano il w ottimale.

Il calcolo ha però bisogno tutti i vettori.

Il set di addestramento conta sul vincolo (la corretta classificazione sul ≥ 1) e poi l'algoritmo lavora sul minimo.

Seconda formulazione della funzione di decisione

- Per classificare un nuovo elemento (vettore) **x** ci interessa il segno di <**w**, **x**> + b, che possiamo scrivere come:

$$\operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b) = \operatorname{sgn}(\left\langle \sum_{i \in Q} \alpha_i \mathbf{x}_i, \mathbf{x} \right\rangle + b) = \operatorname{sgn}(\sum_{i \in Q} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b)$$

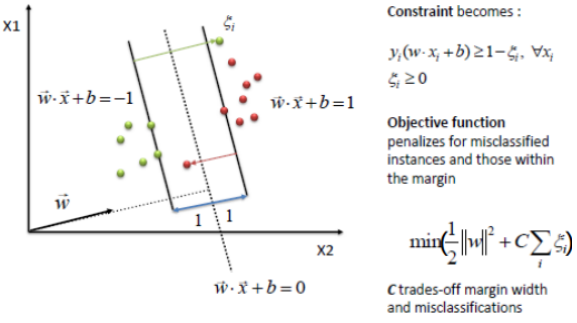
- Quindi la funzione di decisione associata alla soluzione può essere scritta in termini del prodotto interno tra i vettori di supporto x_i e il vettore da classificare **x**

Prendiamo il vettore w delle ipotesi, facciamo il prodotto interno su x che produce uno scalare, aggiungi lo spostamento b e dal risultato prendi il segno ottenendo +1 o -1.

Poi diventa il vettore w calcolato con gli alpha.

E poi la formula viene trasformata ancora linearmente. Questo ci da una riscrittura in base al prodotto interno di x con ciascuno dei vettori di supporto (sommatoria dei vettori di supporto, per ciascuno prodotto interno). In questa formula vengono usati solo i vettori di supporto.

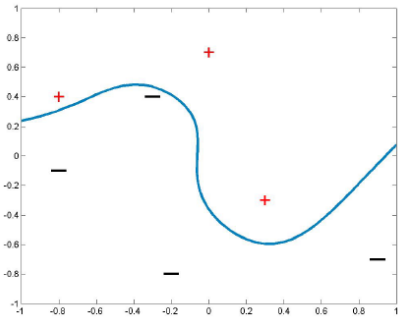
In laboratorio useremo dei metodi diversi che sanno gestire dei casi particolari. Nell'esempio qui se dovessi classificare correttamente tutti i punti troverei un'ipotesi estrema perché con una pendenza alta rispetto alla maggior parte dei punti e quindi ho un rischio di



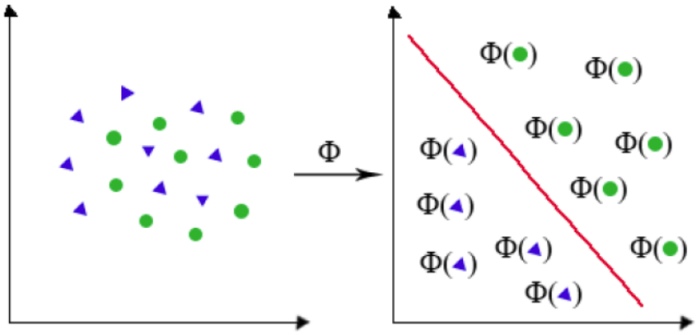
classificazione più alto per il futuro (dopo l'addestramento).

Viene usata una formula diversa che introduce uno "slack" che produce un'ipotesi con dei parametri più liberi. La flessibilità ha un costo C degli errori, quindi conto anche questi quando minimizzo.

Punti non linearmente separabili



Facendo dei trucchi di cambio di rappresentazione dei dati si può fare. Prendo le x e y che stanno a sinistra e creo delle nuove istanze.



Questo esempio legge i punti come uno spirale che cresce in senso antiorario. La coppia di distanza e angolo da un punto centrale quindi aumenta costantemente. Si può supporre che per i blue questo andamento è diverso rispetto ai verdi. Viene quindi rappresentato in un piano separato dalla diagonale.

Posso quindi dire di aver appreso un criterio. è raro che si riesca a mantenere lo spazio bidimensionale nella trasformazione, spesso si può fare questa trasformazione ma porta ad uno spazio enorme (che può anche dare problemi di calcolo)

- Quindi dobbiamo trovare una funzione Φ: $\mathbb{R}^n \rightarrow \mathbb{R}^m$ (con m>n) che mappi i dati iniziali non linearmente separabili in uno spazio di dimensione superiore in cui siano linearmente separabili
- Nel nuovo spazio, la funzione di decisione per classificare un nuovo elemento x avrà la forma

$$\operatorname{sgn}(\sum_{i \in Q} \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + b)$$

- Il calcolo delle immagini $\Phi(x_i)$ è in genere computazionalmente oneroso, ma viene semplificato nel caso particolare delle funzioni kernel

Riscrivo la formula precedente con la fi.

Quando la fi ha delle caratteristiche speciali, quel prodotto interno che applicando le x era facile, anche se la fi ci spara in dimensioni enormi, hanno comunque delle caratteristiche che fanno in modo che il prodotto interno si può fare velocemente, anche se abbiamo dimensioni altissime.

Funzione Kernel

Definiamo una funzione K (Kernel) che ha come risultato del calcolo il prodotto interno dei vettori mappati.

Questa funziona K va quindi trovata (se esiste).

Si sostituisce quindi fi con K che sarà una funzione più veloce che darà lo stesso risultato ma più velocemente.

