# Progetto Elaborazione Linguaggio Naturale: Tecniche di Clustering

Giuseppe De Palma

Alma Mater Studiorum - Università di Bologna
giuseppe.depalma@studio.unibo.it
Matricola: 854846

## 1 Introduzione

Il *clustering* (o analisi dei gruppi) è una forma di *machine learning* non supervisionato che permette di raggruppare in *cluster* elementi non annotati dati in input. Un cluster è una collezione di oggetti "simili" tra loro che sono "dissimili" rispetto agli oggetti degli altri cluster. Questo tipo di machine learning è ottimo per partizionare un insieme di dati in diverse "categorie", quindi poter eseguire diverse analisi ed ottenere nuove informazioni. Applicazioni tipiche in cui il clustering viene molto usato è il riconoscimento di email di spam (le email a scopi pubblicitari o di frode), oppure per l'aggregazione di notizie (Google News ne è un esempio).

Il clustering trova possibili applicazioni anche nel campo dell'elaborazione del linguaggio naturale. Oltre alle nuove possibili analisi sui corpora ed al fornire una visualizzazione grafica delle parole raggrupate, un interessante utilizzo è quello della **generalizzazione** delle parole.

Possiamo considerare i vari cluster delle classi di equivalenza. Per questo motivo, se avessimo un dataset su cui comporre i cluster fatto di frasi e parole, allora si potrebbe assumere che una qualche parola che compare in una frase può essere sostituita con un'altra dello stesso cluster lasciando intatta la correttezza della frase. Ad esempio, se avessimo nel nostro dataset "per Lunedì", "per Martedì", "per Mercoledì", "per Sabato", "per Domenica", senza avere "per Giovedì" e "per Vernedì", e avessimo un cluster in cui i giorni della settimana sono raggruppati insieme, allora potremmo generalizzare l'utilizzo della preposizione "per" con Giovedì e Vernedì.

Il clustering, quindi, può essere molto utile anche nell'elaborazione del linguaggio naturale. Nel progetto in studio vengono testate le capacità di alcune tecniche di clustering da cui si derivano dei risultati per mostrarne le differenze, i pregi e i difetti. I dati utilizzati negli esperimenti, comunque, non sono parti di testo, ma semplici dataset di vettori numerici 2D in modo tale da poter facilmente visualizzare i grafici relativi ai cluster e determinare le caratteristiche di ogni tecnica.

### 1.1 Outline

Nella prossima sezione vengono discussi i diversi algoritmi di clustering in studio, dove per ognuno sono descritte le proprietà, gli utilizzi e la logica dietro il loro funzionamento. In seguito è presentata la sessione sperimentale dove alcuni test e considerazioni sono mostrati. Infine si conclude spiegando le differenze più marcate tra le tecniche.

## 2 Clustering

Ci sono numerosi algoritmi per effettuare clustering, ma essi sono classificabili in poche tipologie: il clustering partizione e il clustering gerarchico. Il clustering partizionale consiste nell'ottenere dei cluster, di solito in modo iterativo, ma spesso senza determinare una vera relazione tra gli elementi. Si inizia con un insieme di cluster iniziale ed iterativamente si riassegnano gli oggetti nei giusti cluster. Il clustering gerarchico, invece, forma un albero (la gerarchia) degli elementi dove un nodo rappresenta un sotto-cluster del nodo padre e le foglie sono i singoli oggetti.

