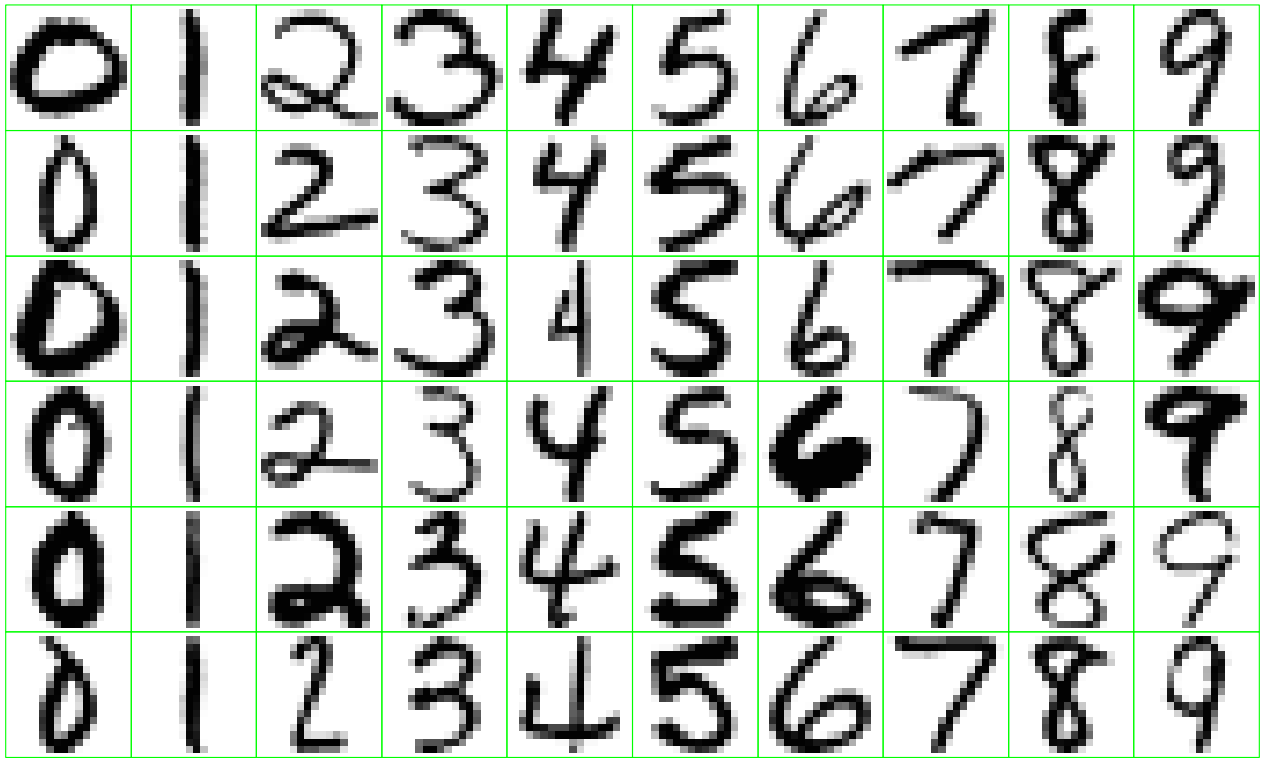
IN550 Machine Learning

Introduzione a modelli e metodi di classificazione

Vincenzo Bonifaci

Esempio: Riconoscimento di cifre scritte a mano

Input: immagine 28×28 in scala di grigi
Output: la cifra decimale (0–9) rappresentata dall'immagine



Dataset MNIST: 60,000 (training set) + 10,000 (test set) immagini etichettate

Problemi di predizione: input e output

- Spazio degli input $\mathcal{X}$
  Es.: insieme delle possibili immagini 28×28
- Spazio degli output $\mathcal{Y}$
  Es.: $\{0, 1, 2, \ldots, 9\}$

Dopo aver visto un certo numero di esempi (x, y), vogliamo trovare una *regola di predizione* (o *ipotesi*)

$$h : \mathcal{X} \to \mathcal{Y}$$

che ricostruisca in maniera accurata la relazione ingresso-uscita

Nei problemi di *regressione* l'output è quantitativo

Nei problemi di *classificazione* l'output è qualitativo

Funzioni di costo [loss functions]

Come quantificare l'accuratezza di una regola di predizione $h : \mathcal{X} \to \mathcal{Y}$ su un particolare esempio?

Una *funzione di costo* $\ell : \mathcal{Y} \times \mathcal{Y}_0 \to \mathbb{R}$ riceva la predizione $\hat{y} = h(x)$ e l'etichetta corretta y, e restituisce un reale nonnegativo

$$\ell(\hat{y}, y) \in \mathbb{R}_+$$

Una funzione di costo per la classificazione

- *Funzione costo 0-1*:

$$\ell(\hat{y}, y) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{se } \hat{y} = y \\ 1 & \text{se } \hat{y} \neq y \end{cases}$$

Il *rischio empirico* diventa la frazione di esempi di training non correttamente classificati:

$$\text{RE}_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(\hat{y}^{(i)}, y^{(i)}) = \frac{|\{i \in S : \hat{y}^{(i)} \neq y^{(i)}\}|}{|S|}$$

Il *rischio atteso* diventa la probabilità che un nuovo esempio non sia correttamente classificato (*inaccuratezza* del classificatore):

$$\text{RA}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(\hat{y}, y)] = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$$

Classificazione Nearest-Neighbor

Classificazione Nearest Neighbor

Immagini di training $x^{(1)}, x^{(2)}, \ldots, x^{(60000)}$
Etichette $y^{(1)}, y^{(2)}, \ldots, y^{(60000)}$ (numeri nel range 0–9)

Come classifichiamo una nuova immagine x?
Approccio Nearest Neighbor:

- Trova l'esempio più "simile" ad x tra gli $x^{(i)}$
- Restituisci la corrispondente etichetta

Lo spazio dei dati

Come misuriamo la distanza tra immagini?

- Dimensioni 28×28 (784 pixel totali)
- Ogni pixel è in scala di grigi: 0–255

Un vettore 784-dimensionale per ogni immagine

- Spazio degli input $\mathcal{X} = \mathbb{R}^{784}$
- Spazio degli output (e delle etichette) $\mathcal{Y} = \{0, 1, \ldots, 9\}$

La distanza euclidea tra x e x' è $\|x - x'\| = \sqrt{\sum_k (x_k - x'_k)^2}$

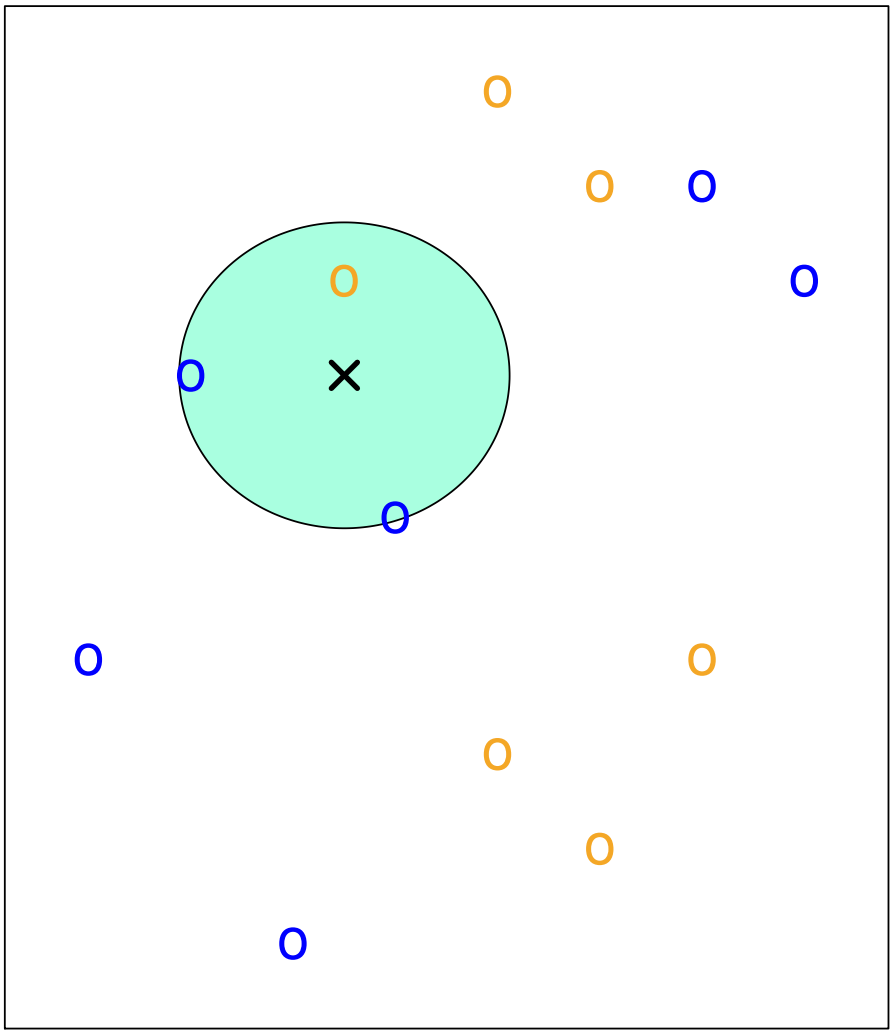
Classificazione K-Nearest Neighbor (K-NN)

