Scuola universitaria professionale della Svizzera italiana
Bachelor di Ingegneria Informatica

SUPSI

Machine Learning

Lezione 7 - Support Vector Machines

Loris Cannelli, Ricercatore, IDSIA-SUPSI
loris.cannelli@supsi.ch

IDSIA-SUPSI, Polo universitario Lugano - Dipartimento Tecnologie Innovative

Supervised Learning

- Immaginiamo di avere N oggetti/dati $\mathbf{x}_1, \ldots, \mathbf{x}_N$, ognuno di dimensione D

Supervised Learning

- Immaginiamo di avere N oggetti/dati $\mathbf{x}_1, \ldots, \mathbf{x}_N$, ognuno di dimensione D
- Ipotizziamo che esistano 2 diverse classi di appartenenza e che ogni oggetto $\mathbf{x}_n$ sia associato ad una delle classi $t_n \in \{-1, 1\}$ (*supervised learning*)

Supervised Learning

- Immaginiamo di avere N oggetti/dati $\mathbf{x}_1, \ldots, \mathbf{x}_N$, ognuno di dimensione D
- Ipotizziamo che esistano 2 diverse classi di appartenenza e che ogni oggetto $\mathbf{x}_n$ sia associato ad una delle classi $t_n \in \{-1, 1\}$ (*supervised learning*)
- Vogliamo capire a quale classe t_{new} appartiene un nuovo oggetto $\mathbf{x}_{\text{new}}$

Supervised Learning

- Immaginiamo di avere N oggetti/dati $\mathbf{x}_1, \ldots, \mathbf{x}_N$, ognuno di dimensione D
- Ipotizziamo che esistano 2 diverse classi di appartenenza e che ogni oggetto $\mathbf{x}_n$ sia associato ad una delle classi $t_n \in \{-1, 1\}$ (*supervised learning*)
- Vogliamo capire a quale classe t_{new} appartiene un nuovo oggetto $\mathbf{x}_{\text{new}}$

$\Rightarrow$ Classificazione

Supervised Learning

- SVM fu proposto per la prima volta da Vapnik nel 1965

Supervised Learning

- SVM fu proposto per la prima volta da Vapnik nel 1965
- Dato un training set, l'idea di Vapnik era di trovare un contorno/confine che dividesse lo spazio in due aree, una per una classe e una per un'altra

Supervised Learning

- SVM fu proposto per la prima volta da Vapnik nel 1965
- Dato un training set, l'idea di Vapnik era di trovare un contorno/confine che dividesse lo spazio in due aree, una per una classe e una per un'altra
- Il più semplice modo di individuare due aree è tracciare una linea (in due dimensioni) o un iperpiano (in più dimensioni):

$$\mathbf{w}^T \mathbf{x}_{\text{new}} + b$$

Supervised Learning

- SVM fu proposto per la prima volta da Vapnik nel 1965
- Dato un training set, l'idea di Vapnik era di trovare un contorno/confine che dividesse lo spazio in due aree, una per una classe e una per un'altra
- Il più semplice modo di individuare due aree è tracciare una linea (in due dimensioni) o un iperpiano (in più dimensioni):

$$\mathbf{w}^T \mathbf{x}_{\text{new}} + b$$

- La classe di appartenenza di $\mathbf{x}_{\text{new}}$ sarà +1 o -1 a seconda di dove l'oggetto si colloca rispetto al contorno di demarcazione:

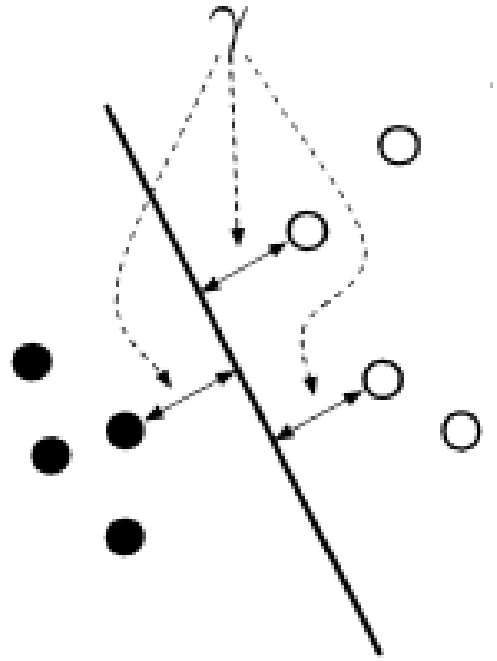
$$t_{\text{new}} = \text{sign}\left(\mathbf{w}^T \mathbf{x}_{\text{new}} + b\right)$$

Margini

- Come posizionare il contorno di demarcazione (cioè, come scegliere i valori di $\mathbf{w}$ e b) è cruciale per la classificazione

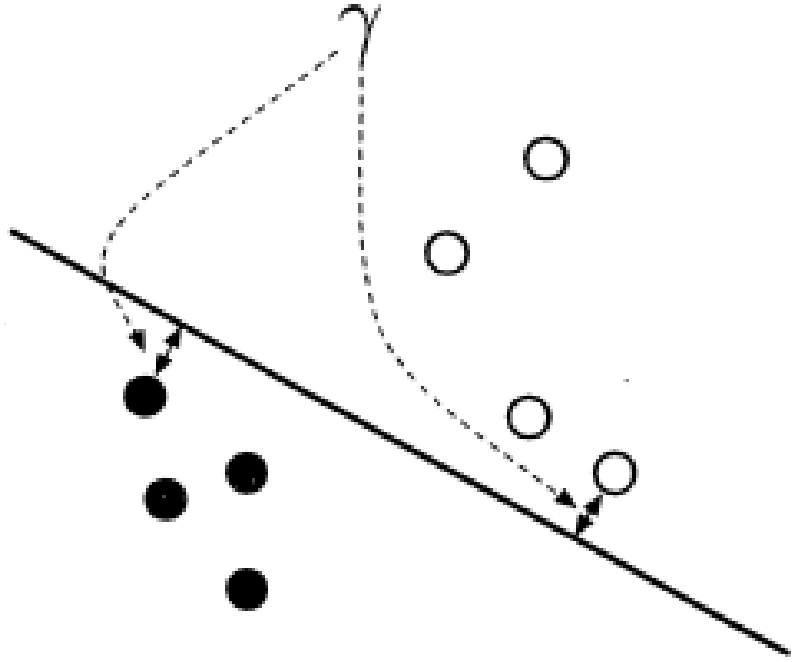
Margini

- Come posizionare il contorno di demarcazione (cioè, come scegliere i valori di $\mathbf{w}$ e b) è cruciale per la classificazione



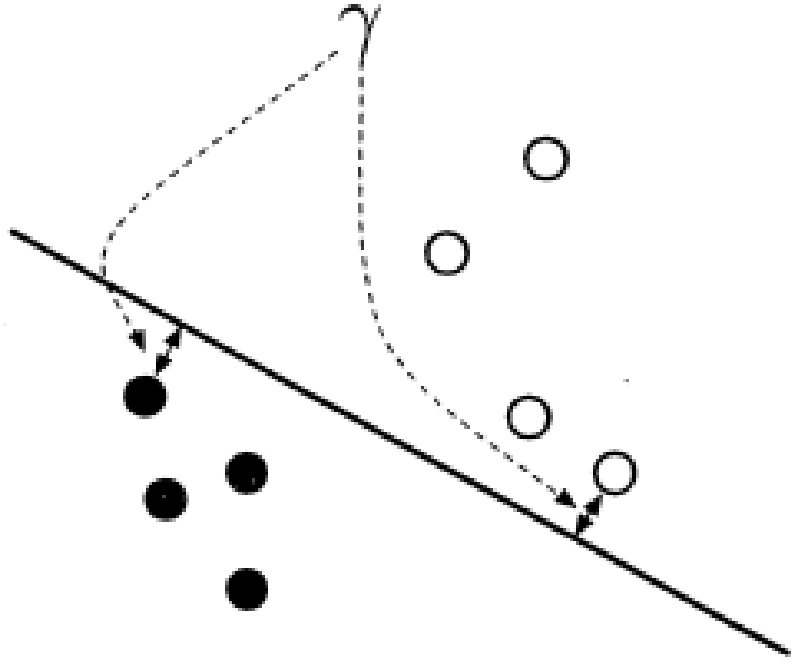
Margini

- Come posizionare il contorno di demarcazione (cioè, come scegliere i valori di $\mathbf{w}$ e b) è cruciale per la classificazione



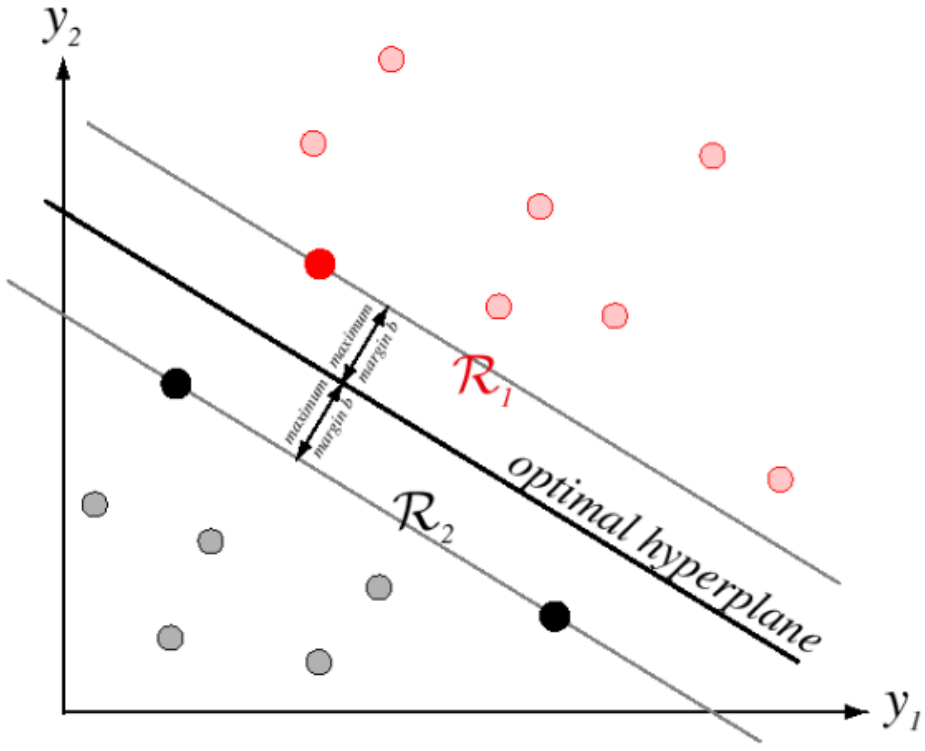
Margini

- Come posizionare il contorno di demarcazione (cioè, come scegliere i valori di $\mathbf{w}$ e b) è cruciale per la classificazione