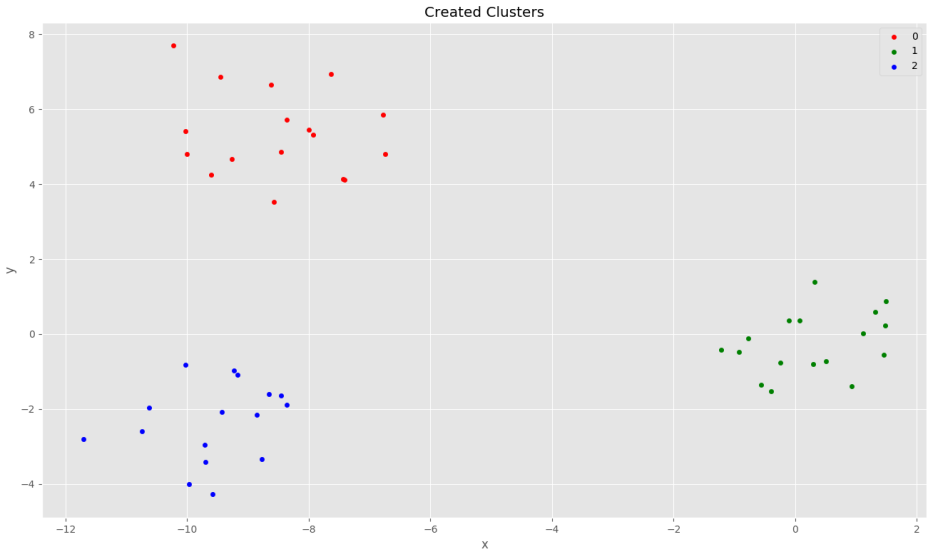
Un'altra importante distinzione tra gli algoritmi di clustering è il *soft clustering* e *hard clustering*. Nel primo caso, ogni oggetto può essere assegnato a più cluster secondo un qualche grado di appartenenza, mentre nel secondo caso ogni oggetto è assegnato ad un unico cluster. In questo progetto vedremo quattro diversi algoritmi, due della classe di clustering gerarchico, due del clustering partizionale. I primi tre eseguono hard clustering mentre l'ultimo (EM) soft clustering.

Di seguito sono elencati i metodi implementati e testati:

- Clustering **gerarchico**
  1. Aggregativo
  2. Divisivo
- Clustering **partizionale**
  1. K-means
  2. EM

### 2.1 Gerarchico

Andando più in dettaglio sulle diverse tecniche, abbiamo detto che la prima classe di clustering permette di creare degli alberi con i cluster e sotto-cluster. Questo può essere ottenuto con un approcco *bottom-up* che è il clustering **aggregativo**, il quale inizia dai singoli oggetti e ne raggruppa i più simili, per poi raggruppare i gruppi più simili e così via, fino ad ottenere un unico gruppo che sarà la radice dell'albero. Un altro approccio è quello *top-down*, il clustering **divisivo**, che in modo inverso dal precedente parte dal gruppo comprendente tutti gli elementi e lo divide in sotto-gruppi in modo da massimizzare la similarità intrinseca dei gruppi, fino ad arrivare ai singoli elementi.



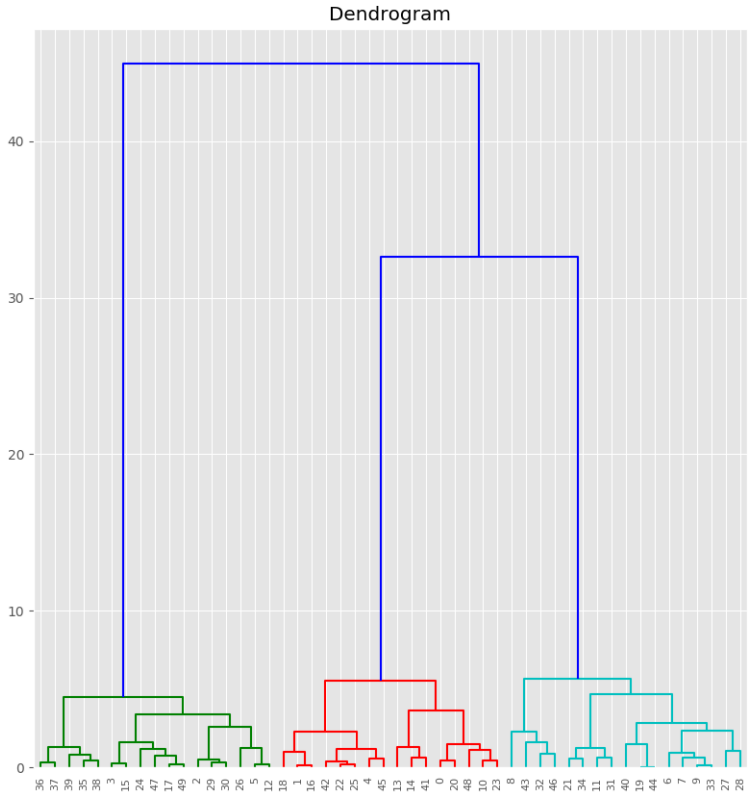




Figura 1: (a) insieme di 50 punti pre-divisi in 3 gruppi distinti, su cui si applica il clustering aggregativo con la funzione di similarità ward. Se ne ricava un dendogramma, in (b), che mostra le relazioni dei dati. Si noti come tagliando orizzontalmente il cluster per le linee verticali più alte dove non appaiono linee orizzontali, rimangono proprio 3 sotto-cluster.

