Computazione per l'interazione naturale: Classificatori non probabilistici

Corso di Interazione uomo-macchina II

Prof. Giuseppe Boccignone

Dipartimento di Scienze dell'Informazione
Università di Milano

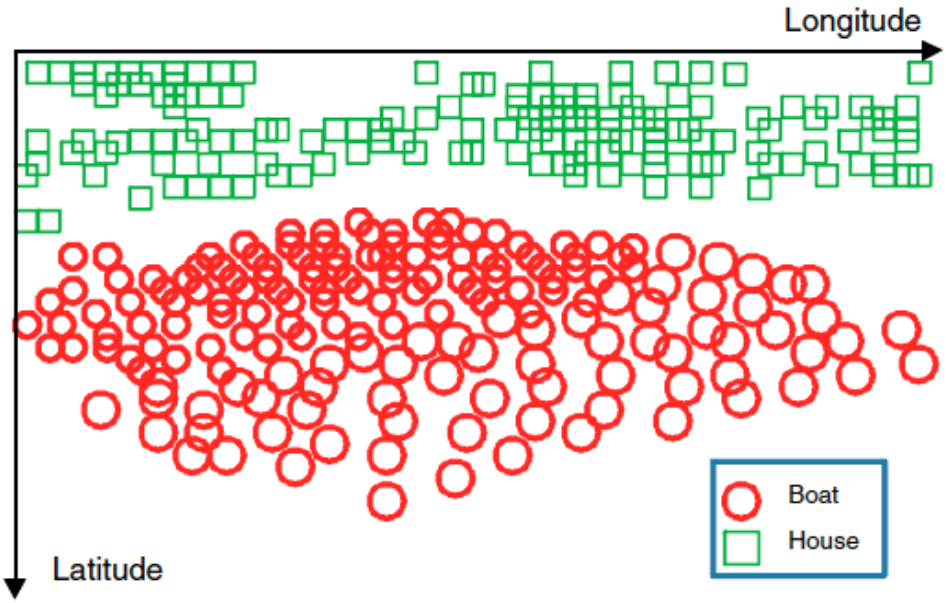
boccignone@di.unimi.it
http://boccignone.di.unimi.it/IUM2_2014.html

Apprendimento supervisionato //Classificazione

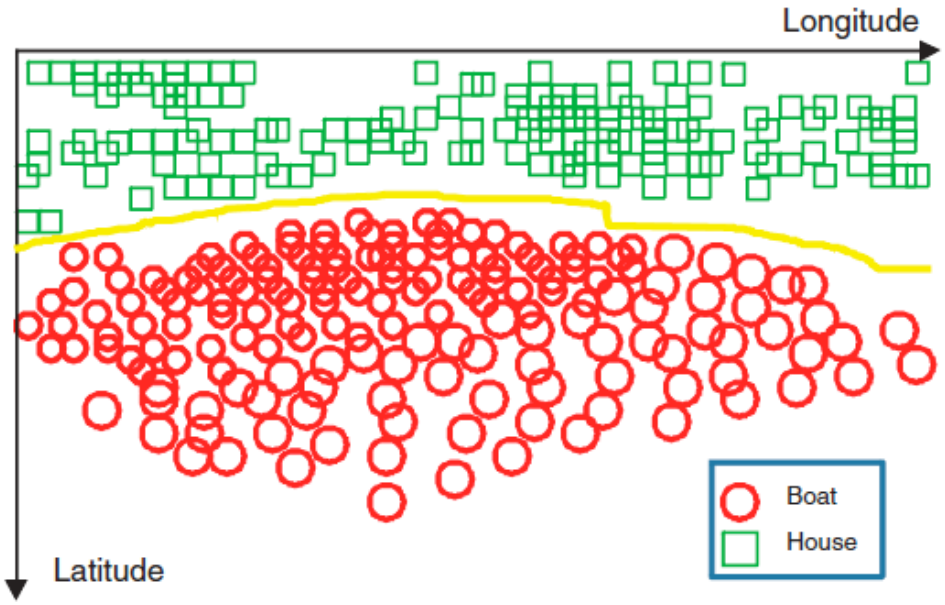
	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization Discrete Output y $\in \{1, \ldots, K\}$	clustering
Continuous	regression Continuous Output $y \in R$	dimensionality reduction

Apprendimento supervisionato //Classificazione

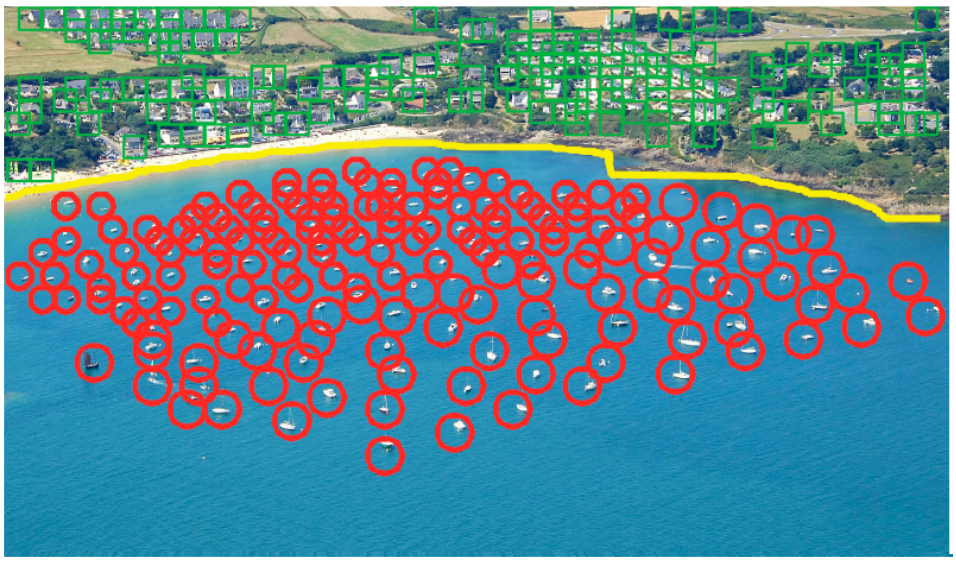
Apprendimento supervisionato //Classificazione



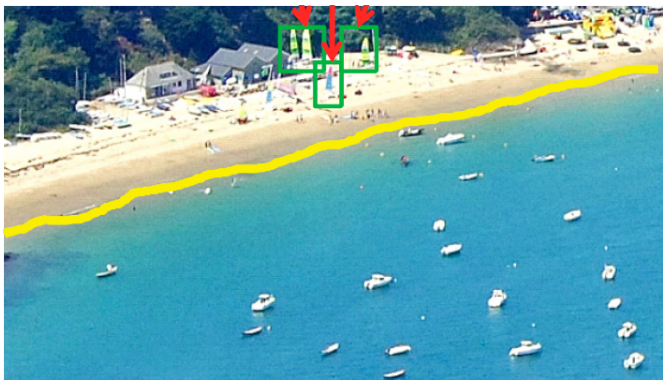
Apprendimento supervisionato //Classificazione



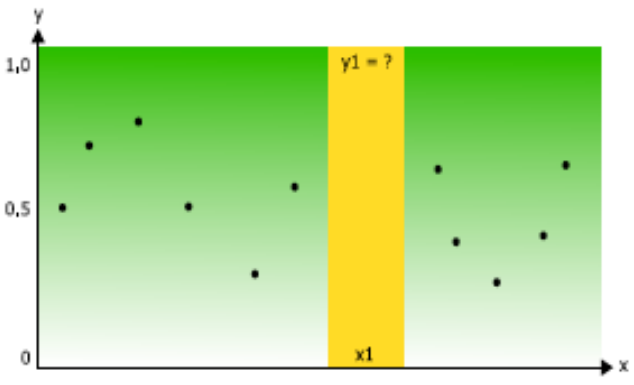
Apprendimento supervisionato //Classificazione



