Data Mining Algorithms In R/Clustering/K-Means

Introduction

Clustering techniques have a wide use and importance nowadays. This importance tends to increase as the amount of data grows and the processing power of the computers increases. Clustering applications are used extensively in various fields such as artificial intelligence, pattern recognition, economics, ecology, psychiatry and marketing.

The main purpose of clustering techniques is to partitionate a set of entities into different groups, called clusters. These groups may be consistent in terms of similarity of its members. As the name suggests, the representative-based clustering techniques uses some form of representation for each cluster. Thus, every group has a member that represents it. The motivation to use such clustering techniques is the fact that, besides reducing the cost of the algorithm, the use of representatives makes the process easier to understand. There are many decisions that have to be made in order to use the strategy of representative-based clustering. For example, there is an obvious trade-off between the number of clusters and the internal cohesion of them. If there are few clusters, the internal cohesion tends to be small. Otherwise, a large number of clusters makes them very close, so that there is little difference between adjacent groups. Another decision is whether the clusters should be mutually exclusive or not, that is, if an entity can co-exist in more than one cluster at the same time.

Technique to be discussed

In this work, we focus on K-Means algorithm, which is probably the most popular technique of representative-based clustering. In the first section, we give a brief explanation of how the algorithm works.

Algorithm

K-Means is an simple learning algorithm for clustering analysis. The goal of K-Means algorithm is to find the best division of n entities in k groups, so that the total distance between the group's members and its corresponding centroid, representative of the group, is minimized. Formally, the goal is to partition the n entities into k sets S_i, $i=1, 2, ..., k$ in order to minimize the within-cluster sum of squares (WCSS), defined as:

$$\sum_{j=1}^{k} \sum_{i=1}^{n} \| x_i^j - c_j \|^2$$

where term $\| x_i^j - c_j \|$ provides the distance between an entity point and the cluster's centroid.

The most common algorithm, described below, uses an iterative refinement approach, following these steps:

- Define the initial groups' centroids. This step can be done using different strategies. A very common one is to assign random values for the centroids of all groups. Another approach is to use the values of K different entities as being the centroids.
- Assign each entity to the cluster that has the closest centroid. In order to find the cluster with the most similar centroid, the algorithm must calculate the distance between all the entities and each centroid.
- Recalculate the values of the centroids. The values of the centroid's fields are updated, taken as the average of the values of the entities' attributes that are part of the cluster.
- Repeat steps 2 and 3 iteratively until entities can no longer change groups.

The K-Means is a greedy, computationally efficient technique, being the most popular representative-based clustering algorithm. The pseudocode of the K-Means algorithm is shown below.

Algorithm 1: K-Means Algorithm

Input: $E = \{e_1, e_2, \ldots, e_n\}$ (set of entities to be clustered)
k (number of clusters)
$MaxIters$ (limit of iterations)
Output: $C = \{c_1, c_2, \ldots, c_k\}$ (set of cluster centroids)
$L = \{l(e) \mid e = 1, 2, \ldots, n\}$ (set of cluster labels of E)

```
foreach c_i ∈ C do
  | c_i ← e_j ∈ E (e.g. random selection)
end
foreach e_i ∈ E do
  | l(e_i) ← argminDistance(e_i, c_j) j ∈ {1...k}
end

changed ← false;
iter ← 0;
repeat
    foreach c_i ∈ C do
      | UpdateCluster(c_i);
    end
    foreach e_i ∈ E do
      | minDist ← argminDistance(e_i, c_j) j ∈ {1...k};
      | if minDist ≠ l(e_i) then
      |   | l(e_i) ← minDist;
      |   | changed ← true;
      | end
    end
    iter++;
until changed = true and iter ≤ MaxIters ;
```

Implementation

In order to use the K-Means algorithm in R, one must install *cluster* package. This package includes a function that performs the K-Mean process, according to different algorithms. These algorithms are described below:

- **Lloyd**

  Given any set of k centers Z, for each center z in Z, let V(z) denote its neighborhood. That is the set of data points for which z is the nearest neighbor. Each stage of Lloyd's algorithm moves every center point z to the centroid of V(z) and then updates V(z) by recomputing the distance from each point to its nearest center. These steps are repeated until convergence. Note that Lloyd's algorithm can get stuck in locally minimal solutions that are far from the optimal. For this reason it is common to consider heuristics based on local search, in which centers are swapped in and out of an existing solution (typically at random). Such a swap is accepted only if it decreases the average distortion, otherwise it is ignored.

