Soft Clustering e Rough K-Means

Elia Mercatanti

Università degli Studi di Firenze
Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Magistrale in Informatica - Data Science

Progetto MASL, Gennaio 2021

Cluster Analysis

- È uno degli strumenti più utilizzati nell'analisi dei dati.
- Negli ultimi decenni c'è stato un crescente interesse verso tali tecniche, in particolare il **soft clustering**.

Cluster Analysis

- Mira a determinare un piccolo numero k di gruppi omogenei da un insieme di n oggetti secondo un data misura della dissomiglianza basata su p variabili osservate $X_1, \ldots, X_p$.
- Ha il compito di suddividere un set di dati in gruppi (**clusters**) in modo significativo ed utile.

Soft Clustering

- Esistono però oggetti che hanno caratteristiche intermedie tra i cluster, spesso non possono essere chiaramente assegnati.
- L'approccio classico (*hard*) al raggruppamento porta ad un'assegnazione non realistica, gli oggetti sono costretti ad appartenere a un solo cluster.

Soft Clustering

- L'approccio **soft** nasce per superare questo inconveniente.
- Idea: ogni dato può appartenere a più di un cluster.
- Esistono almeno quattro tipi di approcci di clustering *soft*: *fuzzy*, possibilistico, basato su modelli e *rough*.

Soft Clustering - Quattro Approcci

- **Fuzzy**: assegna dati ai cluster secondo un certo grado, *membership degree*, compreso tra 0 a 1.
- **Possibilistico**: rilassa i vincoli somma-unità dei *membership degrees*, aggiungendo un termine di penalizzazione.
- **Model-Based**: producono una partizione soft degli oggetti e la probabilità a posteriori di un componente/cluster viene presa come dei *membership degrees*.
- **Rough**: si basa sulla nozione di *rough set*. Ciascuno *rough set* è composto da una *lower approximation* (LA) e una *upper approximation* (UA). Dalle distanze tra ogni oggetto e ogni prototipo, un dato appartiene a una LA o a due o più UA.

Fuzzy Clustering - Fuzzy K-Means

- I dati vengono assegnati ai cluster in base ad un grado di appartenenza.
- Grado ≈ 1, oggetto simile al prototipo di quel cluster, grado 0 ne indica la non appartenenza, lontano dal relativo prototipo.
- Estensione dell'algoritmo *K-Means* al caso *fuzzy* (**FkM**).
- Ha lo scopo di determinare una partizione *fuzzy* di n oggetti in k cluster risolvendo il seguente problema di minimizzazione:

$$\min_{\mathrm{U},\mathrm{H}} J_{FkM} = \sum_{i=1}^{n} \sum_{g=1}^{k} u_{ig}^{m} d^2(\mathrm{x}_i, \mathrm{h}_g),$$

$$\text{s.t.} \quad u_{ig} \in [0,1], \quad \sum_{g=1}^{k} u_{ig} = 1, \quad i = 1, \ldots, n, \quad g = 1, \ldots, k,$$

Fuzzy Clustering - Fuzzy K-Means

- $m > 1$ è il parametro di **fuzziness**, di solito fissato tra 1.5 e 2.
- La soluzione è ottenuta mediante un algoritmo iterativo. Con il metodo del moltiplicatore Lagrangiano (λ).
- Impostando le derivate parziali della funzione Lagrangiana (L) rispetto a u_{ig} e λ uguale a 0.
- Solo l'ultimo vincolo viene utilizzato, il primo è soddisfatto automaticamente.
- Fissando u_{ig}, otteniamo h_g ponendo uguale a 0 le derivate parziali di L rispetto a h_g. La soluzione iterativa è la seguente:

$$u_{ig} = \frac{1}{\sum_{g'=1}^{k} \left(\frac{d^2(\mathrm{x}_i, \mathrm{h}_g)}{d^2(\mathrm{x}_i, \mathrm{h}_{g'})} \right)^{\frac{1}{m-1}}} \qquad \mathrm{h}_g = \frac{\sum_{i=1}^{n} u_{ig}^m \mathrm{x}_i}{\sum_{i=1}^{n} u_{ig}^m}.$$

Fuzzy Clustering - Pro e Contro

Pro:

- I gradi di appartenenza sono inversamente correlati alle relative dissomiglianze tra gli oggetti e i centroidi.
- Per questo motivo, i gradi di appartenenza possono essere interpretati come gradi di condivisione.
- Semplice ed efficiente.

