

# Classificazione

## ■ Classificatore di Bayes

- Approccio parametrico (distribuzione Multinormale)
- Approccio non parametrico (Parzen Window)

## ■ Nearest Neighbor

- $k$ -NN
- Metriche

## ■ SVM

- Lineari: pattern linearmente separabili e non
- Non lineari
- Caso multclasse

## ■ Multi classificatori

- Fusione a livello di decisione
- Fusione a livello di confidenza
- Random Forest (Bagging)
- AdaBoost (Boosting), Gradient Boosting

# Approccio Bayesiano

Il problema è posto in termini probabilistici. Se tutte le distribuzioni in gioco sono note l'approccio Bayesiano costituisce la migliore regola di classificazione possibile: **soluzione OTTIMA !** Non è possibile fare previsioni migliori.

I pattern sono singoli dati, come una email

num. di features per descrivere il pattern

- Sia  $\mathbf{V}$  uno spazio di pattern  $d$ -dimensionali e  $W = \{w_1, w_2 \dots w_s\}$  un insieme di  $s$  classi disgiunte costituite da elementi di  $\mathbf{V}$
- Per ogni  $\mathbf{x} \in \mathbf{V}$  e per ogni  $w_i \in W$ , indichiamo con  $p(\mathbf{x}|w_i)$  la **densità di probabilità condizionale** (o condizionata) di  $\mathbf{x}$  data  $w_i$ , ovvero la densità di probabilità che il prossimo pattern sia  $\mathbf{x}$  sotto l'ipotesi che la sua classe di appartenenza sia  $w_i$
- Per ogni  $w_i \in W$ , indichiamo con  $P(w_i)$  la **probabilità a priori** di  $w_i$  ovvero la probabilità, indipendentemente dall'osservazione, che il prossimo pattern da classificare sia di classe  $w_i$
- Per ogni  $\mathbf{x} \in \mathbf{V}$  indichiamo con  $p(\mathbf{x})$  la **densità di probabilità assoluta** di  $\mathbf{x}$ , ovvero la densità di probabilità che il prossimo pattern da classificare sia  $\mathbf{x}$

$$p(\mathbf{x}) = \sum_{i=1}^s p(\mathbf{x}|w_i) \cdot P(w_i) \quad \text{dove} \quad \sum_{i=1}^s P(w_i) = 1$$

- Per ogni  $w_i \in W$  e per ogni  $\mathbf{x} \in \mathbf{V}$  indichiamo con  $P(w_i|\mathbf{x})$  la **probabilità a posteriori** di  $w_i$  dato  $\mathbf{x}$ , ovvero la probabilità che avendo osservato il pattern  $\mathbf{x}$ , la classe di appartenenza sia  $w_i$ . Per il **teorema di Bayes**:

$$P(w_i|\mathbf{x}) = \frac{p(\mathbf{x}|w_i) \cdot P(w_i)}{p(\mathbf{x})}$$

assegna un nuovo pattern  $x$  alla classe  $w_i$  che ha la massima probabilità a posteriori  $P(w_i | x)$ .  
 Regola di Bayes: Classifica  $x$  come  $w_i$  se  $P(w_i | x) > P(w_j | x)$  per tutti i  $j \neq i$ .

# Classificatore di Bayes

Dato un pattern  $x$  da classificare in una delle  $s$  classi  $w_1, w_2 \dots w_s$  di cui sono note:

- le probabilità a priori  $P(w_1), P(w_2) \dots P(w_s)$
- le densità di probabilità condizionali  $p(x|w_1), p(x|w_2) \dots p(x|w_s)$

la regola di classificazione di Bayes assegna  $x$  alla classe  $b$  per cui è massima la probabilità a posteriori:

$$b = \operatorname{argmax}_{i=1..s} \{ P(w_i | x) \}$$

È intuitivo assegnare il pattern alla classe in cui è più probabile che ricada.

Massimizzare la probabilità a posteriori significa massimizzare la densità di probabilità condizionale tenendo comunque conto della probabilità a priori delle classi.

Esempio: Se un pattern  $x$  ha una densità condizionata  $p(x | w_1)$  alta, ma la classe  $w_1$  è estremamente rara (cioè  $P(w_1)$  molto basso), il prodotto potrebbe non essere il massimo. Il classificatore di Bayes tiene conto di questa rarità.

- La regola si dimostra ottima in quanto minimizza l'errore di classificazione. Ad esempio nel caso di 2 classi e  $d = 1$ :

somma delle probabilità di errore su tutte le regioni di decisione.

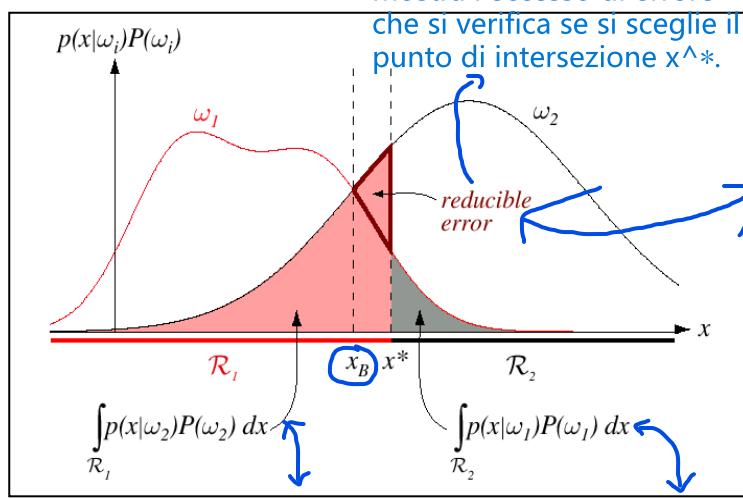
$$P(\text{error}) = \int_{\mathcal{R}_1} p(x|w_2)P(w_2)dx + \int_{\mathcal{R}_2} p(x|w_1)P(w_1)dx$$

mostra l'eccesso di errore che si verifica se si sceglie il punto di intersezione  $x^*$ .

Il Classificatore di Bayes è l'ottima regola perché sceglie le regioni di decisione  $\mathcal{R}_1$  e  $\mathcal{R}_2$  in modo tale da rendere questa probabilità di errore la più piccola possibile.

Lo spazio dei pattern  $V$  viene diviso in  $s$  regioni disgiunte  $R_1, R_2, \dots, R_s$ . Il classificatore assegna  $x$  alla classe  $w_i$  se  $x$  cade nella regione  $R_i$ .

- In ogni regione  $R_i$ , l'algoritmo decide  $w_i$ .
- L'errore in  $R_i$  si verifica solo se un pattern  $x \in R_i$  appartiene in realtà a una classe diversa da  $w_i$  (cioè  $w_j \neq w_i$ ). La probabilità che ciò accada è  $P(w_j | x)$ .

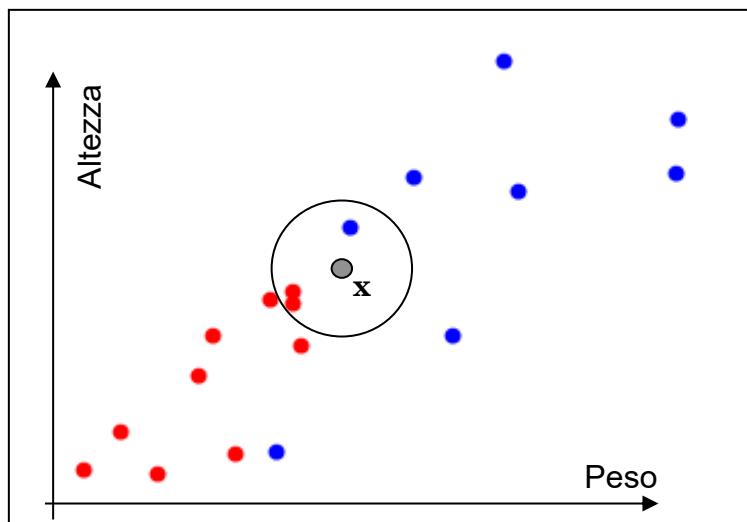


L'errore qui è dato dall'area sotto la curva nera ( $w_2$ )

L'errore qui è dato dall'area sotto la curva rossa ( $w_1$ )

ML

# Esempio



Classificare le persone in maschi/femmine in base a peso e all'altezza, a partire dal training set in figura.

- $V$  è uno spazio a 2 dimensioni ( $d = 2$ )
- $W = \{w_1, w_2\}$   
 $w_1$  = maschi (blu),  
 $w_2$  = femmine (rosso)

Una stima (grossolana) delle probabilità a priori e delle densità può essere effettuata a partire dal training set come segue (si vedranno in seguito tecniche più rigorose per effettuare tale stima):

- **Probabilità a priori:** si considera semplicemente l'occorrenza dei pattern nel training set:  $P(w_1) = 8/18$ ,  $P(w_2) = 10/18$  num. di quelle classi rispetto al totale
- **Densità di probabilità condizionale** per un nuovo pattern  $x$  da classificare: si contano le occorrenze dei pattern del training set delle due classi in un intorno di  $x$ :

$$\begin{aligned} p(x|w_1) &= 1/8 \\ p(x|w_2) &= 2/10 = 1/5 \\ p(x) &= \frac{1}{8} \times \frac{8}{18} + \frac{1}{5} \times \frac{10}{18} = \frac{1}{18} + \frac{2}{18} = \frac{1}{6} \end{aligned}$$

num. di quelle classi nell'intorno rispetto al totale di quelle classi

Si ottiene quindi:

$$\left. \begin{aligned} P(w_1|x) &= \frac{p(x|w_1) \cdot P(w_1)}{p(x)} = \frac{1/18}{1/6} = \frac{1}{3} \\ P(w_2|x) &= \frac{p(x|w_2) \cdot P(w_2)}{p(x)} = \frac{2/18}{1/6} = \frac{2}{3} \end{aligned} \right\}$$

Perché, in totale, ho più probabilità di avere una femmina ed inoltre, nell'intorno di  $x$ , ho più femmine

L'approccio Bayesiano assegna il pattern  $x$  alla classe  $w_2$  (femmine).

# Bayes: approccio parametrico e non-parametrico

- Mentre la stima delle probabilità a priori è abbastanza semplice (se non si hanno elementi si possono ipotizzare le classi equiprobabili), la conoscenza delle densità condizionali è possibile “solo in teoria”; nella pratica due soluzioni:
  - **Approccio parametrico:** si fanno ipotesi sulla forma delle distribuzioni (es. [distribuzione multinormale](#)) e si apprendono i parametri fondamentali ([vettore medio](#), matrice di covarianza) dal training set.
  - **Approccio non parametrico:** si apprendono le distribuzioni dal training set (es. attraverso il metodo [Parzen Window](#)).

Generalmente l'approccio parametrico si utilizza quando, oltre ad avere una ragionevole certezza (o speranza) che la forma della distruzione sia adeguata, la dimensione del training set non è sufficiente per una buona stima della densità.

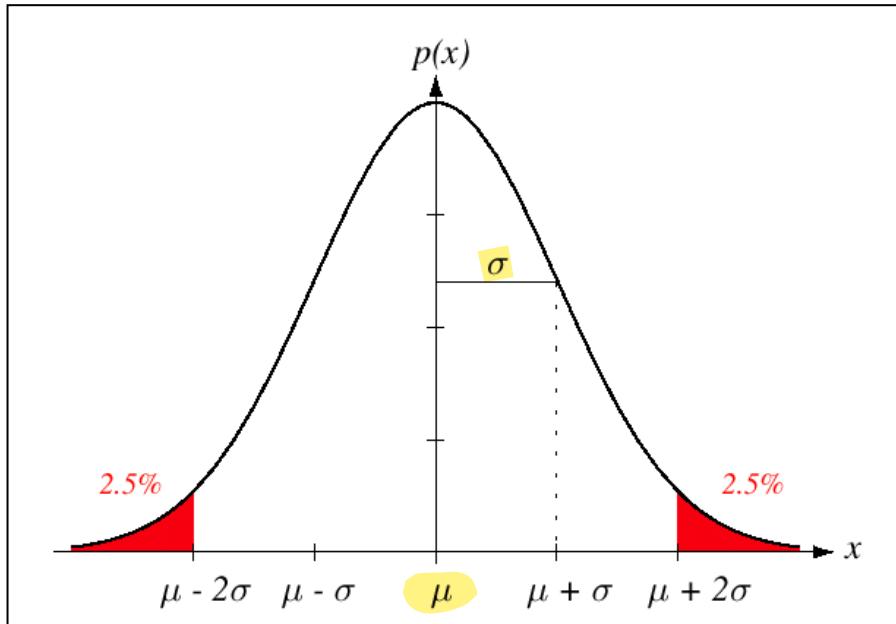
- L'approccio parametrico è infatti generalmente caratterizzato da un minor numero di gradi di libertà e il rischio di overfitting dei dati, quando il training set è piccolo, è minore.

# Distribuzione Normale ( $d=1$ )

- La densità di probabilità della distribuzione normale ( $d = 1$ ) è:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

dove  $\mu$  è il **valor medio** è  $\sigma$  la **deviazione standard** (o **scarto quadratico medio**) e il suo quadrato  $\sigma^2$  la **varianza**.



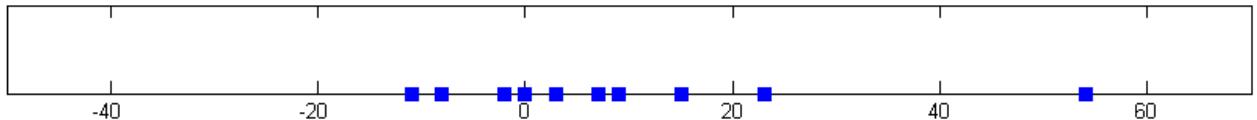
Solo il 5% circa del “volume” è esterno all’intervallo  $[\mu - 2\sigma, \mu + 2\sigma]$ .

Solitamente si assume che la distribuzione valga <sup>circa</sup> 0 a distanze maggiori di  $3\sigma$  dal valore medio.

# Esempio stima di $\mu$ e $\sigma$ (d=1)

- Dato un training set di pattern mono-dimensionali composto da  $n = 10$  elementi:

$$\{3, 7, 9, -2, 15, 54, -11, 0, 23, -8\}$$



- Questo significa che, se assumi che i tuoi dati provengano da una Gaussiana, il miglior modo per descrivere quella curva è usare semplicemente la media e la varianza calcolate direttamente dai dati.
- La stima dei parametri per massima verosimiglianza (**maximum likelihood**) si dimostra [1] essere: è un metodo statistico per determinare i parametri incogniti ( $\theta$ ) di una distribuzione di probabilità (il tuo modello) che si assume abbia generato un certo set di dati osservati ( $x$ ).
- Stima per  $\mu$ : media campionaria dei valori.