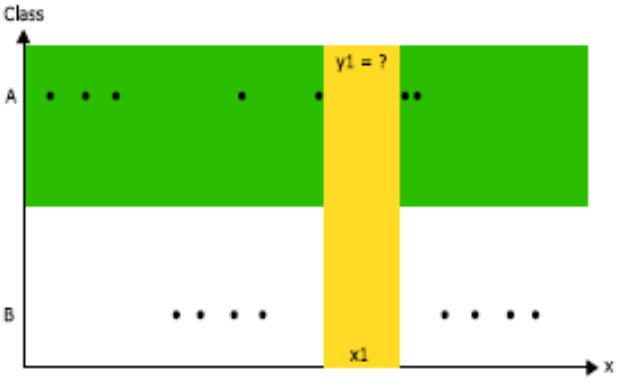
Apprendimento supervisionato //Classificazione



Apprendimento supervisionato //Classificazione vs regressione

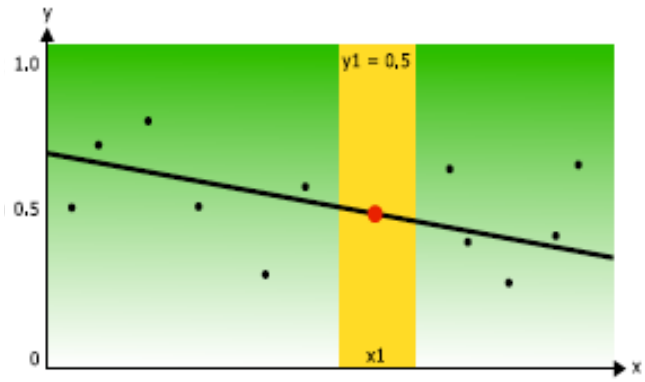


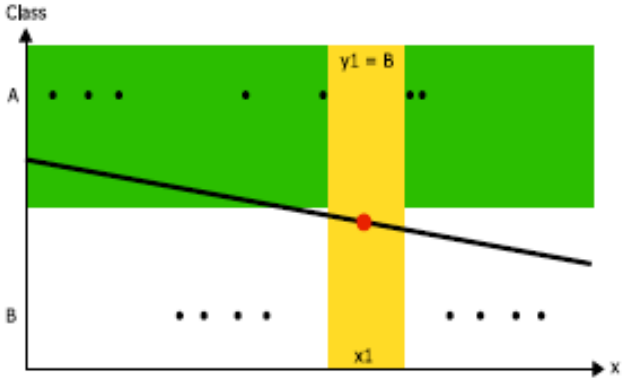
Regressione



Classificazione

Apprendimento supervisionato //Classificazione vs regressione

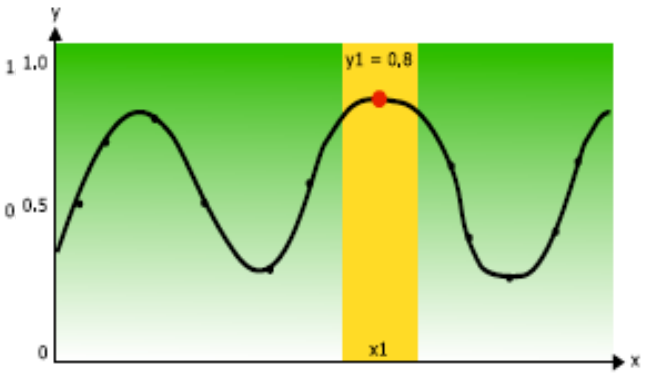


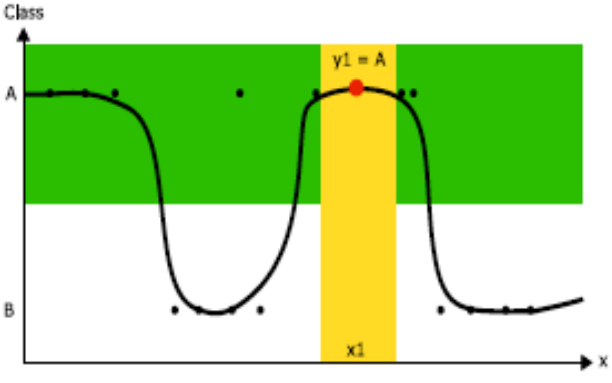


Regressione Classificazione

$$y = ax + b$$

Apprendimento supervisionato //Classificazione vs regressione



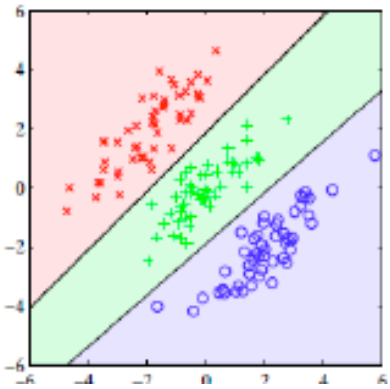


Regressione Classificazione

$$y = w_0x^0 + w_1x^1 + \ldots + w_nx^n$$

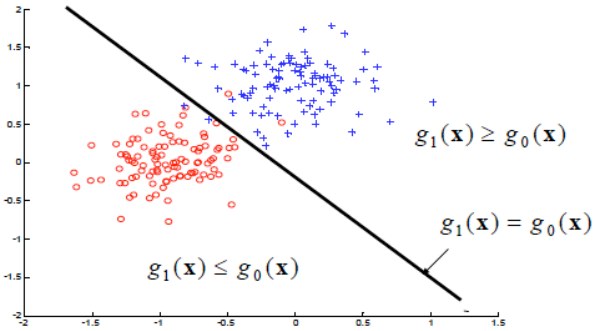
Terminologia generale

- I valori y che si vogliono predire hanno come dominio un insieme discreto,in cui ogni valore denota una **classe** C_i
- Lo spazio di input è diviso in **regioni di decisione** R_i
- Modelli lineari di classificazione: i **confini di decisione** sono funzioni lineari dell'input **x** (iperpiani a m – 1 dimensioni nello spazio m-dimensionale delle features)
- Insiemi di **dati linearmente separabili**: le classi possono essere separate in modo esatto per mezzo di superfici lineari.



Metodologia generale: modelli discriminativi //non probabilistici

- Discriminativi non probabilistici:
  - Trovare f : X → {1, . . . ,K} (**funzione discriminante**) che mappa ogni input x in una classe C_i (con i = f(x))
  - Esempi:
    - k-Nearest Neighbor (kNN)
    - SVM (Support Vector Machine)



Nearest Neighbor (NN)

- Uno dei classificatori più semplici
- Input {altezze, M/F}: $\mathcal{D} = \{(h_1, t_1), (h_2, t_2), \cdots, (h_N, t_N)\}$
- Vogliamo classificare h_{new} come M/F
- Soluzione:
  - Troviamo il valore più vicino (NN) (h_{NN}, t_{NN}) a h_{new} e usiamo la sua label

$$t_{new} = t_{NN}$$

k-Nearest Neighbor

- Generalizziamo NN a kNN:
  - Troviamo i k più vicini a h_{new} e ognuno vota con la propria label
  - Si decide a maggioranza (Majority Vote)
  - (Se non c'e' una maggioranza si fa scelta random)

k-Nearest Neighbor

- Vettore D dimensionale $\mathbf{x} \in \mathbb{R}^D$
- Regola di maggioranza sulle K distanze più piccole

$$\{\delta(\mathbf{x}_{new}, \mathbf{x}_n)\}_{n=1\cdots N}$$

- Distanze

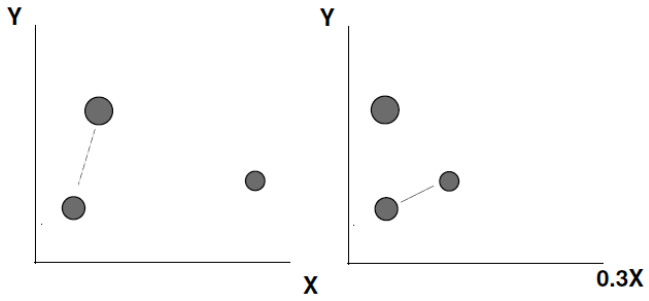
1. **Non-Negativity**: $\delta(\mathbf{x}, \mathbf{y}) \geq 0$
2. **Reflexivity**: $\delta(\mathbf{x}, \mathbf{y}) = 0 \;\; iff \;\; \mathbf{x} = \mathbf{y}$
3. **Symmetry**: $\delta(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{y}, \mathbf{x})$
4. **Triangle Inequality**: $\delta(\mathbf{x}, \mathbf{y}) + \delta(\mathbf{y}, \mathbf{z}) \geq \delta(\mathbf{x}, \mathbf{z})$

