

Clustering

■ Introduzione

- Definizioni
- Criteri
- Algoritmi

■ Clustering Gerarchico

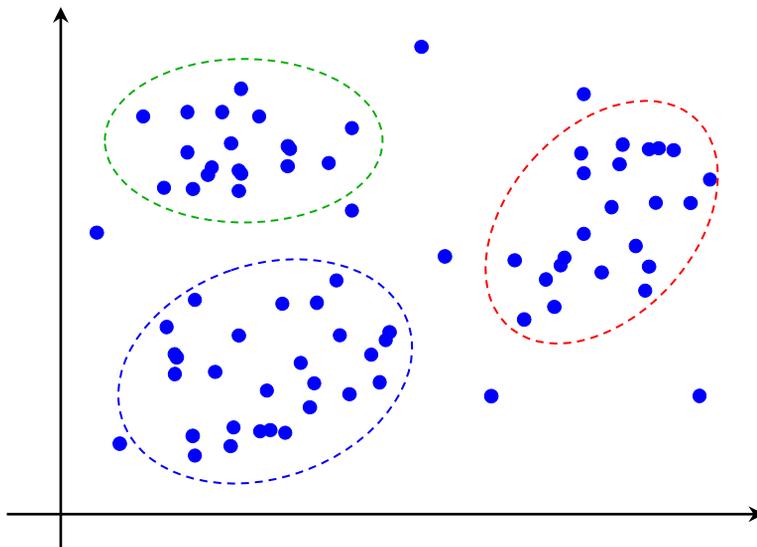
■ Centroid-based Clustering

- K-means
- Fuzzy K-means
- Expectation – Maximization (Gaussian Mixture)

Definizioni

Con il termine Clustering (in italiano «**raggruppamento**») si denota un famiglia di metodi *non supervisionati* in grado di individuare **raggruppamenti intrinseci** (**cluster**) di pattern nello spazio multidimensionale, e (opzionalmente) di **definire in corrispondenza** di tali raggruppamenti le **classi** (incognite).

Il clustering ha **applicazioni** in numerose discipline (*pattern recognition, machine learning, computer vision, data mining, data base, ...*) e pertanto ha sempre ricevuto un **notevole interesse**.



Il problema è **molto complesso** (**NP hard**): determinare la soluzione ottima con ricerca esaustiva è possibile solo nei casi semplici (pochi pattern).

Quante soluzioni?

- dati n pattern, assumendo di conoscere s , il numero di soluzioni è dell'ordine di $\frac{s^n}{s!}$ (approssimazione numero di **Stirling di seconda specie**).
 - https://it.wikipedia.org/wiki/Numeri_di_Stirling
 - Esempio: per $n = 100$, $s = 5$, il numero di soluzioni $\approx 10^{67}$.
- se s non è noto, il numero di soluzioni è dato dal numero di **Bell**, ottenibile sommando i casi precedenti per tutti i valori di s da 1 a n .
 - https://it.wikipedia.org/wiki/Numeri_di_Bell
- Nel caso $s = 2$ è molto semplice dimostrare che il numero di soluzioni è $2^{n-1} - 1$.
 - Esempio: dati i 4 pattern $\{A, B, C, D\}$ e $s = 2$, le soluzioni di clustering sono 7:
 - (A) (B,C,D)
 - (B) (A,C,D)
 - (C) (A,B,D)
 - (D) (A,B,C)
 - (A,B) (C,D)
 - (A,C) (B,D)
 - (A,D) (B,C)

Criteri di Clustering

- **Criteri** e **algoritmi** di clustering sono due cose **ben distinte**: i primi descrivono **cosa** si vuol ottenere specificando il grado di **ottimalità** di ogni soluzione ammissibile; i secondi, dato un criterio di clustering, forniscono una **procedura algoritmica** per determinare soluzioni che lo ottimizzano.
- La maggior parte dei criteri di clustering sono definiti sulla base delle due **osservazioni** seguenti:
 - 1) i pattern all'**interno** dello stesso cluster devono essere tra loro **più simili** rispetto a pattern appartenenti a cluster diversi
 - 2) i cluster sono costituiti da **nuvole di punti** a **densità** relativamente **elevata** separate da zone dove la **densità** è più **bassa**.
- Tra i diversi **criteri** possibili:
 - **minimizzazione distanze dai centroidi**: minimizza la somma dei **quadrati delle distanze** dei pattern \mathbf{x} dai centroidi (i.e. baricentri) delle classi.

$$J_e = \sum_{i=1..s} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \bar{\mathbf{x}}_i\|^2, \quad \bar{\mathbf{x}}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

dove C_i è l' i -esimo cluster, n_i il numero di pattern che contiene e $\bar{\mathbf{x}}_i$ il suo centroide (media).

È un **buon criterio per cluster a simmetria radiale** (i.e., circolari), ma penalizza forme allungate o cluster innestati (i.e. un cluster a forma di anello con all'interno un altro cluster).