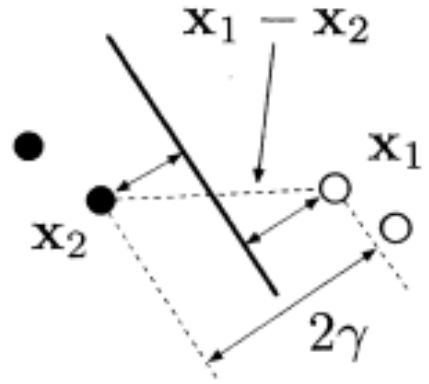
$\Rightarrow$ la proposta di Vapnik è stata quella di utilizzare il classificatore che separa le classi con il maggior margine possibile

Massimizzare il margine

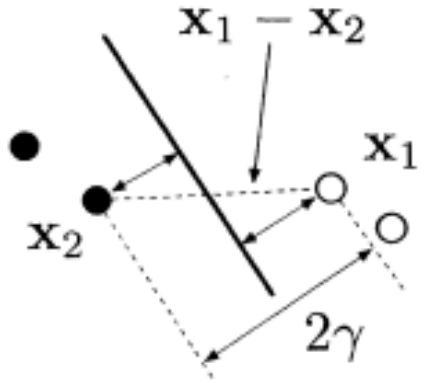


Se i dati del training set sono classificati con ampio margine si può "sperare" che anche nuovi dati vicini al confine tra le classi siano gestiti correttamente

Margine

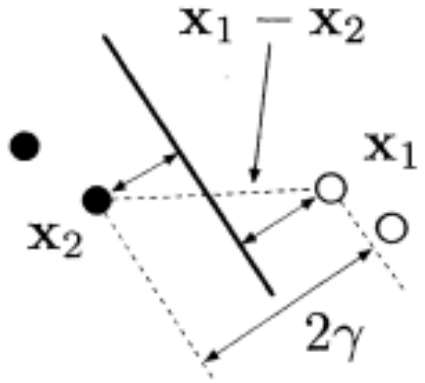


Margine



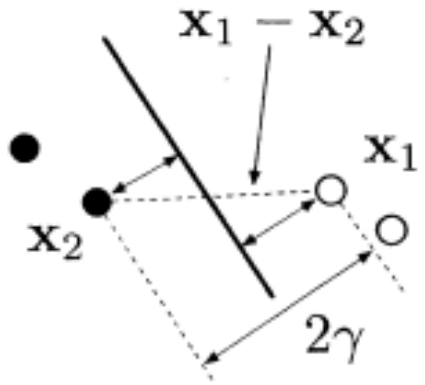
- Il vettore unitario perpendicolare all'iperpiano è $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$

Margine



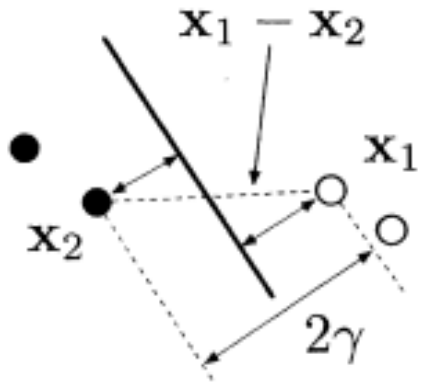
- Il vettore unitario perpendicolare all'iperpiano è $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$
- Con un po' di trigonometria si dimostra che: $2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$

Margine



- Il vettore unitario perpendicolare all'iperpiano è $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$
- Con un po' di trigonometria si dimostra che: $2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$
- Quindi:

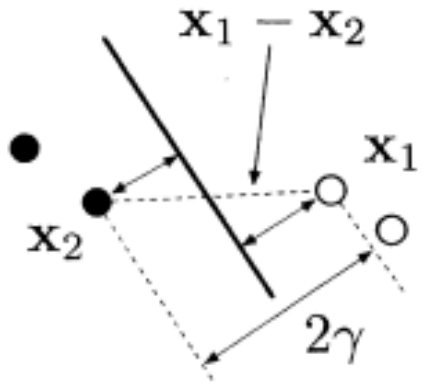
Margine



- Il vettore unitario perpendicolare all'iperpiano è $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$
- Con un po' di trigonometria si dimostra che: $2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$
- Quindi:

$$2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$$

Margine

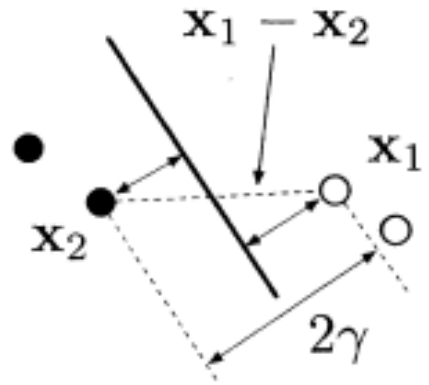


- Il vettore unitario perpendicolare all'iperpiano è $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$
- Con un po' di trigonometria si dimostra che: $2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$
- Quindi:

$$2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$$

$$2\gamma = \frac{\mathbf{w}^T\mathbf{x}_1 - \mathbf{w}^T\mathbf{x}_2}{\|\mathbf{w}\|_2}$$

Margine



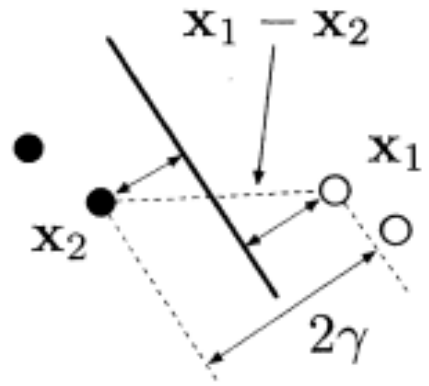
- Il vettore unitario perpendicolare all'iperpiano è $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$
- Con un po' di trigonometria si dimostra che: $2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$
- Quindi:

$$2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$$

$$2\gamma = \frac{\mathbf{w}^T\mathbf{x}_1 - \mathbf{w}^T\mathbf{x}_2}{\|\mathbf{w}\|_2}$$

$$2\gamma = \frac{\mathbf{w}^T\mathbf{x}_1 + b - \mathbf{w}^T\mathbf{x}_2 - b}{\|\mathbf{w}\|_2}$$

Margine



- Il vettore unitario perpendicolare all'iperpiano è $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$
- Con un po' di trigonometria si dimostra che: $2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$
- Quindi:

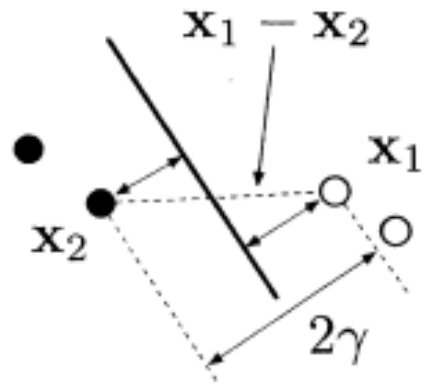
$$2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$$

$$2\gamma = \frac{\mathbf{w}^T\mathbf{x}_1 - \mathbf{w}^T\mathbf{x}_2}{\|\mathbf{w}\|_2}$$

$$2\gamma = \frac{\mathbf{w}^T\mathbf{x}_1 + b - \mathbf{w}^T\mathbf{x}_2 - b}{\|\mathbf{w}\|_2}$$

$$2\gamma = \frac{1 + 1}{\|\mathbf{w}\|_2}$$

Margine



- Il vettore unitario perpendicolare all'iperpiano è $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$
- Con un po' di trigonometria si dimostra che: $2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$
- Quindi:

$$2\gamma = \frac{\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{w}\|_2}$$

$$2\gamma = \frac{\mathbf{w}^T\mathbf{x}_1 - \mathbf{w}^T\mathbf{x}_2}{\|\mathbf{w}\|_2}$$

$$2\gamma = \frac{\mathbf{w}^T\mathbf{x}_1 + b - \mathbf{w}^T\mathbf{x}_2 - b}{\|\mathbf{w}\|_2}$$

$$2\gamma = \frac{1+1}{\|\mathbf{w}\|_2}$$

$$\Rightarrow \gamma = \frac{1}{\|\mathbf{w}\|_2}$$

Ottimizzare il margine

- Abbiamo capito che l'espressione del margine è $\frac{1}{\|\mathbf{w}\|_2}$

Ottimizzare il margine

- Abbiamo capito che l'espressione del margine è $\frac{1}{\|\mathbf{w}\|_2}$
- Vogliamo massimizzarlo, imponendo anche che ogni dato del training set sia correttamente classificato dal contorno risultante

Ottimizzare il margine

- Abbiamo capito che l'espressione del margine è $\frac{1}{\|\mathbf{w}\|_2}$
- Vogliamo massimizzarlo, imponendo anche che ogni dato del training set sia correttamente classificato dal contorno risultante
- Il problema di ottimizzazione da risolvere è quindi:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$t_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) \geq 1, \;\; n = 1, \ldots, N$$

Ottimizzare il margine

- Abbiamo capito che l'espressione del margine è $\frac{1}{\|\mathbf{w}\|_2}$
- Vogliamo massimizzarlo, imponendo anche che ogni dato del training set sia correttamente classificato dal contorno risultante
- Il problema di ottimizzazione da risolvere è quindi:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$t_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) \geq 1, \;\; n = 1, \ldots, N$$

- Questo è un problema di ottimizzazione convesso e vincolato noto come *quadratic programming*