- Esempio: $\delta(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{d=1}^{D}(x_d - y_d)^2}$

$$\delta(\mathbf{x}, \mathbf{y}) = \left(\sum_{d=1}^{D}(x_d - y_d)^p\right)^{\frac{1}{p}}$$

Normalizzazione dei dati

- Problema della scalatura degli assi:



$$\widetilde{\mathbf{x}} \leftarrow \mathbf{x} - \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

```
X = X - repmat(mean(X), size(X,1),1);
```

$$\widetilde{\widetilde{\mathbf{x}}} \leftarrow \frac{N\widetilde{\mathbf{x}}}{\sum_{n=1}^{N}\widetilde{\mathbf{x}}_n^2}$$

```
X = X ./ repmat(std(X), size(X,1),1);
```

k-Nearest Neighbor

$N \times D$

```
function [error,tpred] = knn_multi_class(X, t, Xtest, ttest, k)
```

$N \times 1$

```
Ntest = size(Xtest,1);
N = size(X,1);
tpred = zeros(Ntest,1);

C = max(t);

for n=1:Ntest

    Dist = sum(( repmat(Xtest(n,:),N,1) - X ).^2,2 );

    [sorted_list,sorted_index] = sort(Dist);

    [max_k, index_max_k] = max(histc(t(sorted_index(1:k)),1:C));%conteggia le classi con histc
                                                                % e prende il max
    tpred(n) = index_max_k;
end
error = 100*sum(tpred ~= ttest)/Ntest;
```

$$\delta(\mathbf{x}_{test}, \mathbf{x}_n) = \sqrt{\sum_{d=1}^{D} (x_{test,d} - x_{nd})^2} \quad \mathcal{O}(DN).$$

$\mathcal{O}(N \log N$

Support Vector Machine (SVM) //Cos'è un buon Decision Boundary?

- Due classi linearmente separabili
- Soluzione discriminativa:

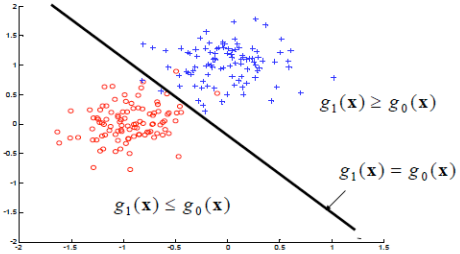
$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^{\mathsf{T}} \phi(\mathbf{x})$$

  - caso binario:

$$g(\mathbf{x}; w_2, w_1, w_0) = w_2 x_2 + w_1 x_1 + w_0 = \mathbf{w}^{\mathsf{T}} \mathbf{x} + w_0$$

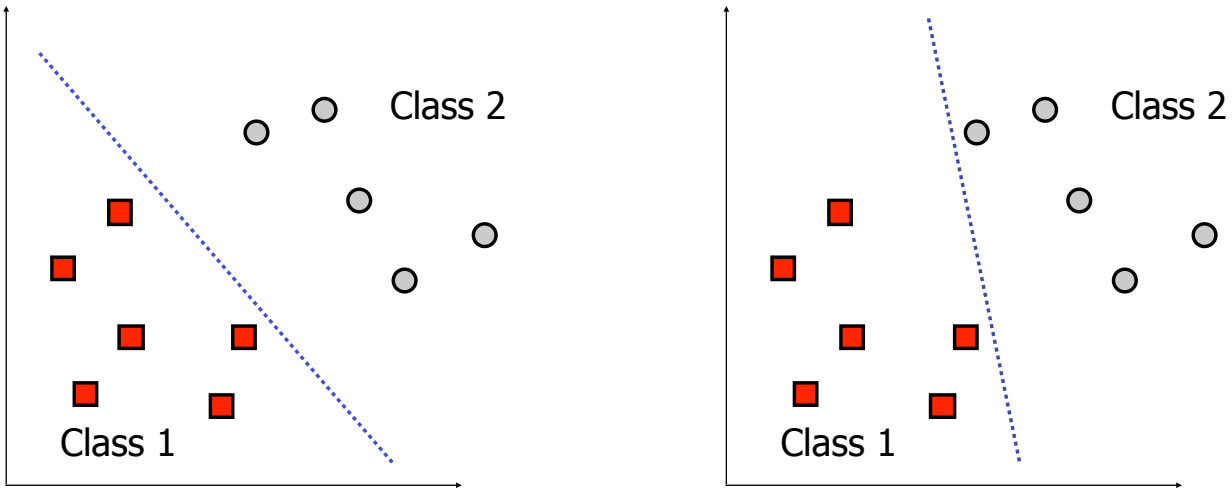
  - target +1 / -1

$$f(\mathbf{x}; w_2, w_1, w_0) = \text{sign}(\mathbf{w}^{\mathsf{T}} \mathbf{x} + w_0)$$



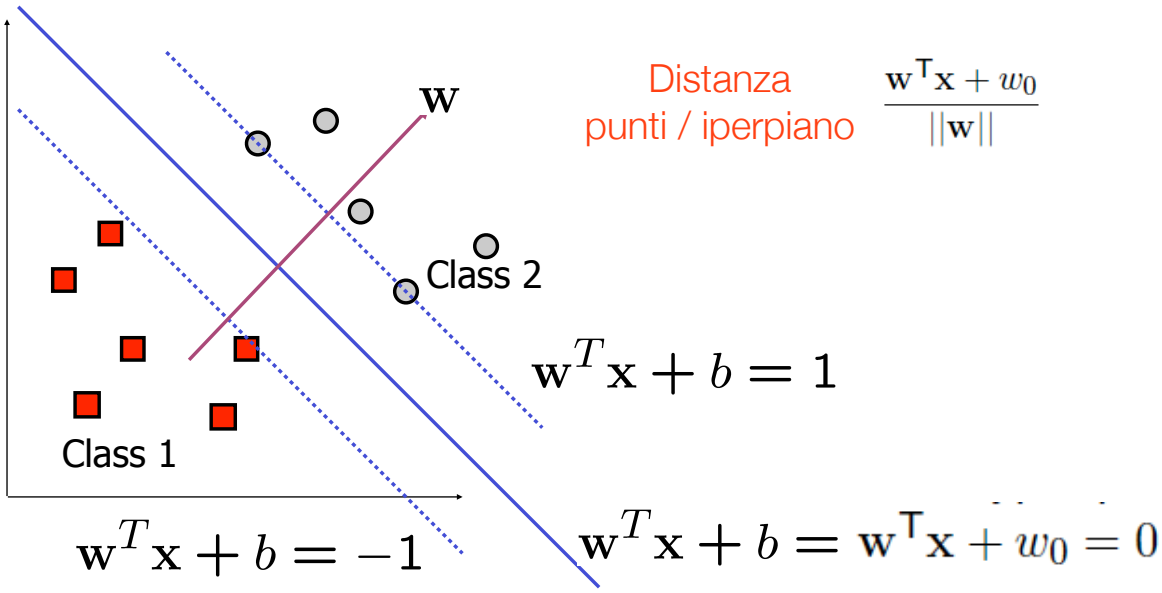
$$(\mathbf{x}_1, t_1) \cdots (\mathbf{x}_N, t_N) \qquad t_n(\mathbf{w}^{\mathsf{T}} \mathbf{x} + w_0) > 0 \;\; \forall \;\; n = 1 \cdots N$$

Support Vector Machine (SVM)
// Cos'è un buon Decision Boundary?



