STAT6494 Project

Yuming Shao

Abstract

This is a project report for statistical learning projects. In this rreport, we review the problem of clustering problem and the overview of the methods of clustring. Then, we may go to the detail about k-means clustering and extend to the BFR algorithm. I want to use an example to demonstrate K-means clustering with the power of the **rmr2** package and visualization of Data with the **Rtsne** package.

1. Introduction

Due to development of internet era and big data, clustring problem are becoming trendy research topics. For example, clustering problem in galaxies cluster into similiar objects, e.g., galaxies, nearby stars, quasars,etc. Another example is about Music CDs. Represent a CD by a set of customers who bought it. Similar CDs have similar sets of customers,and vice-versa.

Firstly, I want to describe what is clustering.

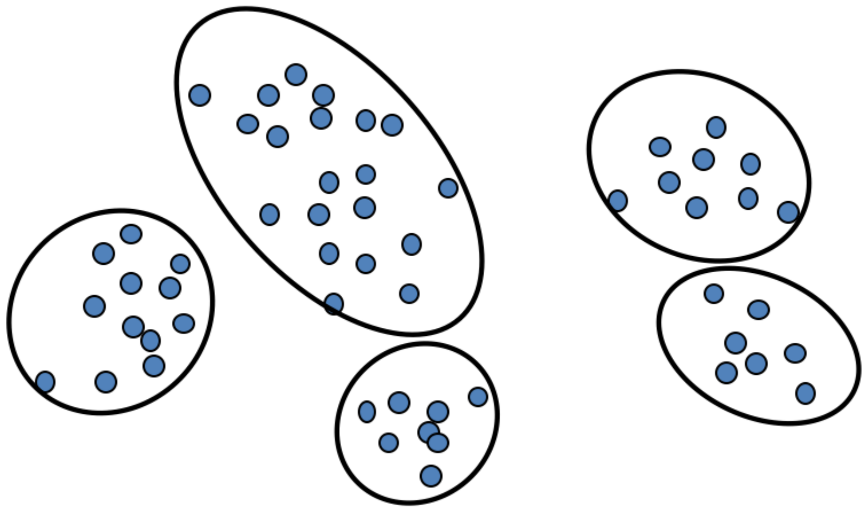
Then,to clustering problem, there are two clustering stratagies, which include Hierarchical clustering algorithms and Point-assignment clustering algorithms. I want to go detail about Hierarchical clustering algorithms and use a R example to illurstrate.

In the third part, I want to introduce the K-means including K-means Algorithm(s) and BFR Algorithm which is extension of K-means to large data.

The final step is about data visualzation of classification by a example.

2. What is clustering

The meaning of cluster is the process of grouping a set of objects into classes of similar objects.



3. Clustering Stratagies

There are two main ways of clustering stratrgies, Hierarchical clustering algorithms and Point-assignment clustering algorithms.

For Hierarchical clustering algorithms, it begin with all points in a cluster of their own, and nearby clusters are merged iteratively. Another way, points are considered in some order, and each one is assigned to the cluster into which it fit best.

Algorithm of hierarchical clustering

- Step 1: Compute distance matrix (In our examples, all distances are Euclidean distance)
- Step 2: Let each data point be a cluster
- **Repeat**
  - Step 3: Combine the two closest clusters
  - Step 4: Update the distance matrix
- **Stop** when further combination leads to clusters that are undesirable for some reasons2*.

About algorithm

- 1. How to determine "closest" clusters?

  There are several ways to define inter-cluster distance. Such as "single", "complete", "average", and "centroid".

- 2. When to stop?

