Data Mining in R
Approccio non supervisionato
ANALISI CLUSTER
3. Metodi non gerarchici

Laura Grassini

Tan, Steinbach, Kumar: Introduction to data mining, 2006, Addison Wesley
http://www-users.cs.umn.edu/~kumar/dmbook/index.php

Indice

I metodi non gerarchici

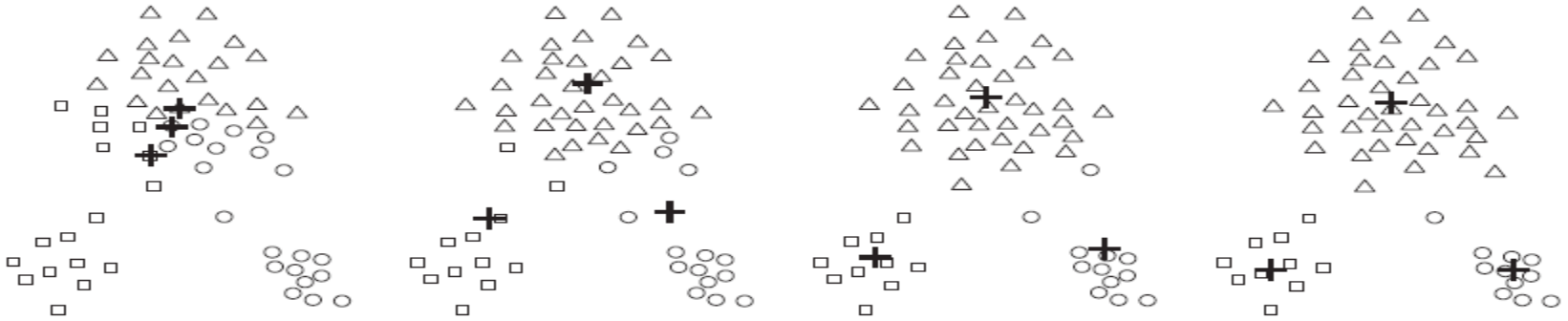
Dato il numero di cluster, arrivano ad una partizione utilizzando una funzione obiettivo da ottimizzare

- K-means
- Partitioning around medoids (PAM)

→ Si tratta di metodi **prototype based**

K-means: algoritmo base del metodo

1. Seleziona il numero di cluster *k*
2. Individua i *centroidi* iniziali (i prototipi dei cluster)
3. Ripeti se i centroidi cambiano ad ogni step
   a) Forma i *k* cluster assegnando ogni unità al centroide più vicino
   b) Ricalcola i centroidi (le **medie** delle variabili) dei k cluster (→in genere dopo il compimento dell'intero step di assegnazione)



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

K-means: assegnare l'unità al centroide più vicino

- Si usa in genere il **quadrato della distanza euclidea**
- Si **minimizza SSE** dove SSE è la somma dei quadrati delle distanze euclidee di ogni punto dal centroide – cioè il punto che ha come coordinate la media delle variabili – del gruppo di appartenenza (equivale, in pratica, alla **devianza interna** o **within**)

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} (dist(c_i, x))^2$$

$$c_i = \frac{1}{n_i} \sum_{x \in C_i} x$$

K-means: le distanze e le funzioni da ottimizzare più usate

Proximity Function	Centroid	Objective Function
Manhattan (L_1)	median	Minimize sum of the L_1 distance of an object to its cluster centroid
Squared Euclidean (L_2^2)	mean	Minimize sum of the squared L_2 distance of an object to its cluster centroid

K-means: caratteristiche

1) La variabilità interna, diminuisce ad ogni step
2) L'algoritmo converge sempre, non importa quali siano i centroidi iniziali scelti (ci vogliono meno di K^n iterazioni)
3) I cluster risultanti dipendono dai centroidi iniziali. E' quindi opportuno applicare l'algoritmo più volte e scegliere la soluzione con SSE minore (i software lo fanno automaticamente)
4) In ogni caso, si trova un minimo locale.

K-means: esempio in R

```
library(cluster)
data(mtcars)
x<-mtcars[,1:7]   # prime 7 colonne del dataset
## standardizziamo le variabili (problema già visto prima)
z<-scale(x,center = TRUE, scale = TRUE)
# voglio 4 gruppi
# lascio a R la scelta dei primi 4 centroidi
# uso la min di SSE
gruppi4<-kmeans(z, 4, method= 'Lloyd')
gruppi4
## dimensione dei 4 gruppi
table(gruppi4$cluster)
```

K-medoids

- E’ simile al k-means ma i centroidi dei cluster sono osservazioni effettive (medoids) e non sono creati dall’aggregazione dei dati individuali (media)
- Scelti *k* medoids, si spostano le unità in modo da massimizzare la somma della distanza euclidea al quadrato dai medoids.
- E’ computazionalmente più complesso e lungo del *k*-means, perché ad ogni step bisogna trovare il medoid nuovo fra gli oggetti che si trovano nel gruppo (mentre nel *k*-means si calcola il centroide facendo la medie di ogni variabile).