Support Vector Machine (SVM)
// Large-margin Decision Boundary

- Decision boundary: deve essere lontano dai dati di entrambe le classi
  - Massimizzare il margine, m
  - Distanza tra l' origine e la retta $\mathbf{w}^t\mathbf{x}=k$ è $k/\|\mathbf{w}\|$

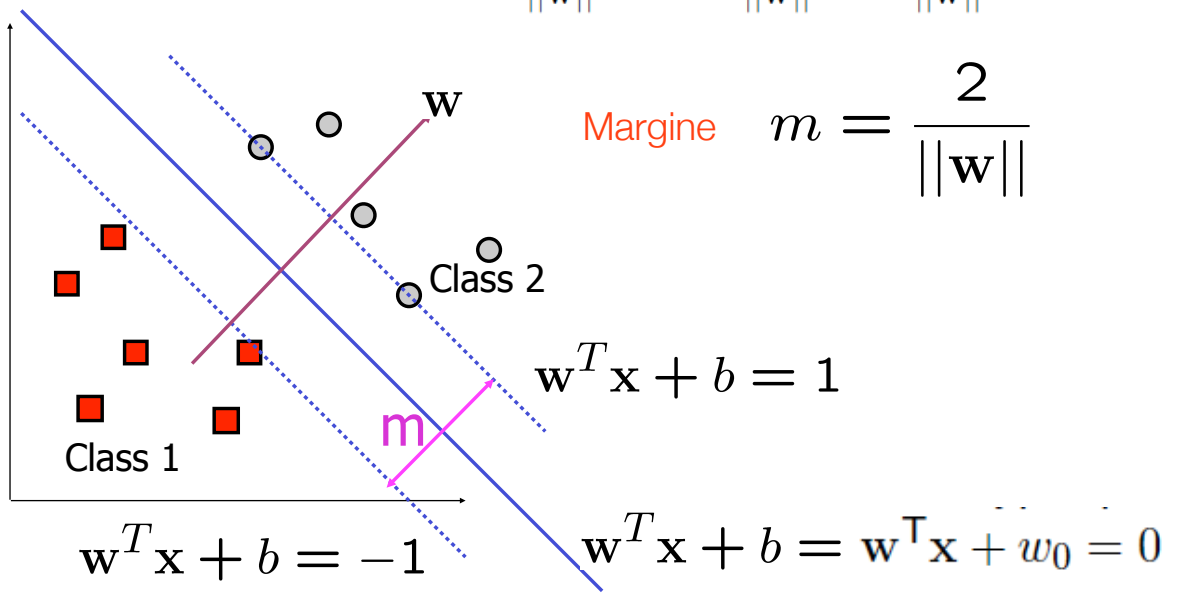


Distanza punti / iperpiano $\frac{\mathbf{w}^T\mathbf{x} + w_0}{\|\mathbf{w}\|}$

Support Vector Machine (SVM)
// Large-margin Decision Boundary

- Decision boundary: deve essere lontano dai dati di entrambe le classi
  - Punti più vicini al margine $\mathbf{w}^T\mathbf{x}_1^* + w_0 = 1 \quad \mathbf{w}^T\mathbf{x}_2^* + w_0 = -1$

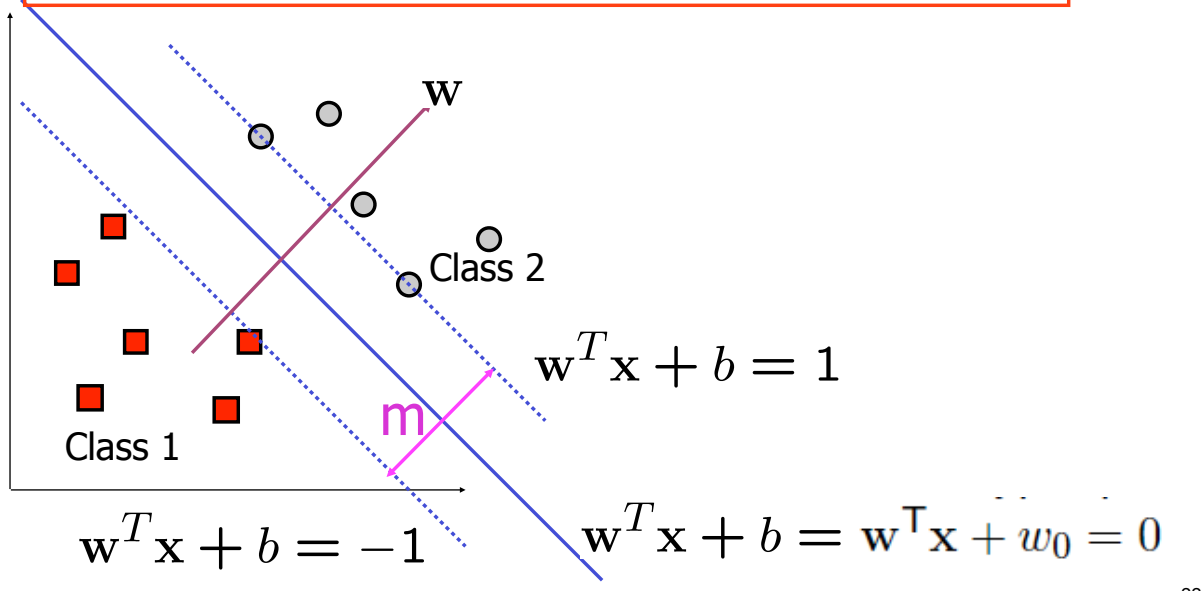
$$\frac{\mathbf{w}^T\mathbf{x}_1^* + w_0}{||\mathbf{w}||} - \frac{\mathbf{w}^T\mathbf{x}_2^* + w_0}{||\mathbf{w}||} = \frac{\mathbf{w}^T}{||\mathbf{w}||}(\mathbf{x}_1^* - \mathbf{x}_2^*)$$



Margine $m = \frac{2}{||\mathbf{w}||}$

Support Vector Machine (SVM)
// Trovare il Decision Boundary

- Decision boundary: deve essere lontano dai dati di entrambe le classi
  - Massimizzare il margine $m = \frac{2}{||\mathbf{w}||}$ → minimizzare $||\mathbf{w}||$



Support Vector Machine (SVM)
// Trovare il Decision Boundary

- $\{x_1, ..., x_n\}$ data set osservato, $y_i \in \{1,-1\}$ la label di classe per x_i
- Il decision boundary dovrebbe classificare perfettamente $\Rightarrow$ $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1,$
- Problema di ottimizzazione vincolata:

Margin minimization

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

Richiami di ottimizzazione vincolata

- Minimizzare f(**x**) sotto il vincolo (s.t.) g(**x**) = 0
- Condizione necessaria affinchè $\mathbf{x}_0$ sia una soluzione:

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}}\Big(f(\mathbf{x}) + \alpha g(\mathbf{x})\Big)\Big|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0} \\ g(\mathbf{x}) = 0 \end{cases}$$

- α: moltiplicatore di Lagrange (Lagrange multiplier)
- Vincoli multipli (multiple constraints) $g_i(\mathbf{x}) = 0$, i=1, …, m, un moltiplicatore α_i per ciascun vincolo

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}}\Big(f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i g_i(\mathbf{x})\Big)\Big|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0} \\ g_i(\mathbf{x}) = 0 \qquad \text{for } i = 1, \ldots, m \end{cases}$$

Richiami di ottimizzazione vincolata

- Per vincoli del tipo $g_i(\mathbf{x}) \leq 0$ il Lagrange multiplier α_i deve essere positivo
- Se $\mathbf{x}_0$ è soluzione per il problema

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \ldots, m$$

- esistono $\alpha_i \geq 0$ per i=1, …, m tali che $\mathbf{x}_0$ soddisfa

Lagrangiana

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}} \left(f(\mathbf{x}) + \sum_i \alpha_i g_i(\mathbf{x}) \right) \Big|_{\mathbf{x}=jx_0} = \mathbf{0} \\ g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \ldots, m \end{cases}$$

- La funzione $f(\mathbf{x}) + \sum_i \alpha_i g_i(\mathbf{x})$ è la Lagrangiana; il suo gradiente deve essere messo a **0**

Problema primale e duale

Lagrangiana primale

Lagrangiana duale

$$L_D(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\mathbf{x}} L_P(\mathbf{x}, \lambda, \mu)$$

Lagrangiana duale

$$\begin{aligned} \underset{\boldsymbol{\lambda}, \boldsymbol{\mu}}{\text{maximize}} \quad & L_D(\boldsymbol{\lambda}, \boldsymbol{\mu}) \\ \text{subject to} \quad & \boldsymbol{\mu} \geq \mathbf{0} \end{aligned}$$