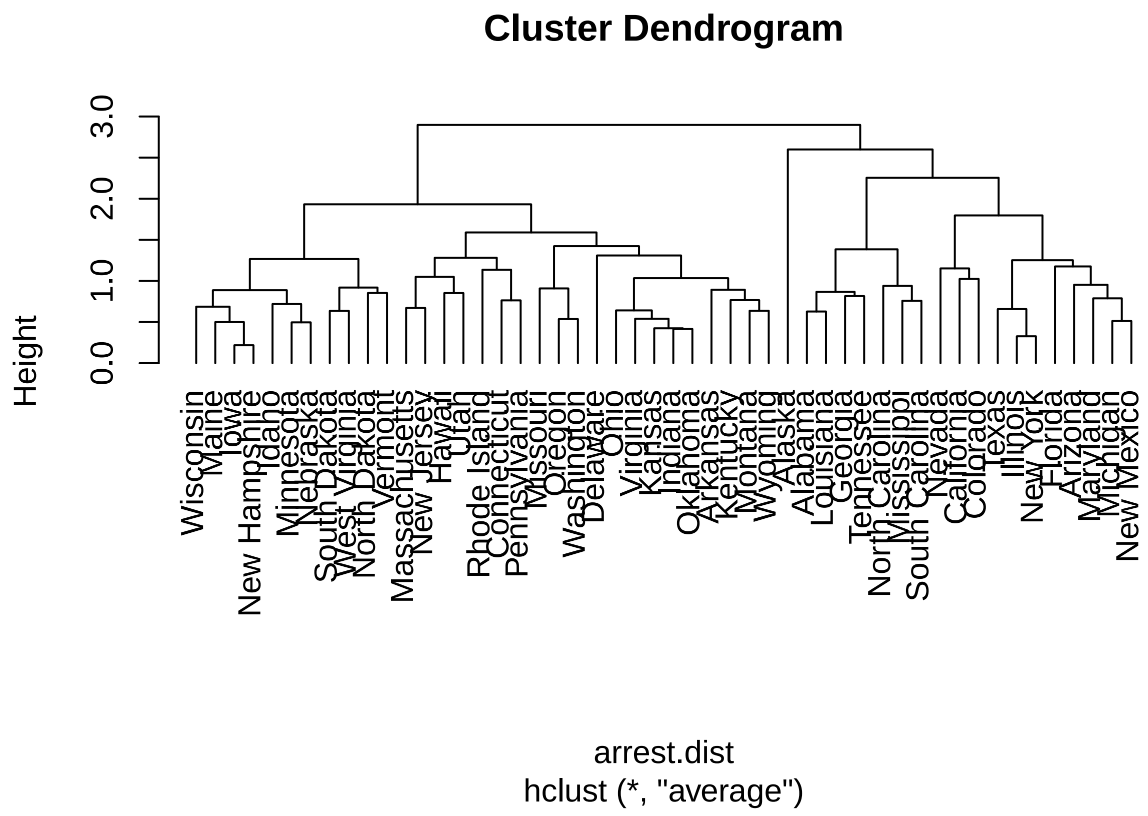
  In practice we repeatedly combine clusters until only one remains. And then choose the number of clusters by certain principle.

4. Example of hierarchical clustering

I got the data from USArrests which is already added in R.

There are 50 observations each represent a state on 4 variables, in rests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973.

```
### Standardize data
median<-apply(USArrests,2,median)
mads<-apply(USArrests,2,mad)
arrest<-scale(USArrests,center=median,scale=mads)
### clustering
arrest.dist<-dist(arrest)
arrest.hclust<- hclust(arrest.dist, method="average")
plot(arrest.hclust,hang=-1)
```



```r
group.6<-cutree(arrest.hclust, 6)
table(group.6)
```

```
## group.6
##  1  2  3  4  5  6
##  7  1  8 20  3 11
```

```r
sapply(unique(group.6),function (g) rownames(USArrests)[group.6==g])
```

```
## [[1]]
## [1] "Alabama"        "Georgia"        "Louisiana"      "Mississippi"
## [5] "North Carolina" "South Carolina" "Tennessee"
##
## [[2]]
## [1] "Alaska"
##
## [[3]]
## [1] "Arizona"    "Florida"    "Illinois"   "Maryland"   "Michigan"
## [6] "New Mexico" "New York"   "Texas"
##
## [[4]]
##  [1] "Arkansas"      "Connecticut"   "Delaware"      "Hawaii"
##  [5] "Indiana"       "Kansas"        "Kentucky"      "Massachusetts"
##  [9] "Missouri"      "Montana"       "New Jersey"    "Ohio"
## [13] "Oklahoma"      "Oregon"        "Pennsylvania"  "Rhode Island"
## [17] "Utah"          "Virginia"      "Washington"    "Wyoming"
##
## [[5]]
## [1] "California" "Colorado"   "Nevada"
```

```
##
## [[6]]
##  [1] "Idaho"         "Iowa"          "Maine"         "Minnesota"
##  [5] "Nebraska"      "New Hampshire" "North Dakota"  "South Dakota"
##  [9] "Vermont"       "West Virginia" "Wisconsin"
```