Criteri di Clustering (2)

- **minimizzazione distanze intra-classe:**

$$J_e = \sum_{i=1..s} n_i \cdot \bar{s}_i, \quad \bar{s}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} f_s(\mathbf{x}, C_i)$$

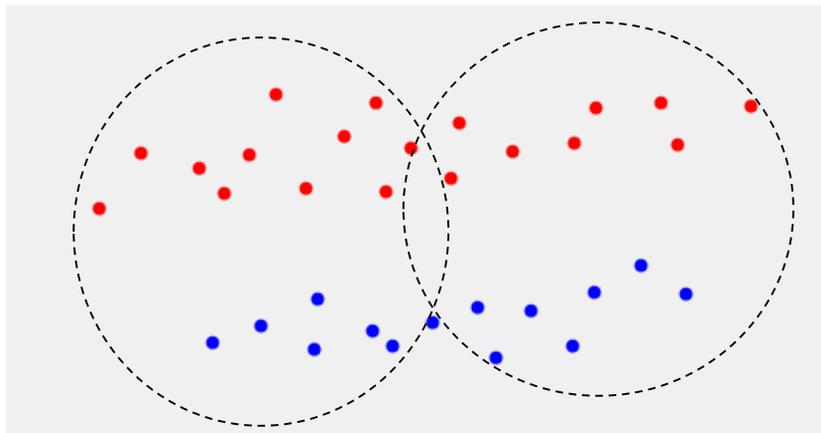
dove $f_s(\mathbf{x}, C_i)$ è una misura di distanza tra \mathbf{x} e il cluster cui appartiene. Ad esempio:

1. $f_s(\mathbf{x}, C_i) = \frac{1}{n_i} \sum_{\mathbf{x}' \in C_i} \|\mathbf{x} - \mathbf{x}'\|^2$

criterio simile alla minimizzazione distanze dai centroidi (slide precedente)

2. $f_s(\mathbf{x}, C_i) = \min_{\substack{\mathbf{x}' \in C_i \\ \mathbf{x}' \neq \mathbf{x}}} \|\mathbf{x} - \mathbf{x}'\|^2$

non penalizza cluster allungati, infatti affinché $f_s(\mathbf{x}, C_i)$ assuma valore ridotto, non è necessario che tutti i (o gran parte dei) pattern di C_i siano vicini a \mathbf{x} , ma è sufficiente un vicino. Nell'esempio sotto, per $s = 2$, (1) favorirebbe l'aggregazione come da cerchi tratteggiati, mentre (2) come da colori (rosso e blu).



Algoritmi di Clustering

- Le principali famiglie di algoritmi sono:
 - **Clustering gerarchico**: attraverso operazioni tipicamente «bottom-up», che aggregano pattern in base a una misura di distanza, si organizzano i dati in struttura ad albero (dendrogramma).
 - **Clustering basato su centroidi**: attraverso euristici (iterativi) si individuano i cluster cercando di minimizzare la distanza dei pattern dai centroidi dei cluster cui appartengono:
 - K-means
 - Fuzzy K-means
 - Expectation – Maximization (Gaussian Mixture)
 - **Clustering basato sulla densità**: i cluster individuati sono regioni connesse in aree ad elevata densità. L'approccio più noto è DBSCAN (<https://en.wikipedia.org/wiki/DBSCAN>)
- Distinguiamo inoltre:
 - **Clustering hard (esclusivo)**: un pattern è assegnato (in modo esclusivo) a un solo cluster.
 - **Clustering soft (fuzzy)**: i pattern appartengono ai diversi cluster con un certo grado di appartenenza (es. tra 0 e 1).

È più efficace nel gestire pattern vicino al bordo di due o più clusters e outliers. L'assegnazione può diventare esclusiva scegliendo, per ogni pattern, il cluster verso cui il grado di appartenenza è massimo.

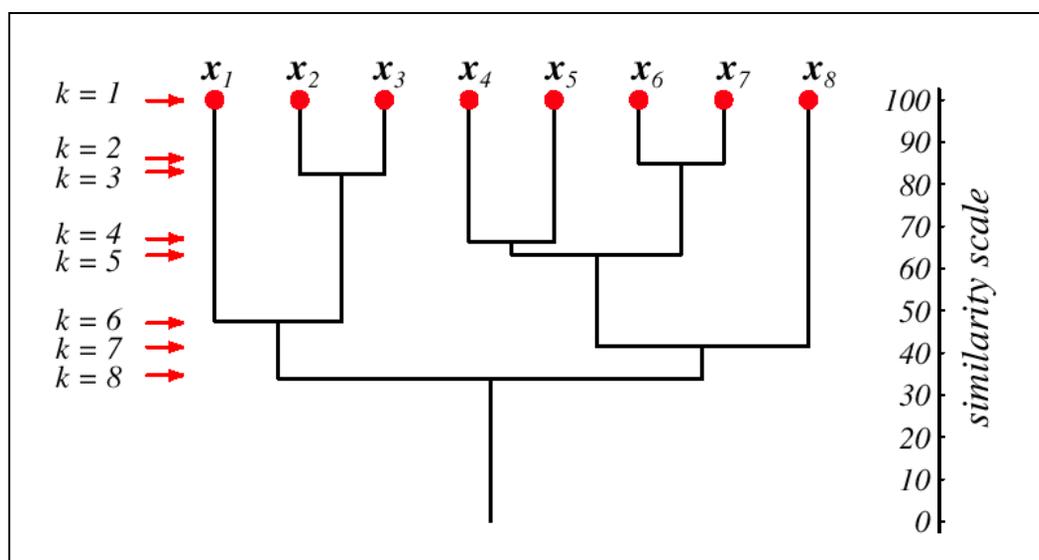
Clustering Gerarchico

Simile al modo di eseguire **classificazione** in **tassonomia biologica**:

- gli insetti sono **gerarchicamente** classificati **specializzandone** le specie a partire da famiglie molto **ampie** fino a famiglie più **ridotte**.