Richiami di ottimizzazione vincolata

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to } 1-y_i(\mathbf{w}^T\mathbf{x}_i+b) \leq 0 \qquad \text{for } i = 1,\ldots,n$$

- Minimizzo la funzione Lagrangiana (problema primale):

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

  - con $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

- Calcolo gradiente di $\mathcal{L}$ rispetto a **w** e b e pongo a zero

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L} \Rightarrow \mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial}{\partial w_0}\mathcal{L} \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

Il problema duale

- Sostituiamo $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ in $\mathcal{L}$

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b)\right)$$
$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i y_i \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T\mathbf{x}_i - b\sum_{i=1}^{n} \alpha_i y_i$$
$$= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{n} \alpha_i$$

$W(\boldsymbol{\alpha})$ $\qquad$ $\sum_{i=1}^{n} \alpha_i y_i = 0$

- E' funzione solo di α_i

Il problema duale

- Problema duale: se conosco **w**, conosco α_i; se conosco α_i, conosco **w**
- Massimizzo la nuova funzione obbiettivo (problema duale)

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Il problema duale

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- E' un problema di programmazione quadratica (quadratic programming, QP)
  - Esiste sempre un massimo globale rispetto a α_i
- **w** si calcola poi come: $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$

Il problema QP

- Molti metodi di QP
  - Loqo, cplex, etc. (http://www.numerical.rl.ac.uk/qp/qp.html)
- Per SVM, sequential minimal optimization (SMO) è il più popolare
  - QP: triviale da risolvere con due variabili
  - Iterativamente: si seleziona una coppia (α_i,α_j) e si risolve QP; si ripete fino a convergenza
- Per i nostri fini il QP solver è una “black-box” che utilizziamo e basta!

Support Vector Machine (SVM) // Soluzione

- Predizione rispetto a un dato nuovo $\mathbf{z} = \mathbf{x}_{new}$
  - Calcoliamo $\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$
  - Classifichiamo **z** nella classe 1 se la somma è positiva, altrimenti classe 2
- Nota: **w** non deve essere calcolato esplicitamente

$$
\begin{aligned}
f(\mathbf{x}_{new}; \mathbf{w}, w_0) &= \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}_{new} + w_0) \\
&= \text{sign}\left(\sum_{n=1}^{N} t_n \alpha_n \mathbf{x}_n^\mathsf{T}\mathbf{x}_{new} + w_0\right) \\
&= \text{sign}\left(\sum_{n \in SV} t_n \alpha_n \mathbf{x}_n^\mathsf{T}\mathbf{x}_{new} + w_0\right)
\end{aligned}
$$

Support Vector Machine (SVM)
// Algoritmo SVM di base

- Create $\mathbf{H}$, where $H_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$.
- Find α so that

$$\sum_{i=1}^{L} \alpha_i - \frac{1}{2}\alpha^T \mathbf{H} \alpha$$

is maximized, subject to the constraints

$$\alpha_i \geq 0 \quad \forall_i \text{ and } \sum_{i=1}^{L} \alpha_i y_i = 0.$$

This is done using a QP solver.

- Calculate $\mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i$.
- Determine the set of Support Vectors S by finding the indices such that $\alpha_i > 0$.
- Calculate $b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s)$.
- Each new point $\mathbf{x}'$ is classified by evaluating $y' = sgn(\mathbf{w} \cdot \mathbf{x}' + b)$.

Support Vector Machine (SVM)
// Forma matriciale

$$\begin{aligned} f(\mathbf{x}_{new}; \mathbf{w}, w_0) &= \text{sign}(\mathbf{w}^\mathsf{T} \mathbf{x}_{new} + w_0) \\ &= \text{sign}\left(\sum_{n=1}^{N} t_n \alpha_n \mathbf{x}_n^\mathsf{T} \mathbf{x}_{new} + w_0 \right) \\ &= \text{sign}\left(\sum_{n \in SV} t_n \alpha_n \mathbf{x}_n^\mathsf{T} \mathbf{x}_{new} + w_0 \right) \\ &= \text{sign}\left(\sum_{n \in SV} t_n \alpha_n K(\mathbf{x}_n, \mathbf{x}_{new}) + w_0 \right) \end{aligned}$$

$$\mathbf{K} = K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{X}\mathbf{X}^\mathsf{T}$$ kernel

$$\mathbf{\Lambda} = \text{diag}(\mathbf{t})$$

$$\max \boldsymbol{\alpha}^\mathsf{T} \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^\mathsf{T} \mathbf{\Lambda} \mathbf{K} \mathbf{\Lambda} \boldsymbol{\alpha}$$

$$\begin{aligned} \alpha_n &\geq 0 \quad \forall \; n = 1 \cdots N \\ \boldsymbol{\alpha}^\mathsf{T} \mathbf{t} &= 0 \end{aligned}$$

Support Vector Machine (SVM)
// Un esempio semplificato

```
clear
Step=0.5;
N = 50;
C=1000;

X = [randn(N/2,2);randn(N/2,2)+[ones(N/2,1).*6 zeros(N/2,1)]];
t=[ones(N/2,1);-ones(N/2,1)];

[alpha,w_0,alpha_index]= mioqp(diag(t)*X*X'*diag(t),ones(N,1),t,C,1e-6);

%Griglia dei contorni
mn = min(X);
mx = max(X);
[x1,x2]=meshgrid(floor(mn(1)):Step:ceil(mx(1)),floor(mn(2)):Step:ceil(mx(2)));
[n11,n12]=size(x1);
[n21,n22]=size(x2);
XG=[reshape(x1,n11*n12,1) reshape(x2,n21*n22,1)];

f = (t(alpha_index).*alpha)'*X(alpha_index,:)*XG' + w_0;

plot(X(alpha_index,1),X(alpha_index,2),'go');
hold
plot(X(1:N/2,1),X(1:N/2,2),'.')
plot(X(N/2+1:N,1),X(N/2+1:N,2),'r.')
contour(x1,x2,reshape(f,[n11,n12]),[0 0]);
hold off
```

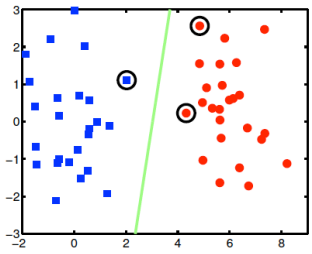
box constraint

$0 \leq \alpha_n \leq C$

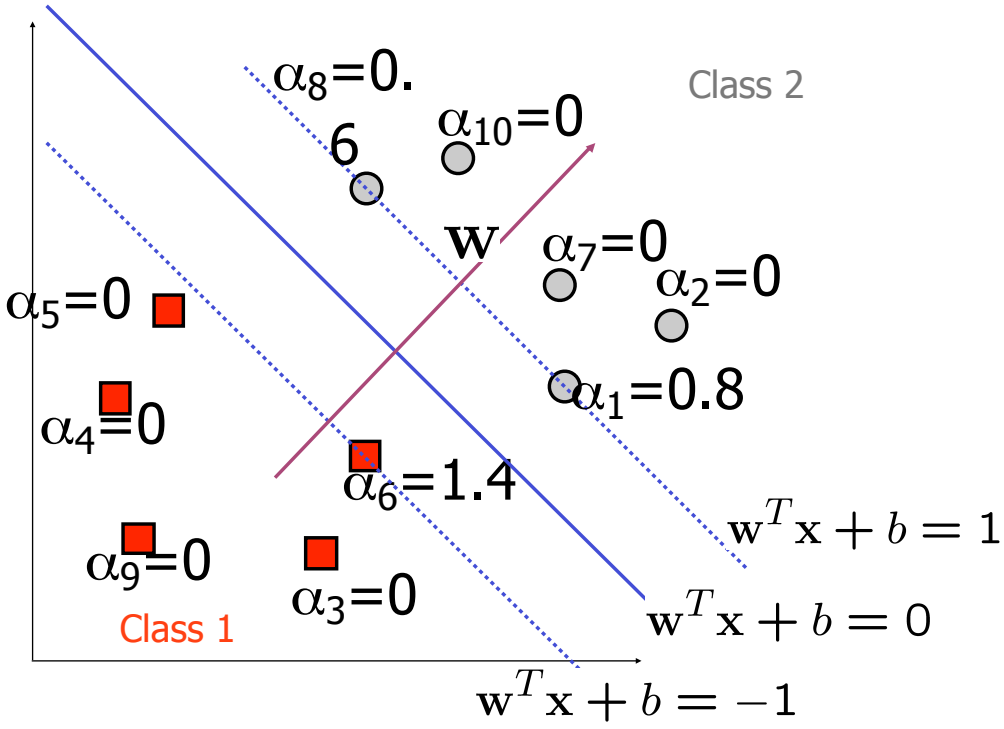
ΛΚΛ 1 t

$\mathbf{x}_{new}$

$t_n \alpha_n \mathbf{x}_n^{\mathrm{T}} \mathbf{x}_{new} + w_0$



Support Vector Machine (SVM)
// Caratteristiche della soluzione



Support Vector Machine (SVM)
// Caratteristiche della soluzione

- Molti α_i sono nulli
  - **w** è combinazione lineare di pochi punti
  - Rappresentazione sparsa: data compression