Ottimizzare il margine

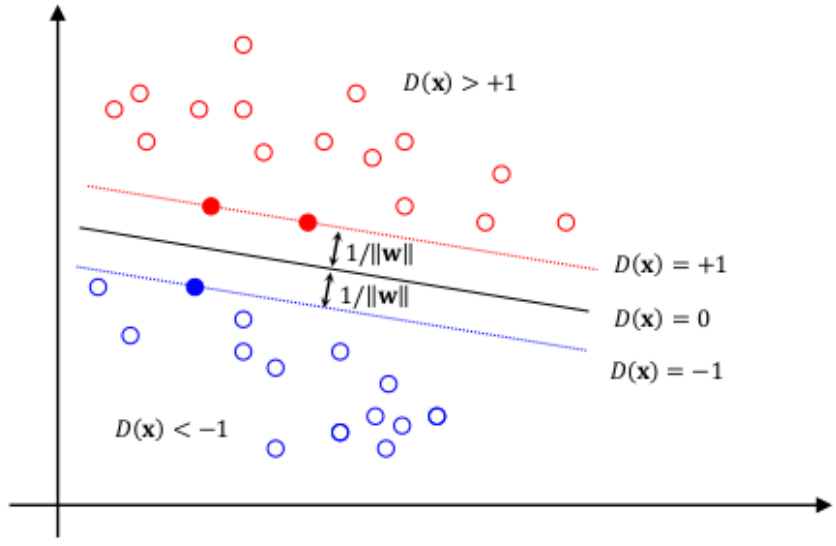
- Abbiamo capito che l'espressione del margine è $\frac{1}{\|\mathbf{w}\|_2}$
- Vogliamo massimizzarlo, imponendo anche che ogni dato del training set sia correttamente classificato dal contorno risultante
- Il problema di ottimizzazione da risolvere è quindi:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$t_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) \geq 1, \;\; n = 1, \ldots, N$$

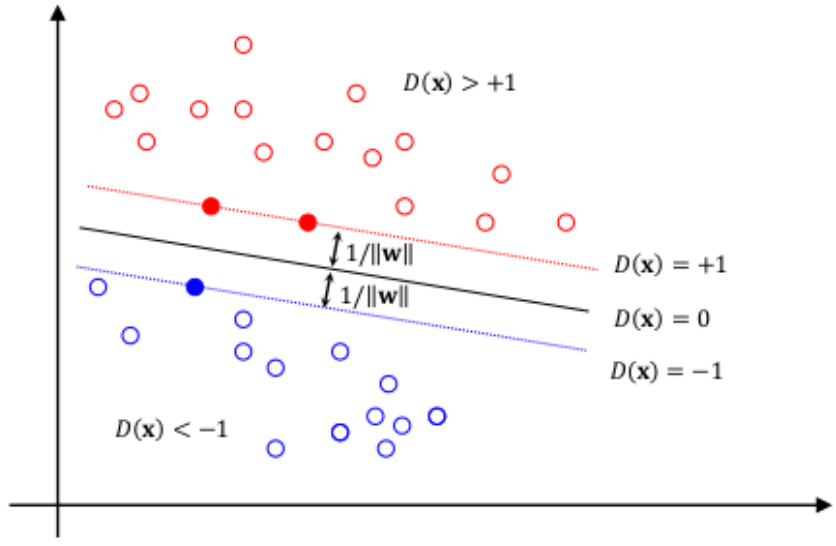
- Questo è un problema di ottimizzazione convesso e vincolato noto come *quadratic programming*
- Non è possibile risolverlo in formula chiusa, servono delle librerie software (es. Scikit-Learn) che generano i valori ottimali di $\mathbf{w}$ e b

Margini



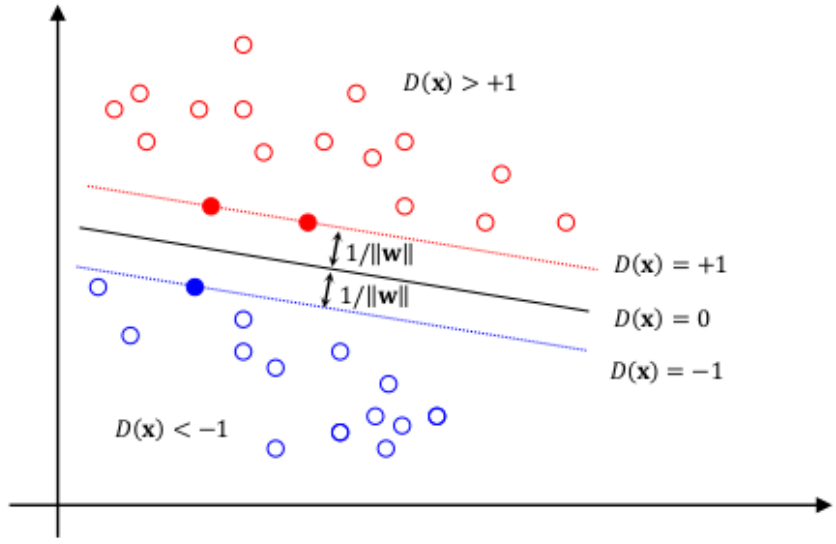
- Per definire il margine abbiamo considerato i dati più vicini al confine, cioè quelli che vanno distanziati maggiormente

Margini



- Per definire il margine abbiamo considerato i dati più vicini al confine, cioè quelli che vanno distanziati maggiormente
- Studiando il problema di ottimizzazione precedente con maggiore dettaglio, si può dimostrare analiticamente che solo quei dati del training set sono necessari per calcolare la soluzione ottimale $\Rightarrow$ support vectors!

Margini



- Per definire il margine abbiamo considerato i dati più vicini al confine, cioè quelli che vanno distanziati maggiormente
- Studiando il problema di ottimizzazione precedente con maggiore dettaglio, si può dimostrare analiticamente che solo quei dati del training set sono necessari per calcolare la soluzione ottimale $\Rightarrow$ support vectors!
- $\Rightarrow$ potremmo addirittura scartare tutti gli altri dati dal training set ed ottenere lo stesso classificatore

SVM: vantaggi

- Definizione della soluzione sulla base di un numero ridotto di support vector (solitamente pochi)

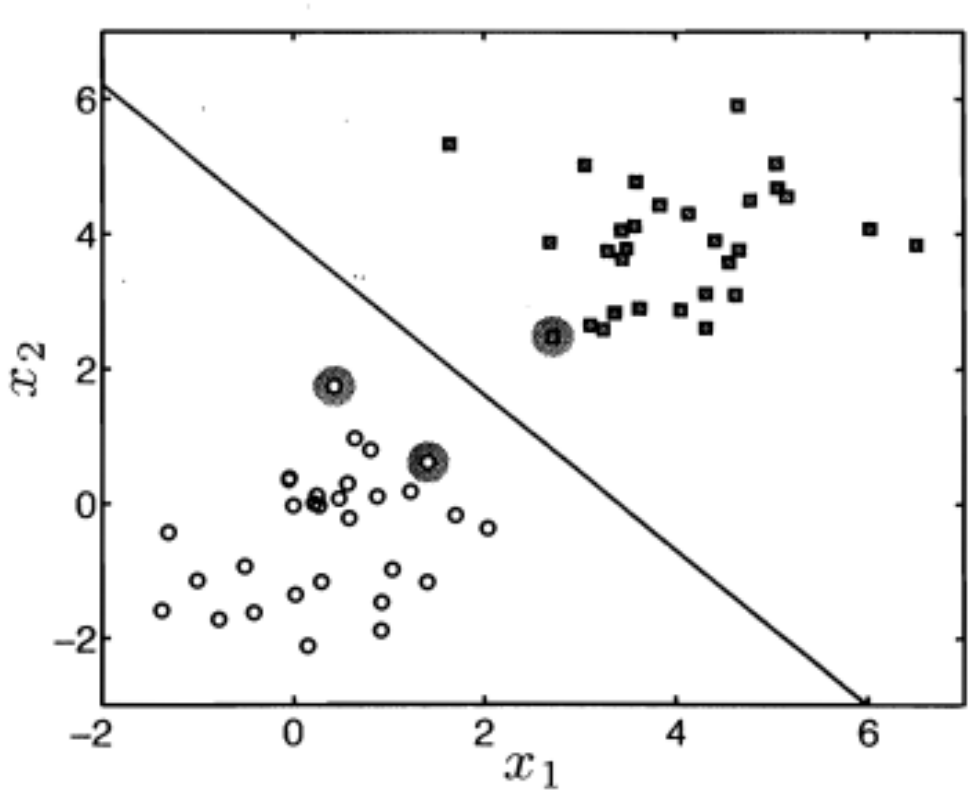
SVM: vantaggi

- Definizione della soluzione sulla base di un numero ridotto di support vector (solitamente pochi)
- Il numero di support vector N_s indica la complessità del problema e può essere dimostrato che l'errore medio (sui possibili training set) è limitato da $\frac{N_s}{N}$

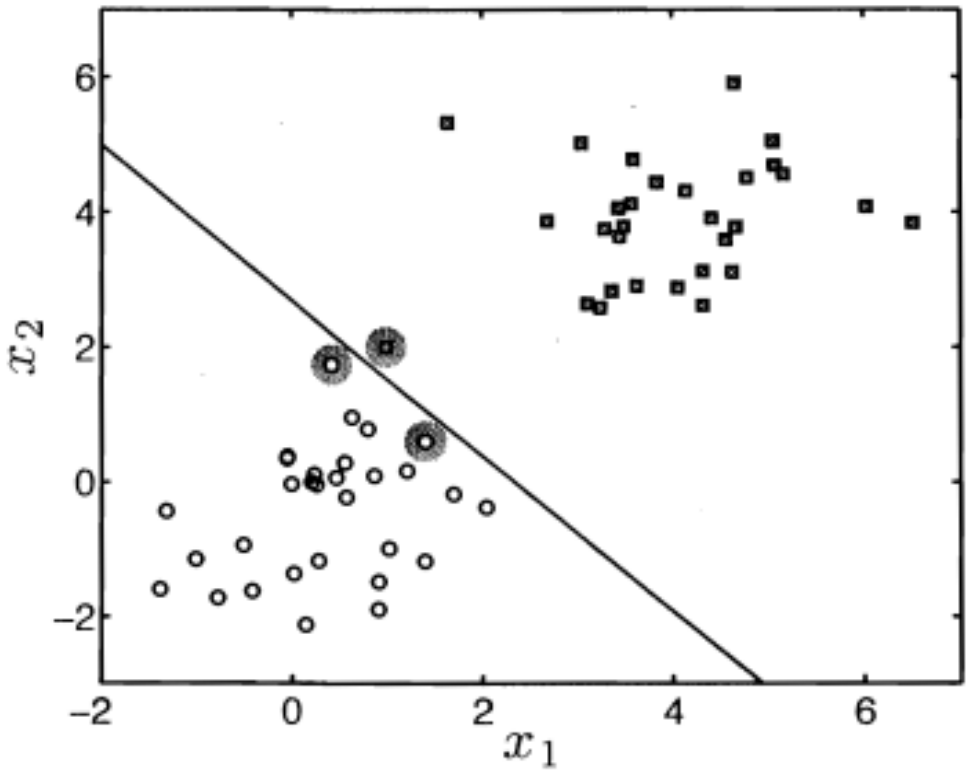
SVM: vantaggi

- Definizione della soluzione sulla base di un numero ridotto di support vector (solitamente pochi)
- Il numero di support vector N_s indica la complessità del problema e può essere dimostrato che l'errore medio (sui possibili training set) è limitato da $\frac{N_s}{N}$
- SVM scala molto bene rispetto alla dimensionalità D dei dati. La complessità computazionale è quadratica rispetto al numero N di dati nel training set. In pratica, il problema può essere risolto per $D = 10^7$ e per N fino a 10^4