- **Forgy**

  Forgy's algorithm is a simple alternating least-squares algorithm consisting of the following steps:

  - Initialize the codebook vectors. (Suppose that when processing a given training case, N cases have been previously assigned to the winning codebook vector.)
  - Repeat the following two steps until convergence:
    1. Read the data, assigning each case to the nearest (using Euclidean distance) codebook vector.
    2. Replace each codebook vector with the mean of the cases that were assigned to it.

- **MacQueen**

  This algorithm works by repeatedly moving all cluster centers to the mean of their respective Voronoi sets.

- **Hartigan and Wong**

  Given n objects with p variables measured on each object x(i,j) for i = 1,2,...,n; j = 1,2,...,p; K-means allocates each object to one of K groups or clusters to minimize the within-cluster sum of squares:

$$Sum(k) = \sum_{i=0}^{n} \sum_{j=0}^{p} (x(i,j) - \overline{x(k,j)})^2$$

where $\overline{x(k,j)}$ is the mean variable j of all elements in group K.

In addition to the data matrix, a K x p matrix giving the initial cluster centers for the K clusters is required. The objects are then initially allocated to the cluster with the nearest cluster mean. Given the initial allocation, the procedure is to iteratively search for the K-partition with locally optimal within-cluster sum of squares by moving points from one cluster to another.

The K-Means function, provided by the *cluster* package, might be used as follow:

```
kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"))
```

where the arguments are:

- **x:** A numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).
- **centers:** Either the number of clusters or a set of initial (distinct) cluster centers. If a number, a random set of (distinct) rows in x is chosen as the initial centers.
- **iter.max:** The maximum number of iterations allowed.
- **nstart:** If *centers* is a number, *nstart* gives the number of random sets that should be chosen.
- **algorithm:** The algorithm to be used. It should be one of "Hartigan-Wong", "Lloyd", "Forgy" or "MacQueen". If no algorithm is specified, the algorithm of Hartigan and Wong is used by default.

If everything goes OK, an object of class *kmeans* is returned. This object has the following components:

- **cluster:** A vector of integers indicating the cluster to which each point is allocated.
- **centers:** A matrix of cluster centers.
- **whithnss:** The within-cluster sum of squares for each cluster.
- **size:** The number of points in each cluster.

View

There are actually two ways of viewing the result of a K-Means use. Both of them use the object of class *kmeans* returned by the function application.

The first way is to plot the object, creating a chart that represents the data. Thus, if there are N objects divided into K clusters, the chart must contain N points representing the objects, and those points must be colored in K different colors, each one representing a cluster set. For example, given the object *km*, which is a result of the function *kmeans* application, all one has to do in order to plot the object is:

```
plot(km)
```

The second way of viewing the result of a K-Means application is to simply print the components of the object of class *kmeans*. For example, given the same object *km* of the previous example, one could print its components using:

```
print(km)
```

Example

Suppose we have four objects and each object have two attributes (or features), as shown in table below.

Table 1: ***Table representing objects***

Object	Attribute X	Attribute Y
A	1	1
B	2	1
C	4	3
D	5	4

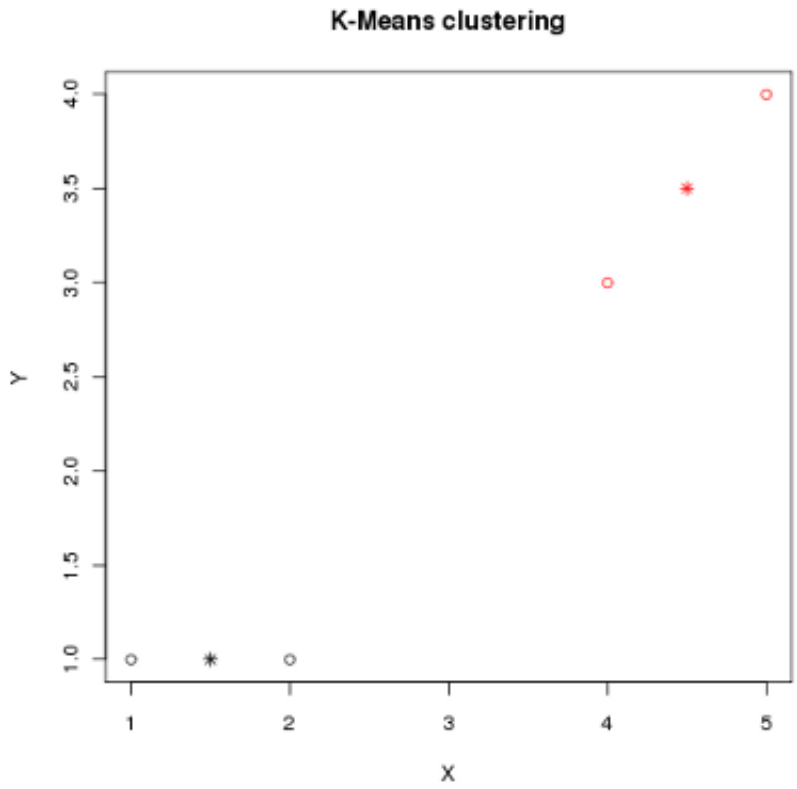
Our goal is to group these objects into K=2 groups based on their two features. The function K-Means can be used to define the groups as follow:

```
# prepare matrix of data
cells <- c(1, 1, 2, 1, 4, 3, 5, 4)
rnames <- c("A", "B", "C", "D")
cnames <- c("X", "Y")
x <- matrix(cells, nrow=4, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames))

# run K-Means
km <- kmeans(x, 2, 15)

# print components of km
print(km)

# plot clusters
plot(x, col = km$cluster)
# plot centers
points(km$centers, col = 1:2, pch = 8)
```