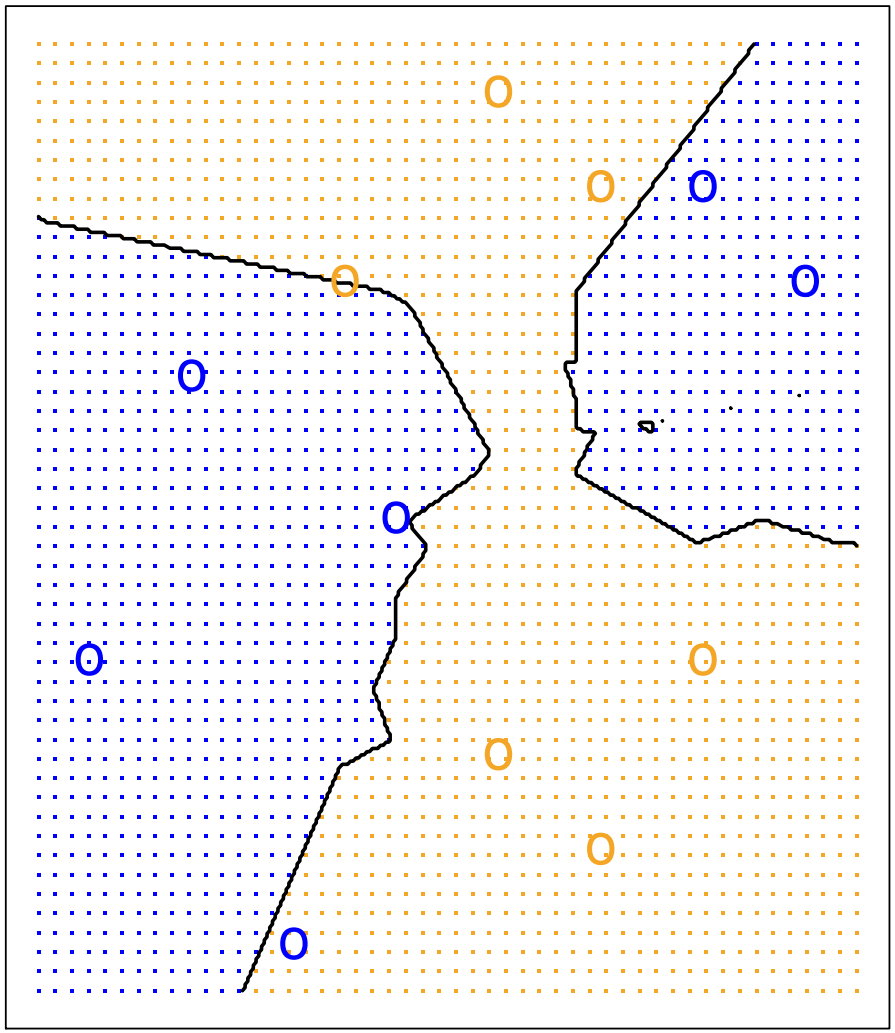
Sia $K \geq 1$ e sia x il punto di cui si vuole stimare l'etichetta

1. Identifica i K esempi $x^{(1)}, \ldots, x^{(K)}$ più vicini ad x (in termini di distanza euclidea)
2. Restituisci l'etichetta più frequente per quegli esempi (la *moda*): $h(x) = \text{argmax}_{y \in \mathcal{Y}} |\{i = 1, \ldots, K \, : \, y^{(i)} = y\}|$

Quando $|\mathcal{Y}| = 2$, l'ultimo passo equivale a restituire l'etichetta di maggioranza

K-NN: Esempio ($K = 3$)





Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo

Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo
- Il rischio atteso stimato (errore di test) di 1-NN è 3.08%

Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo
- Il rischio atteso stimato (errore di test) di 1-NN è 3.08%
- Rischio atteso di un classificatore totalmente aleatorio?

Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo
- Il rischio atteso stimato (errore di test) di 1-NN è 3.08%
- Rischio atteso di un classificatore totalmente aleatorio? 90%

Accuratezza di NN per il dataset MNIST

Applicando 1-NN al dataset MNIST si osserva quanto segue:

- Il rischio empirico (errore di training) di 1-NN è nullo
- Il rischio atteso stimato (errore di test) di 1-NN è 3.08%
- Rischio atteso di un classificatore totalmente aleatorio? 90%

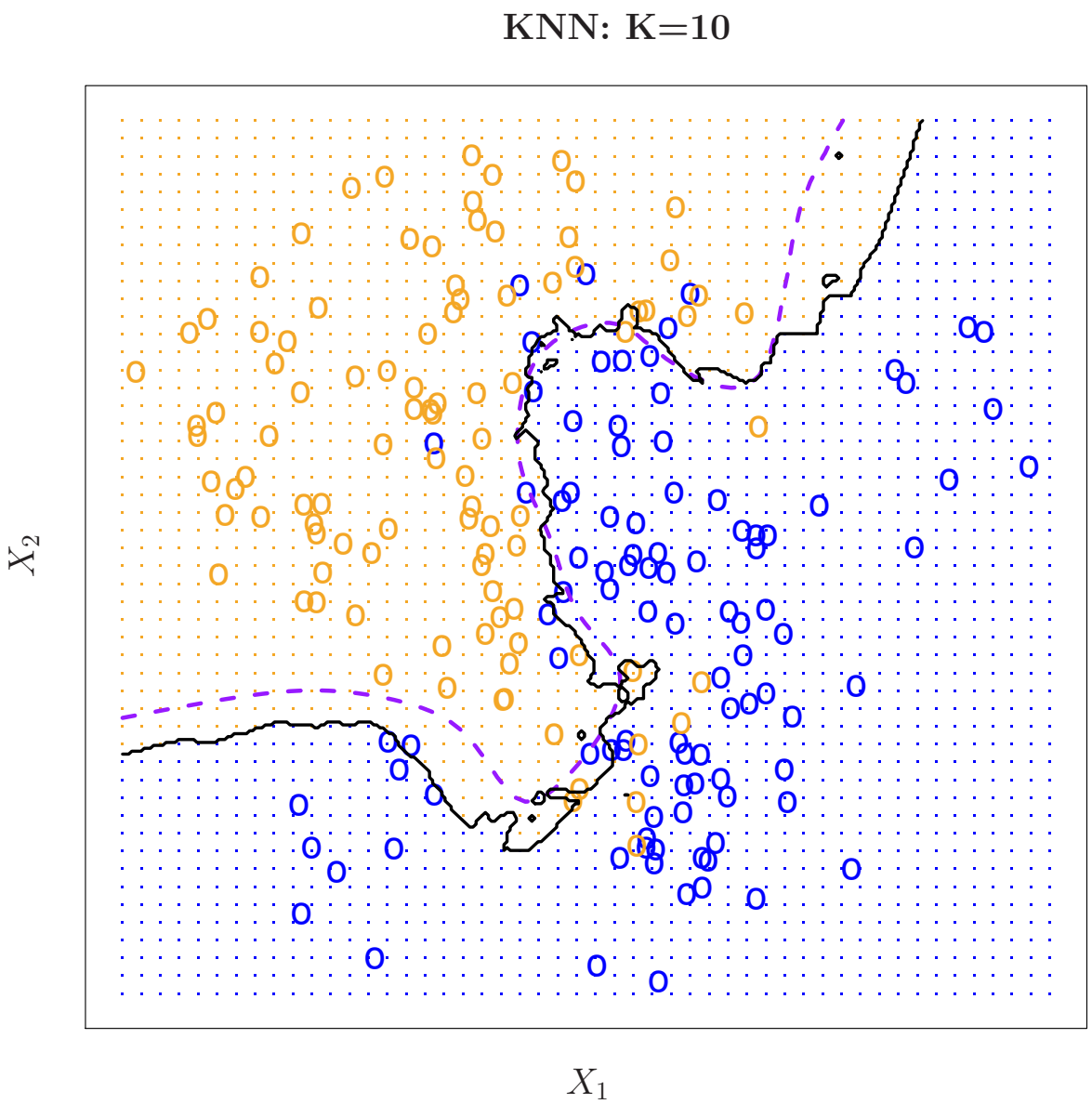
Esempi di errori:

Migliorare l'accuratezza di K-NN: scelta di K

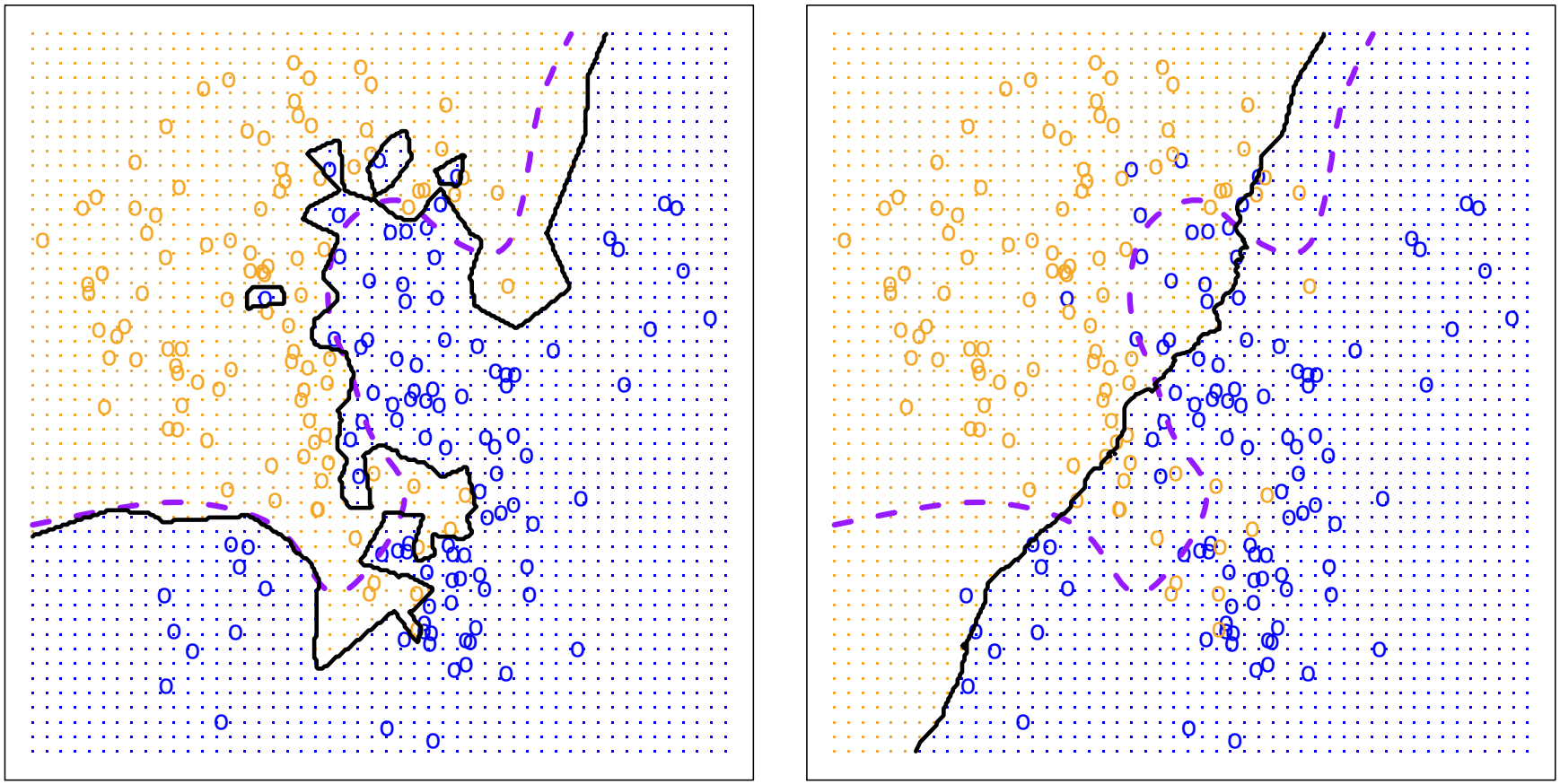
Cosa succede variando K?

K	1	3	5	7	9	11
Errore sul validation set	3.09%	2.94%	3.13%	3.10%	3.43%	3.34%

Effetto della variazione di K

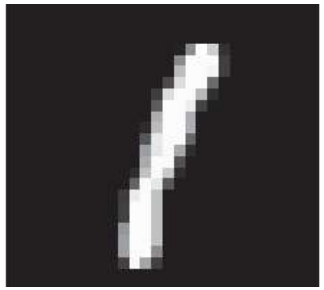
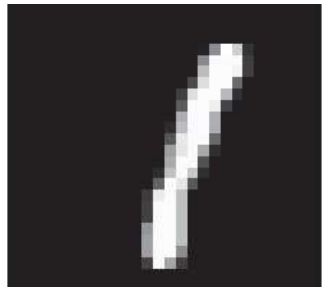


Effetto della variazione di K



Migliorare l'accuratezza di K-NN: la funzione distanza

La distanza euclidea (ℓ_2) tra queste due immagini è molto alta!

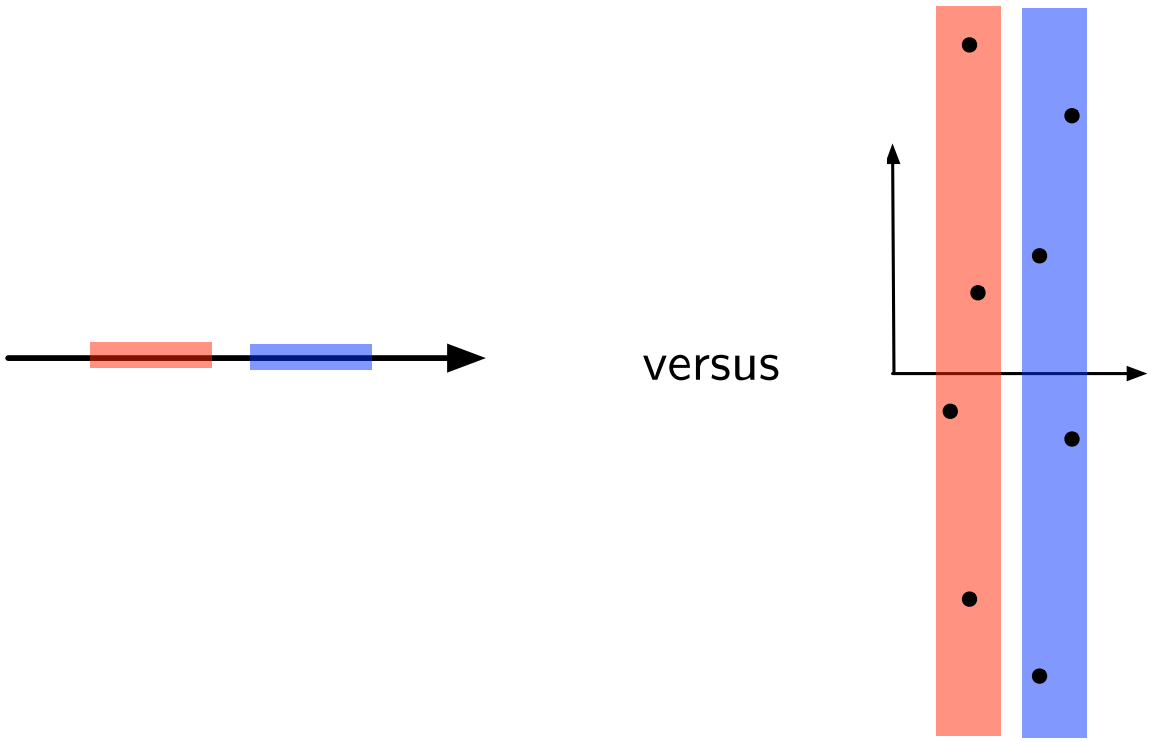
Idea migliore: usare funzioni distanza *invarianti* rispetto a:

- Piccole traslazioni e rotazioni: es. *tangent distance*
- Una classe più ampia di deformazioni naturali: es. *shape context*

Distanza	ℓ_2	tangent distance	shape context
Errore sul validation set	3.09%	1.10%	0.63%

K-NN: L'impatto di variabili rumorose

Una buona *feature selection* è essenziale prima di applicare NN:
anche solo una variabile poco significativa può avere effetti deleteri!