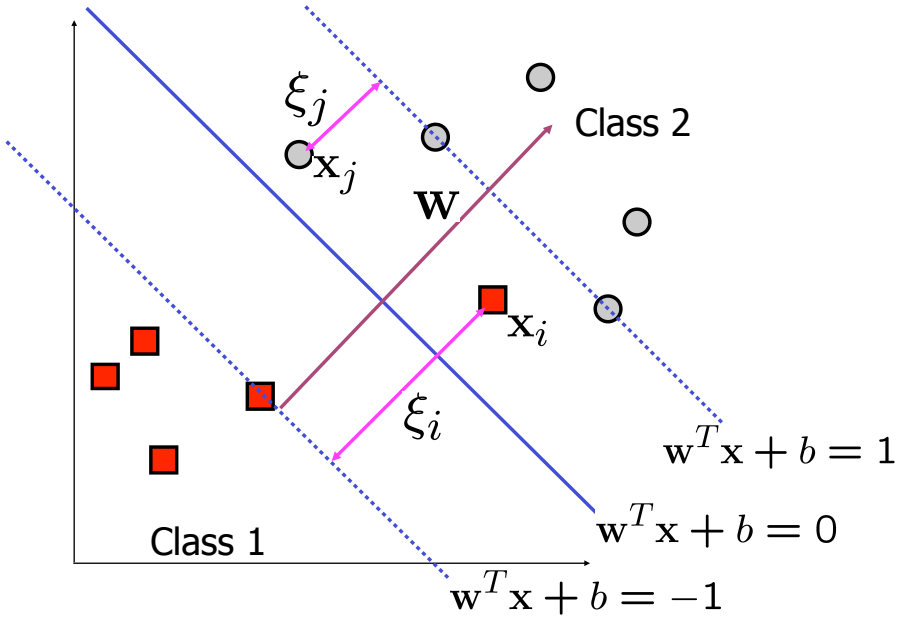
Sparse kernel machines

- $\mathbf{x}_i$ corrispondenti ad α_i diversi da zero: support vectors (SV)
  - Il decision boundary è determinato unicamente dai SV
  - Siano t_j (j=1, ..., s) gli indici dei SV:

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

Da dove esce il parametro C
//Non-linearly Separable Problems

- Consentiamo un "errore" ξ_i nella classificazione;
- ξ_i approssima il numero di sample misclassificati



Support Vector Machine (SVM)
// Iper-piano a margine “soft”

- Minimizziamo $\sum_i \xi_i$, ξ_i si calcolano come

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

  - ξ_i sono “slack variables”
  - Se ξ_i=0 non c’e’ errore per $\mathbf{x}_i$
  - ξ_i upper bound al numero di errori

- Minimizziamo

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

  - C : tradeoff parameter tra errore e margine

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Support Vector Machine (SVM)
// Iper-piano a margine “soft”

- Duale:

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$
$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n}\alpha_i y_i = 0$$

- **w** si ottiene come: $\mathbf{w} = \sum_{j=1}^{s}\alpha_{t_j} y_{t_j}\mathbf{x}_{t_j}$

Support Vector Machine (SVM)
// Problemi di ottimizzazione soft vs. hard

Soft margin

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Hard margin

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Support Vector Machine (SVM)
// Algoritmo SVM soft (2)

- Create $\mathbf{H}$, where $H_{ij} = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$.
- Choose how significantly misclassifications should be treated, by selecting a suitable value for the parameter C.
- Find $\boldsymbol{\alpha}$ so that

$$\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha}$$

  is maximized, subject to the constraints

$$0 \leq \alpha_i \leq C \ \ \forall_i \text{ and } \sum_{i=1}^{L} \alpha_i y_i = 0.$$

  This is done using a QP solver.

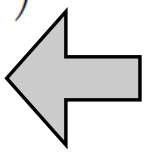
- Calculate $\mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}_i$.
- Determine the set of Support Vectors S by finding the indices such that $0 < \alpha_i \leq C$.
- Calculate $b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s)$.
- Each new point $\mathbf{x}'$ is classified by evaluating $y' = sgn(\mathbf{w} \cdot \mathbf{x}' + b)$.

Support Vector Machine (SVM)
// Estensione a Decision Boundary non lineare

- Abbiamo considerato un classificatore con decision boundary lineare
- Come generalizzare al caso non lineare?

$$
\begin{aligned}
f(\mathbf{x}_{new}; \mathbf{w}, w_0) &= \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}_{new} + w_0) \\
&= \text{sign}\left(\sum_{n=1}^{N} t_n \alpha_n \mathbf{x}_n^\mathsf{T}\mathbf{x}_{new} + w_0\right) \\
&= \text{sign}\left(\sum_{n \in SV} t_n \alpha_n \mathbf{x}_n^\mathsf{T}\mathbf{x}_{new} + w_0\right) \\
&= \text{sign}\left(\sum_{n \in SV} t_n \alpha_n K(\mathbf{x}_n, \mathbf{x}_{new}) + w_0\right)
\end{aligned}
$$

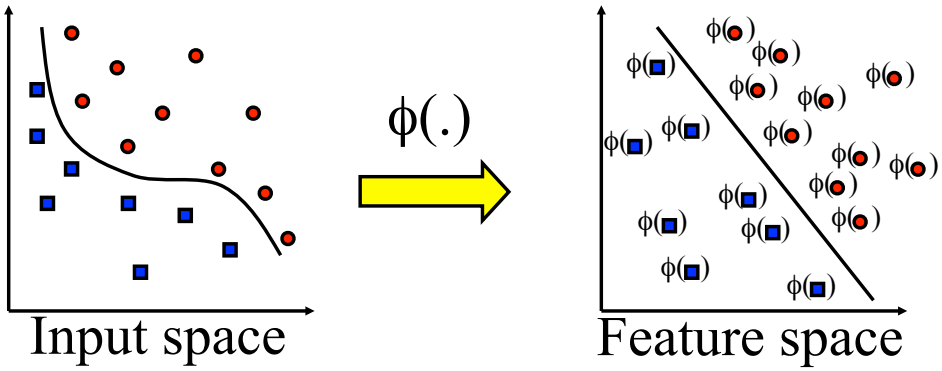
$$\mathbf{K} = K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{X}\mathbf{X}^\mathsf{T}$$ kernel



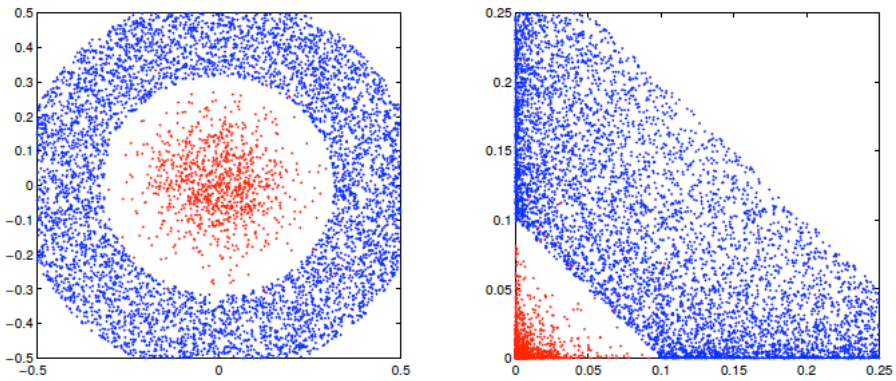
Support Vector Machine (SVM)
// Estensione a Decision Boundary non lineare