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i = \frac{3 + 7 + 9 + (-2) + 15 + 54 + (-11) + 0 + 23 + (-8)}{10} = \frac{90}{10} = 9$$

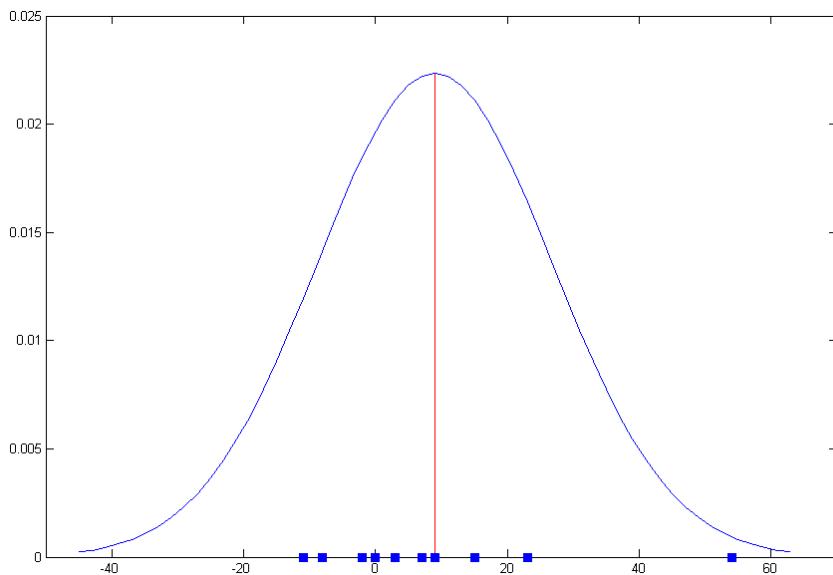
- Stima per  $\sigma^2$ : varianza campionaria dei valori.

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 =$$
$$= \frac{(3-9)^2 + (7-9)^2 + (9-9)^2 + (-2-9)^2 + (15-9)^2 + (54-9)^2 + (-11-9)^2 + (0-9)^2 + (23-9)^2 + (-8-9)^2}{10} = 318.8$$

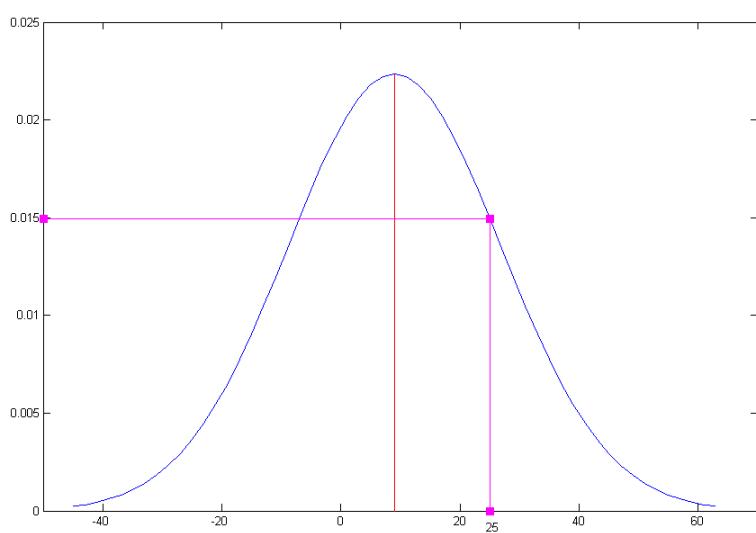
[1] [https://it.wikipedia.org/wiki/Metodo\\_della\\_massima\\_verosimiglianza](https://it.wikipedia.org/wiki/Metodo_della_massima_verosimiglianza)

# ...in forma grafica

dell'esempio



$$p(25) = \frac{1}{17.855 \cdot \sqrt{2 \cdot 3.1416}} \cdot e^{-\frac{(25-9)^2}{2 \cdot 3.1416}} = \frac{1}{44.7559} \cdot e^{-0.4015} = 0.01495$$



**ATTENZIONE SIAMO NEL CONTINUO:**

$p$  è una densità di probabilità:  $p(25)$  non è la probabilità del valore 25 (questa vale 0!) ma la densità di probabilità nel punto 25. Solo considerando un intervallo di valori (anche piccolo) sulla base possiamo parlare di probabilità.  
In altre parole l'intervallo  $[x, x + dx]$  ha probabilità  $p(x)dx$ .

# Distribuzione Normale Multivariata (Multinormale)

■ **Notazione:** per evitare confusione utilizziamo a pedice l'indice del pattern e (ove necessario) ad apice la componente (scalare):

- $\mathbf{x}_i$  pattern i-esimo (vettore)
- $x_i^j$  componente j-esima del pattern i-esimo (scalare)

■ La densità di probabilità nella distribuzione multinormale ( $d > 1$ ):

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2} (\mathbf{x}-\boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

dove  $\boldsymbol{\mu} = [\mu^1, \mu^2 \dots \mu^d]$  è il vettore medio è  $\Sigma = [\sigma^{ij}]$  la matrice di covarianza ( $d \times d$ ).

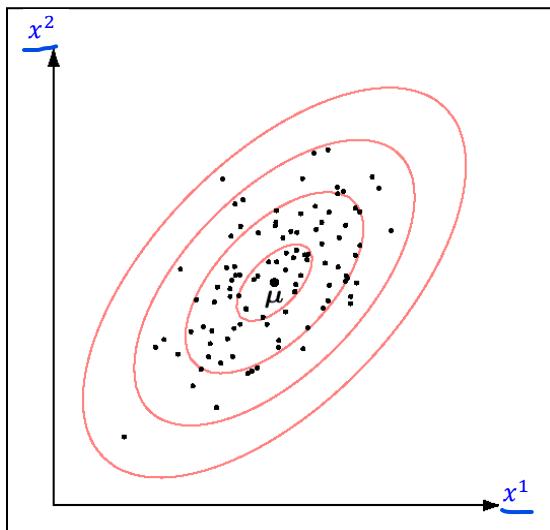
- Si assume che i vettori siano di tipo «colonna». L'apice  $t$  (trasposto) li trasforma in righe.
- $|\Sigma|$  e  $\Sigma^{-1}$  sono rispettivamente il determinante e l'inversa di  $\Sigma$ .
- La matrice di covarianza è sempre simmetrica e definita positiva, pertanto ammette inversa. Essendo simmetrica il numero di parametri che la definisce è  $d \cdot (d + 1)/2$  Perché considero triangolare inferiore e diagonale
- Gli elementi diagonali  $\sigma^{ii}$  sono le varianze dei rispettivi  $x^i$  (ovvero  $(\sigma^i)^2$ ); gli elementi non diagonali  $\sigma^{ij}$  sono le covarianze tra  $x^i$  e  $x^j$ : (quindi mi indica anche le correlazioni tra le varie dimensioni)
  - se  $x^i$  e  $x^j$  sono statisticamente indipendenti  $\sigma^{ij} = 0$
  - se  $x^i$  e  $x^j$  sono correlati positivamente  $\sigma^{ij} > 0$
  - se  $x^i$  e  $x^j$  sono correlati negativamente  $\sigma^{ij} < 0$

# Rappresentazione grafica

## Normale Multivariata

- Per  $d = 2$  la forma della distribuzione è quella di un'ellisse.

Dalla pendenza dell'ellisse si può vedere che  $X_1$  e  $X_2$  sono correlati positivamente



Le diverse ellissi individuano luoghi di punti a densità costante

(Più mi allontano dalla media, più la densità di probabilità si abbassa)

mostrano la dispersione/variabilità dei valori su quelle specifiche dimensioni

- $\mu = [\mu^1, \mu^2]$  controlla la posizione del centro.
- $\sigma^{11}$  e  $\sigma^{22}$  determinano l'allungamento sui due assi dell'ellisse.
- $\sigma^{12} = \sigma^{21}$  controlla la rotazione dell'ellisse rispetto agli assi cartesiani.
  - se  $= 0$  (matrice di covarianza diagonale), la distribuzione multinormale è definita come prodotto di  $d$  normali monodimensionali. In tal caso gli assi dell'ellisse sono paralleli agli assi cartesiani (es. Naive Bayes Classifier).
  - Se  $> 0$  (come nel caso della figura)  $x^1$  e  $x^2$  sono positivamente correlate (quando aumenta  $x^1$  aumenta anche  $x^2$ ).
  - Se  $< 0$   $x^1$  e  $x^2$  sono negativamente correlate (quando aumenta  $x^1$  cala  $x^2$ ).
- Gli assi dell'ellisse sono paralleli agli autovettori di  $\Sigma$ .

Le lunghezze (o meglio, il quadrato delle lunghezze) di questi assi sono proporzionali agli autovalori di  $\Sigma$ .

La Distanza di Mahalanobis migliora la semplice distanza euclidea, rendendola adatta per misurare la somiglianza tra un punto ( $x$ ) e il centro di una distribuzione ( $\mu$ ), tenendo conto delle caratteristiche interne di quella distribuzione: la varianza di ogni dimensione e la correlazione tra le dimensioni.

## Distanza Mahalanobis

- La distanza di Mahalanobis  $r$  tra  $x$  e  $\mu$ , definita dall'equazione:

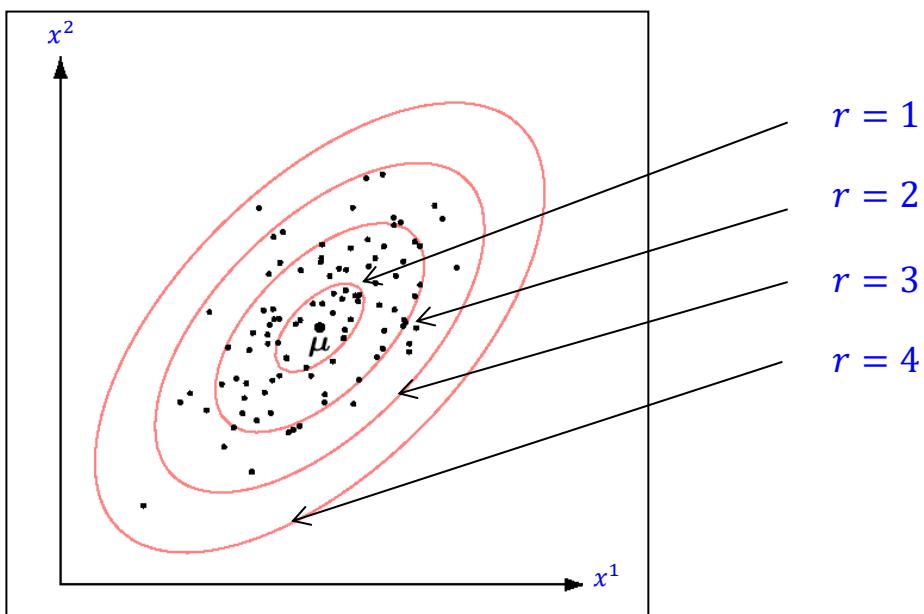
$$r^2 = (x - \mu)^t \Sigma^{-1} (x - \mu)$$

Che sarebbe parte dell'esponente della  $e$  nella formula della distribuzione normale multivariata

definisce i bordi a densità costante in una distribuzione multinormale. Tale distanza viene spesso utilizzata in sostituzione della distanza euclidea, essendo in grado di "pesare" le diverse componenti tenendo conto dei relativi **spazi di variazione** e della loro **correlazione**.

(tiene conto della variabilità delle dimensioni e correlazione tra le varie dimensioni)

Punti che si trovano sulla stessa ellisse hanno lo stesso valore di densità di probabilità (likelihood).

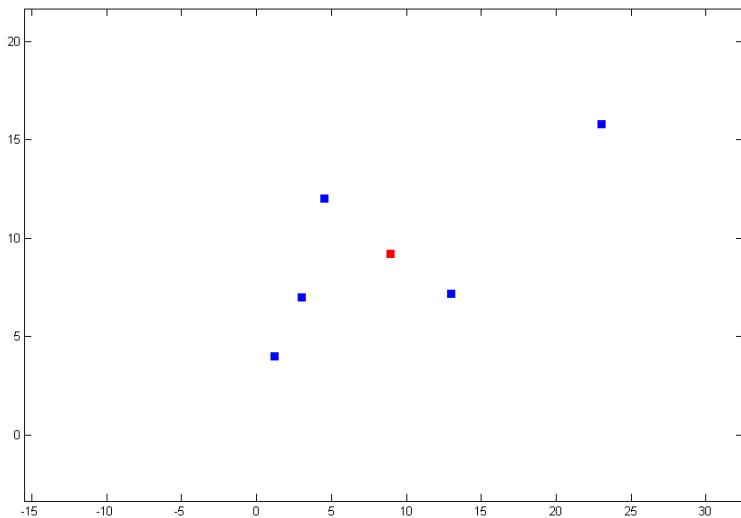


- Un semplice approccio di «**anomaly detection**» consiste nel fitting gaussiano (calcolo di  $\mu$  e  $\Sigma$ ) dai soli pattern di funzionamento normale del sistema. I nuovi pattern, la cui distanza di Mahalanobis eccede una soglia data, sono considerati anomalie. È efficace però solo quando la distribuzione dei dati è gaussiana.