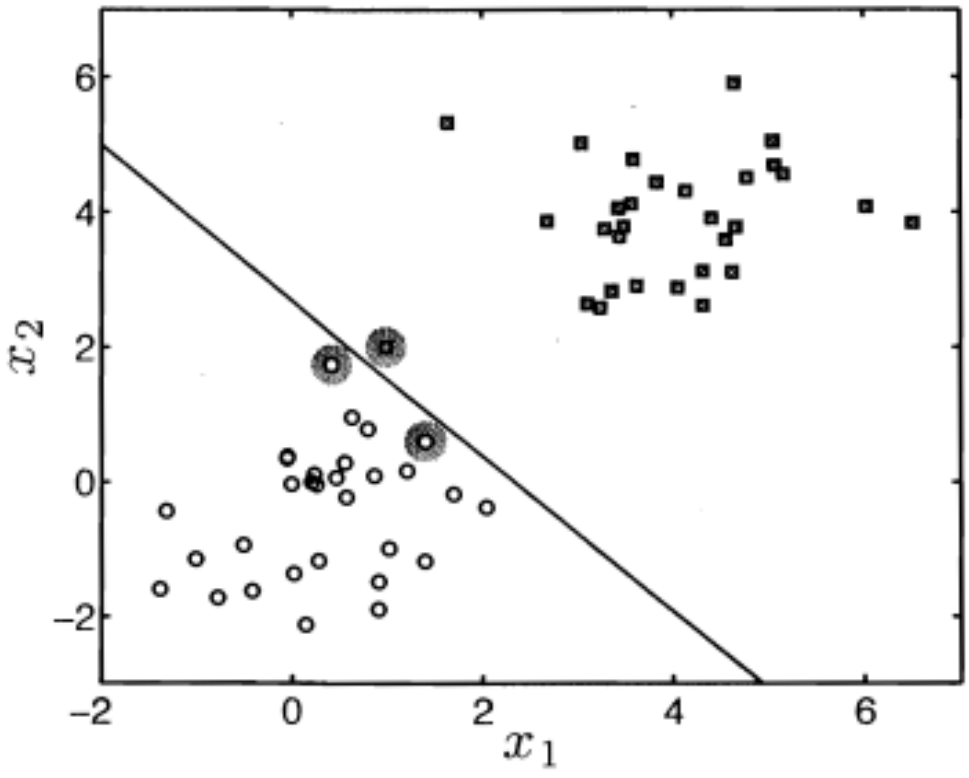
SVM: limitazioni



SVM: limitazioni

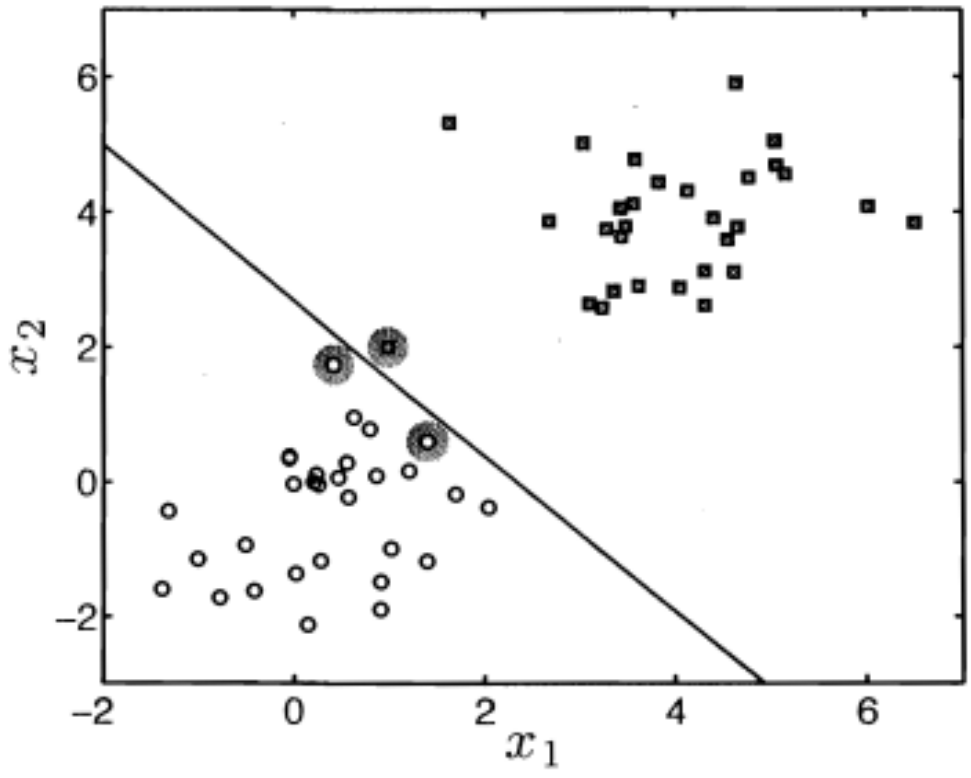


SVM: limitazioni



$\Rightarrow$ questo è un esempio di overfitting

SVM: limitazioni



$\Rightarrow$ questo è un esempio di overfitting

$\Rightarrow$ stiamo basandoci troppo sugli elementi del training set

SVM: pattern non separabili

- Il problema nasce dal fatto che nel nostro problema di ottimizzazione abbiamo imposto che ogni dato del training set dovesse essere classificato correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1$$

SVM: pattern non separabili

- Il problema nasce dal fatto che nel nostro problema di ottimizzazione abbiamo imposto che ogni dato del training set dovesse essere classificato correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1$$

- Ora vogliamo lasciare la possibilità che alcuni dati del training set siano classificati non correttamente:

SVM: pattern non separabili

- Il problema nasce dal fatto che nel nostro problema di ottimizzazione abbiamo imposto che ogni dato del training set dovesse essere classificato correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1$$

- Ora vogliamo lasciare la possibilità che alcuni dati del training set siano classificati non correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1 - \xi_n$$

SVM: pattern non separabili

- Il problema nasce dal fatto che nel nostro problema di ottimizzazione abbiamo imposto che ogni dato del training set dovesse essere classificato correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1$$

- Ora vogliamo lasciare la possibilità che alcuni dati del training set siano classificati non correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1 - \xi_n$$

$\Rightarrow$ slack variables ξ_n, $n = 1, \ldots, N$

SVM: pattern non separabili

- Il problema nasce dal fatto che nel nostro problema di ottimizzazione abbiamo imposto che ogni dato del training set dovesse essere classificato correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1$$

- Ora vogliamo lasciare la possibilità che alcuni dati del training set siano classificati non correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1 - \xi_n$$

$\Rightarrow$ slack variables ξ_n, $n = 1, \ldots, N$

- Ovviamente vogliamo avere la possibilità di decidere quanti dati vogliamo classificare non correttamente. Quindi regolarizziamo il problema di ottimizzazione aggiungendo un iperparametro:

SVM: pattern non separabili

- Il problema nasce dal fatto che nel nostro problema di ottimizzazione abbiamo imposto che ogni dato del training set dovesse essere classificato correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1$$

- Ora vogliamo lasciare la possibilità che alcuni dati del training set siano classificati non correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1 - \xi_n$$

$\Rightarrow$ slack variables $\xi_n,\ n = 1, \ldots, N$

- Ovviamente vogliamo avere la possibilità di decidere quanti dati vogliamo classificare non correttamente. Quindi regolarizziamo il problema di ottimizzazione aggiungendo un iperparametro:

$$\arg\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{n=1}^{N} \xi_n$$

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1, \ \ n = 1, \ldots, N$$

SVM: pattern non separabili

- Il problema nasce dal fatto che nel nostro problema di ottimizzazione abbiamo imposto che ogni dato del training set dovesse essere classificato correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1$$

- Ora vogliamo lasciare la possibilità che alcuni dati del training set siano classificati non correttamente:

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1 - \xi_n$$

$\Rightarrow$ slack variables $\xi_n,\ n = 1, \ldots, N$

- Ovviamente vogliamo avere la possibilità di decidere quanti dati vogliamo classificare non correttamente. Quindi regolarizziamo il problema di ottimizzazione aggiungendo un iperparametro:

$$\arg\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{n=1}^{N} \xi_n$$

$$t_n \left(\mathbf{w}^T \mathbf{x}_n + b \right) \geq 1,\ \ n = 1, \ldots, N$$

il problema è sempre un quadratic programming convesso vincolato

SVM: pattern non separabili

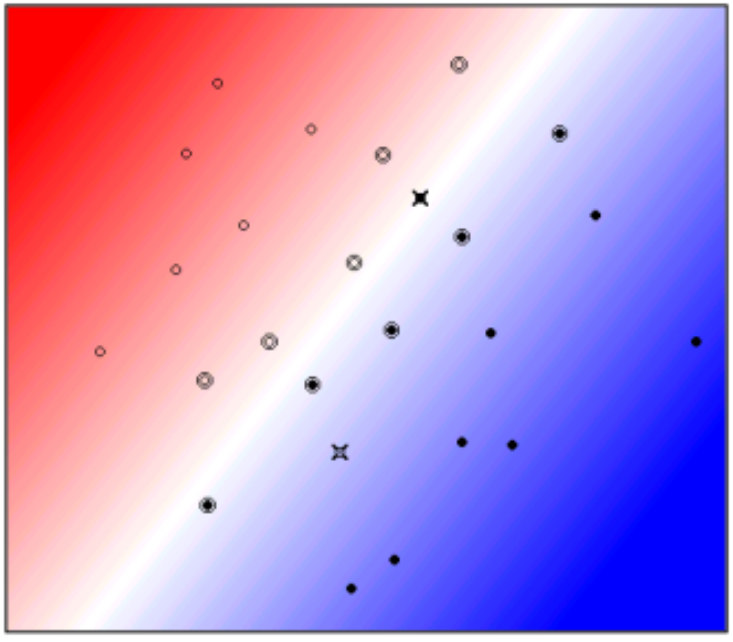
$C \to +\infty$: ritorniamo al caso iniziale