Result of printing components of *km*:

```
K-means clustering with 2 clusters of sizes 2, 2

Cluster means:
    X   Y
1 1.5 1.0

2 4.5 3.5

Clustering vector:
A B C D
1 1 2 2

Within cluster sum of squares by cluster:
[1] 0.5 1.0

Available components:
[1] "cluster"  "centers"  "withinss" "size"
```

Result of plotting:



Case Study

In this section, we illustrate a case study using K-Means.

Scenario

The differences between countries go far beyond the physical and territorial aspects. Hence, for analytical purposes, it is very common to classify countries in groups based on some of their attributes. A traditional classification divides countries into developed, emerging and underdeveloped. In this division, many criteria, such per capita income and life expectancy, can be considered.

The k-means algorithm is a technique for grouping entities according to the similarity of their attributes. As the presenting problem consists of dividing countries into similar groups, it is plausible that K-means can be applied to this task.

Let's consider the scenario where countries need to be classified into the three already mentioned groups: developed, emerging and underdeveloped. To analyze their similarity and assign them to the groups, the following attributes should be taken into account:

- per capita income;
- literacy;
- infant mortality;
- life expectancy;

Input data

The input data is a table containing the numeric values of attributes for each considered country. The table consists of nineteen lines, representing countries and five columns (including the first containing the name of the country), representing the attributes. The table can be loaded from a spreadsheet or from a text file. To preserve the semantics of the clusters, all the values used in this example are real statistics of the countries.

Table 2: ***Input Data***

Country	Per capita income	Literacy	Infant mortality	Life expectancy
Brazil	10326	90	23.6	75.4
Germany	39650	99	4.08	79.4
Mozambique	830	38.7	95.9	42.1
Australia	43163	99	4.57	81.2
China	5300	90.9	23	73
Argentina	13308	97.2	13.4	75.3
United Kingdom	34105	99	5.01	79.4
South Africa	10600	82.4	44.8	49.3
Zambia	1000	68	92.7	42.4
Namibia	5249	85	42.3	52.9
Georgia	4200	100	17.36	71
Pakistan	3320	49.9	67.5	65.5
India	2972	61	55	64.7
Turkey	12888	88.7	27.5	71.8
Sweden	34735	99	3.2	80.9
Lithuania	19730	99.6	8.5	73
Greece	36983	96	5.34	79.5
Italy	26760	98.5	5.94	80
Japan	34099	99	3.2	82.6

Execution

The function "kmeans" can be used to define the groups of countries as follow:

```
# import data
x <- read.table("data.txt")

# run K-Means
km <- kmeans(x, 3, 15)

# print components of km
print(km)

# plot clusters
plot(x, col = km$cluster)
```

```
# plot centers
points(km$centers, col = 1:2, pch = 8)
```