Gli algoritmi sono generalmente **bottom-up** (**agglomerativi**):

- si parte cercando di **aggregare** singoli elementi e ad ogni passo (livello) si **aggregano** (*pattern a pattern, pattern a cluster, o cluster a cluster*) gli elementi tra loro più simili (i.e. **meno distanti**) rispetto a una **soglia** (che dipende dal livello).
- Le principali differenze implementative dipendono dalla definizione utilizzate per calcolare le distanze tra cluster e cluster:
 - **Single link**: distanza minima tra due pattern dei cluster
 - **Average link**: distanza media tra i pattern dei due cluster
 - **Complete link**: distanza massima tra due pattern dei cluster



K-means

K-means (o C-means) è un metodo **computazionalmente molto semplice** e altrettanto semplice da **implementare**: per questo motivo è spesso la prima scelta per risolvere problemi di clustering.

- Minimizza «implicitamente» le **distanze dai centroidi**.
- Richiede in **input** il **numero di cluster** (s) e una **soluzione iniziale**. Produce **buoni risultati** a patto di fornire una ragionevole soluzione iniziale e un numero adeguato di classi.
- Il tipo di **ottimizzazione** è **iterativa** e **locale**; pertanto il metodo può convergere a massimi locali della soluzione. La **convergenza** si ottiene solitamente in pochi passi: < 10).
- Identifica cluster **iper-sferici** nel caso in cui venga utilizzata la **distanza euclidea** come misura di distanza tra i pattern o cluster **iper-ellissoidali** nel caso di **distanza di Mahalanobis**.
- Nella sua **versione base**, l'algoritmo può essere così descritto:

Inizializza n, s

Scegli casualmente s pattern da utilizzare come centroidi iniziali

do { **assegna** ogni pattern al cluster per cui è **minima la distanza**
dal centroide.

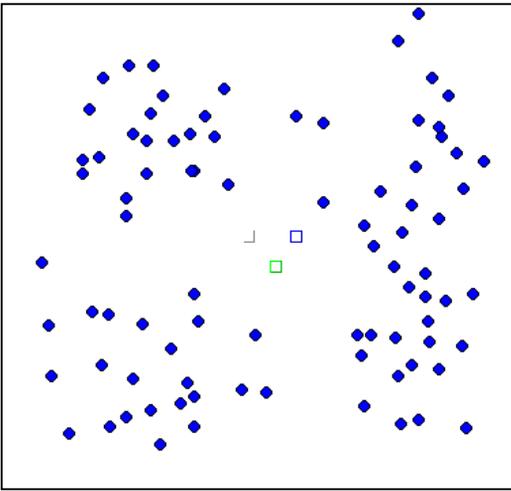
calcola i centroidi dei cluster come media dei rispettivi pattern

} while (*i cluster sono stati modificati & iteration < max*)

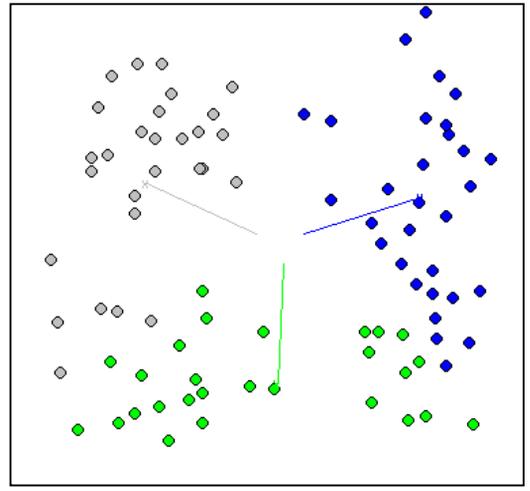
I cluster sono **modificati iterativamente** a seguito del ricalcolo del loro centroide. L'algoritmo **termina** (converge) quando i centroidi sono stabili e quindi le partizioni non **cambiano**.

K-means: esempio ($n = 86, s = 3$)