SVM: pattern non separabili

$C \to +\infty$: ritorniamo al caso iniziale

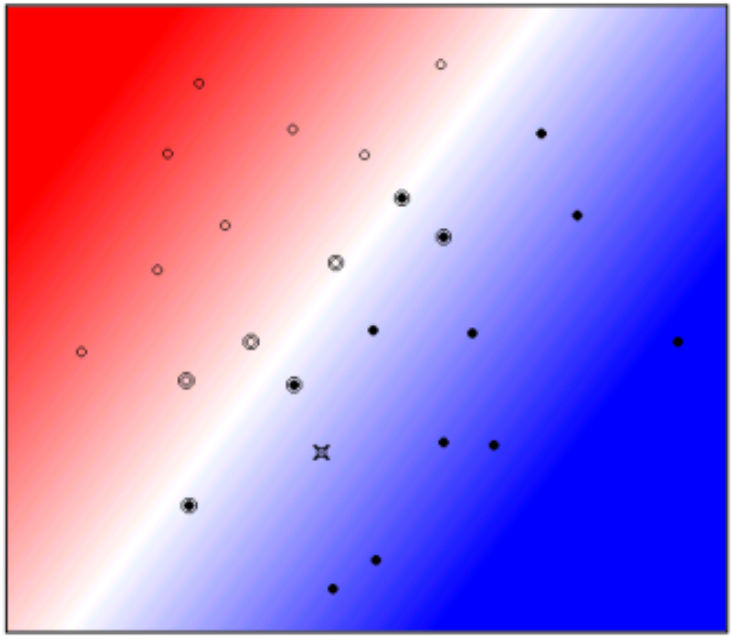
$C = 10$

2 errori, margine maggiore



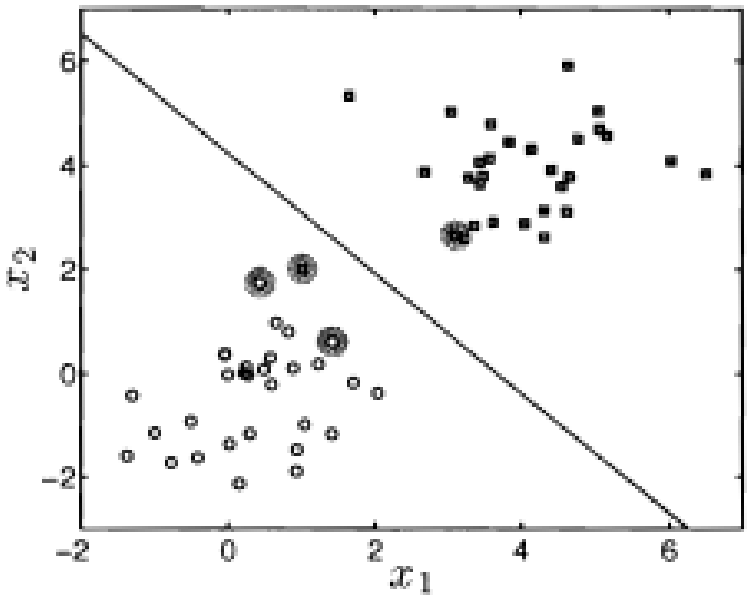
$C = 200$

1 solo errore, margine minore

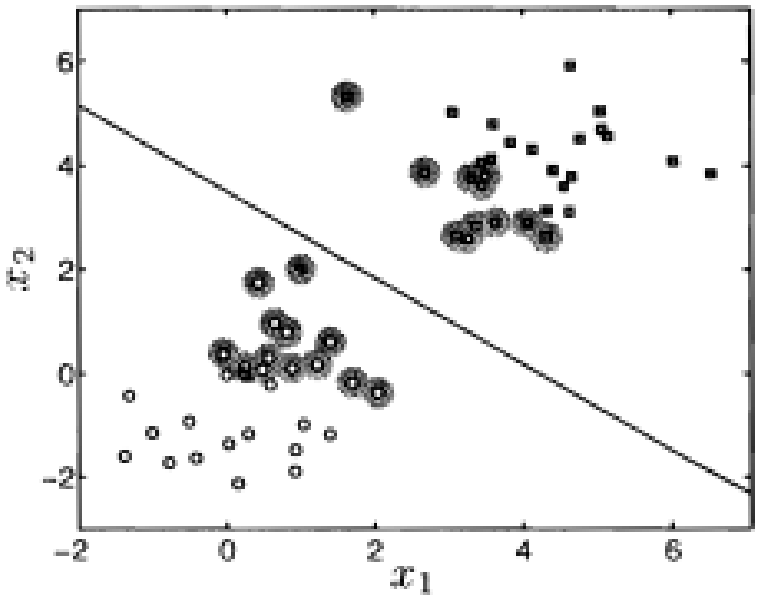


SVM: pattern non separabili

$C \to +\infty$: ritorniamo al caso iniziale

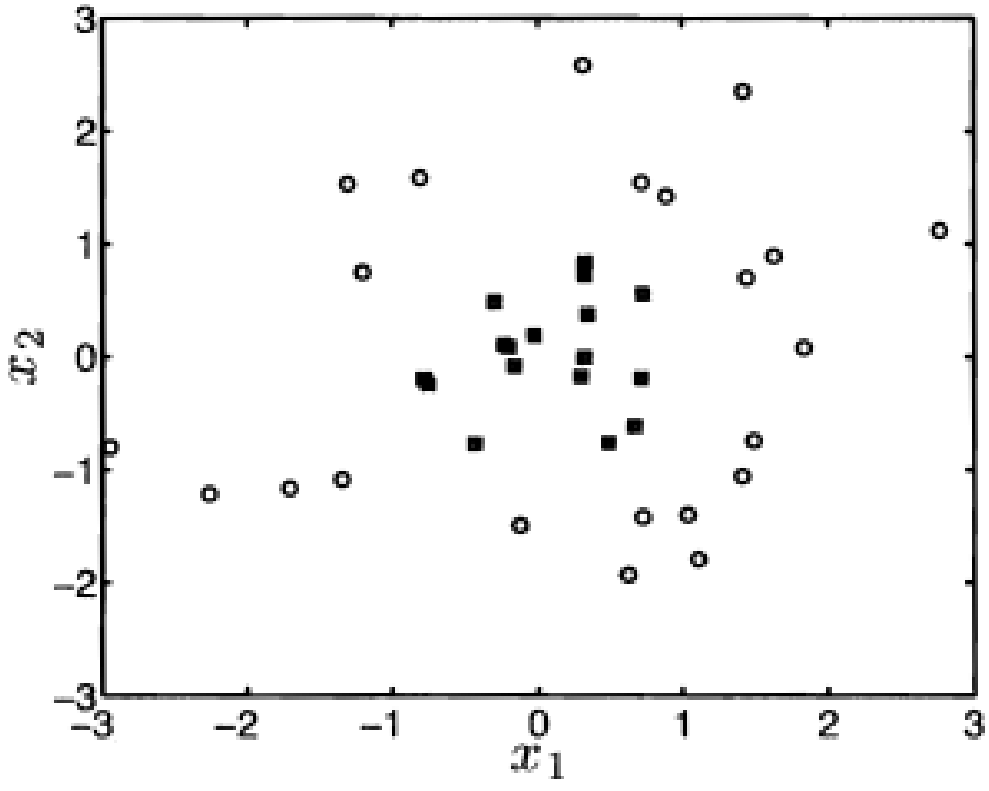


(a) $C = 1$



(b) $C = 0.01$

SVM: altre limitazioni



Un classificatore lineare non può funzionare su un dataset di questo tipo

SVM nonlineare

- Si definisce un mapping Φ nonlineare dei dati verso uno spazio con più alta dimensionalità:

$$\Phi : \mathbb{R}^D \rightarrow \mathbb{R}^M, \ \Phi(\mathbf{x}_i) \triangleq [g_1(\mathbf{x}_i), \ldots, g_M(\mathbf{x}_i)]$$

SVM nonlineare

- Si definisce un mapping Φ nonlineare dei dati verso uno spazio con più alta dimensionalità:

$$\Phi : \mathbb{R}^D \rightarrow \mathbb{R}^M, \ \Phi(\mathbf{x}_i) \triangleq [g_1(\mathbf{x}_i), \ldots, g_M(\mathbf{x}_i)]$$

- In uno spazio $\mathbb{R}^M$ con maggiore dimensionalità è più facile trovare un iperpiano che linearmente classifichi correttamente tutti i dati

SVM nonlineare

- Si definisce un mapping Φ nonlineare dei dati verso uno spazio con più alta dimensionalità:

$$\Phi : \mathbb{R}^D \rightarrow \mathbb{R}^M, \ \Phi(\mathbf{x}_i) \triangleq [g_1(\mathbf{x}_i), \ldots, g_M(\mathbf{x}_i)]$$