## Esempio stima di $\mu$ e $\sigma$ (d=2)

- Dato un training set di pattern bi-dimensionali composto da  $n = 5$  elementi:

$$\{[3,7]^t, [4.5,12]^t, [13,7.2]^t, [1,4]^t, [23,15.8]^t\}$$



- La stima dei parametri per massima verosimiglianza (maximum likelihood) è:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu^1 \\ \mu^2 \\ \dots \\ \mu^d \end{bmatrix}, \quad \mu^i = \frac{1}{n} \sum_{k=1 \dots n} x_k^i \quad \boldsymbol{\mu} = \begin{bmatrix} \frac{3+4.5+13+1+23}{5} \\ \frac{7+12+7.2+4+15.8}{5} \end{bmatrix} = \begin{bmatrix} 8.9 \\ 9.2 \end{bmatrix}$$

o, in notazione vettoriale:

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1 \dots n} \mathbf{x}_i$$

# ...prosegue

$$\Sigma = \begin{bmatrix} \sigma^{11} & \sigma^{12} & \dots & \sigma^{1d} \\ \sigma^{21} & \sigma^{22} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \sigma^{d1} & \dots & \dots & \sigma^{dd} \end{bmatrix}, \quad \begin{array}{l} \text{Quando } i \neq j, \text{ covarianza} \\ \text{Quando } i = j, \text{ varianza} \end{array}$$

$$\sigma^{ij} = \frac{1}{n} \sum_{k=1 \dots n} (x_k^i - \mu^i) \cdot (x_k^j - \mu^j)$$

o, in notazione vettoriale:

$$\Sigma = \frac{1}{n} \sum_{i=1 \dots n} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^t$$

*calcolata come somma di matrici, ciascuna ottenuta come vettore colonna per vettore riga.*

$$\Sigma = \frac{1}{n} \mathbf{X} \mathbf{X}^t$$

*...è possibile scriverla in modo ancora più compatto (moltiplicazione di matrici) dove  $X$  è la matrice contenente i pattern nelle colonne (a cui è stata già tolta la media).*

$$\Sigma = \begin{bmatrix} \sigma^{11} & \sigma^{12} \\ \sigma^{21} & \sigma^{22} \end{bmatrix} = \begin{bmatrix} 66.44 & 25.32 \\ 25.32 & 17.456 \end{bmatrix}$$

$$\sigma^{11} = (\sigma^1)^2 = \frac{(3-8.9)^2 + (4.5-8.9)^2 + (13-8.9)^2 + (1-8.9)^2 + (23-8.9)^2}{5} = 66.44$$

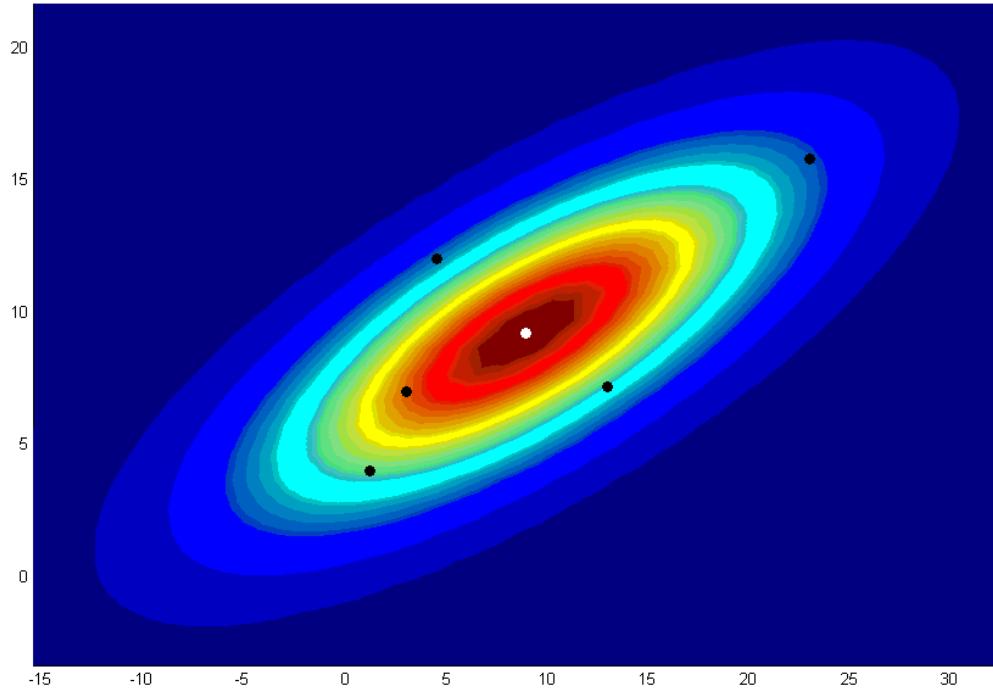
$$\sigma^{12} = \sigma^{21} = \frac{(3-8.9) \cdot (7-9.2) + (4.5-8.9) \cdot (12-9.2) + (13-8.9) \cdot (7.2-9.2) + (1-8.9) \cdot (4-9.2) + (23-8.9) \cdot (15.8-9.2)}{5} = 25.32$$

$$\sigma^{22} = (\sigma^2)^2 = \frac{(7-9.2)^2 + (12-9.2)^2 + (7.2-9.2)^2 + (4-9.2)^2 + (15.8-9.2)^2}{5} = 17.456$$

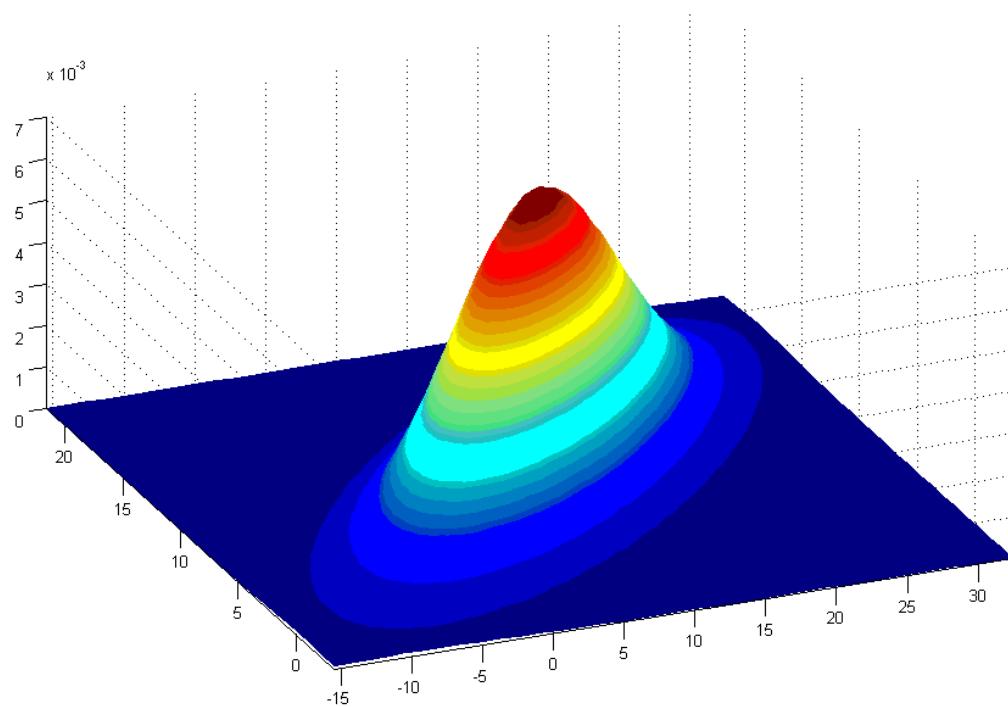
$$|\Sigma| = (66.44 \cdot 17.456) - (25.32 \cdot 25.32) = 518.674$$

$$\Sigma^{-1} = \begin{bmatrix} 0.0337 & -0.0488 \\ -0.0488 & 0.1281 \end{bmatrix}$$

# in forma grafica



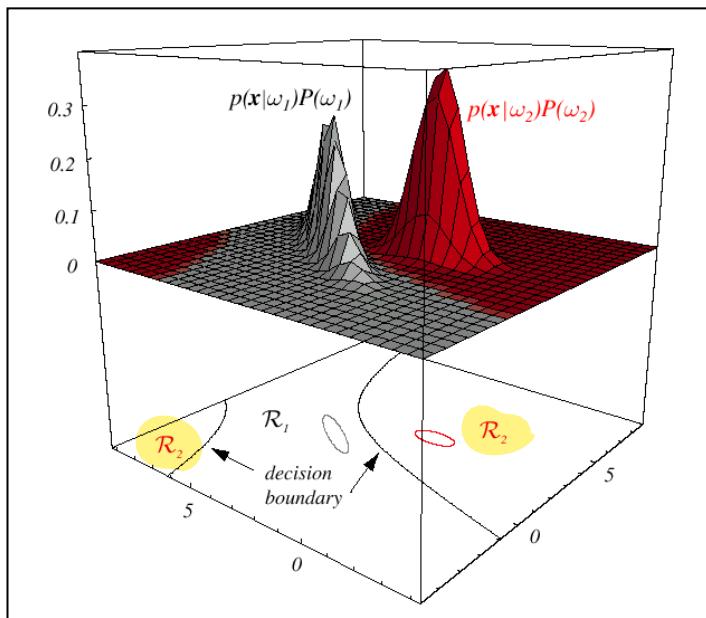
vista  
dall'alto



vista  
laterale

# Classificatore di Bayes con distribuzioni Multinormali

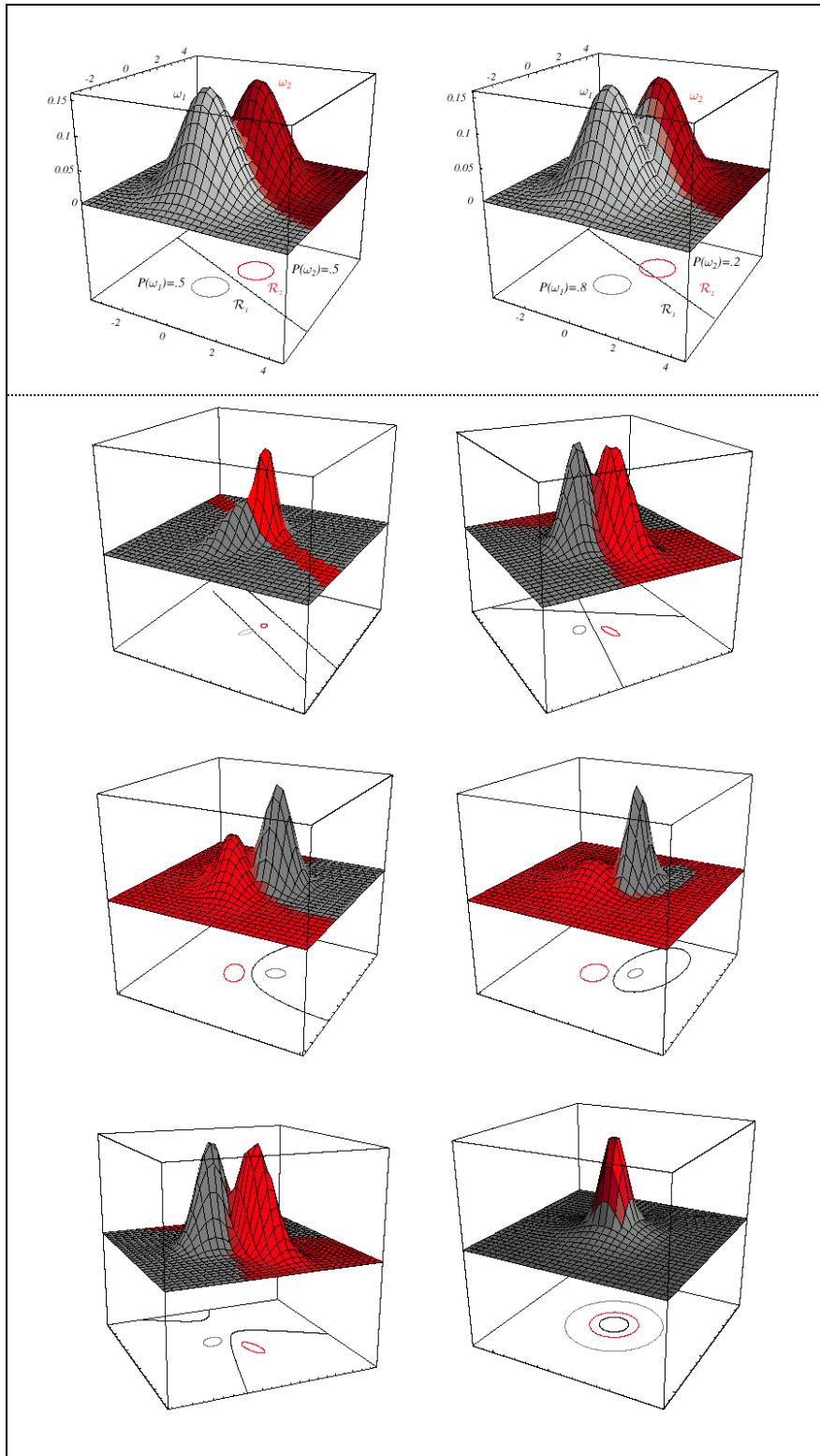
Per ogni classe, tramite approccio parametrico, devo trovare: la probabilità a priori della classe ed i parametri della distribuzione di probabilità condizionata. I parametri di ciascuna distribuzione di probabilità  $p(x | \omega_i)$  vengono trovati tramite il metodo della Massima Verosimiglianza (prendo i pattern di ciascuna classe del training set e calcolo i parametri).



Nell'esempio sono visualizzate le densità condizionali di 2 classi di pattern (distribuiti con distribuzione normale 2-dimensionale) corrette sulla base delle rispettive probabilità a priori.

- La classificazione è eseguita utilizzando la regola Bayesiana. Lo spazio è suddiviso in regioni non connesse. Nel caso specifico  $\mathcal{R}_2$  è costituita da due componenti disgiunte.
- Un **decision boundary** o **decision surface** (superficie decisionale) è una zona di confine tra regioni che il classificatore associa a classi diverse. Sul boundary la classificazione è ambigua.
- Le superfici decisionali possono assumere forme diverse. Nel caso specifico si tratta di due iperboli. In generale:
  - Se le 2 matrici di covarianza sono uguali tra loro: la superficie decisionale è un **iper-piano**.
  - Se le 2 matrici di covarianza sono arbitrarie: la superficie decisionale è un **iper-quadratica**.

# ...altri esempi di superfici decisionali



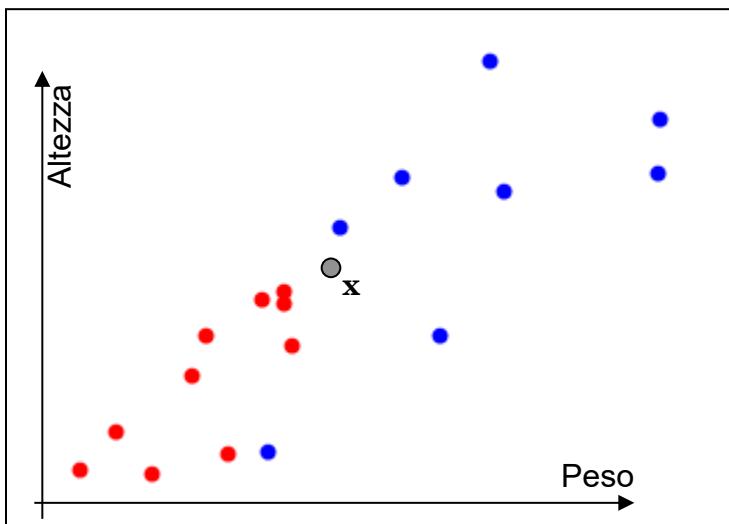
classificatore lineare

Stessa  
matrice  
di  
covarianza:  
iper-piani

classificatore  
non lineare  
**Differenti**  
matrici  
di covarianza:  
iper-  
quadratiche