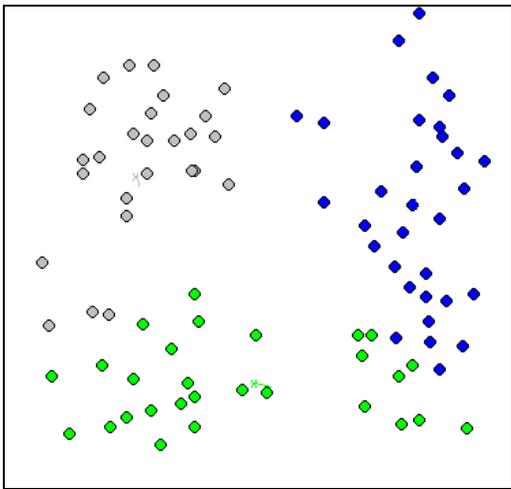
soluzione iniziale



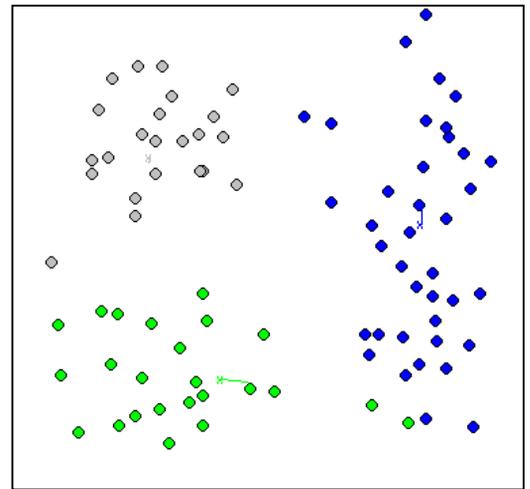
iterazione 1



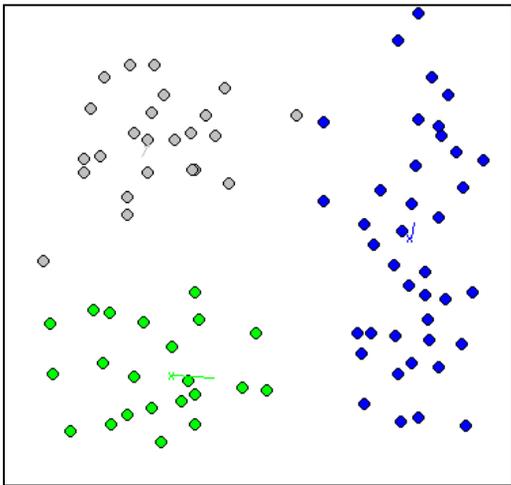
iterazione 2



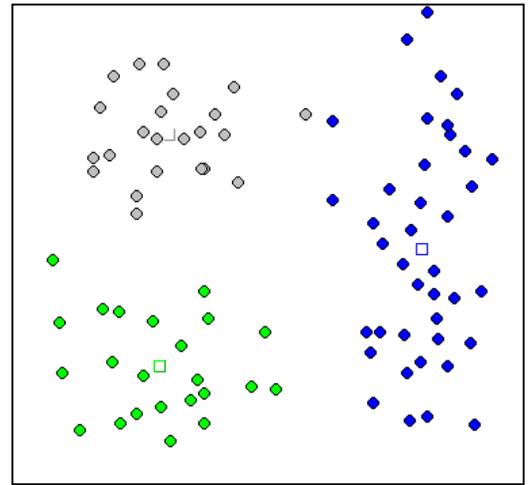
iterazione 4



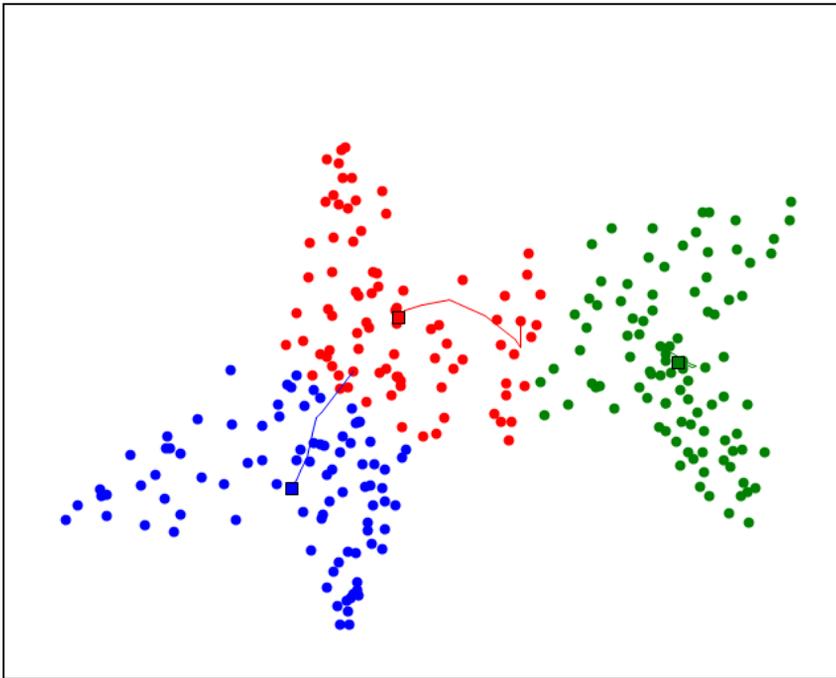
iterazione 6



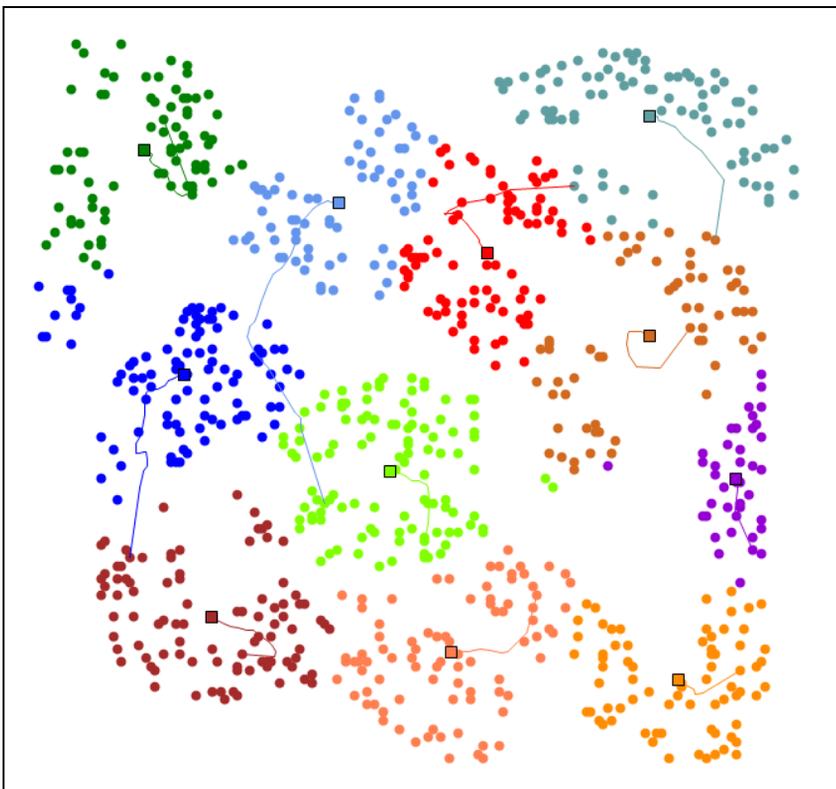
iter. 7 (convergenza)