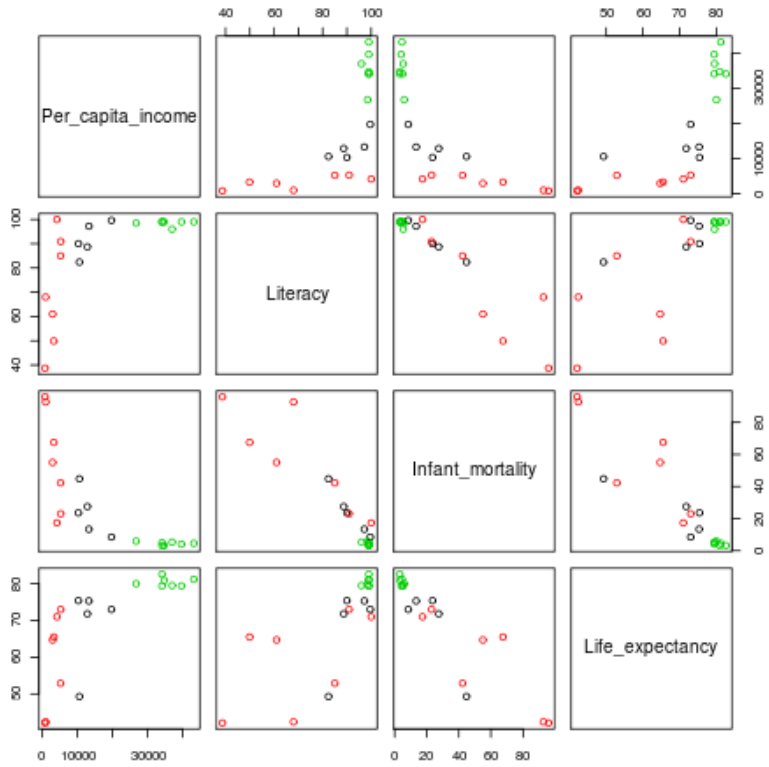
The value of the second parameter of "kmeans" was passed as the number 3, because we want to get three groups of countries.

Output

The result of printing the components of the class returned by the function application is shown below:

```
K-means clustering with 3 clusters of sizes 5, 7, 7

Cluster means:
  Per_capita_income Literacy Infant_mortality Life_expectancy
1         13370.400    91.58        23.560000        68.96000
2          3267.286    70.50        56.251429        58.80000
3         35642.143    98.50         4.477143        80.42857

Clustering vector:
        Brazil        Germany     Mozambique      Australia          China
             1              3              2              3              2
     Argentina United_Kingdom   South_Africa         Zambia        Namibia
             1              3              1              2              2
       Georgia       Pakistan          India         Turkey         Sweden
             2              2              2              1              3
     Lithuania         Greece          Italy          Japan
             1              3              3              3

Within cluster sum of squares by cluster:
[1]  57626083  20109876 158883600

Available components:
[1] "cluster"  "centers"  "withinss" "size"
```

The result of plotting the class returned by the function application is shown below:



Analysis

The implementation of k-means generated three clusters, relatively homogeneous, consisting of 5, 7 and 7 countries. Analyzing the cluster means, we can relate each group with each of the three classes of countries:

- the cluster formed by Germany, United Kingdom, Greece, Australia, Italy and Sweden, has the highest per capita income, literacy and life expectancy and the lowest infant mortality. So, this cluster represents the developed countries.
- the cluster formed by Mozambique, Georgia, Pakistan, India, Zambia and Namibia has the lowest values for all attributes and, therefore, represents the undeveloped countries.
- the cluster formed by the other countries, Brazil, South Africa, Turkish, Argentina and Lithuania represents the group of emerging countries.

To enhance the quality of the classification made by K-Means, the resulting division of the groups was compared with the classification of all countries by Human Development Index as included in the United Nations Development Program's Human Development Report released on October 5, 2009, compiled on the basis of data from 2007. The Human Development Index (HDI) is a comparative measure of well-being that considers aspects like life expectancy, literacy, and education. Compared with the division by the HDI, only four countries have been classified into different groups: Namibia, Georgia, Pakistan and India. These countries should have been placed in the "developing" rather than into "undeveloped" group.

References

1. Meira Jr., W.; Zaki, M. Fundamentals of Data Mining Algorithms. [1]
2. Cluster R package. [2]
3. J. B. MacQueen. 1967. Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press.
4. Hartigan, J.A. (1975), Clustering Algorithms, New York: John Wiley & Sons, Inc.
5. A. K. Jain, M.N. Murthy and P.J. Flynn, Data Clustering: A Review, ACM Computing Reviews, Nov 1999.

References

[1] http://www.dcc.ufmg.br/miningalgorithms/DokuWiki/doku.php
[2] http://cran.r-project.org/web/packages/cluster/index.html

Article Sources and Contributors

Data Mining Algorithms In R/Clustering/K-Means *Source*: http://en.wikibooks.org/w/index.php?oldid=2064526 *Contributors*: Adrignola, Avicennasis, Lucmir, Matheusv, Panic2k4, 1 anonymous edits

Image Sources, Licenses and Contributors

File:Algorithm_kmeans.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Algorithm_kmeans.png *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Lucas Cunha
File:kmeans_plotting.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Kmeans_plotting.png *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Lucas Cunha
File:kmeans_plotting2.png *Source*: http://en.wikibooks.org/w/index.php?title=File:Kmeans_plotting2.png *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Lucas Cunha

License