# Maschi/Femmine

## con Bayes parametrico (multinormali)



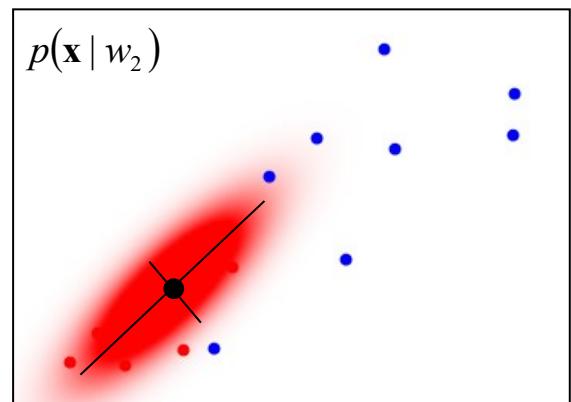
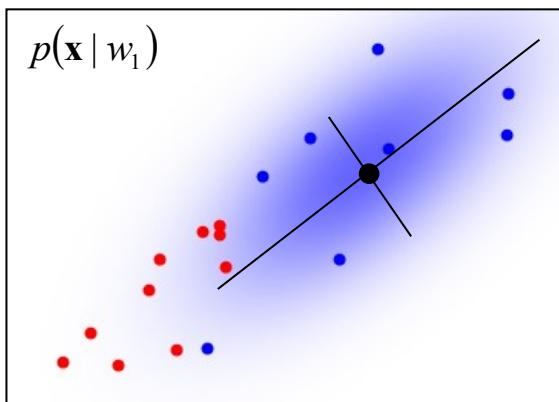
Peso	Altezza	Classe
72	173	w <sub>1</sub>
54	159	w <sub>1</sub>
65	172	w <sub>1</sub>
58	170	w <sub>1</sub>
62	165	w <sub>1</sub>
72	176	w <sub>1</sub>
60	173	w <sub>1</sub>
64	179	w <sub>1</sub>
55	166	w <sub>2</sub>
46	158	w <sub>2</sub>
52	158	w <sub>2</sub>
47	160	w <sub>2</sub>
55	167	w <sub>2</sub>
54	166	w <sub>2</sub>
55	164	w <sub>2</sub>
49	157	w <sub>2</sub>
51	165	w <sub>2</sub>
51	162	w <sub>2</sub>

Stima dei parametri dal training set:

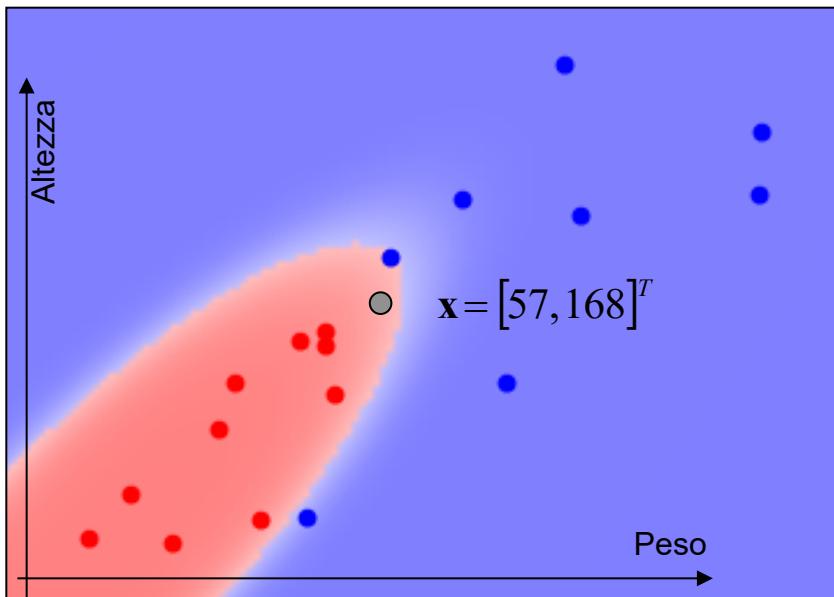
$$\boldsymbol{\mu}_1 = [63.4, 170.9]^T \quad \boldsymbol{\mu}_2 = [51.5, 162.3]^T \quad \Sigma_1 = \begin{bmatrix} 35.2 & 23.3 \\ 23.3 & 34.9 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 10.1 & 8.9 \\ 8.9 & 13.0 \end{bmatrix}$$

$$p(\mathbf{x} | w_1) = \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_1|^{1/2}} \cdot \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^t \Sigma_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1)\right]$$

$$p(\mathbf{x} | w_2) = \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_2|^{1/2}} \cdot \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^t \Sigma_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2)\right]$$



## ... continua



Supponendo di non avere altre informazioni, si possono stimare le probabilità a priori come:  $P(w_1) = 8/18$ ,  $P(w_2) = 10/18$

$$p(\mathbf{x} | w_1) = \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_1|^{1/2}} \cdot \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^t \Sigma_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1)\right] = 0.0033$$

$$p(\mathbf{x} | w_2) = \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_2|^{1/2}} \cdot \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^t \Sigma_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2)\right] = 0.0045$$

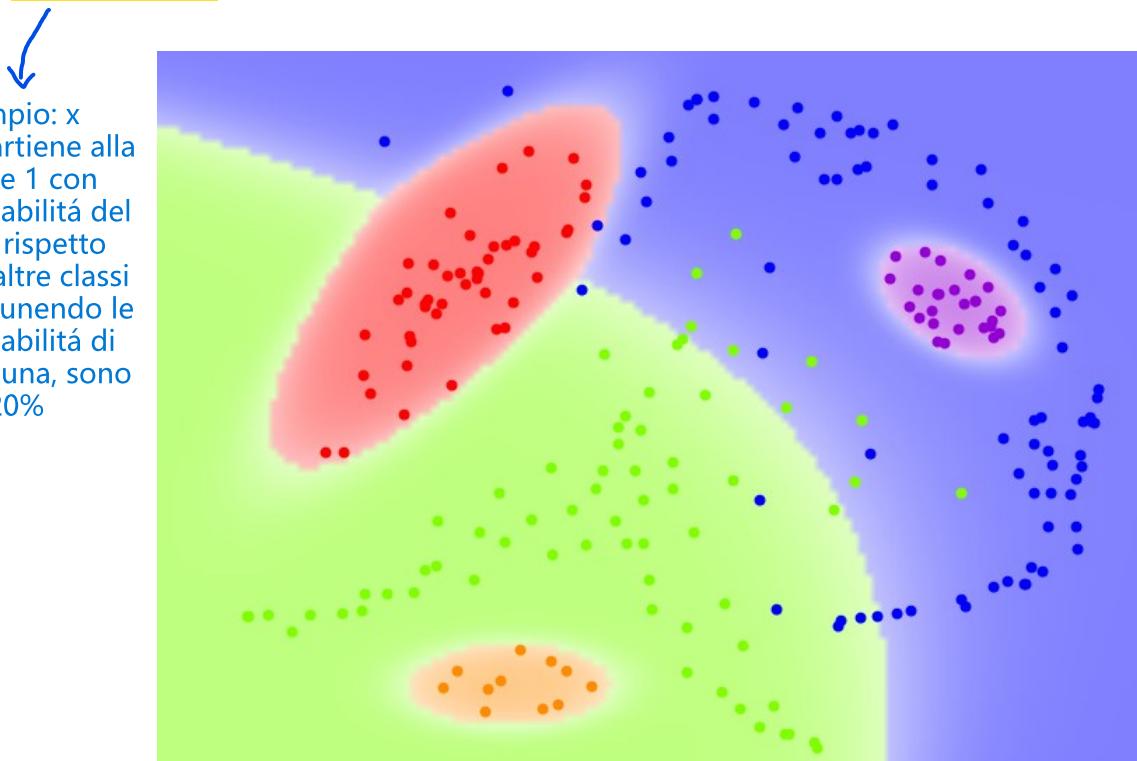
$$p(\mathbf{x}) = \sum_{i=1}^s p(\mathbf{x} | w_i) \cdot P(w_i) = 0.0040$$

$$P(w_1 | \mathbf{x}) = \frac{p(\mathbf{x} | w_1) \cdot P(w_1)}{p(\mathbf{x})} \cong 0.36$$

$$P(w_2 | \mathbf{x}) = \frac{p(\mathbf{x} | w_2) \cdot P(w_2)}{p(\mathbf{x})} \cong 0.64$$

# Bayes e confidenza di classificazione

Un grande vantaggio del classificatore di Bayes, rispetto ad altri classificatori, è legato al fatto che esso produce un valore di output probabilistico (un vero e proprio valore di probabilità tra 0 e 1, con somma 1 sulle diverse classi) che può essere utilizzato come confidenza (visualizzata nella figura come sfumatura colore):



Infatti, un classificatore può assegnare un pattern  $x$  a una classe  $w_i$  con diversi livelli di certezza (o confidenza) che possono essere impiegati per:

Se la massima probabilità a posteriori di un pattern  $x$  è inferiore a una soglia prestabilita (es.  $P(\max)<0.7$ ), il classificatore può decidere di scartare il pattern,

etichettandolo come "sconosciuto" o "anomalia", invece di forzarlo in una delle classi.

- scartare pattern in applicazioni open-set con soglia
- costruire un multi-classificatore

Se ogni classificatore produce un vettore di probabilità a posteriori, questi vettori possono essere combinati (ad esempio, facendo la media pesata o un prodotto) prima di prendere la decisione finale. Questo processo è più informativo e robusto della semplice fusione a livello di decisione finale.

Se non si è interessati alla confidenza, nella formula di Bayes non è necessario dividere per  $p(x)$  il numeratore, e la regola di Bayes è semplicemente:

$p(x)$  normalizza il calcolo della probabilità a posteriori

$$b = \underset{i=1..s}{\operatorname{argmax}} \{ p(x|w_i) \cdot P(w_i) \}$$

# Bayes parametrico in pratica

(o sul tipo di distribuzione)

- Molto spesso si fanno ipotesi azzardate sulla normalità delle densità di probabilità delle classi del problema senza aver sperimentalmente eseguito nessuna verifica; ciò porta ad ottenere cattivi risultati di classificazione.

Pertanto, dato un problema con  $s$  classi e dato un training set (significativo), deve essere innanzitutto **valutata la rispondenza alla "normalità" delle  $s$  distribuzioni**; questo può essere fatto:

- in modo formale (es: test statistico di **Malkovich - Afifi [1]** basato sull'indice di Kolmogorov - Smirnov)
- in modo empirico, visualizzando in vari modi le **nuvole dei dati** (esistono dei tool già predisposti per questo tipo di analisi fino a 3D) o gli **istogrammi sulle diverse componenti** e confrontandoli con le curve teoriche.
- Una volta provata una (seppur vaga) normalità delle distribuzioni, si stimano a partire dai dati, vettore medio  $\mu$  e matrice di covarianza  $\Sigma$  (maximum likelihood). (Metodo di Massima Verosimiglianza)
- Per quanto riguarda le probabilità a priori queste possono essere estratte dalle percentuale di campioni che nel training set appartengono alle diverse classi, o in caso di assenza di informazioni possono essere poste tutte uguali tra loro.
- Ogni nuovo pattern da classificare, è assegnato a una delle possibili classi in accordo con la regola di Bayes nella quale media e covarianza sono ora note.

[1] K. Fukunaga, *Statistical Pattern Recognition*, Academic Press, 1990.

# Approcci non parametrici e stima della Densità

tipologie di

Non vengono fatte ipotesi sulle distribuzioni dei pattern e le densità di probabilità sono **stimate direttamente** dal training set.

Il problema della stima accurata della densità è ritenuto da molti un problema più complesso della classificazione. Pertanto perché risolvere come sotto-problema un problema che è più complesso dell'intero compito di classificazione ?

In generale la stima della densità è affrontabile in **spazi a dimensionalità ridotta** (es.  $d = 3$ ) e diventa critica al crescere della dimensionalità (**curse of dimensionality**): il volume dello spazio aumenta così tanto che i pattern diventano troppo sparsi.

La classificazione richiede solo di trovare la frontiera decisionale, ovvero il confine nello spazio delle feature che separa una classe dall'altra.

La frontiera decisionale può spesso essere trovata in modo più semplice, ad esempio con SVM, senza la necessità di modellare l'intera distribuzione. Stime accurate della densità sono molto più difficili da ottenere e spesso richiedono molti più dati.

Esempio:

- In un cubo 3D di lato 1, loro distanza media di due punti scelti a caso è 0.66
- In un ipercubo con 1M di dimensioni, la distanza media di due punti scelti a caso è 408.25 !

In uno spazio dove i pattern sono così sparsi occorrono molti più esempi per fare stime di densità affidabili.

La riduzione di dimensionalità (ne parleremo in seguito) è una delle tecniche possibili per contrastare la curse of dimensionality

Il motivo principale per cui la stima della densità è difficile è legato alla dimensionalità dello spazio delle feature. All'aumentare del numero di dimensioni ( $d$ ) che descrivono i pattern, il volume dello spazio cresce in modo esponenziale. Quindi, i dati di addestramento disponibili, anche se numerosi, diventano estremamente sparsi in questo vasto spazio.

utilizzando un approccio non parametrico, in particolare il metodo basato sul Finestra Scorrivole (Parzen Window) o sui k-Vicini più Prossimi (k-NN), che condividono lo stesso fondamento matematico.

## Stima della Densità

La probabilità che un pattern  $\mathbf{x}$  cada all'interno di  $\mathfrak{R}$  è:

l'integrale della densità di probabilità sulla regione  $\mathfrak{R}$ .

$$P_1 = \int_{\mathfrak{R}} p(\mathbf{x}') d\mathbf{x}'$$

un'area

Dati  $n$  pattern indipendenti, la probabilità che  $k$  di questi cadano nella regione  $\mathfrak{R}$  è calcolabile attraverso la distribuzione binomiale:

indica il numero di modi in cui si possono scegliere  $k$  successi da  $n$  prove.

$$P_k = \binom{n}{k} P_1^k (1 - P_1)^{n-k}$$

di pattern che cadono in  $\mathfrak{R}$

il cui valor medio è  $k = n P_1$  (e quindi  $P_1 = k/n$ )

Assumendo che la regione  $\mathfrak{R}$  (di volume  $V$ ) sia piccola e che quindi  $p(\cdot)$  non vari significativamente all'interno di essa:

quasi costante

$$P_1 = \int_{\mathfrak{R}} p(\mathbf{x}') d\mathbf{x}' \approx p(\mathbf{x}) \cdot V$$

possiamo approssimare l'integrale con l'altezza della densità nel punto  $\mathbf{x}$  moltiplicata per il volume

$k$ : Il numero di punti di addestramento che sono caduti nella regione  $\mathfrak{R}$  (vicini a  $\mathbf{x}$ )

$n$ : Il numero totale di punti di addestramento.

$V$ : Il volume della regione  $\mathfrak{R}$ .

$$p(\mathbf{x}) = \frac{P_1}{V} = \frac{k}{n \cdot V}$$

è il cuore della stima della densità in un contesto non parametrico.

Questa formula costituisce la base di due importanti metodi non parametrici:

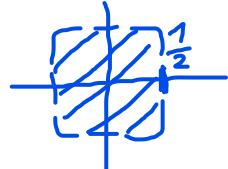
Metodo della Finestra Fissa (Parzen Window): Si fissa il volume  $V$  (quindi la dimensione della regione  $\mathfrak{R}$ ) e si calcola il  $k$  risultante.

Metodo dei k-NN (k-Nearest Neighbors): Si fissa il numero di vicini  $k$  (e quindi la "qualità" statistica della stima) e si calcola il volume  $V$  risultante necessario per racchiudere quei  $k$  vicini.

Questi metodi permettono di "disegnare" la forma della distribuzione di probabilità direttamente dai dati, senza assumere a priori che sia, ad esempio, Gaussiana.

# Parzen Window

La regione  $\mathcal{R}$ , denominata finestra (Window), è costituita da un ipercubo  $d$ -dimensionale, definito dalla funzione  $\varphi$ :



$$\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \leq \frac{1}{2}, j = 1 \dots d \\ 0 & \text{altrimenti} \end{cases}$$

Questa funzione è un indicatore: vale 1 se il vettore di input  $u$  si trova all'interno dell'ipercubo centrale nell'origine, e 0 altrimenti.