K-means: limitazioni



3 classi – 8 iterazioni
Minimizzando le distanze dai centroidi, K-means non è in grado di identificare cluster dalla forma non sferica.



11 classi, 31 iteraz.
anche in questo caso non tutti i cluster hanno forma sferica ...

K-means: soluzione iniziale e validazione

Diverse **varianti** sono state proposte per **generare buone soluzioni iniziali** e di **determinare il numero di classi** (**clustering validation**).

- Per **minimizzare il rischio** di **convergenza** verso **minimi locali**, l'algoritmo può essere eseguito **più volte** a partire da **soluzioni iniziali diverse**:
 - random
 - prodotte da un (diverso) euristico
 - o magari da un metodo evoluzionistico (algoritmo genetico).
- Le tecniche di **validazione** tendono a valutare a posteriori la **bontà** delle **soluzioni prodotte** per diversi valori di s , e a sceglierne una sulla base di un **criterio di validazione** che tenga conto sia della **bontà** della soluzione sia della sua **complessità**:
 - **Problema**: considerando il criterio di «minimizzazione distanze dai centroidi» e lanciando K-means con diversi valori di s , è molto più facile ottenere valori ridotti di J_e per valori elevati di s . Quanto vale J_e per $s = n$ (ogni cluster un solo pattern) ?
 - **Possibile soluzione**: penalizzare le soluzioni con molti cluster; ad esempio:

$$J_e^* = J_e + \text{penalty} \cdot s$$

- **Altra possibilità**: cercare discontinuità (scalini verso il basso) nel valore di J_e (o di J_e^*) al variare di s . Infatti, se i pattern formano raggruppamenti evidenti e ben identificabili, scegliendo il corretto valore di s , dovremmo riscontrare una discontinuità nel grafico di J_e^* al variare di s .

Fuzzy K-means

La variante **Fuzzy** del K-means consente a un pattern di appartenere con un **certo grado di probabilità** a diverse classi. Il criterio di ottimizzazione in questo caso è:

$$J_{fuz} = \sum_{i=1..s} \sum_{j=1..n} [P(C_i | \mathbf{x}_j, \Theta)]^m \cdot \|\mathbf{x}_j - \boldsymbol{\theta}_i\|^2$$

dove $P(C_i | \mathbf{x}_j, \Theta)$ è la **probabilità** che dato il pattern \mathbf{x}_j e l'insieme $\Theta = \{\boldsymbol{\theta}_1 \dots \boldsymbol{\theta}_s\}$ di centroidi che definiscono la **soluzione attuale**, il cluster di **appartenenza** sia C_i .

I **centrodi** invece che come semplice media dei pattern si calcolano come **media pesata** rispetto alle probabilità di appartenenza:

$$\boldsymbol{\theta}_i = \frac{\sum_{j=1..n} P(C_i | \mathbf{x}_j, \Theta) \cdot \mathbf{x}_j}{\sum_{j=1..n} P(C_i | \mathbf{x}_j, \Theta)} \quad \text{Eq. (1)}$$

Le **probabilità di appartenenza** si calcolano come:

$$P(C_i | \mathbf{x}_j, \Theta) = \frac{1}{\sum_{k=1..s} \left(\frac{\|\mathbf{x}_j - \boldsymbol{\theta}_i\|}{\|\mathbf{x}_j - \boldsymbol{\theta}_k\|} \right)^{\frac{2}{m-1}}} \quad \text{Eq. (2)}$$

$m > 1$ è un parametro che definisce quanto l'appartenenza ai diversi cluster debba essere sfumata. Valore tipico $m = 2$.

Fuzzy K-means (2)

Il pseudocodice dell'algoritmo è molto simile al K-means:

Inizializza n, s

Inizializza casualmente $P(C_i | \mathbf{x}_j, \Theta)$, $i = 1 \dots s, j = 1 \dots n$

do { calcola i centroidi $\Theta = \{\theta_1 \dots \theta_s\}$ // Eq. (1)

calcola i gradi di appartenenza

$P(C_i | \mathbf{x}_j, \Theta)$, $i = 1 \dots s, j = 1 \dots n$ // Eq. (2)