Contro:

- Le prestazioni dei metodi basati sul *K-Means* sono influenzati dai valori anomali (*outliers*).
- Dovuto ai vincoli di somma unitaria dei *membership degrees*. Gli oggetti anomali assegnati ai cluster influenzano i centroidi.
  - *Soluzione*: uso del *noise cluster*

- Per cercare di superare l'inconveniente dei valori anomali.
- Rilassare i vincoli della somma unitaria sui *membership degrees*, aggiungendo un termine di penalizzazione.
- Gradi di appartenenza interpretati come gradi di compatibilità degli oggetti con i cluster.
- L'algoritmo di clustering possibilistico più noto è il metodo di clustering **PkM**, che può essere formalizzato come di seguito:

$$\min_{\mathrm{T},\mathrm{H}} J_{PkM} = \sum_{i=1}^{n} \sum_{g=1}^{k} t_{ig}^{\eta} d^2(\mathrm{x}_i, \mathrm{h}_g) + \sum_{g=1}^{k} \gamma_g \sum_{i=1}^{n} (1 - t_{ig})^{\eta},$$

$$\text{s.t. } t_{ig} \in [0, 1], \quad i = 1, \ldots, n, \quad g = 1, \ldots, k,$$

Possibilistic Clustering - Possibilistic K-Means

- γ_g è *cluster-specific*, regola l'importanza dei cluster:

$$\gamma_g = \gamma \frac{\sum_{i=1}^{n} u_{ig}^m d^2(\mathrm{x}_i, \mathrm{h}_g)}{\sum_{i=1}^{n} u_{ig}^m}, \quad g = 1, \dots, k,$$

- $\gamma = 1$, h_g e i vari u_{ig} sono ottenuti dal FkM.
- I γ_g rappresentano il peso relativo del secondo termine della funzione obbiettivo rispetto al primo.
- η (> 1) è il **fuzzifier**.
- Il secondo termine evita inoltre la soluzione banale con $\mathrm{T} = 0$.
- Soluzione iterativa del PkM:

$$t_{ig} = \frac{1}{1 + \left(\frac{d^2(\mathrm{x}_i, \mathrm{h}_g)}{\gamma_g}\right)^{\frac{1}{\eta - 1}}} \qquad \mathrm{h}_g = \frac{\sum_{i=1}^{n} t_{ig}^{\eta} \mathrm{x}_i}{\sum_{i=1}^{n} t_{ig}^{\eta}}.$$

Possibilistic Clustering - Pro e Contro

Pro:

- I gradi di tipicità, t_{ig}, sono inversamente correlati alle differenze tra le osservazioni e i centroidi.
- Calcolati considerando solo la dissomiglianza tra l'osservazione e il centroide più vicino.
- I valori anomali risultano lontani da tutti i centroidi. Di solito dunque hanno gradi di tipicità vicini allo 0.

Contro:

- Può soffrire del rischio di cluster coincidenti, dovuto all'assenza del vincolo alla somma dei gradi di tipicità.
  - *Rimedio*: l'uso di centroidi iniziali razionali (non casuali).
  - *Rimedio*: aggiungere un termine di repulsione tra i centroidi.