Per calcolare la densità in un punto  $x$ , si centra l'ipercubo in  $x$ .

Dato un generico ipercubo centrato in  $x$  e avente lato  $h_n$  (e quindi volume  $V_n = h_n^d$ ) il numero di pattern del training set che cadono all'interno dell'iper-cubo è dato da:

$k_n$  è il numero di pattern che cadono all'interno dell'ipercubo centrato in  $x$

$$k_n = \sum_{i=1}^n \varphi\left(\frac{\mathbf{x}_i - \mathbf{x}}{h_n}\right)$$

trasforma la posizione del punto  $x_i$  in modo che, se e solo se  $x_i$  cade all'interno dell'ipercubo di lato  $h_n$  centrato in  $x$ , la funzione  $\varphi$  restituirà 1 per quel punto. La sommatoria conta tutti gli "1", fornendo così il numero  $k_n$ .

sostituendo  $k_n$  (vedi lucido precedente) si ottiene:

$$p_n(\mathbf{x}) = \frac{1}{n \cdot V_n} \sum_{i=1}^n \varphi\left(\frac{\mathbf{x}_i - \mathbf{x}}{h_n}\right), \quad \text{dove } V_n = h_n^d$$

$h_n$  e quindi  $V_n$  sono costanti

Questa è l'espressione formale della stima della densità di Parzen Window. L'indice  $n$  in  $p_n(x)$  e  $V_n$  indica che la stima dipende dalla dimensione del training set  $n$ .

Ovviamente, specie nel caso in cui il numero di pattern non sia elevato, la dimensione della finestra  $V_n$  (e quindi il lato  $h_n$ ) ha un forte impatto sul risultato, infatti:

- ➡ Solo pochi punti  $k$  cadono all'interno. La stima è molto sensibile alle fluttuazioni casuali del train set.
- ➡ Se la finestra è **piccola**, la stima risulta piuttosto "rumorosa", molto attratta dai campioni e statisticamente instabile.

Tanti punti  $k$  cadono all'interno, rendendo la stima stabile. Tuttavia, la media in un grande volume "livella" i dettagli locali.

- ➡ Se la finestra è **grande** la stima è più stabile ma piuttosto **vaga** e sfuocata.

Per garantire che, all'aumentare dei dati, la stima  $p_n(x)$  converga alla vera densità  $p(x)$ ,

Si dimostra che per ottenere convergenza, la dimensione della finestra deve essere calcolata tenendo conto del numero di campioni del training set:

$$V_n = \frac{V_1}{\sqrt{n}}, \quad \text{dove } V_1 \text{ (o } h_1\text{)} \text{ è un iperparametro}$$

iniziale che non dipende da  $n$

ML

23



Il metodo Parzen Window "standard" che abbiamo visto prima utilizzava una funzione finestra a forma di ipercubo. Questa funzione è hard, perché un punto o contribuisce completamente (se è dentro) o non contribuisce affatto (se è fuori).

# Parzen Window con Soft kernel

I pattern vicini a  $x$  contribuiscono molto alla stima della densità, mentre quelli lontani contribuiscono poco.

Nella pratica, invece di funzioni finestra ipercubo si utilizzano **kernel function** più **soft** grazie alle quali ogni pattern  $x_i$  contribuisce alla stima di densità in un intorno di  $x$  in accordo con la distanza da  $x$ . In questo modo le superfici decisionali risultano molto più regolari (**smoothed**). L'uso di un kernel soft evita i salti e le discontinuità nel calcolo della densità, che sono tipici dell'ipercubo hard.

Le kernel function devono essere funzioni densità (sempre  $\geq 0$  e con integrale su tutto lo spazio uguale a 1). Utilizzando la funzione **multinormale** (con  $\mu = [0\dots 0]$  e  $\Sigma = I$ ):

Questa formula è per una Gaussiana standard unitaria centrata nell'origine

$$\varphi(\mathbf{u}) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{\mathbf{u}^t \mathbf{u}}{2}}$$

Questa funzione assegna il peso massimo (densità) all'origine ( $u=0$ , ovvero quando  $x_i=x$ ) e il peso decade rapidamente all'aumentare della distanza dall'origine.

Aumentando  $n$  (numero di dati): La stima diventa più affidabile e robusta (varianza minore).

Diminuendo  $h$  (kernel stretto): Si catturano più dettagli locali, ma si aumenta il rischio di rumore (alta varianza).

$n=15$

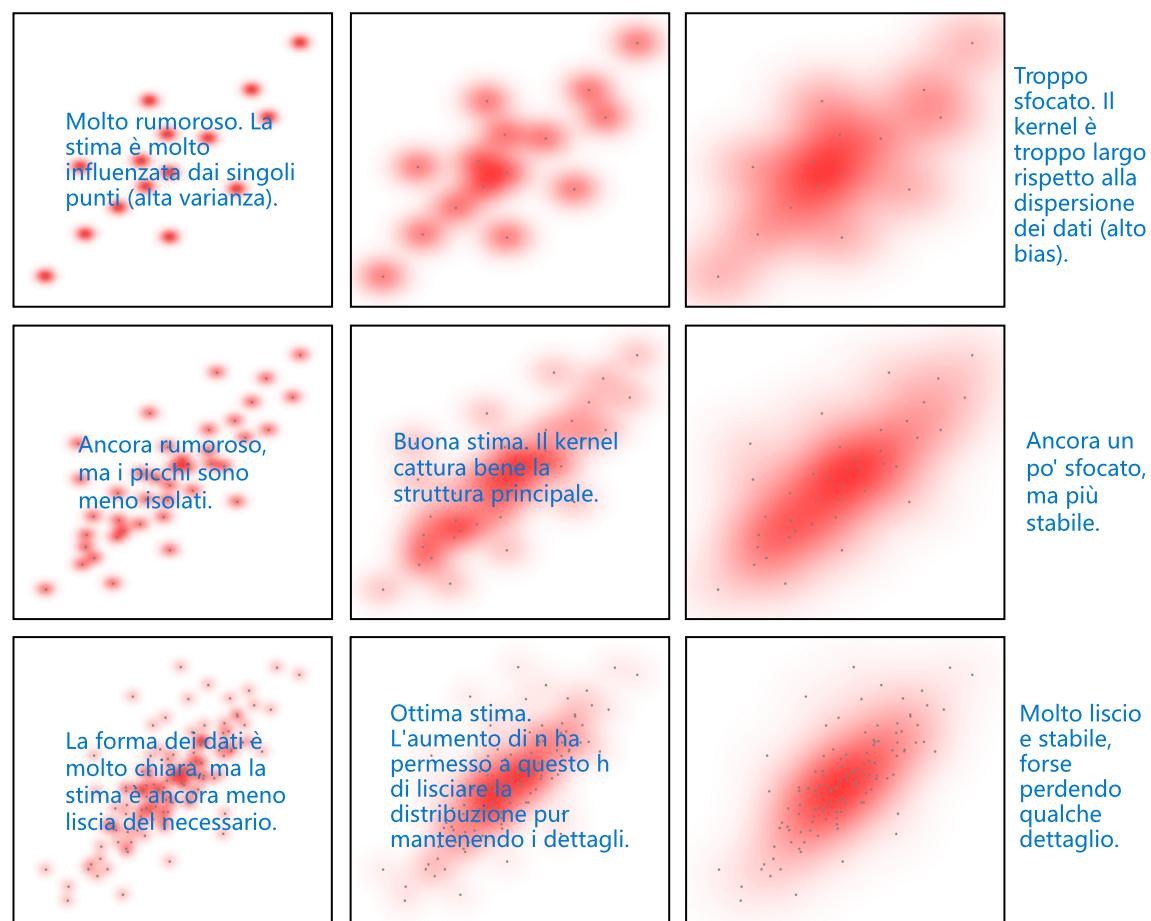
Aumentando  $h$  (kernel largo): Si ottiene una stima più liscia e stabile, ma si rischia di perdere dettagli importanti (alto bias).

La densità finale  $p(x)$  in qualsiasi punto dello spazio è data dalla somma (la sovrapposizione) di tutte queste singole campane.

$h=3$

$h=8$

$h=15$



ML

24

Nel metodo della Finestra Hard, la dimensione della regione R era il Volume  $V_n$  (con lato  $h_n$ ), e tutti i punti dentro contribuivano in modo uguale. Nel metodo del Kernel Soft, non c'è una regione definita, ma il Potere di Influenza è dato dalla larghezza di banda  $h$ .

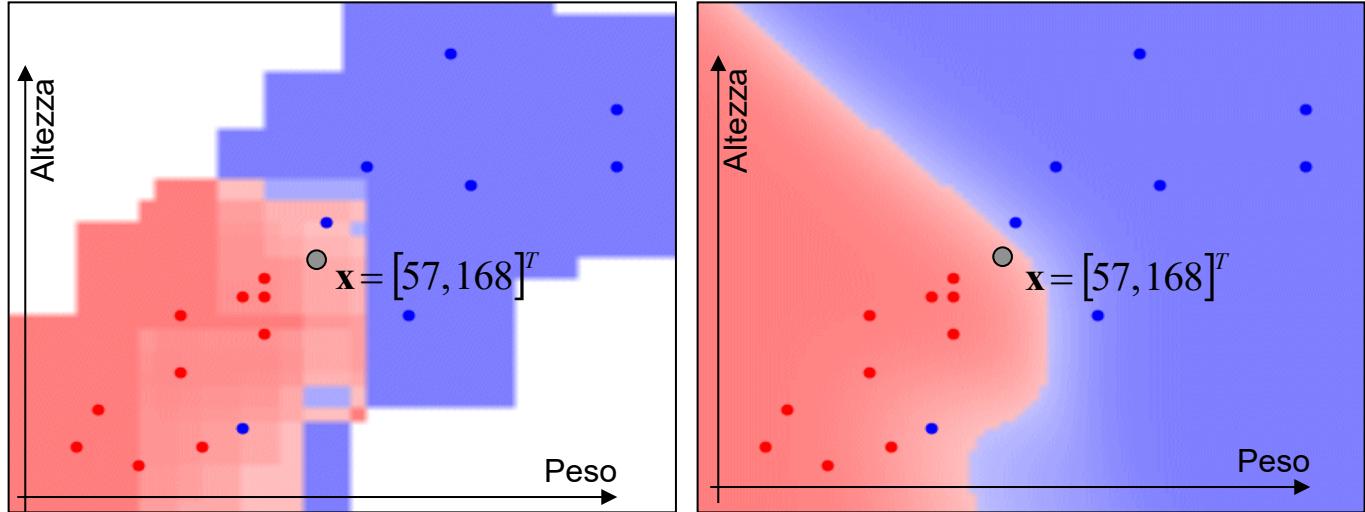


$h$  è l'equivalente funzionale di  $h_n$  (e quindi di  $V_n$ ) perché regola la porzione di spazio che contribuisce alla stima in  $x$ .

## Maschi/Femmine

### con Bayes + Parzen Window

L'uso del Kernel Normale (Soft) produce una frontiera decisionale liscia e curva (grafico a destra), molto più robusta e generalizzabile rispetto alle frontiere a gradino e "a blocchi" prodotte dall'Ipercubo (grafico a sinistra). Questo è il motivo per cui i kernel soft sono preferiti nella pratica.



Stima non-parametrica della densità attraverso Parzen Window  
(nell'ipotesi che  $P(w_1) = 8/18$ ,  $P(w_2) = 10/18$ )

➤ Funzione Kernel *iper cubo* con  $h=10$  (grafico a sinistra)

$$p(\mathbf{x} | w_1) = 0.0038 \quad p(\mathbf{x} | w_2) = 0.0040$$

Densità di Probabilità Condizionata

$$P(w_1 | \mathbf{x}) = \frac{p(\mathbf{x} | w_1) \cdot P(w_1)}{p(\mathbf{x})} \approx 0.43$$

Stimata con Parzen Window con Funzione Kernel Ipercubo (Hard)

Densità di probabilità totale (la somma delle densità condizionate pesate per le a priori).

$$p(\mathbf{x}) = \sum_{i=1}^s p(\mathbf{x} | w_i) \cdot P(w_i) = 0.0039$$

$$P(w_2 | \mathbf{x}) = \frac{p(\mathbf{x} | w_2) \cdot P(w_2)}{p(\mathbf{x})} \approx 0.57$$

➤ Funzione Kernel *normale* con  $h=3$  (grafico a destra)

$$p(\mathbf{x} | w_1) = 0.0024 \quad p(\mathbf{x} | w_2) = 0.0041 \quad p(\mathbf{x}) = \sum_{i=1}^s p(\mathbf{x} | w_i) \cdot P(w_i) = 0.0033$$

$$P(w_1 | \mathbf{x}) = \frac{p(\mathbf{x} | w_1) \cdot P(w_1)}{p(\mathbf{x})} \approx 0.32$$

Stimata con Parzen Window con Funzione Kernel Normale (Soft)

$$P(w_2 | \mathbf{x}) = \frac{p(\mathbf{x} | w_2) \cdot P(w_2)}{p(\mathbf{x})} \approx 0.68$$

# Classificatore Nearest Neighbor (NN)

Data una metrica  $dist(\cdot)$  nello spazio multidimensionale (es. distanza euclidea) il classificatore **nearest neighbor** (letteralmente "il più vicino tra i vicini"), classifica un pattern  $\mathbf{x}$  con la stessa classe dell'elemento  $\mathbf{x}'$  ad esso più vicino nel training set TS:

se secondo la metrica  $dist$

Si identifica l'elemento  $\mathbf{x}'$  in TS che minimizza la distanza da  $\mathbf{x}$

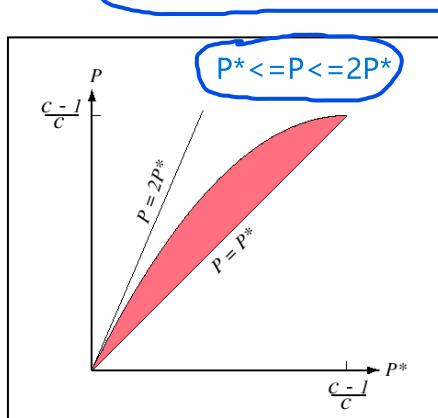
$$dist(\mathbf{x}, \mathbf{x}') = \min_{\mathbf{x}_i \in TS} \{ dist(\mathbf{x}, \mathbf{x}_i) \}$$

■ Invece di derivare dai dati le distribuzioni condizionali delle classi per poi far uso della regola di Bayes per la classificazione, questo classificatore cerca in modo **piuttosto pragmatico** di massimizzare direttamente la probabilità a posteriori; infatti se  $\mathbf{x}'$  è molto vicino a  $\mathbf{x}$  è lecito supporre che:

$$P(w_i | \mathbf{x}) \approx P(w_i | \mathbf{x}')$$

Dato che  $\mathbf{x}'$  è un punto noto del training set, si conosce già la sua classe  $w_i$ , che viene usata come stima per la classe di  $\mathbf{x}$ .