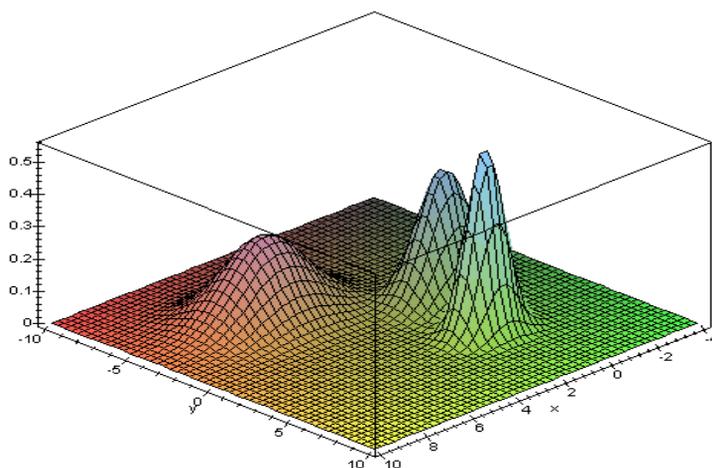
} while (riduzione J_{fuz} significativa & iteration < max)

Rispetto al K-means, questa variante fornisce a volte una convergenza più robusta verso la soluzione finale. D'altro canto uno svantaggio è legato al fatto che l'appartenenza di un pattern a un cluster dipende implicitamente dal numero di cluster, e se questo è specificato in modo incorretto si possono ottenere cattive soluzioni.

Expectation – Maximization (EM)

con Gaussian Mixture

Si ipotizza che i pattern siano stati generati da un *mix di distribuzioni* (“mixture” in inglese): ogni classe ha generato dati in accordo con una *specifica distribuzione*, ma al termine della generazione i pattern appaiono come prodotti da *un'unica distribuzione multi-modale*.



Obiettivo del clustering con EM è *risalire* (a partire dai pattern del training set) *ai parametri delle singole distribuzioni* che li hanno generati. A tal fine si ipotizza *nota la forma* delle distribuzioni e si assume, per semplicità, che esse siano *tutte dello stesso tipo*. Il caso più frequente è quello di *mix* di *s* distribuzioni *multinormali* (gaussiane), di cui si vogliono *stimare* i parametri di definizione (*s* *vettori medi* + *s* *matrici di covarianza* + *s* *coefficienti* α):

$$p(\mathbf{x}|\Theta) = \sum_{i=1..s} \alpha_i \cdot p_i(\mathbf{x}|\theta_i) \quad \theta_i = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$$

dove $\Theta = \{\alpha_1, \alpha_2 \dots \alpha_s, \theta_1, \theta_2 \dots \theta_s\}$ è il vettore di parametri che definisce il mix di distribuzioni, $p_i(\cdot|\theta_i)$ è una multinormale con parametri θ_i

EM: Idea

La stima dei parametri avviene secondo il criterio generale del *maximum likelihood* (in italiano “massima verosimiglianza”).

In generale il likelihood \mathcal{L} dei parametri Θ dati i pattern \mathcal{X} corrisponde alla probabilità (densità di probabilità nel continuo) di aver ottenuto i pattern dati i parametri (inversione):

$$\mathcal{L}(\Theta|\mathcal{X}) = p(\mathcal{X}|\Theta)$$

Per semplicità, al posto del likelihood, si massimizza il suo logaritmo. Considerando i pattern indipendenti la probabilità di avere ottenuto i pattern del training set è il prodotto delle probabilità delle singole generazioni:

$$\log p(\mathcal{X}|\Theta) = \log \prod_{j=1 \dots n} p(\mathbf{x}_j|\Theta) = \sum_{j=1 \dots n} \log \left(\sum_{i=1 \dots s} \alpha_i \cdot p_i(\mathbf{x}_j|\theta_i) \right)$$

Purtroppo questa massimizzazione è **molto difficile** a causa della sommatoria dentro al logaritmo. Per eliminare la sommatoria, ci servirebbe sapere ogni pattern \mathbf{x}_j del training set da quale componente $p_i(\cdot)$ della mixture è stato generato.

A tal fine si utilizza **EM**, che è un approccio iterativo nato per il calcolo del **maximum likelihood** nel caso in cui i dati a disposizione $\mathcal{X}=\{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_n\}$ siano **incompleti** per la **manca** di alcuni **valori** $\mathbf{y}=\{\mathbf{y}_1, \mathbf{y}_2 \dots \mathbf{y}_n\}$. Ogni **pattern completo** $\mathbf{z}_j = [\mathbf{x}_j, \mathbf{y}_j], j = 1 \dots n$ è costituito da due parti di cui solo la prima è nota.

Nel **caso di interesse** (derivare i parametri di una gaussian mixture) i dati **sono completi** ma le **potenzialità** di EM sono utilizzate per **rendere trattabile** la massimizzazione: sfruttiamo i valori \mathbf{y} come **indicatori delle componenti ignote** che hanno generato i singoli pattern.

EM: nel caso generale

Obiettivo di EM è dunque la massimizzazione del **log-likelihood** dei **dati completi** o **log-likelihood-completo**:

$$\log \mathcal{L}(\Theta|Z) = \log \mathcal{L}(\Theta|\mathcal{X}, \mathcal{Y}) = p(\mathcal{X}, \mathcal{Y}|\Theta)$$

A tal fine vengono eseguiti **iterativamente** (fino a convergenza) due passi, **Expectation** e **Maximization**:

- **Expectation**: viene calcolato il valore atteso $E(\cdot)$ del log-likelihood-completo, **dato** il **training set** \mathcal{X} e una **stima dei parametri** Θ^g ottenuti all'iterazione precedente:

$$Q(\Theta|\Theta^g) = E(\log p(\mathcal{X}, \mathcal{Y}|\Theta) | \mathcal{X}, \Theta^g)$$