L'albero che si ottiene può essere visualizzato tramite un dendrogramma, come nella figura 1. Questo diagramma ci mostra le relazioni gerarchiche tra gli oggetti, infatti il suo principale utilizzo è quello di mostrare il modo migliore di organizzare i cluster.

Concentrandosi sulle linee verticali possiamo capire il numero ottimale di cluster da prefissare, siccome l'asse $y$ misura la somiglianza dei cluster, mentre l'asse $x$ rappresenta gli oggetti e i cluster. Disegnando una linea orizzontale che "taglia" il dendrogramma, si afferma che i sotto-cluster al di sotto di questa linea saranno i raggruppamenti finali. Poichè linee verticali corte rappresentano gradi di similarità più alti tra gruppi, tagliando il dendrogramma per le linee verticali più alte che sono libere da linee orizzontali (quando c'è una unione di cluster), si ha il numero di cluster ottimale a cui assegnare gli elementi,

**Aggregativo** Questo tipo di clustering è realizzato tramite un algoritmo *greedy* che prende in input un insieme di dati $S$, da cui ogni oggetto è considerato essere un piccolo cluster da un solo elemento. Ad ogni passo l'algoritmo determina i due cluster più simili e li unisce in un nuovo cluster. L'algoritmo termina quando il cluster contenente tutti gli elementi di $S$ viene formato, il quale sarà l'unico cluster rimanente. I modi in cui si possono determinare la similarità dei cluster sono molteplici. L'algoritmo fa uso di una funzione di similarità per calcolare quanto due cluster sono simili tra di loro. Ne esistono diverse, ad esempio alcune molto utilizzate sono:

- *single link*: ottiene la similarità di due membri **più** simili da due cluster diversi, rispettivamente;
- *complete link*: ottiene la similarità dei due membri **meno** simili da due cluster diversi, rispettivamente;
- *group-average*: ottiene la similarità media tra i membri di due cluster.
- *ward*: ottiene la similarità in base al costo di unione di due cluster. Questo costo è ottenuto calcolando la somma dei prodotti tra gli elementi di due cluster e i loro centroidi.

Nonostante le funzioni di similarità possano differire anche ampiamente, una proprietà che tutte devono avere è la monotonia. Per un insieme di dati $S$ e una funzione di similarità $sim$:

$$\forall c, c', \hat{c} \subseteq S.\ min(sim(c, c'), sim(c, \hat{c})) \geq sim(c, c' \cup \hat{c})$$

Questo perché l'operazione di unione garantisce di non aumentare la similarità, quindi una funzione che non obbedisce a questa condizione rovinerebbe la gerarchia in quanto cluster non simili, piazzati in parti lontane nell'albero, potrebbero ritrovarsi ad essere simili in unioni successive e perciò l'essere vicini nell'albero non corrisponderebbe più al concetto di similarità.

**Divisivo** Come per la controparte aggregativa, dietro il clustering divisivo c'è un algoritmo greedy. Invece di iniziare dai singoli elementi, si inizia dal cluster

contenente tutti gli oggetti. Ad ogni iterazione si determina quale cluster è quello **meno** coerente e lo si divide in due. Come prima si utilizzano funzioni di similarità, poichè due cluster con oggetti simili sono più coerenti di cluster con oggetti non simili. L'operazione di divisione di un cluster è anch'essa una operazione di clustering, poichè bisogna trovare due sotto-cluster. Qualsiasi tecnica di clustering può essere usata per la divisione, anche il clustering aggregativo, ma spesso si utilizza K-means (come in questo progetto). Per queste ragioni il clustering divisivo è solitamente meno usato.

### 2.2 Partizionale

Diversamente dal clustering appena discusso, gli algoritmi per quello partizionale spesso iniziano con una divisione casuale del dataset in vari cluster, per poi raffinarli passo dopo passo attraverso la ricollocazione degli oggetti. Da questo procedimento sorgono varie domande:

1. Quando fermarsi con la ricollocazione?
2. In quanti cluster dividere il dataset?
3. In che modo si seleziona il miglior cluster in cui ricollocare un elemento?