- Definizione: Data una trasformazione Φ: $\mathbb{R}^n \rightarrow \mathbb{R}^m$ una funzione kernel è una mappa K: $\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ tale che $K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x}) \bullet \Phi(\mathbf{y})$
- Kernel trick → computare il prodotto interno delle trasformate di due vettori **x** e **y** attraverso Φ senza computare le trasformate
- In questo modo si semplifica il calcolo della funzione di decisione $\operatorname{sgn}(\sum_{i \in Q} \alpha_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + b)$
- Basta infatti sostituire $\Phi(\mathbf{x}_i) \bullet \Phi(\mathbf{x})$ con $K(\mathbf{x}_i,\mathbf{x})$:

$$\operatorname{sgn}(\sum_{i \in Q} \alpha_i K(x_i, x) + b)$$

Esempio

- Supponiamo di mappare un punto (x, y) in $\mathbb{R}^2$ in (r, s, t) in $\mathbb{R}^3$, ponendo:

$$r = x^2,\ s = y^2,\ t = \sqrt{2}xy$$

- Se ora consideriamo due punti $p_1 = (x_1, y_1)$ e $p_2 = (x_2, y_2)$ in $\mathbb{R}^2$, il prodotto interno delle loro immagini in $\mathbb{R}^3$ sarà:

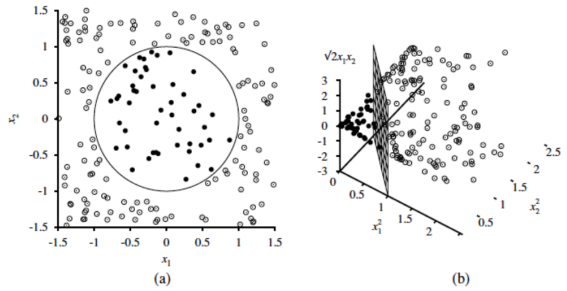
$$(x_1^2, y_1^2, \sqrt{2}x_1y_1) \bullet (x_2^2, y_2^2, \sqrt{2}x_2y_2) =$$
$$(x_1x_2)^2 + (y_1y_2)^2 + 2x_1y_1x_2y_2$$

La funzione trovata è un quadrato di prodotto che dovrebbe essere più semplice rispetto a quella iniziale.

E' facile verificare che questo non è altro che il quadrato del prodotto interno di p_1 e p_2

$$[(x_1, y_1) \bullet (x_2, y_2)]^2 = (x_1x_2 + y_1y_2)^2$$

Quindi possiamo trovare il risultato con un semplice calcolo in $\mathbb{R}^2$, evitando di calcolare le immagini in $\mathbb{R}^3$



Support Vector non sarebbe in grado di trovare quel cerchio, quindi trasformiamo in uno spazio tridimensionale dove c'è un piano che separa i punti.

Quindi grazie allo spazio tridimensionale support vector è in grado di trovare il piano, e tramite la kernel function possiamo convertire i punti da verificare (dopo l'addestramento) nei punti del secondo piano in tempo ragionevole.

Se i dati sono mappati in uno spazio di dimensioni sufficientemente elevate, saranno quasi sempre linearmente separabili.

Quattro dimensioni sono sufficienti per separare linearmente un cerchio in qualsiasi punto del piano. Cinque dimensioni sono sufficienti per separare linearmente qualsiasi ellisse.

In generale (con l'eccezione di alcuni casi speciali) se abbiamo N esempi, questi saranno sempre separabili in spazi di dimensione N - 1 o più. (non serve sapere questa cosa)

- Rappresentano un modo per applicare le SVM in modo efficiente in spazi di dimensione molto alta (o infinita):

  $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$, ove $\Phi: R^n \rightarrow R^m$
- Calcolare $K(x, y)$ può essere molto economico, anche se il calcolo di $\Phi(x)$ è molto costoso (ad esempio, perché è un vettore di dimensioni elevate)
- In tali casi, se abbiamo un modo efficiente per calcolare $K(x, y)$, possiamo addestrare le SVM nello spazio degli attributi di dimensionalità maggiore, senza mai dover trovare o rappresentare esplicitamente i vettori $\Phi(x)$

Anche nell'addestramento c'era un prodotto interno.

Quindi anche qui possiamo semplificare usando K. Quindi usiamo k nell'inferenza (fase di utilizzo del modello allenato) e anche nell'addestramento.

Esempio 2

- Sia $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$, e si consideri la trasformazione

$$\Phi: \mathbb{R}^3 \rightarrow \mathbb{R}^{10}$$

definita da:

$$\Phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3, x_1^2, x_2^2, x_3^2)$$

che sono tutti i monomi fino al grado 2 ottenuti combinando le tre variabili di partenza, con opportuni coefficienti. Allora:

$$\Phi(\mathbf{x}) \bullet \Phi(\mathbf{y}) = 1 + 2\Sigma_1^d x_i y_i + 2\Sigma_1^d x_i^2 y_i^2 + 2\Sigma x_i x_j y_i y_j = (1 + \mathbf{xy})^2$$

Ho ottenuto sia la possibilità di separare i punti muovendomi in 10 dimensioni, che una semplificazione dei calcoli tramite K.

Cose extra

- Alcuni kernel standard:

Lineare: $K(x, y) = x \cdot y$

Polinomiale: $K(x, y) = (1 + x \cdot y)^d$

Radial Basis Function: $K(x, y) = e^{-\gamma \|x-y\|^2}$

Gaussian Radial Basis Function: $K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}$

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

- Teorema: Un kernel definisce una matrice $[k_{i,j}]$ che è simmetrica e definita positiva.
- Teorema (Mercer): Ogni matrice simmetrica e definita positiva è un kernel