K-NN: Velocizzare la ricerca

Ricerca naïf dei K punti più vicini richiede tempo $m \cdot d$ per un dataset di taglia m su d variabili: lenta!

Esistono *strutture dati* che, preprocessando i dati, velocizzano la ricerca:

- Locality sensitive hashing
- Ball trees
- K-d trees

Spesso supportate dalle librerie di Machine Learning

Per esempio, `scikit-learn` offre le strutture `KDTree` e `BallTree`

Accuratezza di NN per la classificazione binaria

Sia $\mathcal{X} = [0,1]^d$, $\mathcal{Y} = \{0,1\}$, $\mathcal{D}$ una distribuzione su $\mathcal{X} \times \mathcal{Y}$

Teorema (Prestazioni di 1-NN)

Sia h_S^{NN} l'ipotesi costruita da 1-NN sull'insieme di training S e sia h^* l'ipotesi che minimizza il rischio atteso $\mathrm{RA}(h)$. Allora

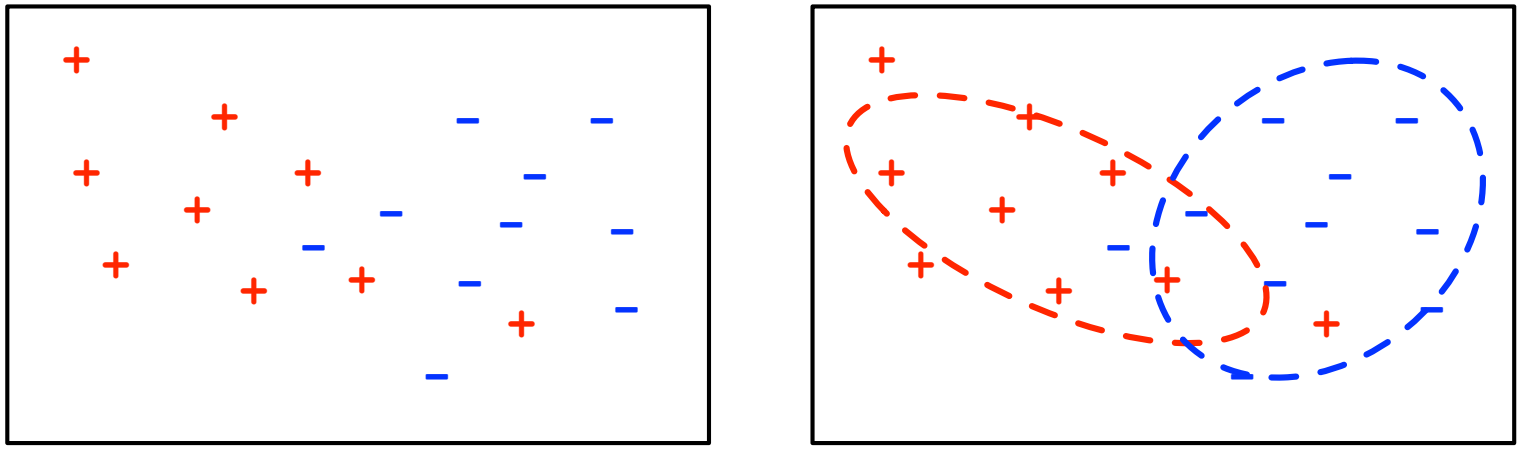
$$\mathbb{E}_S[\mathrm{RA}(h_S^{NN})] \leq 2 \cdot \mathrm{RA}(h^*) + c\sqrt{d}m^{-1/(d+1)}$$

dove c è una costante che dipende solo dalla distribuzione $\mathcal{D}$.

Quando $m \to \infty$, il secondo termine tende a zero e quindi il rischio atteso di 1-NN tende (al più) al doppio del rischio atteso minimo

Classificazione generativa

Approccio generativo alla classificazione



Durante l'apprendimento:

- Fai il fit di una distribuzione di probabilità per ciascuna classe

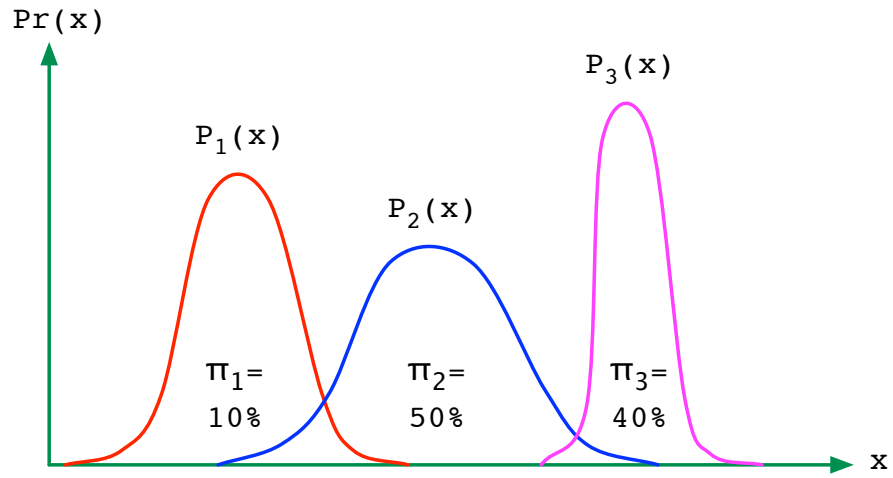
Per classificare un nuovo punto x:

- Determina da quale distribuzione di probabilità è più verosimile che il punto sia stato generato

Modelli generativi

Esempio:

- Spazio di input $\mathcal{X} = \mathbb{R}$
- Spazio di output $\mathcal{Y} = \{1, 2, 3\}$



Per ciascuna classe j, stimiamo:

- la probabilità a priori di quella classe, $\pi_j = \Pr(y = j)$
- la distribuzione degli input in quella classe, $P_j(x) = \Pr(x|y = j)$

Criterio bayesiano

Per classificare un nuovo x: scegli l'etichetta y che massimizza $\Pr(y|x)$

Per definizione di probabilità condizionata,

$$\Pr(y|x) = \frac{\overbrace{\Pr(y)}^{\text{prob. a priori della classe}} \cdot \overbrace{\Pr(x|y)}^{\text{prob. di } x \text{ nella classe } y}}{\underbrace{\Pr(x)}_{\text{non dipende da } y}}$$

Modello *generativo* perché implicitamente apprende la distribuzione congiunta $\Pr(x, y)$ ed quindi anche in grado di generare nuovi esempi (x, y) (più o meno plausibili)

Giustificazione del criterio bayesiano

Ricordiamo che la nostra funzione costo è:

$$\ell(\hat{y}, y) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{se } \hat{y} = y \\ 1 & \text{se } \hat{y} \neq y \end{cases}$$

Il rischio atteso *condizionato* all'osservazione di x è

$$\mathbb{E}[\ell|x] = \Pr[h(x) \neq y|x] = 1 - \Pr[h(x) = y|x]$$

e minimizzarlo equivale a scegliere $h(x) = \hat{y}$ dove $\hat{y}$ massimizza $\Pr(\hat{y}|x)$