Model-Based Clustering

- Presuppone che i dati siano generati da un modello statistico.
- In questo contesto, i dati seguono una mistura di distribuzioni.
- Dato $\mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n) \in \mathbb{R}_p$, si presume che x_i derivi da una mistura finita di funzioni di densità di probabilità:

$$f(\mathrm{x}_i;\ \mathbf{\Phi}) = \sum_{g=1}^{k} \pi_g f(\mathrm{x}_i|\theta_g),$$

$$dove \quad \pi_g,\ g = 1, \ldots, k \quad s.t. \quad \pi_g > 0 \quad e \quad \sum_{g=1}^{k} \pi_g = 1$$

- Stima dei parametri $\mathbf{\Phi}$ con l'approccio della massima verosimiglianza, eseguita applicando l'algoritmo EM.
- La partizione *soft* dei dati ottenuta mediante le probabilità a posteriori, utilizzando la regola del massimo a posteriori.

Model-Based - Finite Mixture of Gaussian Densities

La mistura di distribuzione Gaussiane è la più utilizzata. La miscela finita di densità gaussiane (**FMG**) è quindi data da:

$$f(x_i;\ \mathbf{\Phi}) = \sum_{g=1}^{k} \pi_g \phi(x_i|\mu_g, \mathbf{\Sigma}_g),$$

- I cluster ellissoidali centrati sul vettore medio μ_g sono generati dal modello precedente.
- Il parametro $\mathbf{\Sigma}_g$ controlla le altre proprietà geometriche di ogni cluster.

Le FMG sono sensibili ai dati anomali. Soluzioni:

- Adottate una mistura di distribuzioni t.
- Usare il *trimming*, dove i valori anomali vengono scartati.

Rough Clustering - Proprietà

Un **rough set** C_g è definito mediante la sua **lower approximation** (**LA**) e la sua **upper approximation** (**UA**), ovvero, $\underline{A}(C_g)$ e $\overline{A}(C_g)$. LA e UA di C_g devono soddisfare le seguenti proprietà:

(P1) Un oggetto x_i può far parte al massimo di una LA.

(P2) $\mathrm{x}_i \in \underline{A}(C_g) \Rightarrow \mathrm{x}_i \in \overline{A}(C_g)$.

(P3) x_i non fa parte di alcuna LA $\Leftrightarrow$ x_i appartiene a due o più UA.

- Lo scopo è determinare se un oggetto appartiene alla UA o LA di un cluster.
- Per l'assegnazione di x_i, data una soglia ψ, vengono utilizzati i rapporti $d(\mathrm{x}_i, \mathrm{c}_g)/d(\mathrm{x}_i, \mathrm{c}_{g'})$ con $g = 1, \ldots, k$ e $g' = 1, \ldots, k$.

Rough Clustering - Regole di Assegnazione

Per ricavare le varie UA e LA sono utilizzate le seguenti regole:

1. Se $d(\mathrm{x}_i, \mathrm{h}_g)$ è il minimo per $1 \leq g \leq k$ e $d(\mathrm{x}_i, \mathrm{h}_g)/d(\mathrm{x}_i, \mathrm{h}_{g'}) \leq \psi$ per ogni coppia (g, g') allora $\mathrm{x}_i \in \overline{A}(C_g)$ e $\mathrm{x}_i \in \overline{A}(C_{g'})$.
   - La proprietà (P3) viene dunque soddisfatta.
2. Altrimenti, $\mathrm{x}_i \in \underline{A}(C_g)$ tale che $d(\mathrm{x}_i, \mathrm{h}_g)$ è il minimo per $1 \leq g \leq k$.
   - Per la proprietà (P2), $\mathrm{x}_i \in \overline{A}(C_g)$.
   - Segue che anche la proprietà (P1) è soddisfatta.

- Un cluster è descritto da due approssimazioni *hard*, una LA e una UA (o regione di confine).
- Un dato ha due gradi di appartenenza bivalenti al cluster g, uno per la sua LA e uno per la UA:

$$\mu_{ig}^{LA} \in \{0, 1\} \qquad \mu_{ig}^{UA} \in \{0, 1\}.$$

- LA: oggetti che appartengono al cluster. UA oggetti che potrebbero appartenere al cluster o meno.
- Può essere considerato come un metodo *soft* a causa della regione di confine per gestire l'incertezza.
- I dati vengono assegnati alle LA o UA utilizzando le regole (1) o (2) precedenti.