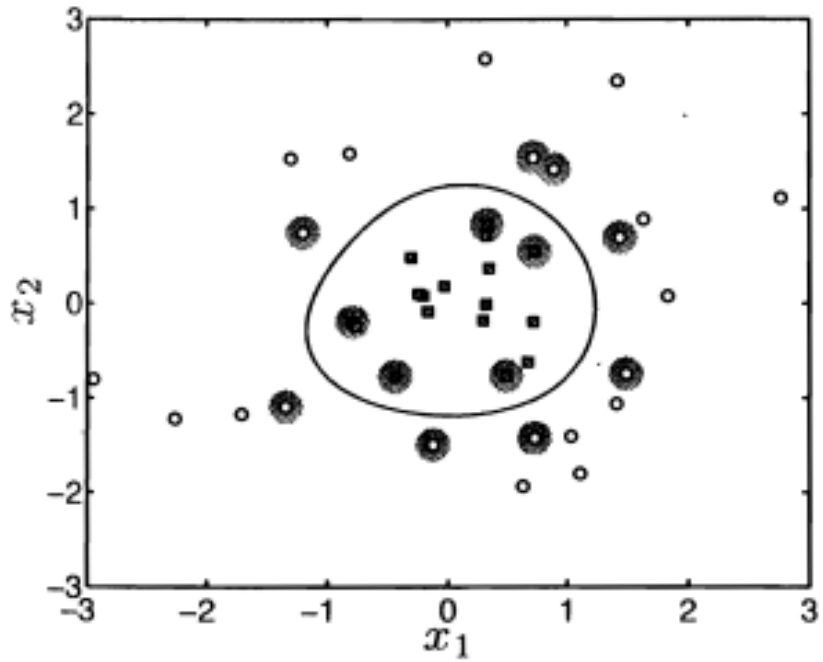
- In uno spazio $\mathbb{R}^M$ con maggiore dimensionalità è più facile trovare un iperpiano che linearmente classifichi correttamente tutti i dati
- L'iperpiano trovato in $\mathbb{R}^M$ quando viene riportato in $\mathbb{R}^D$ diventa una superficie nonlineare arbitrariamente complessa

SVM nonlineare

- Si definisce un mapping Φ nonlineare dei dati verso uno spazio con più alta dimensionalità:

$$\Phi : \mathbb{R}^D \rightarrow \mathbb{R}^M, \ \Phi(\mathbf{x}_i) \triangleq [g_1(\mathbf{x}_i), \ldots, g_M(\mathbf{x}_i)]$$

- In uno spazio $\mathbb{R}^M$ con maggiore dimensionalità è più facile trovare un iperpiano che linearmente classifichi correttamente tutti i dati
- L'iperpiano trovato in $\mathbb{R}^M$ quando viene riportato in $\mathbb{R}^D$ diventa una superficie nonlineare arbitrariamente complessa



SVM nonlineare: kernels

- Quale mapping/trasformazione Φ e quale dimensionalità M scegliere?

SVM nonlineare: kernels

- Quale mapping/trasformazione Φ e quale dimensionalità M scegliere?
- Analizzando la teoria della SVM, è possibile dimostrare che non è veramente necessario definire analiticamente un mapping Φ, ma basta saper calcolare il prodotto $\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \triangleq K(\mathbf{x}_i, \mathbf{x}_j)$, che definiremo kernel

SVM nonlineare: kernels

- Quale mapping/trasformazione Φ e quale dimensionalità M scegliere?
- Analizzando la teoria della SVM, è possibile dimostrare che non è veramente necessario definire analiticamente un mapping Φ, ma basta saper calcolare il prodotto $\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \triangleq K(\mathbf{x}_i, \mathbf{x}_j)$, che definiremo kernel
- Definire un kernel è molto più facile che definire un mapping Φ e permette di lavorare con dimensionalità M maggiori di 10^8 o anche infinite (necessarie per alcuni problemi)

SVM nonlineare: esempi

- Polinomio di grado q: Le componenti $g_i, i = 1, \ldots, M$, del mapping Φ (questo è uno dei pochi casi dove è possibile definire analiticamente Φ) sono tutte le possibili combinazioni di prodotti tra elevamenti a potenza minori di q delle componenti di $\mathbf{x}$. Esempio: $D = 2$, $q = 2$:

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

quindi $M = 9$. Si può dimostrare che il kernel associato è:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \left[\mathbf{x}_i^T \mathbf{x}_j + 1\right]^q$$

SVM nonlineare: esempi

- Polinomio di grado q: Le componenti $g_i, i = 1, \ldots, M$, del mapping Φ (questo è uno dei pochi casi dove è possibile definire analiticamente Φ) sono tutte le possibili combinazioni di prodotti tra elevamenti a potenza minori di q delle componenti di $\mathbf{x}$. Esempio: $D = 2$, $q = 2$:

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

  quindi $M = 9$. Si può dimostrare che il kernel associato è:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \left[\mathbf{x}_i^T \mathbf{x}_j + 1\right]^q$$

- Radial Basis Function (RBF) di ampiezza $\frac{1}{\gamma}$:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\gamma^2 \|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2}}$$

SVM nonlineare: esempi

- Polinomio di grado q: Le componenti $g_i, i = 1, \ldots, M$, del mapping Φ (questo è uno dei pochi casi dove è possibile definire analiticamente Φ) sono tutte le possibili combinazioni di prodotti tra elevamenti a potenza minori di q delle componenti di $\mathbf{x}$. Esempio: $D = 2$, $q = 2$:

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

  quindi $M = 9$. Si può dimostrare che il kernel associato è:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \left[\mathbf{x}_i^T\mathbf{x}_j + 1\right]^q$$

- Radial Basis Function (RBF) di ampiezza $\frac{1}{\gamma}$:

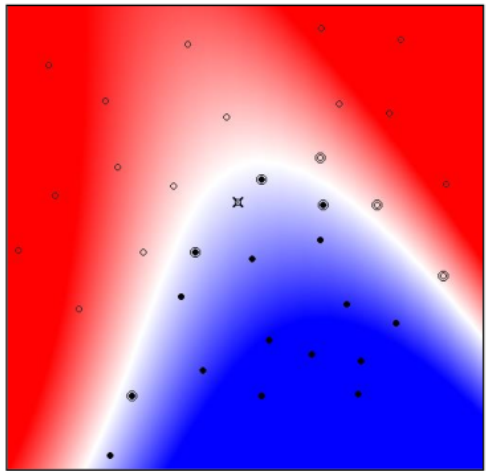
$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\gamma^2\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2}}$$

- 2-layer Neural Network (NN): Risolvere SVN con questo kernel è equivalente a fare training di una NN con pesi e hidden layer dipendenti dai valori degli iperparametri $\eta, \alpha > 0$:

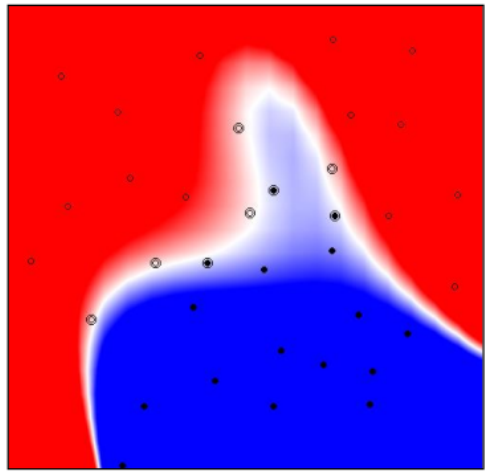
$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\eta(\mathbf{x}_i^T\mathbf{x}_j) + \alpha\right)$$