Il valore atteso (**media**) è calcolato rispetto alla variabile aleatoria \mathcal{Y} , governata dalla distribuzione $f(\mathcal{Y}|\mathcal{X}, \Theta^g)$:

$$E(\log p(\mathcal{X}, \mathcal{Y}|\Theta) | \mathcal{X}, \Theta^g) = \int_{\mathcal{Y} \in \Psi} \log p(\mathcal{X}, \mathcal{Y}|\Theta) \cdot f(\mathcal{Y}|\mathcal{X}, \Theta^g) d\mathcal{Y}$$

- **Maximization**: determina il valore dei **parametri** che massimizzano il valore atteso calcolato al passo di **Expectation**:

$$\Theta^{g+1} = \underset{\Theta}{\operatorname{argmax}} Q(\Theta|\Theta^g)$$

EM: nel caso specifico

Sfruttando le peculiarità di EM, si **assume** l'esistenza di **una componente nascosta** y_j (per ogni pattern) che indica quale delle s distribuzioni gaussiane ha generato il pattern \mathbf{x}_j .

Il **log-likelihood-completo** assume ora la forma:

$$\log \mathcal{L}(\Theta | \mathcal{X}, \mathbf{y}) = \sum_{j=1 \dots n} \log \left(\alpha_{y_j} \cdot p_{y_j}(\mathbf{x}_j | \boldsymbol{\theta}_{y_j}) \right)$$

sebbene gli y_j non siano noti, **una loro stima** (o meglio una stima della **loro distribuzione**) può essere derivata (**teorema di Bayes**) dai parametri Θ^g disponibili all'iterazione (g) corrente.

Per un singolo y_j (e corrispondente \mathbf{x}_j):

$$P(y_j = k | \mathbf{x}_j, \Theta^g) = P(k | \mathbf{x}_j, \Theta^g) = \frac{\alpha_k^g \cdot p_k(\mathbf{x}_j | \boldsymbol{\theta}_k^g)}{\sum_{i=1 \dots s} \alpha_i^g \cdot p_i(\mathbf{x}_j | \boldsymbol{\theta}_i^g)} \quad \text{Eq. (1)}$$

Per un generico vettore \mathbf{y} di osservazioni:

$$P(\mathbf{y} | \mathcal{X}, \Theta^g) = \prod_{j=1 \dots n} P(y_j | \mathbf{x}_j, \Theta^g)$$

Il valore atteso Q della slide precedente può essere scritto come:

$$\begin{aligned} Q(\Theta | \Theta^g) &= \sum_{\mathbf{y} \in \Psi} \log \mathcal{L}(\Theta | \mathcal{X}, \mathbf{y}) \cdot P(\mathbf{y} | \mathcal{X}, \Theta^g) = \\ &= \sum_{\mathbf{y} \in \Psi} \left(\sum_{j=1 \dots n} \log \left(\alpha_{y_j} \cdot p_{y_j}(\mathbf{x}_j | \boldsymbol{\theta}_{y_j}) \right) \cdot \prod_{j=1 \dots n} P(y_j | \mathbf{x}_j, \Theta^g) \right) \end{aligned}$$

EM: formule incrementali

A **seguito** di alcuni passaggi (non banali) [1] durante i quali l'ottimizzazione è eseguita **eguagliando a 0** le **derivate** parziali rispetto ai **parametri incogniti** si ottengono le seguenti formule per l'aggiornamento incrementale dei parametri:

$$P(C_k | \mathbf{x}_j, \Theta^g) = \frac{\alpha_k^g \cdot p(\mathbf{x}_j | \boldsymbol{\theta}_k^g)}{\sum_{i=1 \dots s} \alpha_i^g \cdot p(\mathbf{x}_j | \boldsymbol{\theta}_i^g)} \quad \text{Eq. (1)}$$

$$\alpha_k^{g+1} = \frac{1}{n} \sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g) \quad \text{Eq. (2)}$$

$$\boldsymbol{\mu}_k^{g+1} = \frac{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g) \cdot \mathbf{x}_j}{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g)} \quad \text{Eq. (3)}$$

$$\boldsymbol{\Sigma}_k^{g+1} = \frac{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g) (\mathbf{x}_j - \boldsymbol{\mu}_k^{g+1}) (\mathbf{x}_j - \boldsymbol{\mu}_k^{g+1})^t}{\sum_{j=1 \dots n} P(C_k | \mathbf{x}_j, \boldsymbol{\theta}_k^g)} \quad \text{Eq. (4)}$$

Intuitivamente, trattando le componenti della mixture come cluster:

- Eq (1), stessa della slide precedente: indica la probabilità di appartenenza di \mathbf{x}_j al cluster C_k . Calcolata sfruttando inversione di Bayes. *Ricorda*: $p(\cdot | \boldsymbol{\theta}_k)$ è una multinormale con parametri $\boldsymbol{\theta}_k$
- Eq (2). Calcola il «peso» del cluster C_k in base alla somma delle appartenenze a C_k di tutti i pattern del training set.
- Eq (3) e (4). Calcolano media e matrice di covarianza del cluster C_k pesando i pattern rispetto al loro grado di appartenenza a C_k .

[1] J. A. Bilmes, "A Gentle Tutorial of the EM Algorithm and its Application for Gaussian Mixture and Hidden Markov Models", International Computer Science Institute, 1998.