Quando fermarsi può essere determinato tramite una metrica sulla qualità dei cluster. Questa può essere una funzione di similarità come la group-average (che va a toccare tutti i cluster) e finchè il risultato incrementa di molto, si continua la procedura.

La stessa metrica può essere utilizzata per rispondere alla seconda domanda. In alcuni casi possiamo già avere delle informazioni sul corretto numero di cluster da creare, oppure decidiamo arbitrariamente in quanti cluster dividere il dataset. In altri casi invece, possiamo dividere i dati in $k$ cluster osservando l'andamento della metrica. Per capire un buon numero di cluster da avere, bisogna vedere in quale transizione c'è un incremento sostanziale da $k-1$ a $k$ cluster e un piccolo incremento da $k$ a $k+1$ cluster. Trovando un numero $k$ di cluster che abbia questa proprietà, possiamo avere in modo automatico una buona partizione del dataset.

Per l'ultima questione, sono gli algoritmi per il clustering partizionale che si occupano di come ricollocare gli oggetti, che ora verranno presentati.

**K-means** K-means è un algoritmo di hard clustering che definisce i cluster attraverso i loro punti centrali. Partendo da dei centroidi iniziali, iterativamente si assegna ogni oggetto al cluster il cui centroide è il più vicino, calcolato tramite la distanza euclidea. Una volta che tutti gli elementi sono stati assegnati, si ricalcolano i punti centrali di tutti i cluster e si ripete il procedimento di assegnazione così da raffinare il clustering.

**EM** L'algoritmo EM (Expectation-Maximization) è definibile come la versione di K-means per il soft clustering. In quest'algoritmo possiamo vedere ogni cluster

come una distribuzione di probabilità (in questo progetto vedremo l'algoritmo EM con *Gaussian mixture model*). L'idea è quindi di alternare due step: l'Expectation step dove accade il calcolo delle probabilità per ogni oggetto di essere stato generato da ognuna delle distribuzioni (in K-means questo è analogo ad assegnare ogni punto ad un cluster). Successivamente c'è il Maximization step, in cui si massimizzano i parametri delle distribuzioni e degli oggetti (pesi, covarianza, medie) così da poter ripetere il primo step per raffinare le probabilità. L'equivalente in K-means è muovere i punti centrali dei cluster per poi procedere con un'altra riallocazione.

## 3 Sessione Sperimentale

In questa sezione verranno mostrati vari esperimenti per sottolineare le differenze tra i quattro metodi presentati. Sono stati effettuati diversi test usando diversi dataset di vettori 2D, così da facilmente visualizzare i risultati con grafici. Per l'algoritmo di clustering aggregativo è stato scelto di usare la funzione di similarità ward. Per il clustering divisivo viene usato K-means per creare 2 cluster a partire da un cluster più grande, che è lo step di divisione. Per scegliere quale cluster dividere, sono state vagliate diverse possibilità:

- Un approccio comune è quello di selezionare il cluster con la massima cardinalità. Questo dà priorità alla produzione di cluster bilanciati rispetto alle dimensioni;
- Un altro modo è calcolare il raggio di ogni cluster, il quale è la massima distanza tra i punti e il centroide. Quindi si seleziona il cluster con il raggio maggiore;
- Un altro approccio è calcolare il diametro di ogni cluster, il quale è la massima distanza tra ogni coppia di punti di un cluster. Come prima, si va a scegliere il cluster che ha il diametro maggiore.

Data la sua immediatezza, si è scelto di utilizzare il primo approccio.

Vengono ora mostrati dei test che rappresentano i diversi esperimenti eseguiti. Prima viene mostrato un dataset e dei testi dove il clustering aggregativo e divisivo sono stati adoperati. In seguito i due metodi sono presentati i risultati dei due metodi di clustering partizionale. Infine sarà dato un quadro generale delle differenze riscontrate.

### 3.1 Aggregativo e Divisivo

Il primo dataset, mostrato in figura 2, contiene 1000 elementi distintamente raggruppati con colori diversi in 5 diverse zone, così che sarà ovvio quali cluster dovrebbero essere formati.