Classificatore bayesiano

$$h(x) = \operatorname{argmax}_{\hat{y} \in \mathcal{Y}} \Pr(\hat{y}|x)$$

Analisi del discriminante

Per ogni $x \in \mathcal{X}$ e ogni etichetta $j \in \mathcal{Y}$,

$$\Pr(y = j|x) = \frac{\Pr(y = j) \cdot \Pr(x|y = j)}{\Pr(x)} = \frac{\pi_j P_j(x)}{\Pr(x)}$$

Il termine $\Pr(x)$ non dipende da j

Dato x, l'etichetta j più verosimile è quella che massimizza $\pi_j P_j(x)$

La quantità $\delta_j(x) \stackrel{\text{def}}{=} \log(\pi_j P_j(x))$ è chiamata *discriminante*

Dato x, l'etichetta j più verosimile è quella che massimizza $\delta_j(x)$

Fit di un modello generativo

Esempio: Classificazione di bottiglie di vino in base al tipo di vino (vitigno)

Training set: 130 bottiglie

- Tipo 1 (Nebbiolo): 43 bottiglie
- Tipo 2 (Grignolino): 54 bottiglie
- Tipo 3 (Barbera): 33 bottiglie
- Per ogni bottiglia, 13 feature: `Alcool`, `Acido malico`, `Ceneri`, `Alcalinità delle ceneri`, `Magnesio`, `Fenoli totali`, `Flavonoidi`, `Fenoli non flavonoidi`, `Proantocianina`, `Intensità di colore`, `Tonalità`, `OD280/OD315`, `Prolina`

Test set: 48 bottiglie

Pesi delle classi:

$\pi_1 = 43/130 \approx 0.33 \qquad \pi_2 = 54/130 \approx 0.41 \qquad \pi_3 = 33/130 \approx 0.26$

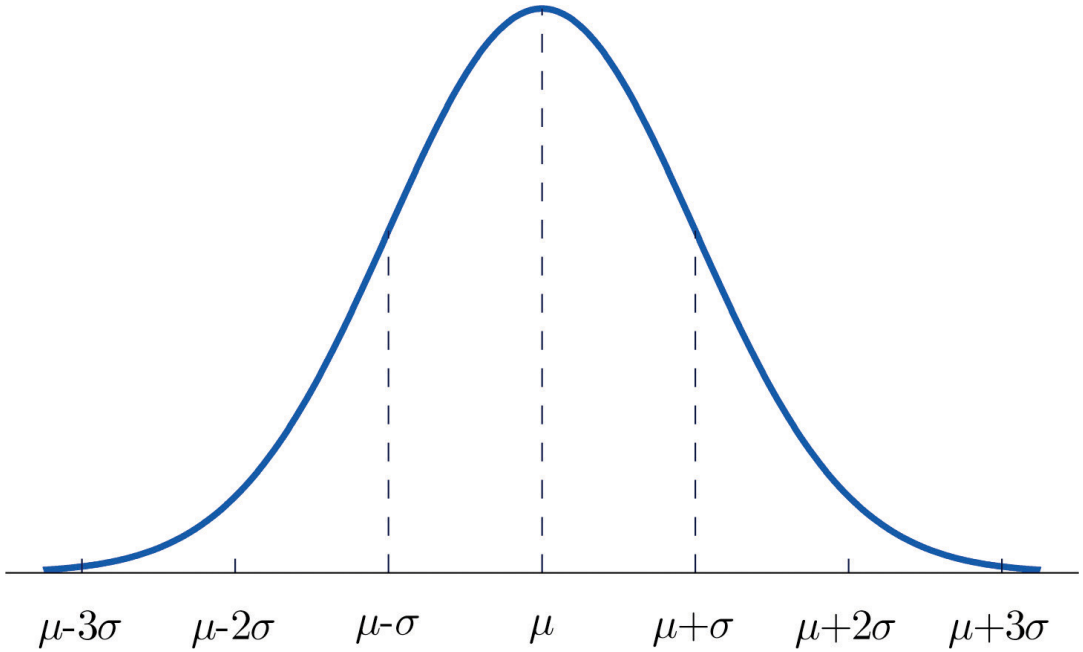
Cerchiamo le distribuzioni P_1, P_2, P_3 delle feature di ogni classe

Supponiamole *gaussiane* e per ora basiamole su un'unica feature: `Alcool`

Gaussiana univariata

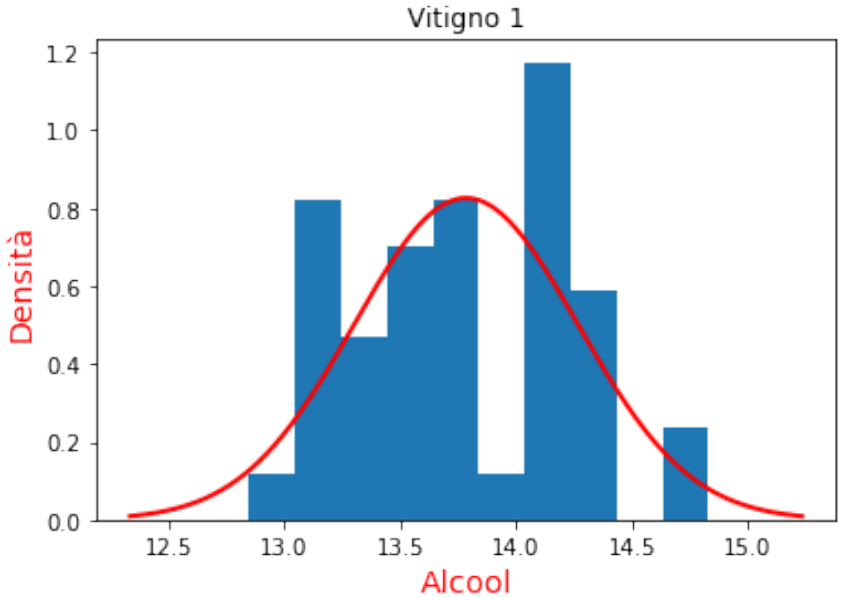
La gaussiana $N(\mu, \sigma^2)$ ha media μ, varianza σ^2, e densità di probabilità

$$p(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



Distribuzione per la classe 1

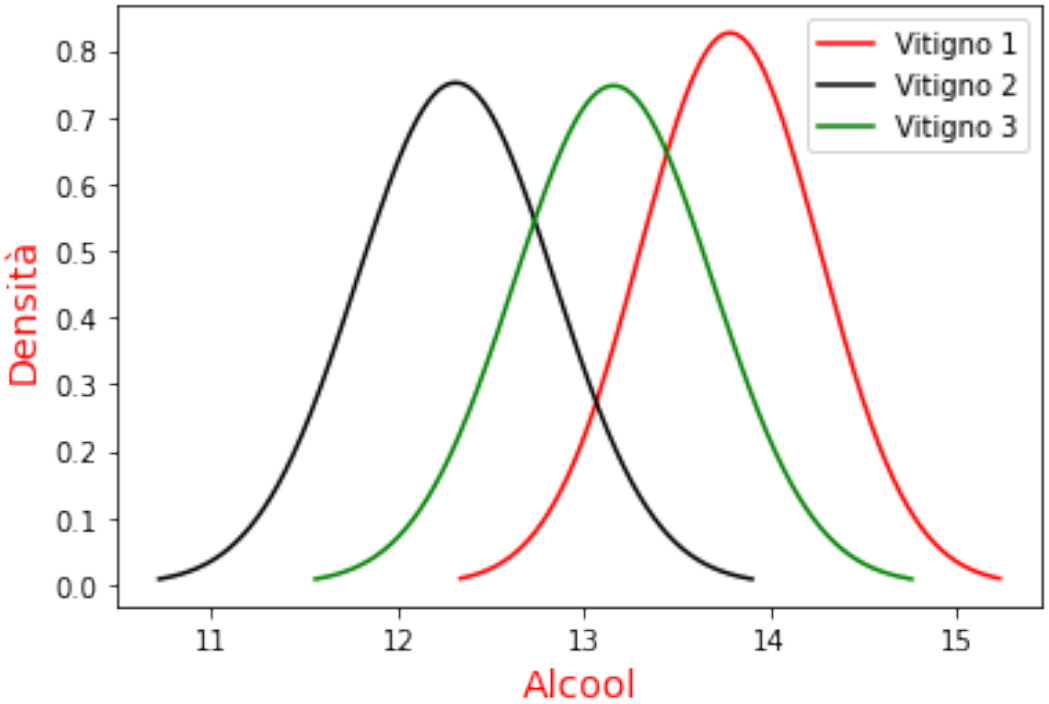
Unica feature che utilizziamo: `Alcool`



Media	$\mathbb{E}\, x$	Media stimata	$(1/m)\sum_i x^{(i)}$
Varianza	$\mathbb{E}\,(x-\mu)^2$	Var. stimata	$(1/m)\sum_i (x^{(i)}-\mu)^2$