SVM nonlineare: esempi

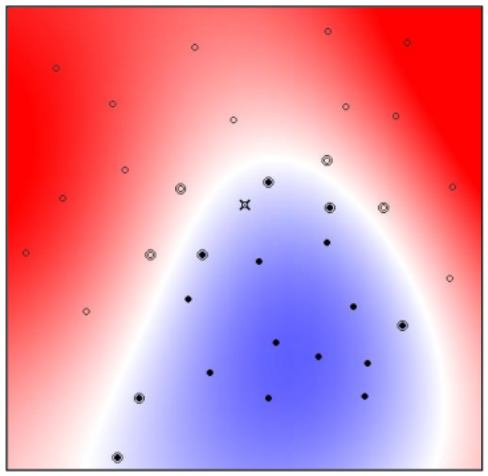
Polinomio $q = 2$



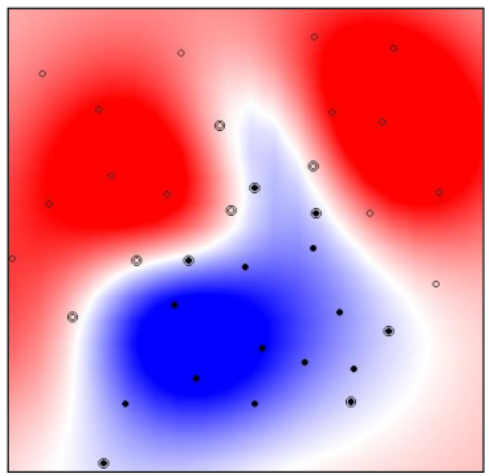
Polinomio $q = 10$



RBF $\sigma = 1$

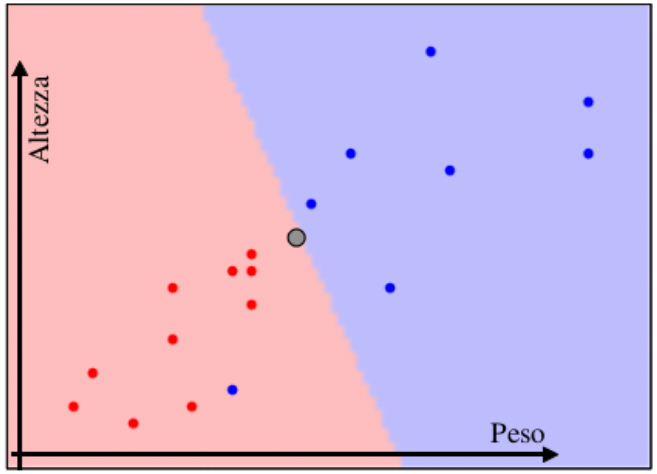


RBF $\sigma = 0.2$

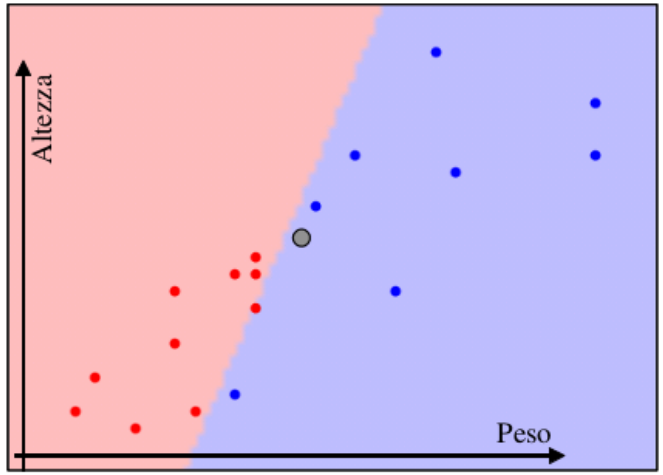


$$\sigma = \frac{1}{\gamma}$$

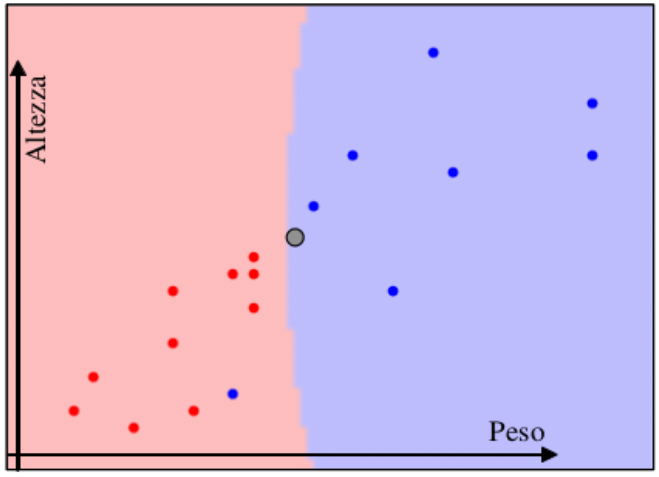
SVM nonlineare: esempi



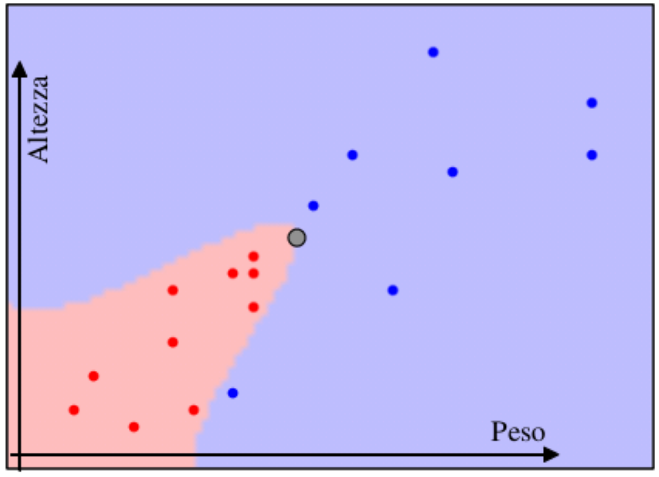
Lineare, $C = 10$



Lineare, $C = 500$

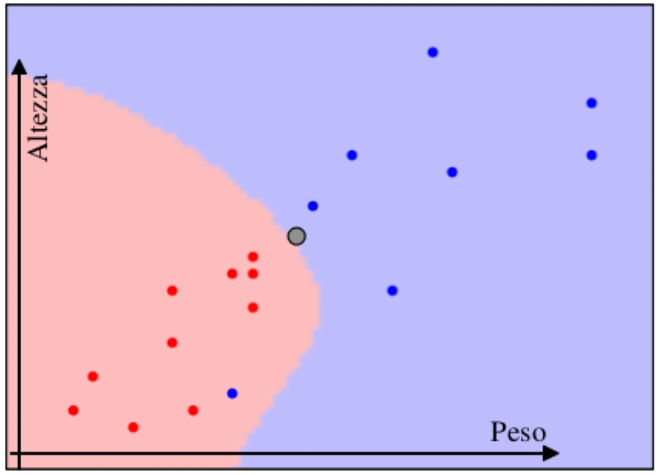


Polinomio $q = 3, C = 10$

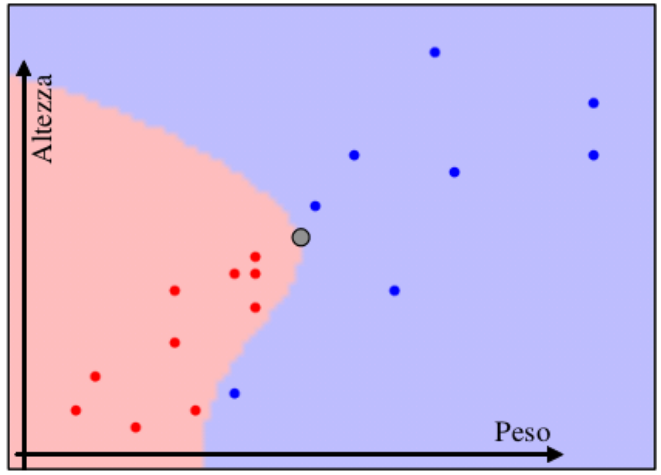


Polinomio $q = 3, C = 500$

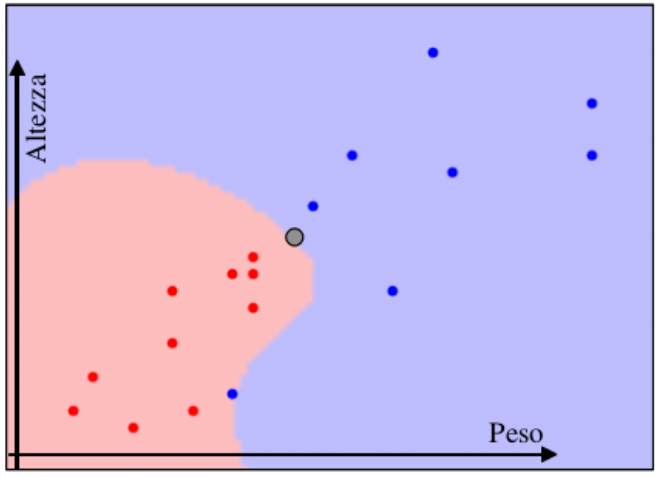
SVM nonlineare: esempi



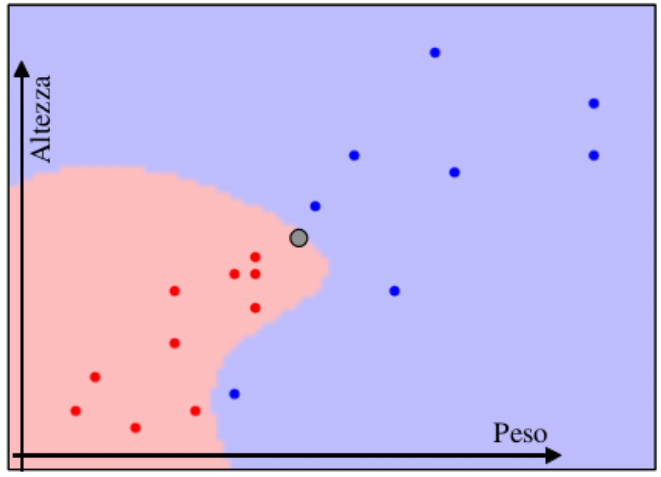
RBF, $\gamma = 5, C = 10$



RBF,$\gamma = 5, C = 500$



RBF,$\gamma = 10, C = 10$



RBF,$\gamma = 10, C = 500$

SVM multiclasse

- Come classificare $\mathbf{x}_{\text{new}}$ quando le classi sono più di 2?
  Immaginiamo di avere T classi $c_1, \ldots, c_T$

SVM multiclasse

- Come classificare $\mathbf{x}_{\text{new}}$ quando le classi sono più di 2?
  Immaginiamo di avere T classi $c_1, \ldots, c_T$

- SVM One-Against-All: per ogni classe c_i, si costruisce un classificatore SVM dividendo il training set in due classi: una classe per i dati appartenenti a c_i e un'altra classe per tutti gli altri dati. Si valuta (secondo qualche metrica) con quale grado di confidenza $\mathbf{x}_{\text{new}}$ appartiene a c_i (se appartiene). Dopo aver ripetuto il confronto su tutte le T classi, si classifica $\mathbf{x}_{\text{new}}$ come appartenente alla classe dove è stato registrato il più alto grado di confidenza

SVM multiclasse

- Come classificare $\mathbf{x}_{\text{new}}$ quando le classi sono più di 2?
  Immaginiamo di avere T classi $c_1, \ldots, c_T$

- SVM One-Against-All: per ogni classe c_i, si costruisce un classificatore SVM dividendo il training set in due classi: una classe per i dati appartenenti a c_i e un'altra classe per tutti gli altri dati. Si valuta (secondo qualche metrica) con quale grado di confidenza $\mathbf{x}_{\text{new}}$ appartiene a c_i (se appartiene). Dopo aver ripetuto il confronto su tutte le T classi, si classifica $\mathbf{x}_{\text{new}}$ come appartenente alla classe dove è stato registrato il più alto grado di confidenza $\Rightarrow$ vengono costruite T SVM

SVM multiclasse

- Come classificare $\mathbf{x}_{\text{new}}$ quando le classi sono più di 2?
  Immaginiamo di avere T classi $c_1, \ldots, c_T$

- SVM One-Against-All: per ogni classe c_i, si costruisce un classificatore SVM dividendo il training set in due classi: una classe per i dati appartenenti a c_i e un'altra classe per tutti gli altri dati. Si valuta (secondo qualche metrica) con quale grado di confidenza $\mathbf{x}_{\text{new}}$ appartiene a c_i (se appartiene). Dopo aver ripetuto il confronto su tutte le T classi, si classifica $\mathbf{x}_{\text{new}}$ come appartenente alla classe dove è stato registrato il più alto grado di confidenza $\Rightarrow$ vengono costruite T SVM

- SVM One-Against-One: si confronta tramite SVM ogni classe c_i con ogni altra possibile classe c_j, $i \neq j$, e si valuta di volta in volta a quale classe viene assegnato $\mathbf{x}_{\text{new}}$. Dopo aver svolto tutti i confronti, si classifica $\mathbf{x}_{\text{new}}$ come appartenente alla classe dove è stato assegnato più volte

SVM multiclasse

- Come classificare $\mathbf{x}_{\text{new}}$ quando le classi sono più di 2?
  Immaginiamo di avere T classi $c_1, \ldots, c_T$

- SVM One-Against-All: per ogni classe c_i, si costruisce un classificatore SVM dividendo il training set in due classi: una classe per i dati appartenenti a c_i e un'altra classe per tutti gli altri dati. Si valuta (secondo qualche metrica) con quale grado di confidenza $\mathbf{x}_{\text{new}}$ appartiene a c_i (se appartiene). Dopo aver ripetuto il confronto su tutte le T classi, si classifica $\mathbf{x}_{\text{new}}$ come appartenente alla classe dove è stato registrato il più alto grado di confidenza $\Rightarrow$ vengono costruite T SVM