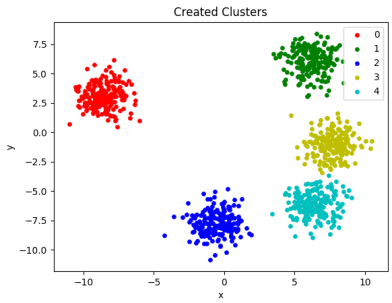
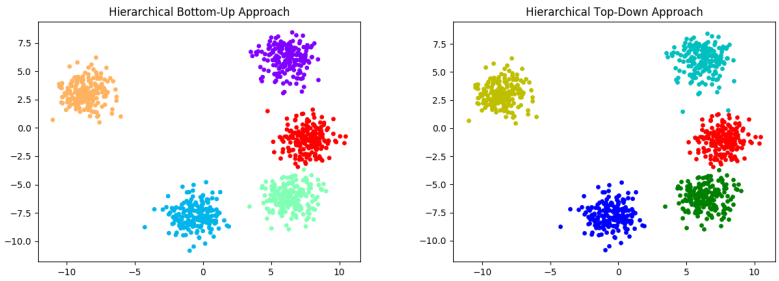



Figura 2: Dataset di 1000 elementi raggruppati in 5 diverse zone, etichettati con colori diversi. Questi saranno i cluster che le varie tecniche formeranno.

Una prima applicazione delle due tecniche è stata quella di lanciare le procedure prefissando il numero di cluster da ottenere. In figura 3 e 4 sono mostrati alcuni risultati. Per entrambe le tecniche, se richiesto di trovare esattamente il numero di cluster che il dataset prevede (in questo caso 5), allora vengono identificati in modo molto simile i vari gruppi. La tecnica aggregativa è magari poco più precisa per quegli elementi ai confini tra due diversi cluster.



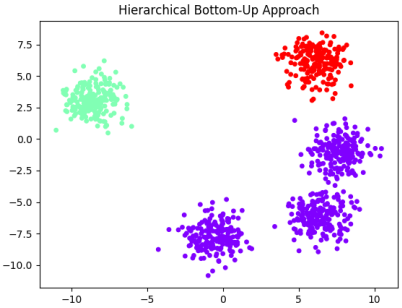

(a) Aggregativo per i 5 cluster.

(b) Divisivo per i 5 cluster.

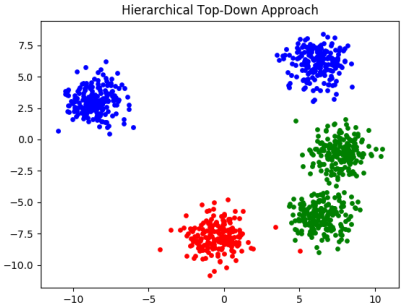
Figura 3: I risultati delle due tecniche per il clustering gerarchico sul dataset mostrato in figura 2. La formazione dei cluster è molto simile per entrambi gli algoritmi, i gruppi principali identificati sono gli stessi. Piccole differenze per gli elementi molto vicini a 2 o più cluster.

Invece iniziano ad esserci delle differenze anche sostanziali quando si vuole avere un numero minore o maggiore di cluster. Le figure 4a e 4b mostrano due istanze in cui si vogliono 3 cluster. In questo caso i cluster trovati sono tutti e 3 diversi per i due algoritmi. Spesso pochi, separati cluster possono risultare diversi, ma a seconda dell'approccio utilizzato per la scelta del cluster da dividere, i risultati possono differire molto da ciò che si ottiene con la tecnica aggregativa.

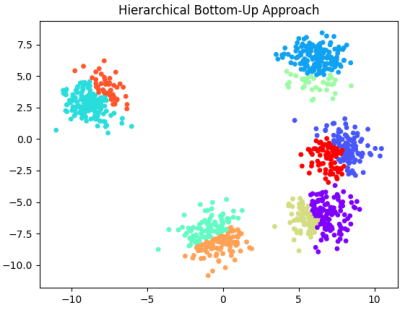
Si è poi raddoppiato il numero di cluster voluto, da 5 a 10, i cui risultati sono mostrati in 4c e 4d. Entrambe le tecniche hanno seguito la strada di dividere in due ciascuna delle 5 zone dove sono raggruppati gli elementi, ottenendo 10 cluster. Qui si riesce a notare bene come il clustering divisivo faccia dei tagli decisamente più netti per dividere i gruppi.