- Abbiamo considerato un classificatore con decision boundary lineare
- Come generalizzare al caso non lineare?
- Idea: trasformare $\mathbf{x}_i$ in uno spazio a dimensionalità maggiore
  - Input space: spazio dei dati $\mathbf{x}_i$
  - Feature space: lo spazio degli $\phi(\mathbf{x}_i)$ dopo la trasformazione
- Perchè?
  - Operazione lineare nel feature space = operazione non lineare nell' input space

Support Vector Machine (SVM)
// Trasformare i dati

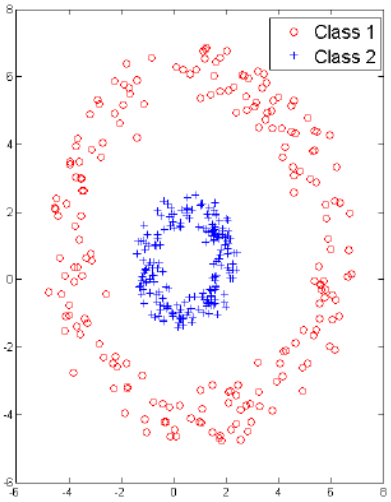


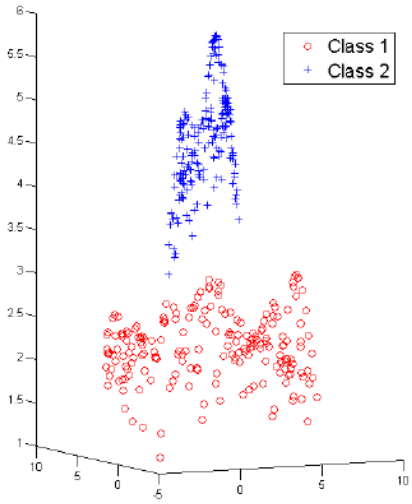
Support Vector Machine (SVM)
// Trasformare i dati



con un polinomio di ordine 2

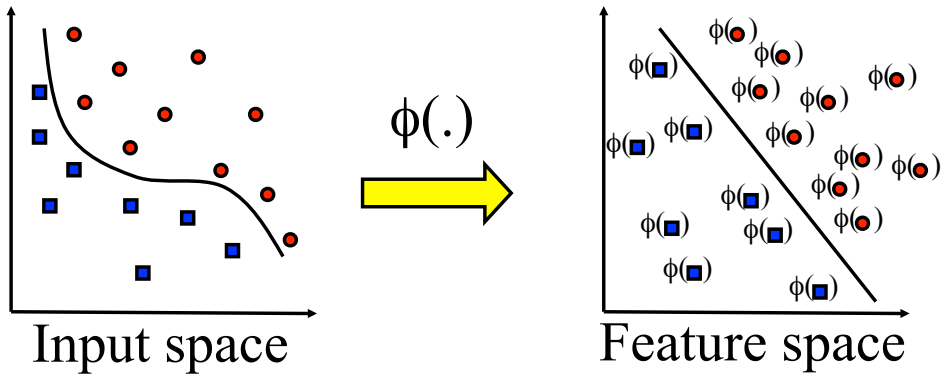
Support Vector Machine (SVM)
// Trasformare i dati





con una Radial Basis Function (RBF)

Support Vector Machine (SVM)
// Trasformare i dati



- Problema: la computazione nel feature space puo' essere costosa per la dimensionalità elevata dello spazio (anche infinita!!)
- Soluzione: kernel trick

Support Vector Machine (SVM)
// II Kernel Trick

- Ricordiamo il problema iniziale

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- I punti compaiono solo come distanza o prodotto scalare (inner product)
- Se dobbiamo calcolare una distanza nel feature space, non abbiamo bisogno del mapping esplicito
- Si definisce la funzione di kernel K come:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Support Vector Machine (SVM)
// II Kernel Trick

$$\mathbf{x} \rightarrow \phi(\mathbf{x})$$

$$\delta(\mathbf{x},\mathbf{y})^2 = (\mathbf{x}-\mathbf{y})^\mathsf{T}(\mathbf{x}-\mathbf{y}) \longrightarrow \delta(\phi(\mathbf{x}),\phi(\mathbf{y}))^2 = (\phi(\mathbf{x})-\phi(\mathbf{y}))^\mathsf{T}(\phi(\mathbf{x})-\phi(\mathbf{y}))$$
$$= \phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{x}) + \phi(\mathbf{y})^\mathsf{T}\phi(\mathbf{y}) - 2\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{y})$$

$$K(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{y})$$

$$\delta(\phi(\mathbf{x}),\phi(\mathbf{y}))^2 = (\phi(\mathbf{x})-\phi(\mathbf{y}))^\mathsf{T}(\phi(\mathbf{x})-\phi(\mathbf{y}))$$
$$= K(\mathbf{x},\mathbf{x}) + K(\mathbf{y},\mathbf{y}) - 2K(\mathbf{x},\mathbf{y})$$

Esempio

$$\phi : \mathbf{x} = (x_1 x_2)^\mathsf{T} \rightarrow \phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^\mathsf{T} \in \mathcal{F} = \mathbb{R}^3$$

$$\begin{aligned}\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{y}) &= (x_1^2, x_2^2, \sqrt{2}x_1x_2)(y_1^2, y_2^2, \sqrt{2}y_1y_2)^\mathsf{T} \\ &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\ &= (x_1y_1 + x_2y_2)^2 \\ &= (\mathbf{x}^\mathsf{T}\mathbf{y})^2\end{aligned}$$

K è calcolato nello spazio di input

Support Vector Machine (SVM)
// Esempi di Kernel Functions

- Polynomial kernel di grado d

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Radial Basis Function (RBF) kernel di ampiezza σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

  - Collegato ai RBF neural networks
  - Feature space infinito

- Sigmoide di parametri κ e θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

Support Vector Machine (SVM)
// Come uso la Kernel Function

- Sostituisco tutti i prodotti interni con kernel functions
- Learning:

Originale

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Con kernel function

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Support Vector Machine (SVM)
// Come uso la Kernel Function

- Predizione: **z** classificato di classe 1 se $f \geq 0$, di classe 2 se $f < 0$

Originale

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

Con kernel function

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

Support Vector Machine (SVM)
// ancora su Kernel Functions

- Poichè il training di SVM richiede solo il valore K($\mathbf{x}_i$, $\mathbf{x}_j$), non vi è restrizione sulla forma di $\mathbf{x}_i$ e $\mathbf{x}_j$
  - $\mathbf{x}_i$ può essere una sequenza o un albero piuttosto che un feature vector
- K($\mathbf{x}_i$, $\mathbf{x}_j$): è solo una misura di similarità fra $\mathbf{x}_i$ e $\mathbf{x}_j$
- Per un nuovo punto **z**, la discriminant function è una somma pesata (weighted sum) della similarità tra z e un insieme di valori pre-selezionati (i support vectors)

$$f(\mathbf{z}) = \sum_{\mathbf{x}_i \in \mathcal{S}} \alpha_i y_i K(\mathbf{z}, \mathbf{x}_i) + b$$

$\mathcal{S}$: the set of support vectors

Support Vector Machine (SVM)
// ancora su Kernel Functions

- Non tutte le misure di similarità sono lecite (matematicamente)
  - La kernel function deve soddisfare il criterio di Mercer, i.e., deve essere “definita positiva”
    - così la (n x n) kernel matrix, con elemento (i,j) uguale a $K(\mathbf{x}_i, \mathbf{x}_j)$, è sempre definita positiva
    - QP è convesso e computabile in tempo polinomiale