■ In effetti, si può **dimostrare** (solo però nel caso di TS popolato da infiniti campioni) che la probabilità di errore  $P$  (nella figura sotto) della regola nearest neighbor non è mai peggiore del doppio del minimo errore possibile  $P^*$  (quello Bayesiano).



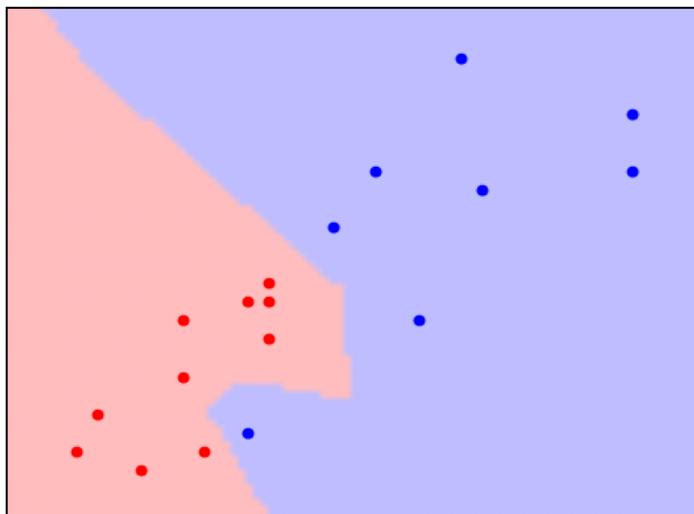
È l'errore minimo teoricamente possibile. Si ottiene solo se si conoscono esattamente le vere distribuzioni di probabilità.

**Bayes Reale:** Nella pratica, se il training set è piccolo o lo spazio è ad alta dimensionalità, la stima delle densità con Parzen Window o altri metodi può essere molto imprecisa (come discusso con la Curse of Dimensionality).

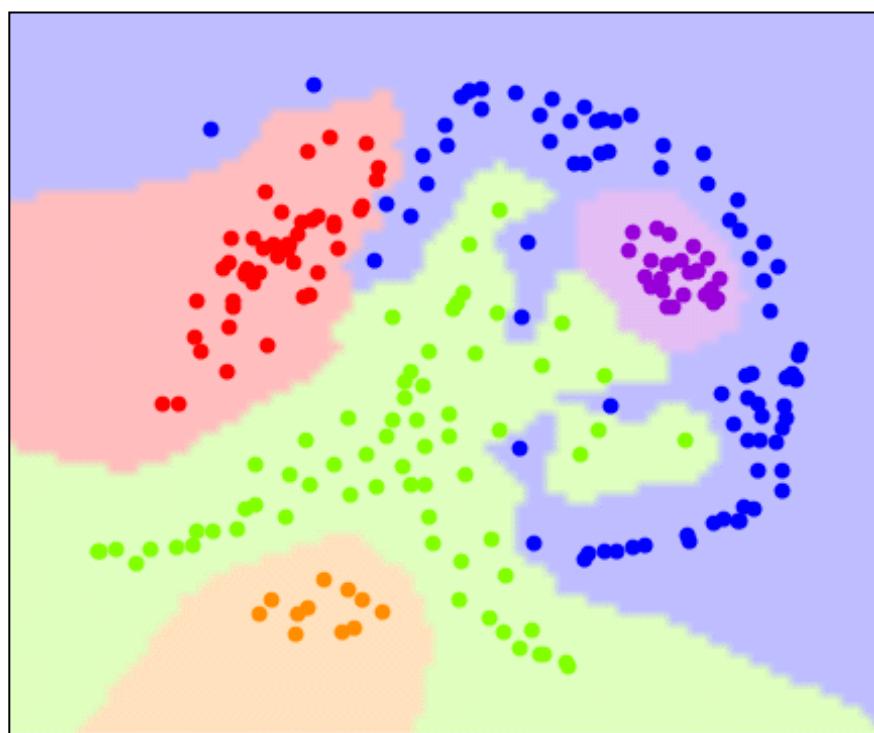
**NN Reale:** In questi casi, la semplicità e la robustezza del NN possono portare a risultati migliori del classificatore Bayesiano, i cui risultati sono stati "inquinati" da stime di densità inaccurate.

■ Nella pratica, questo non significa però che l'approccio Bayesiano fornisca sempre risultati migliori di nearest neighbor, infatti se la stima delle densità condizionali è poco accurata i risultati del classificatore Bayesiano possono essere peggiori.

## Esempi NN



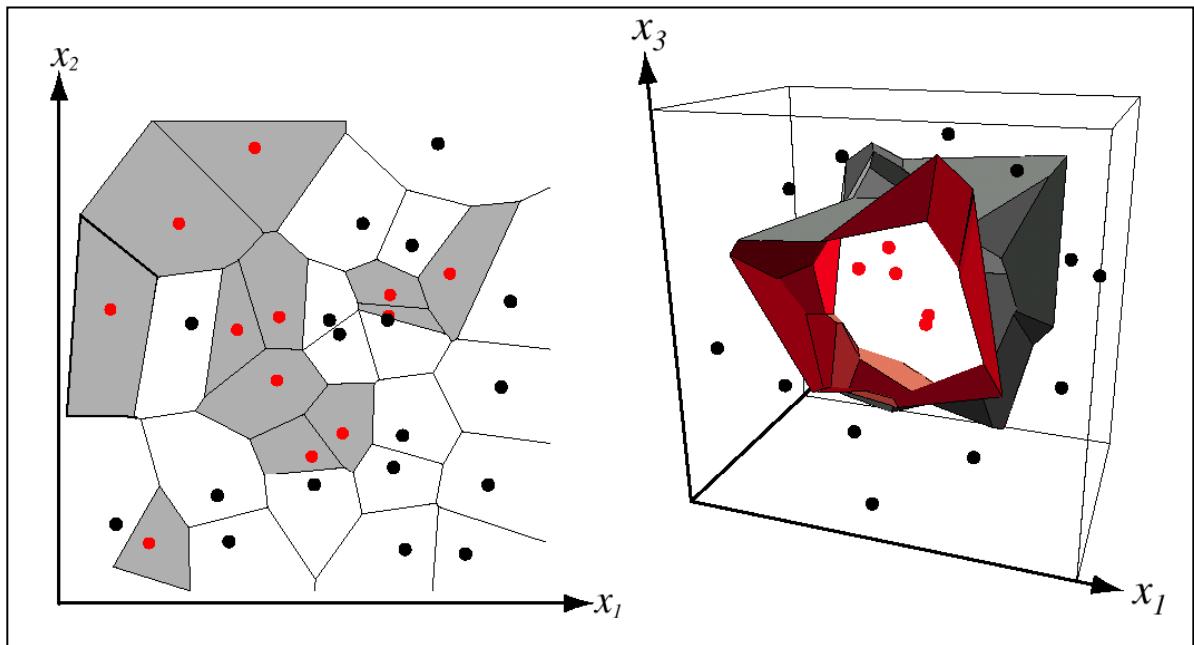
- Nell'esempio visto in precedenza, la regola NN assegna il pattern  $x$  alla classe  $w_1$  (maschi - blu)
- La figura seguente mostra il partizionamento dello spazio operato dalla regola NN su un training set con 5 classi:



# Da NN a k-NN

(regola 1-NN)

- La regola nearest neighbor produce un partizionamento dello spazio, noto come *tassellazione di Voronoi*:



Ogni elemento  $x_i \in TS$  determina un tassello, all'interno del quale i pattern saranno assegnati alla stessa classe di  $x_i$ .

- La regola di classificazione nearest neighbor è piuttosto radicale; infatti basta che un elemento del training set non sia molto "affidabile" (**outlier**) affinché tutti i pattern nelle sue vicinanze siano in seguito etichettati non correttamente.

■ Che errore commette il classificatore NN sul training set? **ZERO**  
Per ogni punto  $x_i$  del training set, il suo vicino più prossimo è  $x_i$  stesso (la distanza è zero). Dato che  $x_i$  viene classificato con la sua stessa classe, l'errore di classificazione sui dati di addestramento è nullo.

- Un modo generalmente più robusto, che può essere visto come estensione della regola nearest-neighbor (in questo caso detta 1-nearest neighbor) è il cosiddetto classificatore **k-nearest neighbor (k-NN)**.

La Regola: Invece di considerare solo il vicino più prossimo, k-NN considera i  $k$  vicini più prossimi a  $x$ .  
Decisione: La classe assegnata a  $x$  è determinata tramite un voto a maggioranza tra le classi di questi  $k$  vicini.

Effetto di Smoothing: Il voto a maggioranza agisce da filtro contro il rumore.  
Immunità agli Outlier: Un singolo outlier etichettato male non può più determinare la classificazione da solo. Per influenzare la decisione, gli outlier dovrebbero essere presenti in numero sufficiente (più della metà dei  $k$  vicini) nella zona di  $x$ .

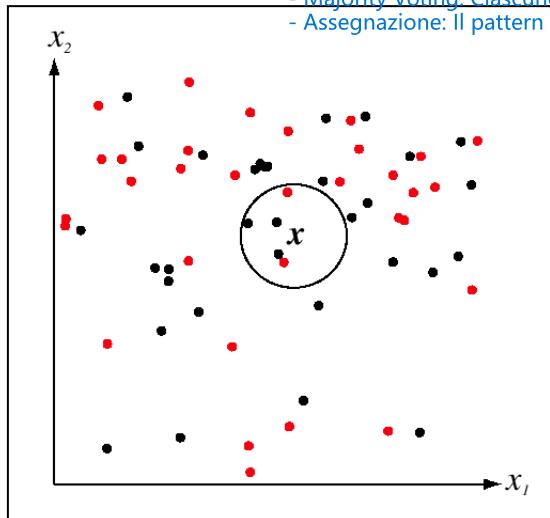
ML

## k-Nearest-Neighbor (k-NN)

- La regola ***k-Nearest Neighbor*** (**k-NN**) determina i  $k$  elementi più vicini al pattern  $x$  da classificare ( $k$  è un iperparametro); ogni pattern tra i  $k$  vicini vota per la classe cui esso stesso appartiene; il pattern  $x$  viene assegnato alla classe che ha ottenuto il maggior numero di voti.

La regola del k-NN per classificare un nuovo pattern  $x$  è:

- Ricerca dei Vicini: Trova gli  $k$  elementi del Training Set (TS) che hanno la distanza minore da  $x$ . Questi  $k$  punti formano l'ipersfera di ricerca attorno a  $x$ .
- Majority Voting: Ciascuno di questi  $k$  vicini vota per la sua classe.
- Assegnazione: Il pattern  $x$  viene assegnato alla classe che ha ottenuto il maggior numero di voti.



nella figura il classificatore 5-NN,  
assegna x alla classe "nera"  
in quanto quest'ultima ha ricevuto 3  
voti su 5.

(numero di classi pari)

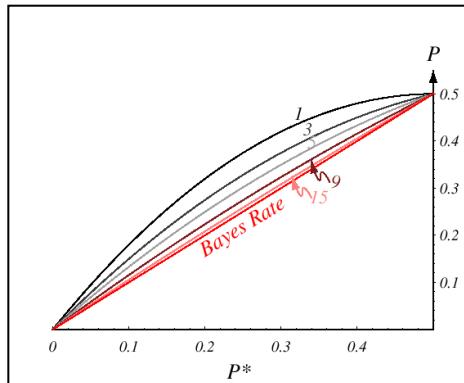
Nel caso di 2 classi è bene scegliere  $k$  dispari per evitare pareggi.

nelle zone a bassa densità, la finestra deve essere adattata per trovare i  $k$  campioni

- Per TS infiniti la regola di classificazione k-NN si dimostra migliore di 1-NN, e **aumentare di  $k$** , l'errore  $P$  converge all'errore Bayesiano.

k Piccolo (es.  $k=1,3$ ): Varianza Alta / Bias Basso: Il modello è molto sensibile al rumore locale e agli outlier, ma mantiene alta la capacità di discriminare dettagli fini nella frontiera decisionale.

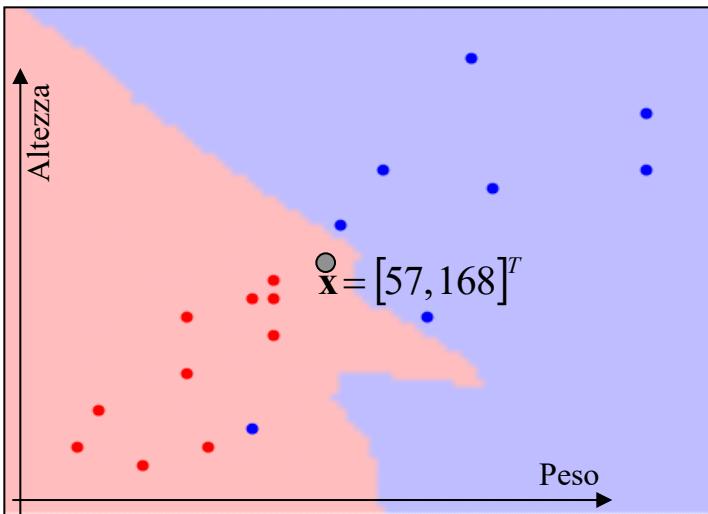
k Grande (es.  $k=15,20$ ): Varianza Bassa / Bias Alto: La decisione è molto stabile e robusta (tante votazioni mitigano il rumore), ma la stima diventa "sfocata" e ignora la struttura locale dei dati. La frontiera decisionale diventa più generica e semplice (rischio di underfitting).



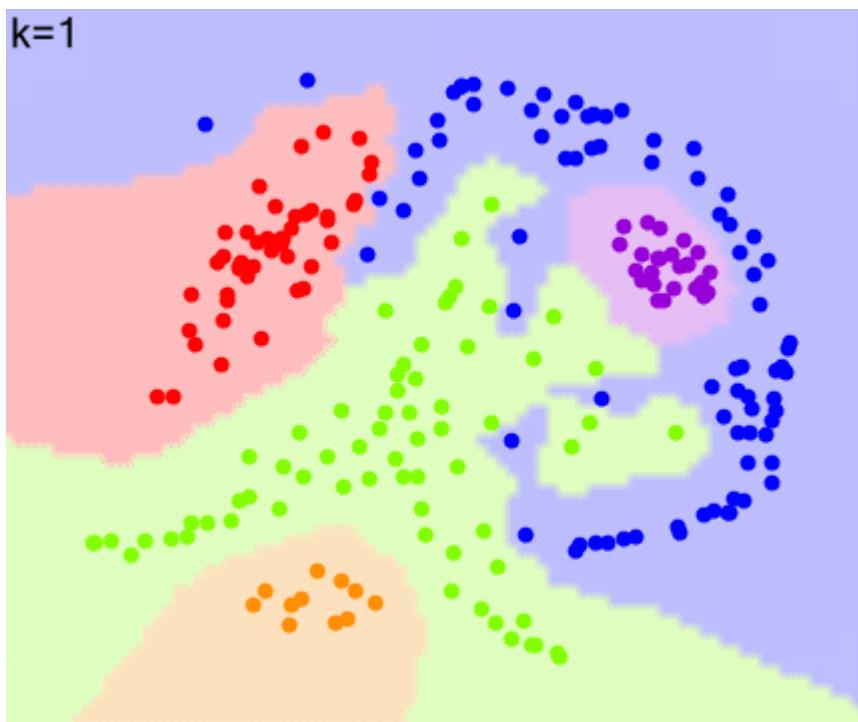
- Nella pratica (TS limitati), **aumentare  $k$**  significa estendere l'ipersfera di ricerca andando a sondare la probabilità a posteriori **lontano dal punto di interesse**; il valore ottimale di  $k$  (solitamente  $< 10$ ) deve essere determinato su un validation set separato.