I centroidi sono calcolati come segue. Se le cardinalità $|\underline{A}(c_g)|$ e $|\overline{A}(c_g) - \underline{A}(c_g)|$ non sono uguali a 0, avremo che:

$$h_g = w_l \times \frac{\sum_{x_i \in \underline{A}(c_g)} x_i}{|\underline{A}(c_g)|} + w_u \times \frac{\sum_{x_i \in \overline{A}(c_g) - \underline{A}(c_g)} x_i}{|\overline{A}(c_g) - \underline{A}(c_g)|},$$

con $w_l > 0$ e $w_u > 0$ tali che $w_l + w_u = 1$ (solitamente $w_l > w_u$).

Se $|\underline{A}(c_g)| = 0$ o $|\overline{A}(c_g) - \underline{A}(c_g)| = 0$ di conseguenza uno dei due rapporti viene annullato.

```r
initialize_means = function(data_matrix, num_clusters, means_matrix) {

  if(is.matrix(means_matrix)) { # means pre-defined # no action required

  }else if (means_matrix == 1) {  # random means

    num_features = ncol(data_matrix)
    means_matrix = matrix(0, nrow=num_clusters, ncol=num_features)
    for (i in 1:num_features) {
      means_matrix[,i] = c(runif(num_clusters, min(data_matrix[,i]),
                                 max(data_matrix[,i])))
    }

  }else if (means_matrix == 2) { # maximum distance means

    means_objects = seq(length=num_clusters, from=0, by=0)
    objects_dist_matrix = as.matrix(dist(data_matrix))

    pos_vector = which(objects_dist_matrix == max(objects_dist_matrix),
                       arr.ind = TRUE)
    means_objects[1] = pos_vector[1,1]
    means_objects[2] = pos_vector[1,2]

    for(i in seq(length=(num_clusters-2), from=3, by=1) ) {
      means_objects[i] = which.max(
        colSums(objects_dist_matrix[means_objects, -means_objects]))
    }

    means_matrix = data_matrix[means_objects, ]

  }
```

```
# Assign object to upper approximation
assign_upper_approx = function(dataset, means_matrix, threshold) {

  num_obs   = nrow(dataset)
  num_clusters = nrow(means_matrix)

  distances_to_clusters = seq(length=num_clusters, from=0, by=0)

  upper_approx_matrix = matrix(0, nrow = num_obs, ncol = num_clusters )

  for (i in 1:num_obs) {

    # distances_to_clusters from object i to all clusters j
    for (j in 1:num_clusters){
      distances_to_clusters[j] = sum( (dataset[i,] - means_matrix[j,] )^2 )
    }

    min_distance = max(min(distances_to_clusters), 1e-99)

    # Includes the closest objects.
    closest_clusters_idx = which((distances_to_clusters / min_distance)
                                 <= threshold)

    upper_approx_matrix[i, closest_clusters_idx] = 1

  }

  return(upper_approx_matrix)
}
```

Implementazione in R - Assegnazioni alle LA

```r
# Assign object to lower approximation out of an upper approximation.
assign_lower_approx = function(upper_approx_matrix) {

  # Initialization of lower_approx_matrix
  lower_approx_matrix = 0 * upper_approx_matrix

  object_idx = which( rowSums(upper_approx_matrix) == 1 )

  lower_approx_matrix[object_idx, ] = upper_approx_matrix[object_idx, ]

  return(lower_approx_matrix)
}
```