Nell'esempio: media stimata $\mu \approx 13.78$, varianza stimata $\sigma^2 \approx 0.23$

Analisi del discriminante unidimensionale



$\pi_1 = 0.33,\ P_1 = N(13.78, 0.23)$

$\pi_2 = 0.41,\ P_2 = N(12.31, 0.28)$

$\pi_3 = 0.26,\ P_3 = N(13.15, 0.28)$

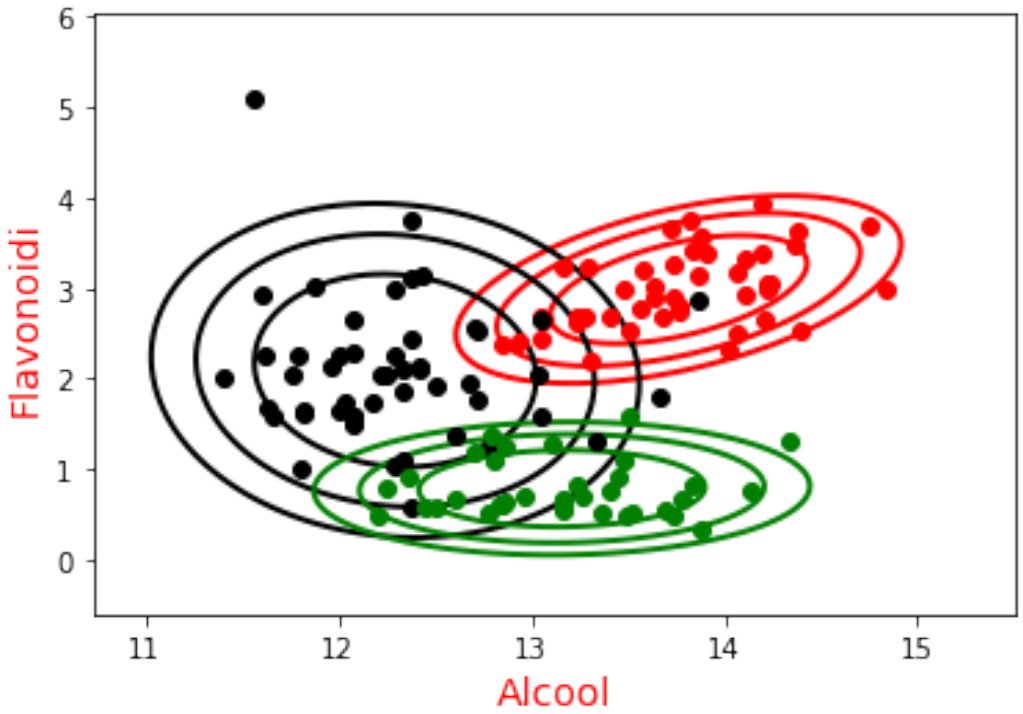
Per classificare x: determina l'etichetta j che massimizza $\pi_j P_j(x)$

Errore di test: $17/48 \approx 35\%$

Aggiunta di feature

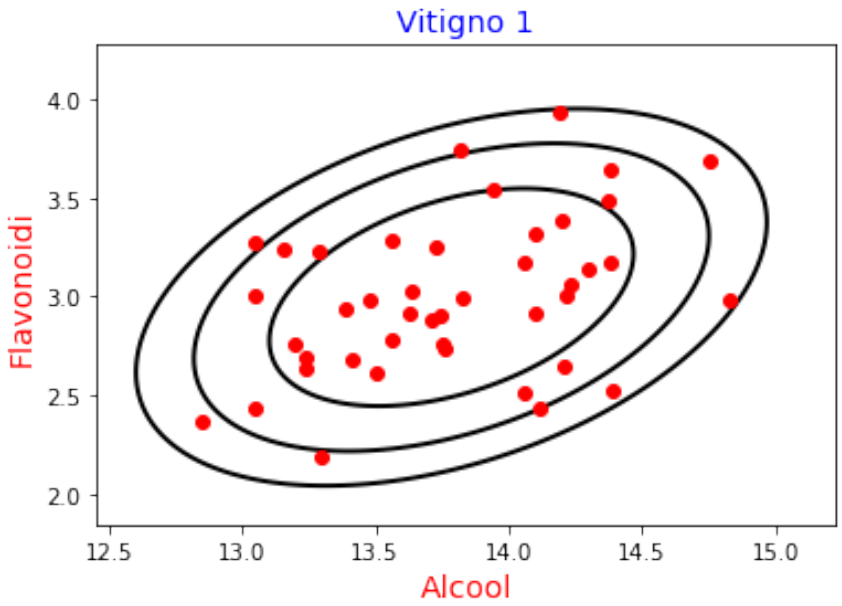
Più feature permettono una maggiore separazione tra le classi

Aggiungiamo la variabile `Flavonoidi`



Errore di test diventa $3/48 \approx 6\%$

Uso di una gaussiana bivariata

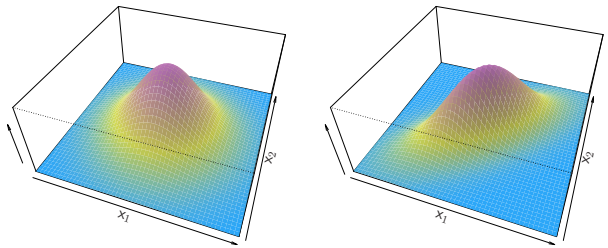


Modelliamo la classe 1 con una gaussiana bivariata:

media $\mu = \begin{pmatrix} 13.7 \\ 2.98 \end{pmatrix}$ matrice di covarianza $\Sigma = \begin{pmatrix} 0.22 & 0.09 \\ 0.09 & 0.17 \end{pmatrix}$

$$\mu_i = \mathbb{E}\, x_i$$
$$\Sigma_{ij} = \mathrm{Cov}(x_i, x_j) = \mathbb{E}[(x_i - \mu_i)(x_j - \mu_j)]$$

Densità della gaussiana bivariata



- Media $\mu = (\mu_1, \mu_2) \in \mathbb{R}^2$
- Matrice di covarianza $\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \in \mathbb{R}^{2\times 2}$

$$p(x) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

- $|\Sigma|$ qui indica il *determinante* di Σ

Gaussiana multivariata

Gaussiana in $\mathbb{R}^d$: $N(\mu, \Sigma)$

- media: $\mu \in \mathbb{R}^d$
- covarianza: $\Sigma \in \mathbb{R}^{d \times d}$
- μ è il vettore dei valori attesi:

$$\mu_1 = \mathbb{E}\, x_1, \mu_2 = \mathbb{E}\, x_2, \ldots, \mu_d = \mathbb{E}\, x_d$$

- Σ è la matrice di covarianza:

$$\Sigma_{ij} = \mathrm{Cov}(x_i, x_j)$$

$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

Analisi del discriminante quadratica (QDA)

1. Calcola le probabilità a priori π_j per ogni classe j
2. Fai il fit di una gaussiana multivariata $P_j(x)$ per ogni classe j:
   - Calcola il vettore di media empirica $\mu^{(j)}$
   - Calcola la matrice di covarianza empirica $\Sigma^{(j)}$
3. Dato x, restituisci j che massimizza $\pi_j P_j(x)$
   (equivalentemente: che massimizza $\delta_j(x)$)

Analisi discriminante quadratica (QDA)

Densità della gaussiana:

$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

L'argomento dell'esponenziale è una funzione quadratica di x:

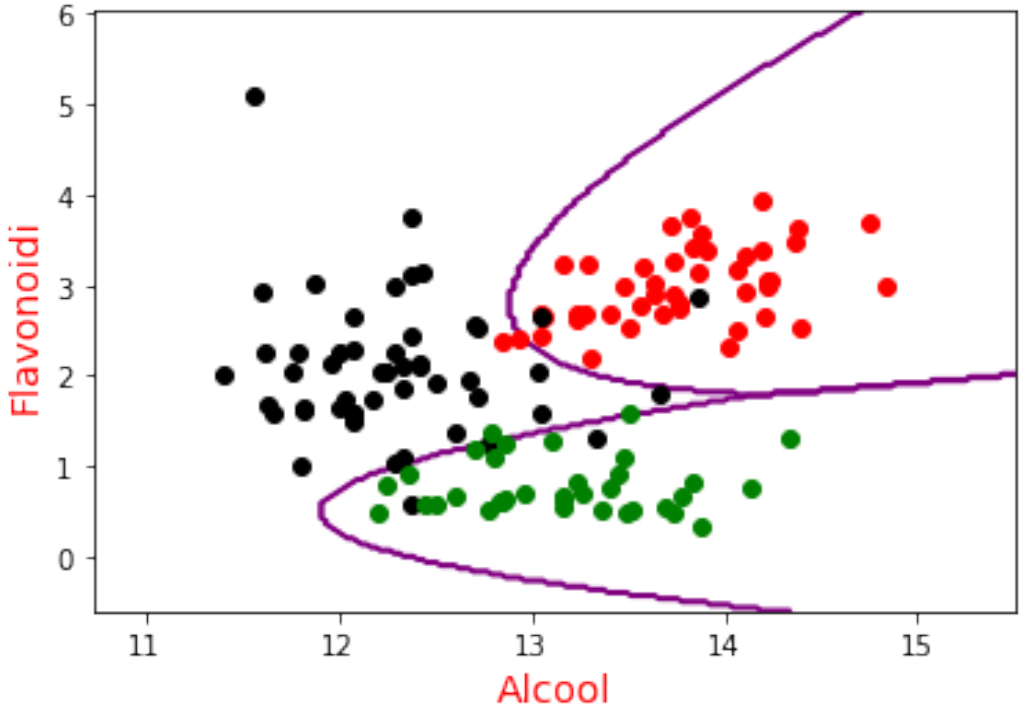
$$\log p(x) = \text{costante} - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)$$

Il discriminante di ogni classe j è una funzione quadratica di x:

$$\begin{aligned}\delta_j(x) &= \log(\pi_j P_j(x)) \\ &= \log \pi_j + \text{costante} - \frac{1}{2}\log|\Sigma^{(j)}| - \frac{1}{2}(x-\mu^{(j)})^\top (\Sigma^{(j)})^{-1}(x-\mu^{(j)})\end{aligned}$$

$\Rightarrow$ Le *frontiere di decisione* sono determinate da equazioni quadratiche in x

QDA per il dataset `wine`



Considerando tutte e 13 le feature, l'errore di test diventa zero

Analisi discriminante lineare (LDA)

Se la matrice di covarianza è la stessa per tutte le classi,
$\Sigma^{(1)} = \Sigma^{(2)} = \ldots = \Sigma$, sviluppando i prodotti abbiamo:

$$\begin{aligned}\delta_j(x) &= \log \pi_j + \text{costante} - \frac{1}{2} x^\top \Sigma^{-1} x + x^\top \Sigma^{-1} \mu^{(j)} - \frac{1}{2} \mu^{(j)\top} \Sigma^{-1} \mu^{(j)} \\ &= \log \pi_j + c(x) + x^\top \Sigma^{-1} \mu^{(j)} - \frac{1}{2} \mu^{(j)\top} \Sigma^{-1} \mu^{(j)}\end{aligned}$$

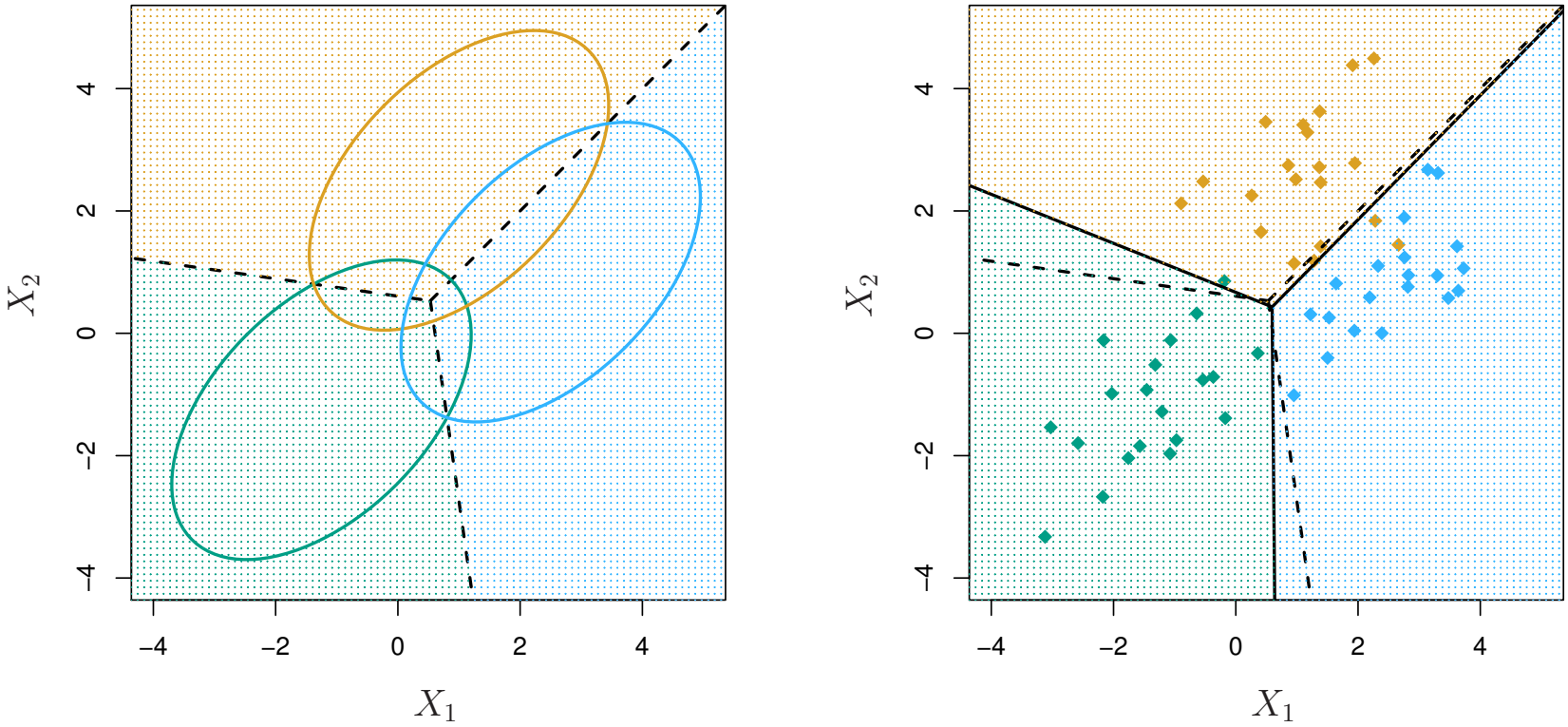
dove $c(x)$ non dipende da j (e quindi è irrilevante per i confronti)

$\Rightarrow$ Le frontiere di decisione sono determinate da equazioni lineari in x

L'analisi discriminante lineare assume che la matrice di covarianza Σ sia comune a tutte le classi (anche se empiricamente si osservano matrici $\Sigma^{(j)}$ distinte):