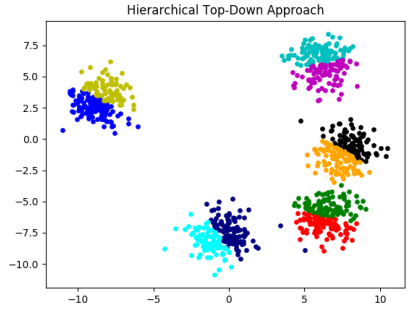

(a) Aggregativo per 3 cluster.




(b) Divisivo per 3 cluster.




(c) Aggregativo per 10 cluster.




(d) Divisivo per 10 cluster.

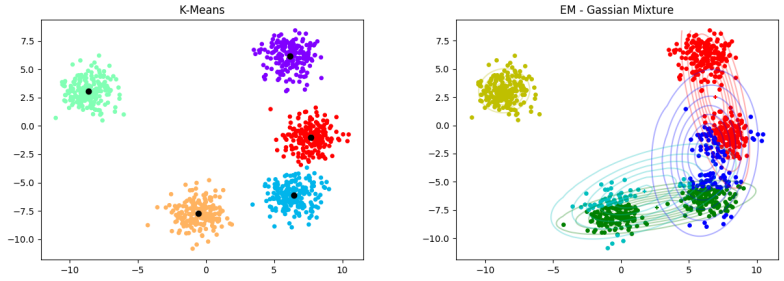
Figura 4: Sempre con lo stesso dataset (dalla figura 2), sono qui mostrati i risultati del clustering gerarchico prefissando l'identificazione di 3 cluster, (a) e (b), e 10 cluster, (c) e (d), invece di 5.

Oltre queste differenze, anche il tempo richiesto di esecuzione è stato abbastanza diverso tra le due tecniche. Il clustering aggregativo è stato decisamente più veloce nel completare i calcoli. Questo è chiaramente dovuto al non utilizzare una seconda tecnica di clustering per formare nuovi cluster, cosa che invece il clustering divisivo fa per dividere gli elementi.

### 3.2 K-means ed EM

Una prima differenza sostanziale di K-means, anche con le tecniche precedenti, è la velocità di esecuzione. In tutti i test il calcolo dei cluster è stato immediato. Decisamente un vantaggio non da poco.

In figura 5 vengono mostrati i due algoritmi applicato al dataset precedente. K-means non solo è stato piuttosto veloce, ma ha formato gli stessi 5 cluster descritti dal dataset. EM invece, essendo un algoritmo di soft clustering, mostra come gli elementi rientrino in più di un cluster. Gli elementi in giallo appartengono ad un unico cluster, essendo essi ben separati dagli altri. Per il resto invece, essendo tra loro più vicini, non c'è una divisione ovvia. Anzi, molti elementi fanno parte di due o più cluster con probabilità molto simili. Ad esempio, i punti rossi che sono molto vicini ai punti blu hanno una probabilità molto alta sia per il cluster blu che per il rosso, ma di poco sono stati classificati come rossi.