Support Vector Machine (SVM)
// esempio

- Abbiamo 5 punti 1D
  - x_1=1, x_2=2, x_3=4, x_4=5, x_5=6, con 1, 2, 6 di classe 1 e 4, 5 di classe 2 ⇒
  - y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1
- Usiamo polynomial kernel di grado 2
  - $K(x,y) = (xy+1)^2$
  - C = 100
- Troviamo α_i (i=1, …, 5):

$$\max. \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Support Vector Machine (SVM)
// esempio

- Con un QP solver, si ottiene
  - α_1=0, α_2=2.5, α_3=0, α_4=7.333, α_5=4.833
  - I constraints sono soddisfatti...
  - Support Vectors: {x_2=2, x_4=5, x_5=6}
- Funzione discriminante

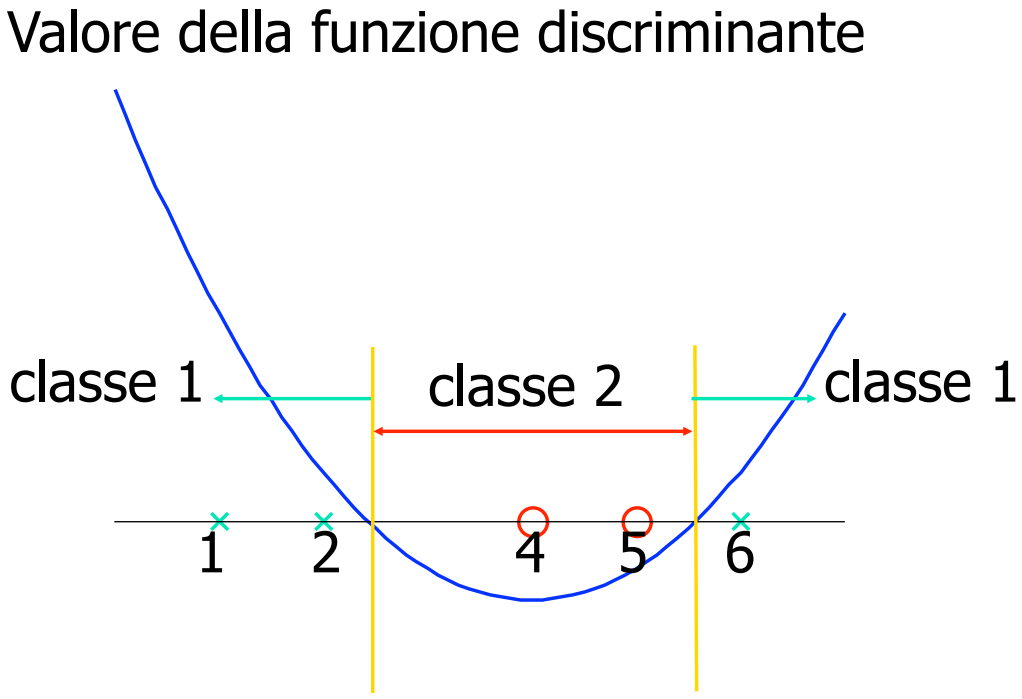
$$f(z) = 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + 4.833(1)(6z+1)^2 + b$$
$$= 0.6667z^2 - 5.333z + b$$

(α_5, y_5, $K(z, x_5)$)

- b si ottiene risolvendo f(2)=1 o f(5)=-1 o f(6)=1, poichè x_2 and x_5 sono sulla retta $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = 1$ e x_4 giace su $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = -1$
- Tutte danno b=9

$$\Longrightarrow f(z) = 0.6667z^2 - 5.333z + 9$$

Support Vector Machine (SVM)
// esempio



Support Vector Machine (SVM) // Algoritmo SVM (3)

The first step, therefore, consists of choosing our kernel and hence the mapping $\mathbf{x} \longmapsto \phi(\mathbf{x})$.

- Create $\mathbf{H}$, where $H_{ij} = y_i y_j \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x}_j)$.
- Choose how significantly misclassifications should be treated, by selecting a suitable value for the parameter C.
- Find $\boldsymbol{\alpha}$ so that

$$\sum_{i=1}^{L} \alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha}$$

  is maximized, subject to the constraints

$$0 \leq \alpha_i \leq C \quad \forall_i \text{ and } \sum_{i=1}^{L} \alpha_i y_i = 0.$$

  This is done using a QP solver.

- Calculate $\mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \phi(\mathbf{x}_i)$.
- Determine the set of Support Vectors S by finding the indices such that $0 < \alpha_i \leq C$.
- Calculate $b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \phi(\mathbf{x}_m) \cdot \phi(\mathbf{x_s}))$.
- Each new point $\mathbf{x}'$ is classified by evaluating $y' = sgn(\mathbf{w} \cdot \phi(\mathbf{x}') + b)$.

Support Vector Machine (SVM) // scegliere la Kernel Function

- Puo' essere complicato.
- La kernel matrix è una rappresentazione di tutti i dati
- Molte soluzioni proposte (diffusion kernel, Fisher kernel, string kernel, …)
- Problematiche di stima del Kernel dai dati
- In pratica: si può cominciare con un polynomial kernel o un RBF kernel di ampiezza ragionevole

Support Vector Machine (SVM) // altri aspetti

- Classificazione multi-class?
  - Si può cambiare la formulazione di QP
  - Oppure combinare multiple binary classifiers
    - one-versus-all classifiers
    - multiple pairwise classifiers “intelligently”
- Come interpretare il valore della SVM discriminant function in termini di probabilità?
  - Effettuare logistic regression sull’output di SVM su validation set
- Usare software SVM (e.g. libsvm) che è multi-class

Support Vector Machine (SVM) // Software

- Implementazioni http://www.kernel-machines.org/software.html
- LIBSVM
- SVMLight: una delle prime
- Matlab toolboxes
- PMTK3 di Murphy

Support Vector Machine (SVM)
// Istruzioni per l'uso

- Preparare la pattern matrix
- Selezionare la kernel function
- Selezionare i parametri della kernel function e il valore di C
  - validation set
- Training per ottenere gli α_i
- Classificare usando α_i e support vectors ricavati dal training

Tuning dei parametri

$$K(\mathrm{x},\mathrm{y}) = \exp(-||\mathrm{x}-\mathrm{y}||^2/(2\sigma^2))$$

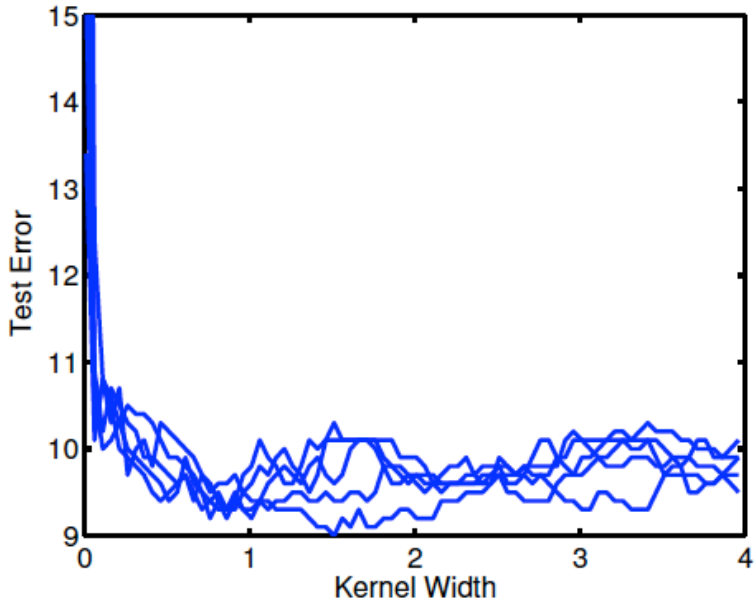


Figure 8: The percentage error achieved by an SVM using a Radial Basis Kernel function with a width parameter ranging from 0.01 to 4.0 in step sizes of 0.05. For each of these ranges a value of C was selected from 1 to 4 and we can see that the minimum test error of 9.0% was achieved with hyper-parameter values of $C = 1$ and $\beta = 1.4$.

Risorse in rete

- http://www.kernel-machines.org/
- http://www.support-vector.net/
- http://www.support-vector.net/icml-tutorial.pdf
- http://www.kernel-machines.org/papers/tutorial-nips.ps.gz
- http://www.clopinet.com/isabelle/Projects/SVM/applist.html