Implementazione in R - Ciclo Principale Rough K-Means

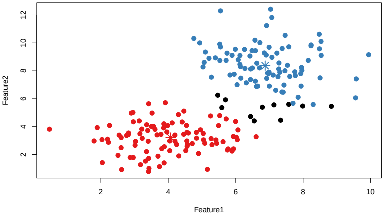
```r
while ( !identical(old_upper_approx_matrix, upper_approx_matrix)
        && iterations < max_iterations ) {

  # Lower Approximation (LA) Matrix and UA-LA matrix (boundary)
  lower_approx_matrix = assign_lower_approx(upper_approx_matrix)
  boundary_matrix = upper_approx_matrix - lower_approx_matrix

  # Calculate sums of observations in LA and boundary in every cluster
  means_matrix_lower = crossprod(lower_approx_matrix, dataset)
  means_matrix_boundary = crossprod(boundary_matrix, dataset)

  # Update means matrix
  for (i in 1:num_clusters) {

    # Dividers means calculation, cardinalities of LA and UA-LA (boundary)
    divider_lower_approx = sum(lower_approx_matrix[, i])
    divider_boundary = sum(boundary_matrix[, i])

    if (divider_lower_approx != 0 && divider_boundary != 0) {
      means_matrix_lower[i,] = means_matrix_lower[i,] /
                               divider_lower_approx
      means_matrix_boundary[i,] = means_matrix_boundary[i,] /
                                  divider_boundary
      means_matrix[i,] = weight_lower*means_matrix_lower[i,] +
                         (1-weight_lower)*means_matrix_boundary[i,]
    } else if (divider_boundary == 0) {
      means_matrix[i,] = means_matrix_lower[i,] / divider_lower_approx
    } else { # if(divider_lower_approx[,i]) == 0)
      means_matrix[i,] = means_matrix_boundary[i,] / divider_boundary
    }
  }
```

- Test su **FkM** (pacchetto R *fclust*), **PkM** (*ppclust*), **FMG** (*mclust*) e **RkM** (implementato).
- Valutazione delle prestazioni su 4 semplici dataset.
- 3 dataset sintetici 2d con 2 cluster: *DemoData*, *G2*, *Synth*.
- Dataset *Iris*, tre cluster, per sfruttarne le etichette di classe.

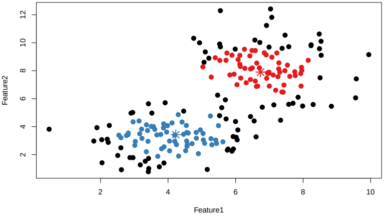
Indici Utilizzati:

- **Indici Interno**: *Fuzzy Silhouette Index*, si basa sul coefficiente di *silhouette* che combina le misure di coesione e separazione.
- **Indici Esterni**:
  - *Purity*, indica la misura in cui i cluster contengono una singola classe, valori da 0 a 1.
  - *Adjusted Rand Index* (ARI), è la versione corretta per il raggruppamento casuale di elementi dell'indice Rand.
- Tempi di esecuzione.

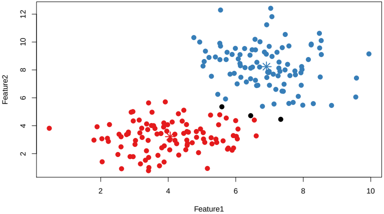
Risultati del Clustering sul Dataset DemoDataC2D2a



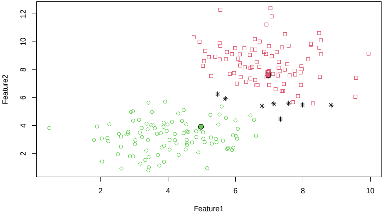
(a) **Fuzzy K-Means**



(b) **Possibilistic K-Means**



(c) **Finite Mixture Gaussian Model**



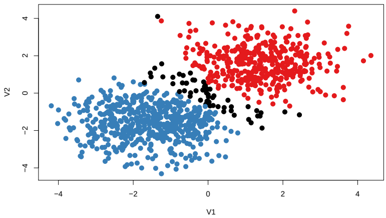
(d) **Rough K-Means**

Prototipi del Dataset *DemoDataC2D2a*

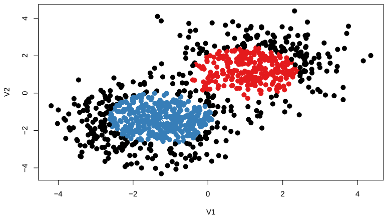
	FkM		PkM		FMG		RkM	
	Clus 1	Clus 2	Clus 1	Clus 2	Clus 1	Clus 2	Clus 1	Clus 2
Feature 1	4.072	6.874	4.230	6.739	4.058	6.909	4.975	6.967
Feature 2	3.213	8.369	3.430	7.905	3.217	8.234	3.911	7.610

