# $k$-MEANS

Abstract. L'obiettivo è implementare l'algoritmo $k$-means. Vengono forniti una descrizione dell'algoritmo ad alto livello, alcuni suggerimenti sulle variabili e sui metodi da utilizzare ed infine lo pseudocodice.

## 1. Introduzione

L'algoritmo $k$-means è un algoritmo di *clustering*, ovvero di raggruppamento di oggetti simili tra loro. Gli oggetti sono descritti da *features* (caratteristiche, attributi), su cui si basa la valutazione della similarità. Si considerino ad esempio i dati contenuti nella seguente tabella.

| Paziente | Febbre | Aritmia |
|---|---|---|
| Rossi | 36.5 | si |
| Ferrari | 39.0 | no |
| Bianchi | 38.4 | no |
| Russo | 36.2 | si |

Volendo raggruppare in due gruppi (*cluster*) i quattro pazienti in base ai sintomi, sembra naturale formare i due gruppi {Rossi, Russo} e {Ferrari, Bianchi} (si noti che nella realtà la situazione è molto più complessa e di difficile soluzione).

L'algoritmo $k$-means divide i dati in $K$ gruppi, cercando di minimizzare le distanze tra il "centro" di ogni gruppo e gli oggetti che vi appartengono.

I passi dell'algoritmo sono i seguenti:

(1) Inizializzare il centro dei $K$ cluster con un oggetto scelto a caso senza reimmissione (ovvero non è possibile che uno stesso oggetto sia il centro di due cluster diversi)
(2) Assegnare ogni oggetto al cluster più vicino
(3) Calcolare il nuovo centro di ogni cluster
(4) Ripetere i passi 2 e 3 fino a quando la funzione obiettivo cambia a meno di un valore di soglia fissato

Per poterlo implementare è necessario specificare tre elementi: la *distanza* utilizzata, come calcolare il *centro* di ogni cluster, come calcolare la *funzione obiettivo*.

Supponendo che i dati siano tutti numerici, la distanza utilizzata è la *distanza euclidea*, ovvero, considerando due oggetti $m_1, m_2 \in M$ e $N$ caratteristiche, la distanza tra i due oggetti è

$$d(m_1, m_2) = \sqrt{\sum_{i=1}^{N} (m_{1,i} - m_{2,i})^2}$$

dove $m_{1,i}$ è il valore che l'oggetto $m_1$ assume per l' $i$-esima caratteristica.
Nell'esempio precedente (supponendo si = 1 e no = 0) avremmo $d$(Rossi,Bianchi) $= \sqrt{(36.5 - 38.4)^2 + (1 - 0)^2} = 2.15$

Il valore di ogni caratteristica $c_i$ del *centro* $c$ di un dato cluster $C$ si calcola come la media delle caratteristiche degli elementi del cluster:

$$c_i = \frac{1}{|C|} \sum_{m \in C} m_i$$

dove $c_i$ e $m_i$ sono i valori della feature $i \in N$ per il centro $c$ e per l'elemento $m$ di un dato cluster $C$. Quindi avremo $N$ valori $c_i$, uno per ogni feature. Il tutto per ognuno dei $K$ cluster.
Le caratteristiche del centro del cluster {Ferrari, Bianchi} sono date dai due valori (38.7,0) ottenuti come $\frac{39+38.4}{2} = 38.7$ e $\frac{0+0}{2}$ (= no).

Infine, avendo a disposizione distanza e centro è possibile calcolare la funzione obiettivo come

$$O = \sum_{i=1}^{K} \sum_{m \in C_i} d(m, c)$$

dove $d(m, c)$ è la distanza tra il centro di un cluster e l'elemento $m$ del cluster stesso. La funzione obiettivo quindi calcola la somma delle distanze di ogni elemento dal centro del cluster a cui appartiene.

## 2. Variabili e costanti

Supponiamo di avere $M$ oggetti descritti ciascuno da $N$ caratteristiche. Sono da prevedere (almeno) le seguenti variabili:

- Una matrice `dati` di dimensione $M \times N$ di tipo double
- Un array `cluster` di interi di dimensione $M$ in cui memorizzare per ogni oggetto il cluster di appartenenza
- Una matrice di double `centri` di dimensione $K \times N$ in cui memorizzare il centro dei $K$ cluster
- Una variabile di tipo double `obiettivo` contiene il valore della funzione obiettivo

Le costanti richieste sono (almeno)

- `k` (di tipo intero): numero di cluster
- `alfa` (di tipo double): soglia di terminazione dell'algoritmo
- `iter` (di tipo intero): numero massimo di iterazioni

*Suggerimento*: mettere come valore di `k` un numero tra 3 e 7, fare diverse prove con valori di `alfa` e `iter` diversi, ad esempio `alfa = 0.1` e `iter = 1000`. Quello che dovrebbe accadere è che diminuendo `alfa`, aumenta il numero di iterazioni necessarie per raggiungere la precisione richiesta.

## 3. Metodi

Sono da prevedere (almeno) i seguenti metodi:

- `InizializzaDati` che deve inizializzare la matrice `dati` con numeri casuali tra 0 e 1
- `InizializzaCluster` che inizializza la matrice `centri` con i valori di $k$ oggetti, scegliendo a caso (senza reimmissione) $k$ righe della matrice `dati` (si suppone quindi $k < M$).
- `AggiornaCentri` che calcola i valori del centro di ogni cluster
- `CalcolaCluster` che per ogni oggetto calcola il cluster di appartenenza e memorizza il risultato in `cluster`

- `CalcolaObiettivo` calcola il valore della funzione obiettivo

## 4. Pseudocodice

Si riporta di seguito un possibile pseudocodice per implementare l'algoritmo. Nota: *precisione* è la differenza della funzione obiettivo tra una iterazione e la successiva. Il ciclo principale termina quando si è raggiunta la precisione richiesta `alfa` o quando si raggiunge il numero massimo di iterazioni `iter`.

**begin**
  Chiedere all'utente i valori di $M$ ed $N$;
  `InizializzaDati`;
  Stampare la matrice `dati`;
  `obiettivo = 0`;
  `InizializzaCluster`;
  **do**
    `CalcolaCluster`;
    `AggiornaCentri`;
    `CalcolaObiettivo`;
  **while** *((precisione* $>$ `alfa`*) && (numero iterazioni* $<$ `iter`*))*;
  Stampare `cluster`;
  Stampare il numero di iterazioni effettuate;
  Stampare il valore della funzione obiettivo delle ultime due iterazioni;
  Stampare la precisione raggiunta;
  Stampare il motivo della terminazione: raggiunta precisione o numero massimo di iterazioni;
**end**