- SVM One-Against-One: si confronta tramite SVM ogni classe c_i con ogni altra possibile classe c_j, $i \neq j$, e si valuta di volta in volta a quale classe viene assegnato $\mathbf{x}_{\text{new}}$. Dopo aver svolto tutti i confronti, si classifica $\mathbf{x}_{\text{new}}$ come appartenente alla classe dove è stato assegnato più volte $\Rightarrow$ vengono costruite $\frac{T(T-1)}{2}$ SVM

SVM multiclasse

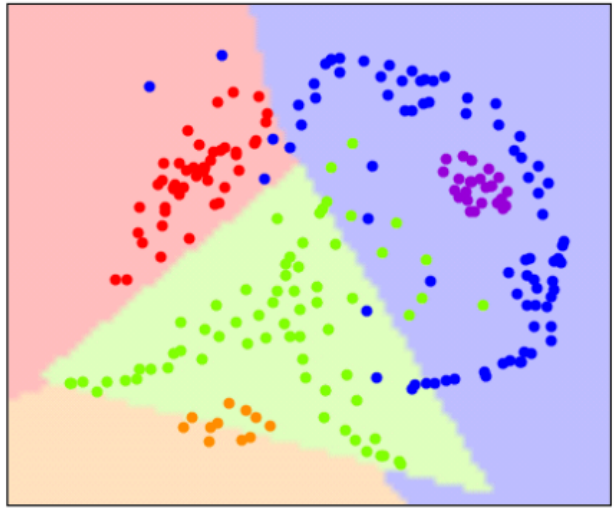
- Come classificare $\mathbf{x}_{\text{new}}$ quando le classi sono più di 2?
  Immaginiamo di avere T classi $c_1, \ldots, c_T$

- SVM One-Against-All: per ogni classe c_i, si costruisce un classificatore SVM dividendo il training set in due classi: una classe per i dati appartenenti a c_i e un'altra classe per tutti gli altri dati. Si valuta (secondo qualche metrica) con quale grado di confidenza $\mathbf{x}_{\text{new}}$ appartiene a c_i (se appartiene). Dopo aver ripetuto il confronto su tutte le T classi, si classifica $\mathbf{x}_{\text{new}}$ come appartenente alla classe dove è stato registrato il più alto grado di confidenza $\Rightarrow$ vengono costruite T SVM

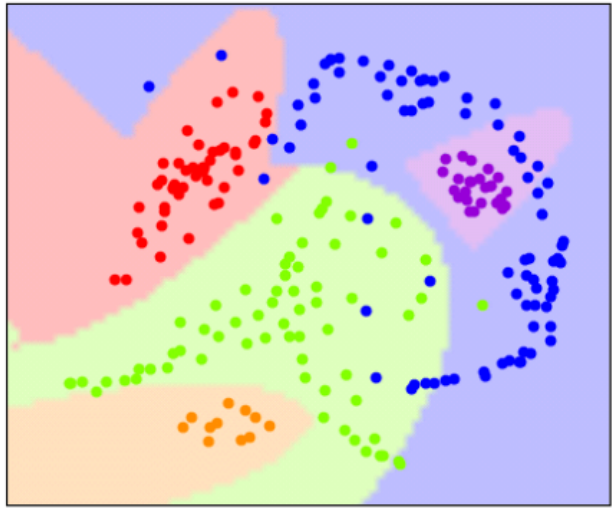
- SVM One-Against-One: si confronta tramite SVM ogni classe c_i con ogni altra possibile classe c_j, $i \neq j$, e si valuta di volta in volta a quale classe viene assegnato $\mathbf{x}_{\text{new}}$. Dopo aver svolto tutti i confronti, si classifica $\mathbf{x}_{\text{new}}$ come appartenente alla classe dove è stato assegnato più volte $\Rightarrow$ vengono costruite $\frac{T(T-1)}{2}$ SVM

$\Rightarrow$ OAO molto più accurato di OAA, ma anche molto più lento

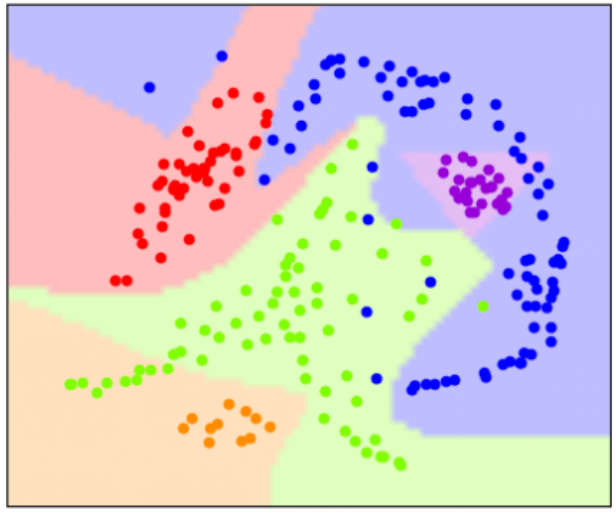
SVM multiclasse: esempio



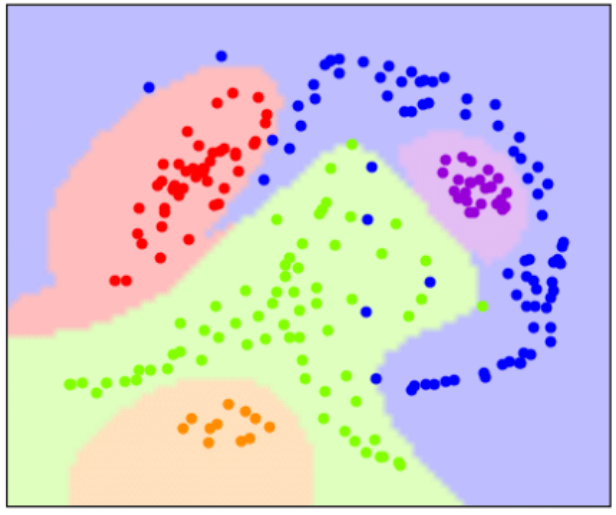
Lineare, $C = 100$



Polinomio $q = 2, C = 100$



Polinomio $q = 7, C = 100$



RBF, $\gamma = 5, C = 100$

SVM: implementazioni

Librerie

- LIBSVM `https://www.csie.ntu.edu.tw/ cjlin/libsvm/`
  - Implementato in Scikit-Learn: sklearn.svm.SVC
  - Utilizza OAO per il multiclasse

- LIBLINEAR `https://www.csie.ntu.edu.tw/ cjlin/liblinear/`
  - Implementato in Scikit-Learn: sklearn.svm.LinearSVC
  - Utilizza OAA per il multiclasse

SVM: implementazioni

Librerie

- LIBSVM `https://www.csie.ntu.edu.tw/ cjlin/libsvm/`
  - Implementato in Scikit-Learn: sklearn.svm.SVC
  - Utilizza OAO per il multiclasse

- LIBLINEAR `https://www.csie.ntu.edu.tw/ cjlin/liblinear/`
  - Implementato in Scikit-Learn: sklearn.svm.LinearSVC
  - Utilizza OAA per il multiclasse

Consigli

SVM: implementazioni

Librerie

- LIBSVM `https://www.csie.ntu.edu.tw/ cjlin/libsvm/`
  - Implementato in Scikit-Learn: sklearn.svm.SVC
  - Utilizza OAO per il multiclasse

- LIBLINEAR `https://www.csie.ntu.edu.tw/ cjlin/liblinear/`
  - Implementato in Scikit-Learn: sklearn.svm.LinearSVC
  - Utilizza OAA per il multiclasse

Consigli

- Quando i dati hanno alta dimensionalità ($D > 5000$) si usa SVM lineare: in questo caso solitamente i dati sono molto sparsi e iperpiani sono sufficienti. L'unico iperparametro da calibrare è C

SVM: implementazioni

Librerie

- LIBSVM `https://www.csie.ntu.edu.tw/ cjlin/libsvm/`
  - Implementato in Scikit-Learn: sklearn.svm.SVC
  - Utilizza OAO per il multiclasse
- LIBLINEAR `https://www.csie.ntu.edu.tw/ cjlin/liblinear/`
  - Implementato in Scikit-Learn: sklearn.svm.LinearSVC
  - Utilizza OAA per il multiclasse

Consigli

- Quando i dati hanno alta dimensionalità ($D > 5000$) si usa SVM lineare: in questo caso solitamente i dati sono molto sparsi e iperpiani sono sufficienti. L'unico iperparametro da calibrare è C
- Quando $D < 20$ si usa solitamente SVM nonlineare con kernel RBF

SVM: implementazioni

Librerie

- LIBSVM `https://www.csie.ntu.edu.tw/ cjlin/libsvm/`
  - Implementato in Scikit-Learn: sklearn.svm.SVC
  - Utilizza OAO per il multiclasse
- LIBLINEAR `https://www.csie.ntu.edu.tw/ cjlin/liblinear/`
  - Implementato in Scikit-Learn: sklearn.svm.LinearSVC
  - Utilizza OAA per il multiclasse

Consigli

- Quando i dati hanno alta dimensionalità ($D > 5000$) si usa SVM lineare: in questo caso solitamente i dati sono molto sparsi e iperpiani sono sufficienti. L'unico iperparametro da calibrare è C
- Quando $D < 20$ si usa solitamente SVM nonlineare con kernel RBF
- La calibrazione degli iperparametri avviene, come sempre, con validation set e cross-validation

SVM: implementazioni

Librerie

- LIBSVM `https://www.csie.ntu.edu.tw/ cjlin/libsvm/`
  - Implementato in Scikit-Learn: sklearn.svm.SVC
  - Utilizza OAO per il multiclasse
- LIBLINEAR `https://www.csie.ntu.edu.tw/ cjlin/liblinear/`
  - Implementato in Scikit-Learn: sklearn.svm.LinearSVC
  - Utilizza OAA per il multiclasse

Consigli

- Quando i dati hanno alta dimensionalità ($D > 5000$) si usa SVM lineare: in questo caso solitamente i dati sono molto sparsi e iperpiani sono sufficienti. L'unico iperparametro da calibrare è C
- Quando $D < 20$ si usa solitamente SVM nonlineare con kernel RBF
- La calibrazione degli iperparametri avviene, come sempre, con validation set e cross-validation
- Nel caso multiclasse, quando il numero di classi è troppo elevato, OAA è una scelta obbligata