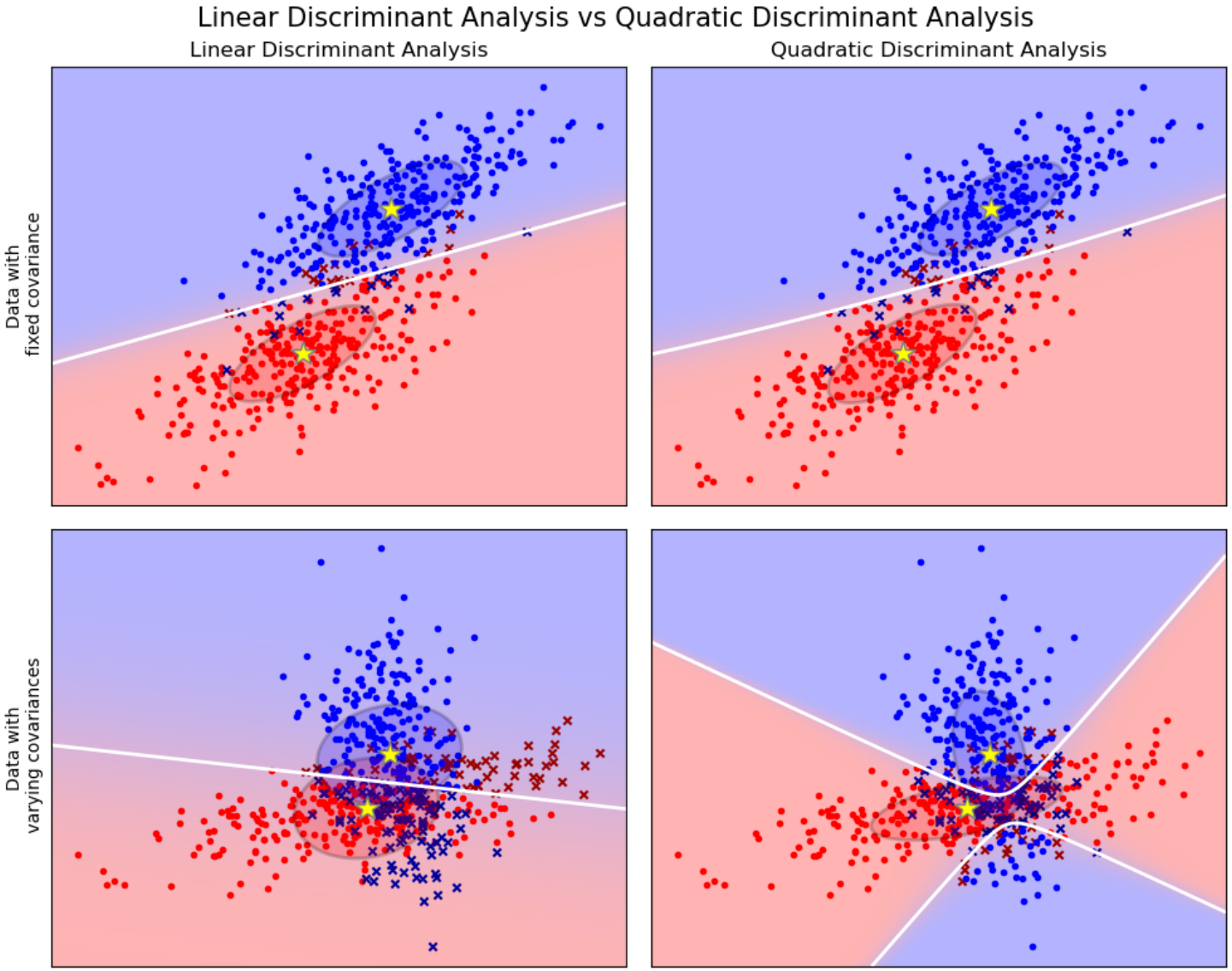
Per stimare Σ utilizziamo la formula $\sum_j \pi_j \Sigma^{(j)}$

LDA: Esempio



- Sinistra: ellissi contenenti il 95% di probabilità per ciascuna delle tre classi
- Destra: le frontiere di decisione determinate da 20 osservazioni

LDA vs. QDA


Linear Discriminant Analysis vs Quadratic Discriminant Analysis
Linear Discriminant Analysis
Quadratic Discriminant Analysis
Data with fixed covariance
Data with varying covariances

Modellazione generativa con altre distribuzioni

La modellazione generativa non è ristretta all'uso di distribuzioni Gaussiane

Altre possibilità (tutti esempi di *famiglie esponenziali*):

- Distribuzione Gamma (supporto positivo)
- Distribuzione di Poisson (supporto numerabile)
- Distribuzione categorica (supporto finito)

Tutte le distribuzioni di famiglie esponenziali possono essere stimate con relativa facilità (seguendo il principio MLE)

Naive Bayes

Se il numero di variabili d è molto alto, l'elaborazione delle matrici di covarianza (matrici $d \times d$) diventa impraticabile

Il metodo *Naive Bayes* offre una alternativa più rozza ma efficiente

Naive Bayes

1. Fai il fit di una distribuzione condizionata $\Pr_i$ per ciascuna variabile x_i, indipendentemente una dall'altra
2. Assumi $\Pr(x|y) = \Pr_1(x_1|y) \cdot \Pr_2(x_2|y) \ldots \cdot \Pr_d(x_d|y)$
3. Dato x, restituisci j che massimizza $\pi_j \Pr(x|y = j)$

Attenzione: L'assunzione di indipendenza porta tipicamente ad una stima inaccurata delle probabilità. Ciononostante, la qualità della classificazione può essere adeguata.

Esempio: Classificazione di testi (spam/no spam)

Dizionario: $D = \{$ `a`, `aardvark`, `...`, `buy`, `...`, `zygmurgy` $\}$
Dimensione: $d = |D| = 5000$

Rappresentiamo un messaggio con un vettore $x \in \{0,1\}^d$:

$$x = \begin{bmatrix} 1 \\ 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{bmatrix}$$

dove $x_k = 1 \Leftrightarrow$ il messaggio contiene la k-esima parola di D

$$y \in \{1\,(\text{spam}), 0\,(\text{no spam})\}$$

Un modello generativo esplicito per una $\Pr(x|y)$ categorica richiederebbe $2^d - 1$ parametri!

Esempio: Classificazione di testi (spam/no spam)

Usando l'assunzione Naive Bayes:

$$\Pr(x_1, \ldots, x_d | y) = \prod_{k=1}^{d} \Pr_k(x_k | y)$$

dove ciascuna $\Pr_k$ è specificata dai due parametri

$$\phi_{k|y=1} = \Pr(x_k = 1 | y = 1), \qquad \phi_{k|y=0} = \Pr(x_k = 1 | y = 0)$$

Inoltre modelliamo le probabilità a priori delle classi:

$$\pi_1 = \Pr(y = 1), \qquad \pi_0 = \Pr(y = 0) = 1 - \pi_1$$

In questo caso i parametri sono solo $2d + 1$

Esempio: Classificazione di testi (spam/no spam)

La Maximum Likelihood Estimation fornisce le seguenti stime:

$$\phi_{k|y=1} = \frac{|\{i = 1, \ldots, m : x_k^{(i)} = 1, y^{(i)} = 1\}|}{|\{i = 1, \ldots, m : y^{(i)} = 1\}|}$$

$$\phi_{k|y=0} = \frac{|\{i = 1, \ldots, m : x_k^{(i)} = 1, y^{(i)} = 0\}|}{|\{i = 1, \ldots, m : y^{(i)} = 0\}|}$$

$$\pi_1 = \frac{|\{i = 1, \ldots, m : y^{(i)} = 1\}|}{m}$$

facili da calcolare con un'unica passata sul dataset

Per classificare x, restituiamo come al solito

$$\operatorname*{argmax}_j \Pr(y = j|x) = \operatorname*{argmax}_j [\pi_j \Pr(x|y = j)]$$

Esempio: Classificazione di testi (spam/no spam)

In alternativa agli stimatori MLE, si utilizza talvolta una loro variante (bayesiana), detta *Laplace smoothing*:

$$\phi_{k|y=1} = \frac{1 + |\{i = 1, \ldots, m : x_k^{(i)} = 1, y^{(i)} = 1\}|}{2 + |\{i = 1, \ldots, m : y^{(i)} = 1\}|}$$

$$\phi_{k|y=0} = \frac{1 + |\{i = 1, \ldots, m : x_k^{(i)} = 1, y^{(i)} = 0\}|}{2 + |\{i = 1, \ldots, m : y^{(i)} = 0\}|}$$

$$\pi_1 = \frac{1 + |\{i = 1, \ldots, m : y^{(i)} = 1\}|}{2 + m}$$

Questo attenua il problema dei "*cigni neri*" (ad es. una parola del dizionario mai osservata nei messaggi di training)

K-NN e classificazione generativa in `scikit-learn`

Approccio	Iperparametri	Interfaccia `scikit-learn`
K-NN	K	`KNeighborsClassifier(n_neighbors)`
LDA	–	`LinearDiscriminantAnalysis()`
QDA	–	`QuadraticDiscriminantAnalysis()`
Naive Bayes (dati binari)	–	`BernoulliNB()`