EM: algoritmo

Inizializza n, s

Inizializza $g = 0, \alpha_k^0, \theta_k^0 = \{\mu_k^0, \Sigma_k^0\}, k = 1 \dots s$

do {

Expectation: calcola

$$P(C_k | \mathbf{x}_j, \Theta^g), \quad k = 1 \dots s, j = 1 \dots n \quad // \text{ Eq. (1)}$$

Maximization: calcola

$$\alpha_k^{g+1}, \mu_k^{g+1}, \Sigma_k^{g+1}, \quad k = 1 \dots s \quad // \text{ Eq. (2) (3) (4)}$$

$$g = g + 1$$

} while (parametri α_k e θ_k non stabili & $g < \max$)

L'algoritmo è simile a fuzzy k-means, ma in questo caso, oltre ai centroidi (qui i vettori medi μ_k^{g+1}), abbiamo maggiori gradi di libertà nella forma dei cluster che possono essere ellissoidali.

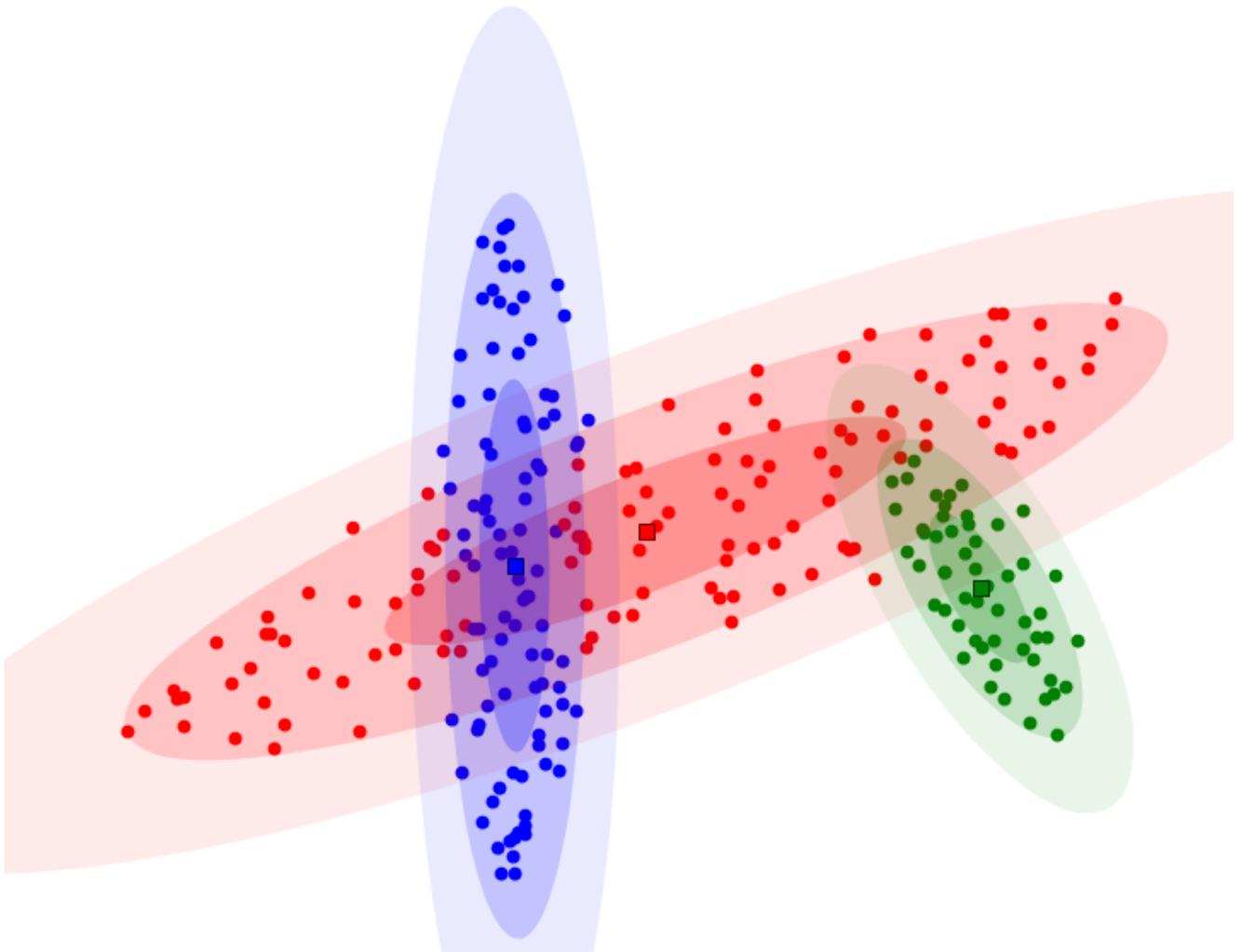
Attenzione all'implementazione:

- Le matrici di covarianza non possono essere generate completamente random (è necessario siano simmetriche e definite positive). *Consiglio:* partire con matrici diagonali.
- Le probabilità di appartenenza possono assumere valori molto piccoli e causare problemi numerici. *Consiglio:* usare costanti moltiplicative se necessario e normalizzare alla fine.

Esempio

3 classi – 40 iterazioni circa
(generato con BioLab)

A differenza di K-means e fuzzy K-means,
EM è in grado di identificare nuvole di punti con
forme ellissoidali



Altro esempio

11 classi – 200 iterazioni circa
(generato con BioLab)

