PARTITION METHOD

È un modo di fare clustering dove si partiziona il Dataset in K cluster distinti, dove K è scelto a priori.
Questi tipi di algoritmi hanno l'obiettivo di minimizzare la seguente cost function:

$$\delta = \sum_{i=1}^{K} \sum_{p \in C_i} (p - c_i)^2$$

Dove con c_i si intende il centroide o medoide del cluster.

K-MEANS

Trova K cluster e definisce ogni centroide come la media dei punti del cluster.

Input: K e D.
Output: K-cluster

PASSI

1) Scegliere arbitrariamente K punti in D che formeranno i K cluster inizialmente.
2) Associare ogni punto in D al cluster con centroide più vicino al punto in analisi.
3) Aggiornare i K centroidi con le nuove medie
4) Ripetere finché non ci sono più cambiamenti nei cluster trovati

Nel punto 2) non abbiamo bisogno che gli attributi siano numerici, sappiamo come valutare la distanza fra valori nominali. Mentre nel punto 3) è necessario che i punti siano numerici per via della media.

PRO

1) È molto efficiente, e ha complessità $O(t \cdot K \cdot n)$ dove t = iterazioni, K = # cluster, $n = |D|$.
Normalmente vale che $K, t \ll n$.

CONTRO

1) È applicabile a oggetti che stanno in uno spazio continuo e n-dimensionale.

2) Bisogna scegliere K a priori. Ma ci sono metodi per determinare il miglior K.

3) Molto sensibile a rumore e outliers.

4) Trova solo cluster di forma convessa per via della definizione di distanza.

VARIAZIONI DI K-MEANS

1) Selezionando i migliori K punti iniziali
2) Valutando differentemente la metriche di dissimilarità
3) Diverse strategie per calcolare la media dei cluster

GESTIRE DATI CATEGORICI: K-MODES

Sappiamo che i dati categorici non possono essere gestiti da K-means per via del calcolo della media. Perciò, non usiamo la media ma la moda (ovvero il valore che occorre più frequentemente nel DB). Bisogna usare un metodo basato sulla frequenza per aggiornare la moda di volta in volta.

K-MEDOIDS (PAM)

Il problema più importante di K-means sono gli outliers che influenzano il risultato.
Invece di usare la media come centroide si usa il medoide, ovvero il punto che è situato più al centro nel cluster.
Ogni oggetto non rappresentativo viene testato al posto del medoide attuale nel cluster.
Si sceglie la configurazione che abbassa la cost function

$$\delta = \sum_{i=1}^{K} \sum_{p \in C_i} |p - c_i|$$

Si sceglie di cambiare configurazione se

$$S = \delta_{old} - \delta_{new} < 0.$$

PROBLEMA DELLA COMPLESSITÀ

Non è un algoritmo scalabile e va in difficoltà con grandi moli di dati. Ha una complessità $O(k(n-k)^2)$ per ogni iterazione.

CLARA

L'idea è quella di usare PAM per un campione del dataset e non su tutto D così da abbassare la complessità. Se usiamo campioni rappresentativi di D possiamo raggiungere buoni risultati, tenendo sempre conto che se il medoide ottimale non è presente nel campione selezionato allora è impossibile raggiungere un risultato ottimale.

Per una migliore approssimazione si usano più campioni e il risultato migliore viene messo in output.

PASSI

Da ripetere per $i = 1 \dots R$:

1) Disegnare un campione con oggetti presi casualmente da D, e eseguire PAM sul campione scegliendo K.
2) Ogni oggetto in D, viene associato a uno dei K medoidi valutati con PAM.
3) Si calcola la dissimilarità media dell'intero D con i cluster trovati precedentemente. Si mantiene il valore se questa metrica è migliore rispetto alle iterazioni precedenti.

COMPLESSITÀ DI CLARA:

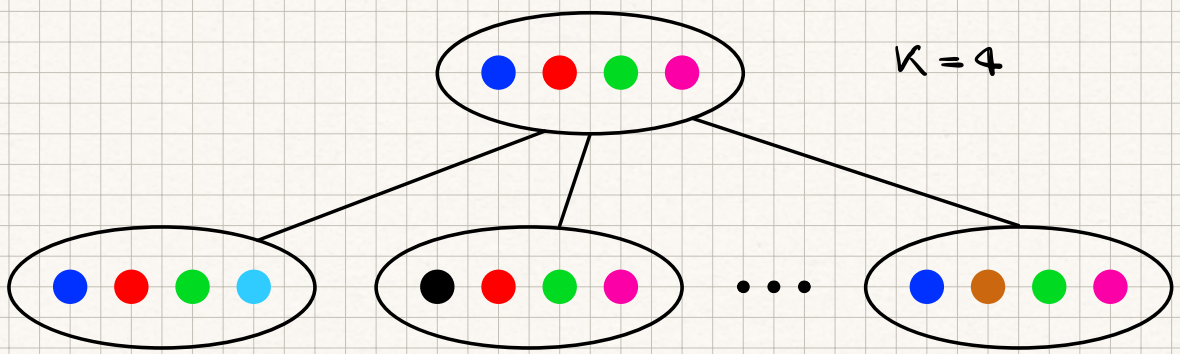
$$O(Ks^2 + K(n-K))$$

Dove s è la dimensione del campione.

CONTRO: È condizionato dal campione preso in considerazione, perché se non è rappresentativo non arriviamo all'ottimo.

CLARANS

La ricerca dei punti significativi può essere ridotta alla ricerca in un grafo dove ogni oggetto di D è considerato come potenziale soluzione.
La regola per costruire il grafo di ricerca è la seguente: due nodi sono considerati vicini se il loro set di medoidi differisce di un solo punto. Ogni nodo del grafo contiene quindi una possibile soluzione in termini di medoidi.



Ad ogni nodo è associato un costo definito come la dissimilarità totale fra ogni punto e il medoide del suo cluster. Bisogna cercare il nodo con costo minimo.

Dato che esaminare tutti i $K(n-K)$ vicini è deleterio, CLARANS a ogni passo costruisce un vicino del nodo corrente e lo analizza, seguendo quella strada finché il costo non raggiunge l'ottimo locale.
Una volta raggiunto l'ottimo CLARANS non si ferma ma ricomincia per un numero massimo di iterazioni espresse dall'utente come parametro.

Differenze fra CLARA e CLARANS

La differenza più lampante è che CLARA lavora su un sample del dataset, creato all'inizio della sua esecuzione confinando di fatti la sua ricerca.
CLARANS, d'altro canto, usa sempre dei sample, ma li crea diversi a ogni iterazioni, esplorando i vicini che migliorano la cost function.

COMPLESSITÀ : $O(n)$

ALGORITMO

```
CLARANS (numLoc, maxneighbor) // Parametri
{
    i = 1
    while (i < numLoc) // finché non faccio tot iterazioni
    {
        minCost = +∞
        currentNode = random
        j = 1
        while (j < maxneighbor) // finché non controllo
                                // tot vicini
        {
            S = randomNeighbor (currentNode)
            if (cost(S) < cost (currentNode))
            {
                currentNode = S
                j = 1
                continue
            }
            else
            {   j++
            }
```

// Ovvero considero un nodo
// che differenzia di solo un
// centroide da quello corrente

```
        }
        // j > maxneighbor
        if (cost(currentNode) < mincost)
        {   mincost = cost(currentNode)
            bestNode = currentNode
        }
    i++
    }
} return bestNode.
```

PARAMETRI DI CLARANS

1) Numero di iterazioni : numLoc
2) Numero di vicini da analizzare : maxneighbor
3) Min Cost = +∞. : mincost