Il valore ottimale di  $k$  deve essere scelto in modo da bilanciare la robustezza (ottenuta con  $k$  grande) e la capacità discriminativa (ottenuta con  $k$  piccolo).

# Esempi k-NN

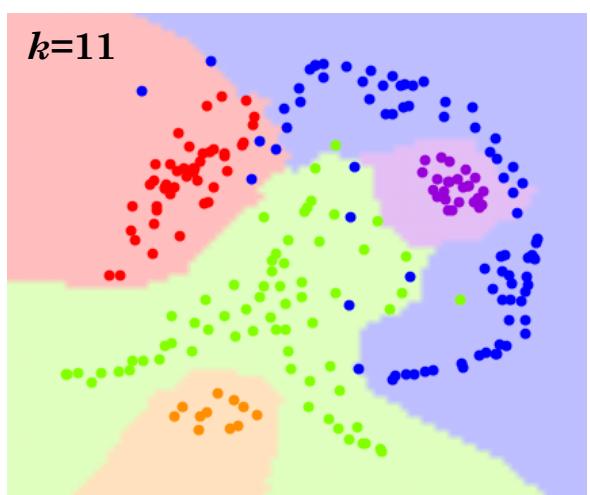
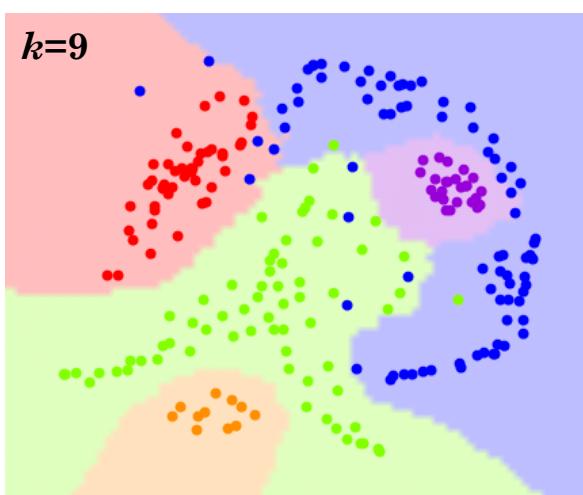
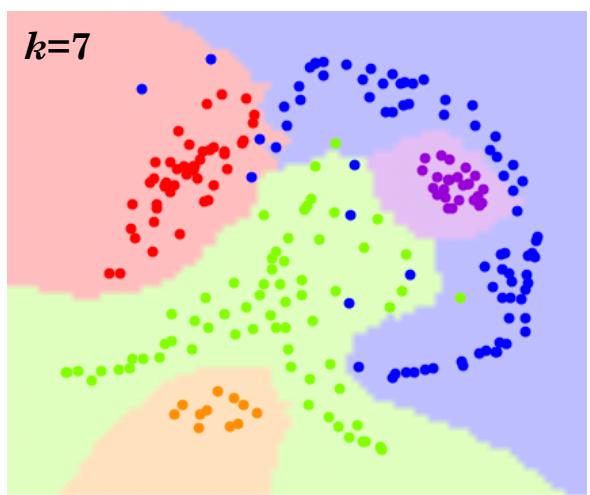
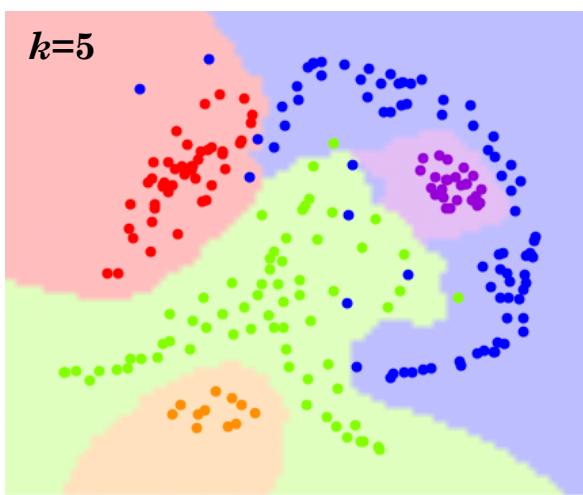
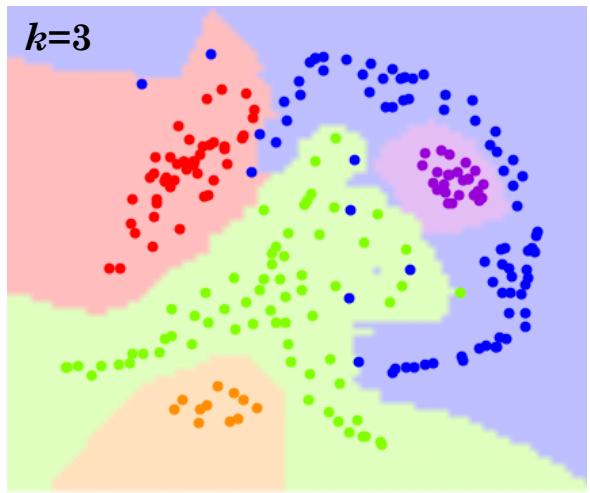
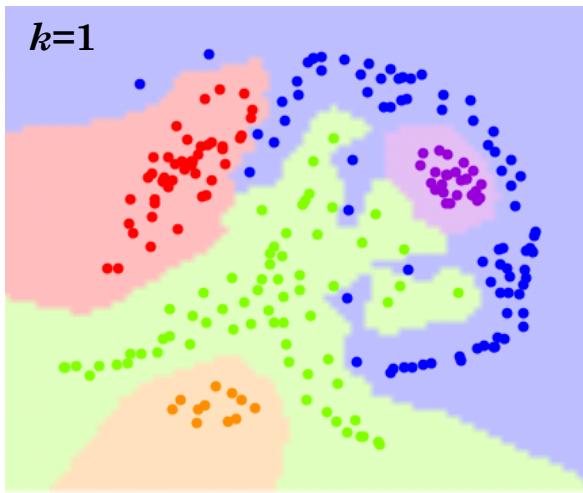


- Nell'esempio visto in precedenza, la regola k-NN con  $k=3$  assegna il pattern  $x$  alla classe  $w_2$  (femmine - rossi)
- L'animazione (scomposta nel lucido successivo) mostra il partizionamento dello spazio operato dalla regola k-NN sul training set con 5 classi visto in precedenza al variare di  $k$



# Espansione dell'animazione

## lucido precedente ( $k=1,3,5,7,9,11$ )



# **k-NN e Confidenza di Classificazione**

Da un classificatore  $k$ -NN risulta piuttosto semplice estrarre una **confidenza** (probabilistica) circa la classificazione eseguita; siano

Grazie alla sua natura basata sul voto a maggioranza, è molto semplice estrarre una confidenza probabilistica sulla decisione di classificazione.

$[v_1, v_2 \dots v_s]$ ,  
conteggi dei voti  
ottenuti dalle rispettive  
classi tra i  $k$  vicini

$$\sum_{i=1}^s v_i = k$$

$s =$  numero di classi

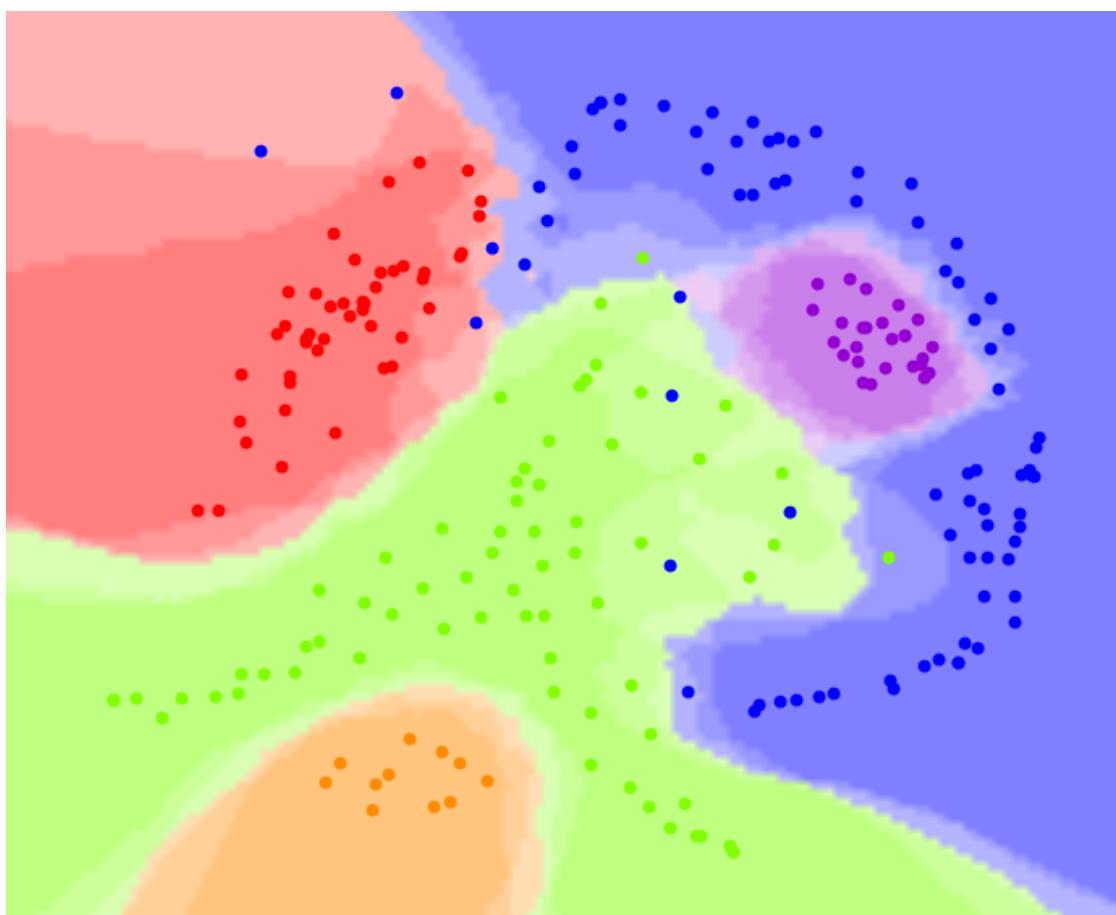
La somma totale dei voti deve essere pari al numero di vicini

i voti ottenuti dal pattern  $x$ , allora le confidenze (vedi sfumature in figura sotto) possono essere semplicemente ottenute dividendo per  $k$  i voti ottenuti:

- Regioni di Colore Pieno: Sono le zone in cui la confidenza di appartenenza a una singola classe è molto alta (ad esempio, la classe Rossa ha ottenuto 5 voti su  $k=5$ , quindi la confidenza è  $5/5=1.0$ ). Queste sono le regioni in cui il modello è molto sicuro.

$$\left[ \frac{v_1}{k}, \frac{v_2}{k} \dots \frac{v_s}{k} \right] \quad \text{Vettore di confidenze}$$

- Sfumature/Regioni Intermedie: Sono le zone vicine alle frontiere decisionali (i confini tra i colori). In queste zone, i  $k$  vicini di  $x$  appartengono a classi diverse, e i voti sono più distribuiti.



Il  $k$ -NN usa la proporzione dei voti per fornire una stima pragmatica della probabilità a posteriori.

# NN e complessità computazionale

Il k-NN è un algoritmo lazy. Non apprende un modello compatto durante la fase di training, ma rimanda tutto il lavoro alla fase di classificazione (inferenza).

L'utilizzo di un classificatore NN o k-NN nel caso di training set di elevate dimensioni può diventare problematico:

- Necessario memorizzare tutti i pattern del Training Set
- Per ogni classificazione è necessario calcolare la distanza del pattern da classificare da tutti i pattern del training set e ordinare (parzialmente le distanze) per ottenere le più piccole

Tecniche di editing/condensing (lucido successivo) possono alleviare questo problema, ma quando l'efficienza è importante è consigliabile indicizzare i dati attraverso strutture spaziali (es. **kd-tree**) che consentono di individuare i vicini senza effettuare una scansione esaustiva.

La libreria **FLANN** (C++) consente di effettuare ricerche nearest neighbor approssimate molto efficientemente.

<http://www.cs.ubc.ca/research/flann/>

→ non garantiscono di trovare esattamente i  $k$  vicini più vicini, ma trovano vicini molto prossimi in un tempo significativamente inferiore.

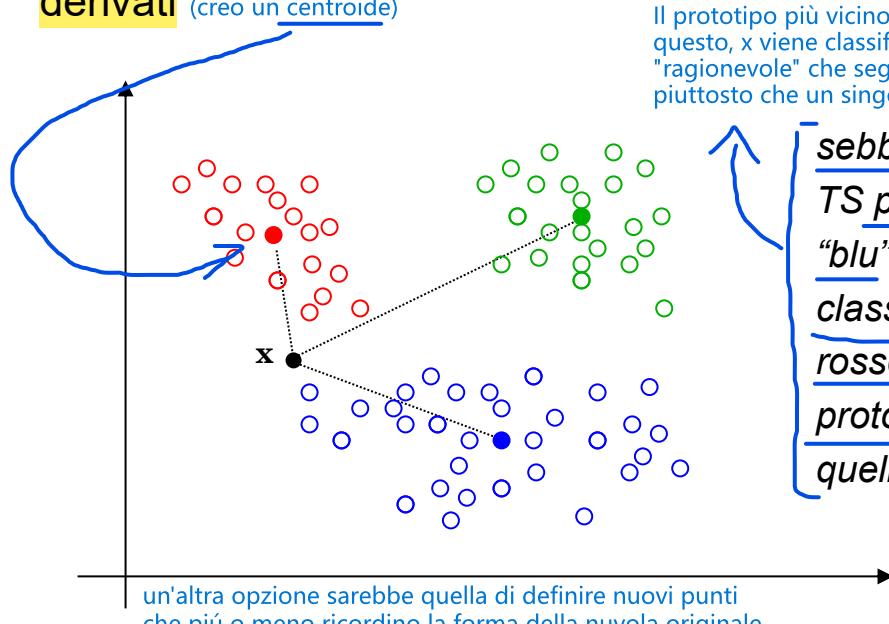
Invece di eseguire una scansione esaustiva ( $O(N)$ ), si usano strutture dati ottimizzate per la ricerca in spazi multidimensionali.

k-d Tree: (dove  $k$  sta per dimensionalità, non per i vicini) È un albero binario che suddivide ricorsivamente lo spazio delle feature. Questo consente all'algoritmo di saltare intere regioni dello spazio che è molto improbabile contengano i  $k$  vicini più prossimi, riducendo la complessità di ricerca a  $O(\log N)$  (o quasi) nella pratica, anziché  $O(N)$ .