K-medoids in R

```
## criterio partition around medoids
## dataset z usato per il k-means
library(cluster)
pam4<-pam(z, 4)  # 4 gruppi
pam4
## dimensione dei 4 gruppi
table(pam4$clustering)
```

I medoids dei 4 gruppi sono rispettivamente

- Mazda RX4 Wag
- Fiat X1-9
- Merc 450SL
- Lincoln Continental

Individuazione di traiettorie con *k*-means

Dati: serie storiche o dati longitudinali (colonne) riferiti a più unità (righe)

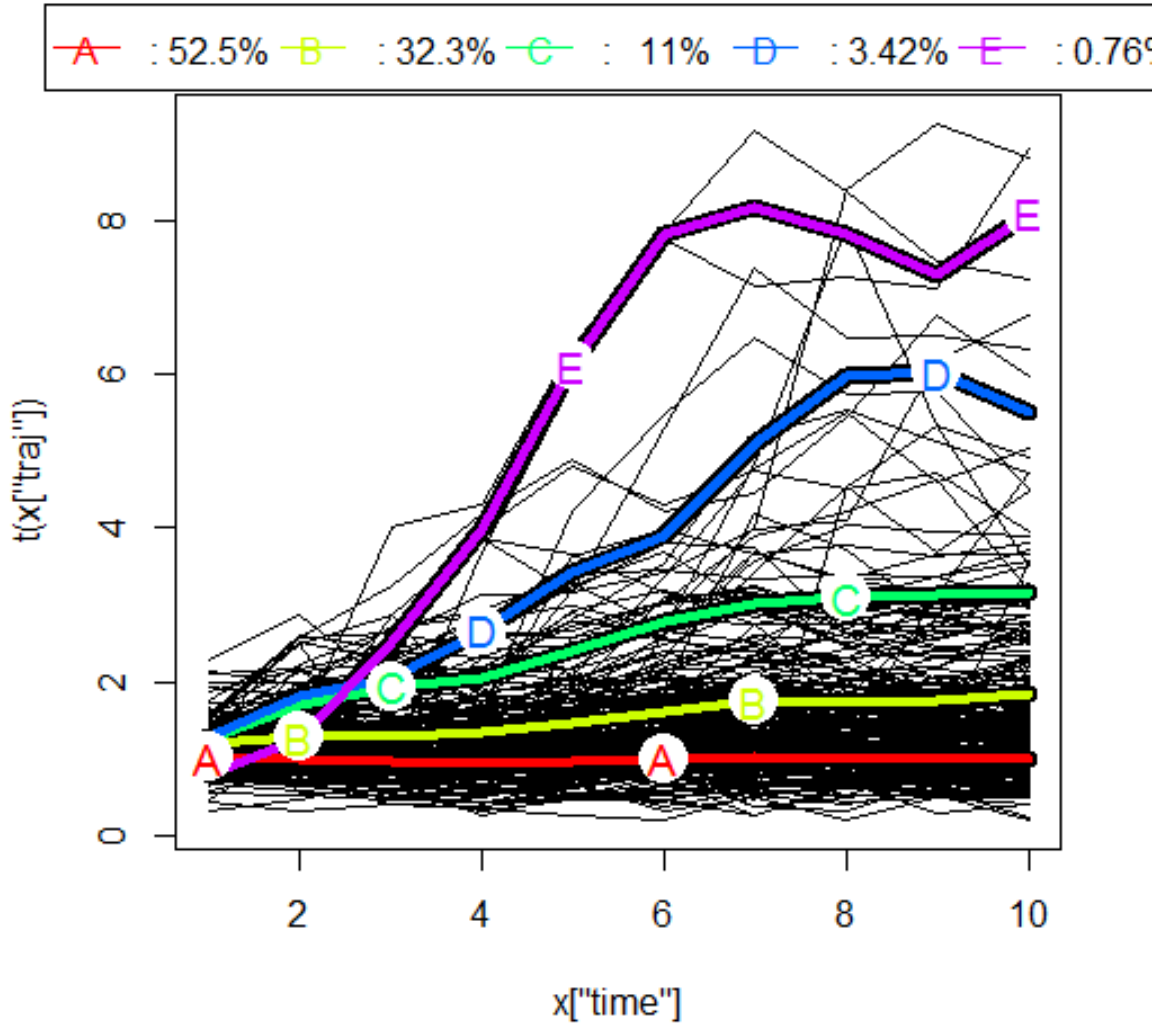
Obiettivo: trovare pattern temporali comuni

Come procede: ogni tempo rappresenta una variabile

Esempio: *arrivi turisti(t)/arrivi turisti(t-1)* nei comuni toscani (*n*=262)

codice_com	A2001	A2002	A2003	A2004	A2005	A2006	A2007	A2008	A2009	A2010
45001	1.175	1.550	1.160	1.044	1.514	1.359	1.167	1.239	1.760	1.985
45002	1.165	1.087	0.951	1.049	0.857	0.819	0.380	0.495	0.655	0.800
45003	1.089	0.997	1.096	0.970	1.015	1.053	0.908	0.738	0.636	0.737
45008	0.636	1.347	1.151	0.978	0.669	0.388	0.586	0.842	1.534	1.335
45010	0.993	1.002	1.010	0.774	0.958	1.001	0.916	0.765	0.882	0.829
45011	1.060	0.806	0.672	0.600	0.674	0.915	0.804	0.943	0.926	0.934

Esempio in R: libreria `kml`



```
### caricare l'area AREA_Cluster.Rdata e caricare la libreria kml
### l'area contiene il datframe «arrivi»
library(kml)
## creiamo il dataset con indicazione dell'ID e dei dati di cluster
DATI=cld(traj=arrivi[,2:11], id=arrivi$codice_com)
## eseguiamo il k-means (ogni anno è come se fosse una variabile)
### non occorre assegnare niente;
### i dati si aggiornano con i risultati dell'analisi
ncl=5  ### numero di cluster scelto
kml(DATI, nbClusters = ncl)
### per vedere i contenuti di DATI
slotNames(DATI)

# per estrarre gli indentificatori dei gruppi
qqq=getClusters(DATI,ncl)
table(qqq)  ## dimensione dei gruppi

choice(DATI)  ### si vedono le traiettorie
```