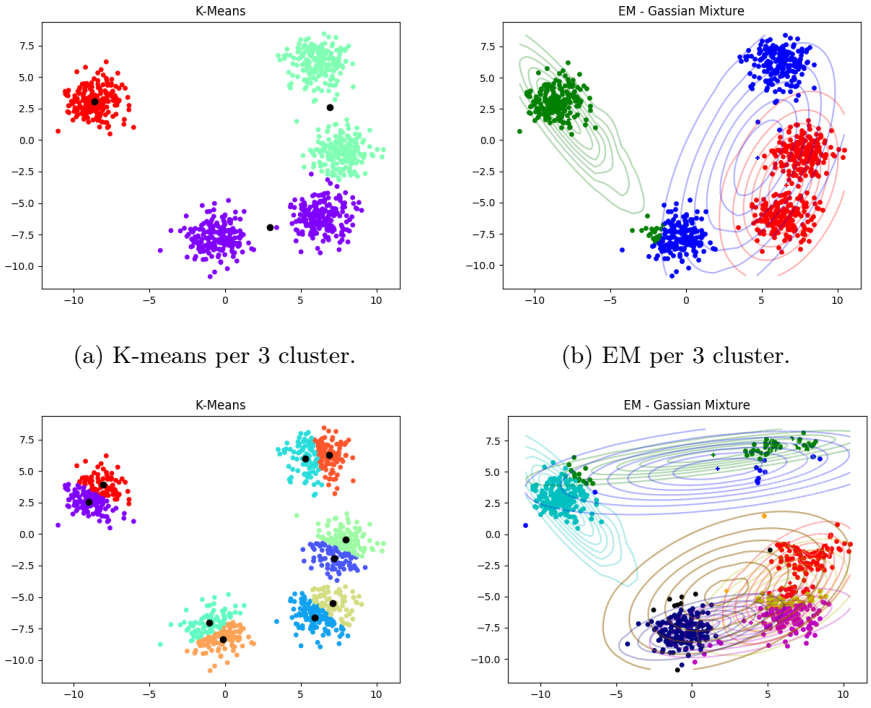

(a) K-means per i 5 cluster.

(b) EM per i 5 cluster.

Figura 5: I risultati delle due tecniche per il clustering partizionale sul dataset mostrato in figura 2. In (a) K-means riesce a partizionare gli elementi esattamente come nel dataset originale, i punti neri sono i centroidi. In (b) EM invece forma 1 cluster ben definito (colore giallo), mentre con gli altri 4 c'è un netto soft clustering, dove alcuni elementi fanno parte perfino di 3 cluster diversi.

Anche per queste due tecniche si è voluto testare l'identificazione prima di 3 cluster, poi di 10. In 6a K-means trova 3 ottimi cluster, di nuovo diversi da quelli trovati dalle tecniche gerarchiche. Uno è chiaramente ben separato dagli altri, mentre gli altri due sono ognuno due gruppi vicini tra di loro. Questo è

probabilmente il miglior modo per suddividere quel dataset in 3 cluster. Invece in 6b EM suddivide il dataset in modo ulteriormente diverso, però i 4 gruppi a destra in realtà condividono l'appartenenza in 2 cluster.




(a) K-means per 3 cluster.

(b) EM per 3 cluster.

(c) K-means per 10 cluster.

(d) EM per 10 cluster.

Figura 6: I Risultati del clustering partizionale prefissando l'identificazione di 3 cluster, (a) e (b), e 10 cluster, (c) e (d).

Per il caso dei 10 cluster da creare, si nota come in 6c c'è una divisione abbastanza netta come nel caso del clustering divisivo, il che è prevedibile in quando la tecnica divisiva implementata utilizza K-means. Da EM, invece, non traspare una chiara struttura dei cluster quando se ne richiede un alto numero, sopratutto se gli elementi sono ben distinti in un numero inferiore di gruppi. Questo ci dice che, ignorando per il momento i tempi di esecuzione più lunghi, in una applicazione concreta questa tecnica va usata più che altro quando c'è un reale bisogno di fare soft clustering con un numero appropriato di cluster da identificare.

## 4 Conclusioni

Tenendo conto sia del funzionamento degli algoritmi sia di questi ed altri test, si è notato che il clustering gerarchico può essere più adatto per analisi più dettagliate, come quando c'è un bisogno di informazioni maggiori sulle relazioni tra gli elementi. L'approccio aggregativo è più facile da implementare di quello divisivo, ma entrambi hanno i propri casi d'uso. Sono però meno efficienti del clustering partizionale, dove K-means è il migliore. Per questo motivo quando c'è un reale bisogno di efficienza, senza dubbio il clustering partizionale è da scegliere. Inoltre, K-means è anche l'algoritmo concettualmente più semplice. Se i suoi risultati sono già sufficienti allora è un'ottima scelta. Infine EM ha i suoi casi d'uso, ad esempio quando è utile fare soft clustering oppure bisogna usare modelli probabilistici anche complessi, anche con dataset sofisticati e non necessariamente rappresentati in spazi euclidei (cosa richiesta da K-means).

## Riferimenti bibliografici


1. Christopher D. Manning, Hinrich Schtüze
   *Foundations of Statistical Natural Language Processing.* The MIT Press Cambridge, Massachusetts 1999.
2. Chris Ding, Xiaofeng He *Cluster merging and splitting in hierarchical clustering algorithms.* University of California, Berkeley,