# NN e Prototipi di Classi

Talvolta nella classificazione nearest neighbor invece di mantenere tutti i pattern del TS e calcolare la distanza da ciascuno di essi, si preferisce selezionare/derivare da essi uno (o più) **prototipi** per ciascuna classe e utilizzare questi ultimi per la classificazione come se fossero i soli elementi di TS: queste tecniche prendono il nome di:

- ➡ Obiettivo: Rimuovere gli outlier per rendere il k-NN più robusto e le frontiere decisionali più regolari.
- **editing**: quando si cancellano solo pattern dal training set, senza derivare nuovi pattern (di solito, vengono cancellati gli outlier o pattern ridondanti per quella classe)
- ➡ Obiettivo: Ridurre drasticamente la dimensione del TS mantenendo un'accuratezza simile.  
**condensing**: se i prototipi non appartenevano al TS e sono stati derivati (creo un centroide)



Il prototipo più vicino a  $x$  è quello di classe Rossa. Per questo,  $x$  viene classificato come Rosso, una decisione più "ragionevole" che segue la densità generale dei dati piuttosto che un singolo vicino anomalo.

sebbene il pattern di TS più vicino a  $x$  sia "blu",  $x$  viene classificato come rosso, in quanto il prototipo più vicino è quello rosso.

un'altra opzione sarebbe quella di definire nuovi punti che più o meno ricordino la forma della nuvola originale

Ciò comporta solitamente i **seguenti vantaggi**:

- non è necessario calcolare un elevato numero di distanze.
- i prototipi sono spesso più affidabili e robusti di singoli pattern (si riduce il rischio di affidarsi ad outlier).

Un singolo prototipo di classe può essere **derivato** come vettore medio dei vettori di quella classe nel TS. Processi di **vector quantization** o **clustering** consentono di ottenere più prototipi per ogni classe.

# NN e Metriche

e le performance

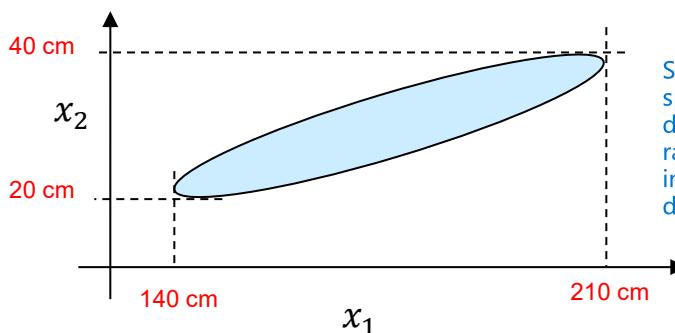
- Il comportamento della regola **k-NN** è strettamente legato alla metrica (funzione distanza) adottata.
- La **distanza euclidea**, che rappresenta il caso  $L_2$  nella definizione di metriche di Minkowski, è sicuramente la metrica più spesso utilizzata.

$$L_k(\mathbf{a}, \mathbf{b}) = \left( \sum_{i=1}^d |a_i - b_i|^k \right)^{1/k}$$

$k=2$  (Distanza L2):  
Distanza Euclidea.

- **Nella pratica**, prima di adottare semplicemente la distanza euclidea è bene **valutare** lo spazio di variazione delle componenti (**o feature**) e la presenza di eventuali forti correlazioni tra le stesse.
- Supponiamo ad esempio di voler classificare le persone sulla base dell'altezza e della lunghezza del piede. Ogni pattern  $\mathbf{x}$  (bidimensionale) risulta costituito da due feature ( $x_1$  = altezza,  $x_2$  = lunghezza del piede).

Una piccola variazione nell'Altezza (che può arrivare fino a 70 cm) ha un impatto molto maggiore sul totale della distanza Euclidea rispetto alla stessa variazione relativa nella Lunghezza del Piede.



Se le diverse features hanno spazi di variazione (range) molto diversi, la componente con il range maggiore peserà inevitabilmente di più nel calcolo della distanza Euclidea.

Lo spazio di variazione dell'altezza ( $210-140 = 70$  cm) risulta maggiore di quello della lunghezza del piede ( $40-20=20$  cm). Pertanto se la similarità tra pattern venisse misurata con semplice distanza euclidea la componente altezza "peserebbe" più della componente lunghezza del piede.

La normalizzazione è il processo di trasformazione delle feature (variabili indipendenti) di un dataset in modo che rientrino in un intervallo standardizzato. L'obiettivo principale è evitare che le feature con intervalli di valori più ampi dominino il calcolo delle distanze o i pesi del modello.

# Normalizzazione

Per evitare i problemi legati a diversi spazi di variazioni delle feature, particolarmente fastidiosi per alcune tecniche (es. reti neurali), si consiglia di normalizzare i pattern.

Le normalizzazioni più comuni sono:

- **Min-Max scaling:** per ogni feature  $i$  – esima si calcolano il massimo  $\max_i$  e il minimo  $\min_i$  e si applica una trasformazione lineare (scaling) che «tipicamente» mappa  $\min_i$  a 0 e  $\max_i$  a 1.

Mantiene la forma originale della distribuzione e non riduce l'importanza degli outlier (i quali però possono distorcere molto il range [0,1])

$$x' = (x - \min_i) / (\max_i - \min_i)$$

Riscallo tutto tra 0 e 1

- **Standardization:** per ogni feature  $i$  – esima si calcola la media  $\text{mean}_i$  e la deviazione standard  $\text{stddev}_i$  e si trasformano i valori come:

Meno sensibile agli outlier rispetto al Min-Max Scaling, poiché gli outlier non alterano drasticamente il range come fanno min e max.

$$(Z-Score) \quad x' = (x - \text{mean}_i) / \text{stddev}_i$$

Deviazione Standard

Dopo la trasformazione tutte le feature hanno (sul training set) media 0 e deviazione standard 1.

Attenzione: i parametri della normalizzazione (es. minimi, massimi) si calcolano **sul solo training set** e la trasformazione si applica sia a tutti i dati (training, validation, test). Nel caso di **k-fold cross-validation** la normalizzazione dovrebbe essere ripetuta k volte (uso di *pipeline* in scikit-learn).

Se calcolassimo i parametri (come la media) usando anche i dati di test, staremmo introducendo una "perdita" di informazione sul set di test nel nostro modello. Staremmo violando l'assunzione che il set di test sia una vera rappresentazione di dati sconosciuti.

Le semplici tecniche sopra descritte operano sulle singole feature indipendentemente. Una tecnica di normalizzazione efficace (ma più costosa) che opera simultaneamente su tutte le feature tenendo conto della loro correlazione è la **Whitening transform** (o **PCA whitening**). Molto in voga prima del deep-learning, oramai poco utilizzata.

Può migliorare significativamente le prestazioni di alcuni modelli rendendo lo spazio delle feature "più sferico" (da qui il nome whitening), ma è più costosa dal punto di vista computazionale.

ML

36

Trasforma l'ellissoide della distribuzione originale in una sfera con assi paralleli a quelli cartesiani. Decorrelazione: Rimuove le correlazioni tra le feature (gli assi dell'ellissoide vengono allineati agli assi cartesiani). Varianza Unitaria: Rende la varianza di ogni nuova componente pari a 1 (diventando una sfera).

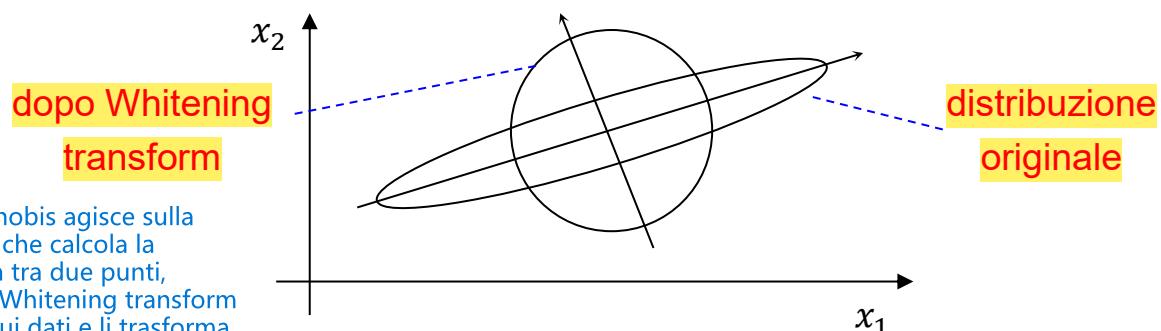
La Whitening Transform e la Distanza di Mahalanobis mirano a eliminare gli effetti negativi della correlazione tra le feature.

## Whitening transform

Un'efficace normalizzazione rispetto agli spazi di variazione, in grado anche di tener conto delle correlazioni tra feature è possibile:

- pre-normalizzando lo spazio delle feature attraverso Whitening transform (che vedremo meglio in seguito)
- ➡ utilizzando come metrica la *distanza di Mahalanobis*.

Le due alternative sono equivalenti. Nel primo caso l'ellissoide corrispondente allo spazio delle feature viene "sferizzato" a priori e viene in seguito usata la distanza euclidea; nel secondo la distanza di Mahalanobis normalizza ogni componente sulla base della matrice di covarianza  $\Sigma$ .



Mahalanobis agisce sulla metrica che calcola la distanza tra due punti, mentre Whitening transform agisce sui dati e li trasforma

Da non sottovalutare l'importanza della correlazione tra features come **aspetto negativo per la classificazione**. Infatti, l'utilizzo di feature correlate **riduce** (anche drasticamente) il potere discriminante. Nel caso **ideale** tutte le feature sono staticamente indipendenti (ellissoide assi paralleli a quelli cartesiani).

Due feature altamente discriminanti se prese *individualmente*, ma tra loro fortemente correlate, sono nel complesso meno discriminanti di una terza feature leggermente più di discriminante di ognuna delle precedenti.

La distanza di Mahalanobis (o la sferizzazione dello spazio) **tiene conto delle correlazioni** e pesa maggiormente feature non correlate.

L'obiettivo del Metric Learning è trovare una trasformazione ottimale degli input in un nuovo spazio dove la distanza Euclidea standard sia più significativa per il compito di classificazione.

Distance Metric Learning

# Metric Learning

Il Metric Learning è un approccio di Machine Learning Supervisionato il cui obiettivo è imparare la funzione di distanza ottimale direttamente dai dati di training.

Un approccio più generale alla scelta della metrica da utilizzare in una determinata applicazione, consiste nel learning supervisionato della metrica stessa dai dati del training set.

Obiettivo è determinare una trasformazione degli input che:

- «avvicini» pattern della stessa classe
- «allontani» pattern di classi diverse

La distanza euclidea nella spazio originale è:

$$dist(a, b) = \sqrt{(\mathbf{a} - \mathbf{b})^t (\mathbf{a} - \mathbf{b})} = \|\mathbf{a} - \mathbf{b}\|_2$$

Versione Vettoriale della somma di Minkowski per la distanza euclidea

Un tipico approccio di metric learning **lineare** determina (con training supervisionato) una matrice  $\mathbf{G}$  che trasforma gli input, e continuare ad applicare la distanza euclidea agli input trasformati

La matrice  $\mathbf{G}$  viene appresa durante il training supervisionato

La matrice  $\mathbf{G}$  impara a pesare e a decorrelare le feature.

$$dist(a, b) = \|\mathbf{G}\mathbf{a} - \mathbf{G}\mathbf{b}\|_2$$

La matrice  $\mathbf{M}$  (chiamata matrice metrica) racchiude tutte le correlazioni e i pesi ottimali che la trasformazione ha appreso per massimizzare la separazione delle classi.

$$\mathbf{M} = \mathbf{G} \cdot \mathbf{T}^T \cdot \mathbf{G}$$

Vedremo una possibile soluzione di questo problema nell'ambito della riduzione di dimensionalità con **LDA** (Linear Discriminant Analysis).

Sono anche possibili approcci non lineari:

$$dist(a, b) = \|\mathbf{G}\phi(\mathbf{a}) - \mathbf{G}\phi(\mathbf{b})\|_2$$

dove  $\phi$  è una funzione non lineare. che mappa i dati originali in uno spazio ad alta dimensionalità

In questo spazio ad alta dimensione, la separazione lineare (data dalla matrice  $\mathbf{G}$ ) può essere trovata anche se la separazione nel dominio originale non lo era (questo è simile al kernel trick usato nelle SVM). 

La distanza coseno indica quanto due vettori differiscono in termini di orientamento, dove 0 indica massima somiglianza e valori più alti indicano maggiore differenza

# Similarità Coseno e Distanza Coseno

Una similarità/distanza piuttosto utilizzata in applicazioni di **information retrieval**, **data mining** e **text mining** è la similarità/distanza **coseno**.

Geometricamente, dati due vettori  $\mathbf{a}$  e  $\mathbf{b}$  la similarità coseno corrisponde al coseno dell'angolo tra di essi:

$$\text{CosSimilarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^t \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}$$

è noto infatti che il prodotto scalare tra due vettori è:

$$\mathbf{a}^t \cdot \mathbf{b} = \|\mathbf{a}\| \cdot \|\mathbf{b}\| \cdot \cos(\theta)$$

Similarità 0: i vettori sono ortogonali

I vettori sono identici in direzione

Due vettori identici hanno similarità 1 e due vettori opposti -1.

La distanza coseno è semplicemente:

$$\text{CosDistance}(\mathbf{a}, \mathbf{b}) = 1 - \text{CosSimilarity}(\mathbf{a}, \mathbf{b})$$

La Similarità Coseno ignora le differenze di lunghezza e valuta solo se la proporzione e l'orientamento dei termini sono simili. Se lo sono, l'angolo è piccolo e la similarità è alta, indicando che i documenti trattano lo stesso argomento.

**Esempio Confronto di testi:** Un testo può essere codificato da un vettore numerico dove ogni dimensione contiene il numero di occorrenze di una certa parola rispetto a un dato dizionario. La similarità di contenuto tra due testi non dipende dal numero assoluto di parole ma dalla frequenza relativa di ciascuna di esse.

La distanza coseno «sconta» la lunghezza dei vettori.

Due testi sul calcio, se uno è più lungo dell'altro, è naturale che contenga più frequentemente termini calcistici, anche se non lo è, per questo bisogna scontare la lunghezza di tale testo

La distanza coseno non è una metrica (es. non rispetta la diseguaglianza triangolare). Se si è interessati a una metrica si può passare alla distanza angolare:

$$\text{AngularDistance}(\mathbf{a}, \mathbf{b}) = \frac{\arccos(\text{CosSimilarity}(\mathbf{a}, \mathbf{b}))}{\pi}$$