Tabella: Prototipi del Dataset *DemoDataC2D2a*

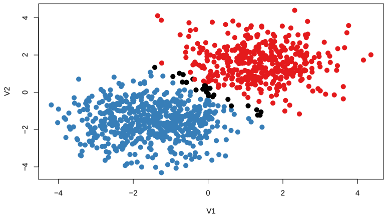
Risultati del Clustering sul Dataset *Synth*



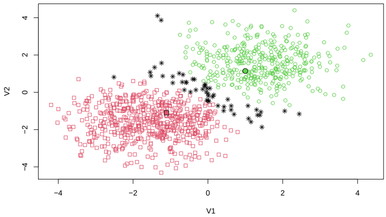
(a) **Fuzzy K-Means**



(b) **Possibilistic K-Means**

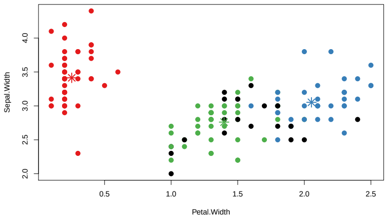


(c) **Finite Mixture Gaussian Model**

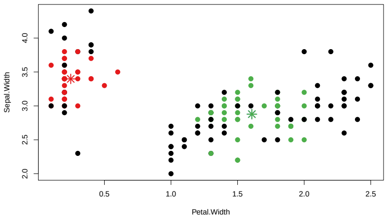


(d) **Rough K-Means**

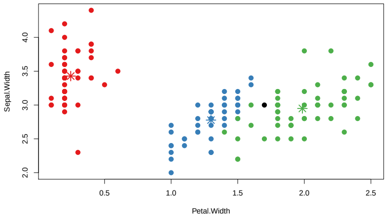
Risultati del Clustering sul Dataset *Iris*



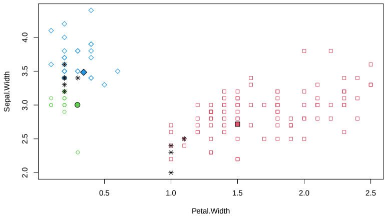
(a) **Fuzzy K-Means**



(b) **Possibilistic K-Means**



(c) **Finite Mixture Gaussian Model**



(d) **Rough K-Means**

Indici di Valutazione e Tempi di Esecuzione

- Fuzzy Silhouette Index -	DemoDataC2D2a	G2	Synth	Iris
Fuzzy K-Means	0.8476	0.8525	0.8223	0.8091
Possibilistic K-Means	**0.8564**	**0.8633**	**0.8230**	**0.9538**
Model-Based Clustering	0.8280	0.8346	0.8038	0.6579
Rough K-Means	0.8370	0.8388	0.8118	0.7532

- External Indices -	Purity	ARI
Fuzzy K-Means	0.9667	0.9039
Possibilistic K-Means	0.6667	0.5681
Model-Based Clustering	0.9667	0.9039
Rough K-Means	0.5200	0.4413

- Execution Times -	DemoDataC2D2a	G2	Synth	Iris
Fuzzy K-Means	**0.0070**	**0.3590**	**0.1940**	**0.0050**
Possibilistic K-Means	0.7640	5.2400	4.7044	1.3802
Model-Based Clustering	0.0240	1.765	0.3060	0.0680
Rough K-Means	0.1170	0.5850	0.3500	0.0140

Grazie dell'Attenzione

Link Progetto GitHub:

https://github.com/elia-mercatanti/soft-clustering-rough-kmeans