Here, in this example, I divided into six group. We can see the result from the output.

5. K-means clustering

K-means clustering is a partitioning method. The function kmeans partitions data into k mutually exclusive clusters, and returns the index of the cluster to which it has assigned each observation. Unlike hierarchical clustering, k-means clustering operates on actual observations (rather than the larger set of dissimilarity measures), and creates a single level of clusters. The distinctions mean that k-means clustering is often more suitable than hierarchical clustering for large amounts of data.

For k-means basics, assume Euclidean space first, it starts by picking k, the number of clusters and initialize clusters by picking one point per cluster. Then, extend k-means to BFR (Bradley-Fayyad-Reina) is a variant of k-means designed to handle very large datasets.

Algorithm of K-means clustering

- Step 1: For each point, place it in the cluster whose current centroid it sis nearest
- Step 2: After all points are assigned, update the locations of centroids of the K clusters
- Step 3: Reassign all points to their closest centroid

Repeat 2 and 3 until convergence, which means points don't move between clusters and centroids stabilize

How to select K?

Try different k,looking at the change in the average distance to centroid as k increases. and averge falls rapidely until right k, then changes little.

6. The BFR Algorithm

BFR [Bradley-Fayyad-Reina] is a variant of k-means designed to handle very large (disk-resident) data sets.

Assumes that clusters are normally distributed around a centroid in a Euclidean space. Standard deviations in different dimensions may vary.

1: Initialize K clusters.

2: Load in a bag points from disk.

3: Assign new points to one of the K original clusters, if they are within some distance threshold of the cluster.

4: Cluster the remaining points, and create new clusters.

5: Try to merge new clusters from step 4 with any of the existing clusters.

6: Repeat steps 2-5 until all points are examined.

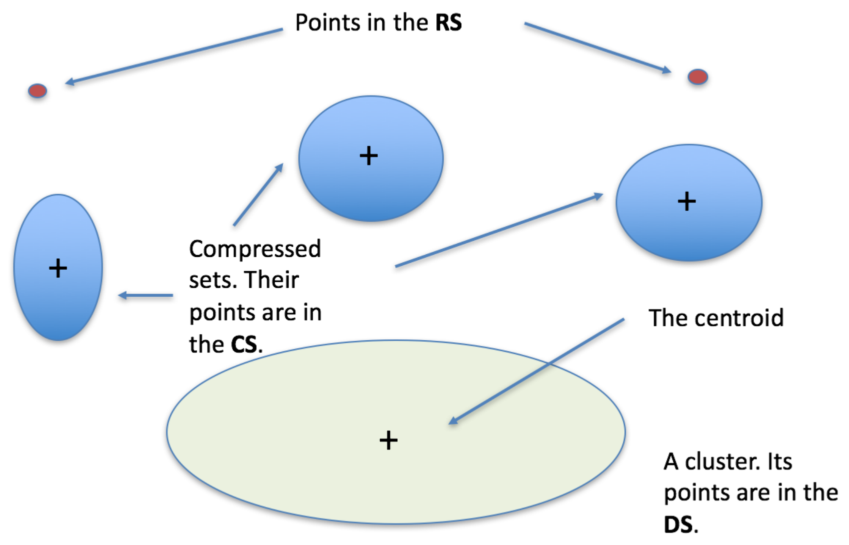
Points are read from disk one main-memory-full at a time. Most points from previous memory loads are summarized by simple statistics.

Step 1 From the initial load we select the initial k centroids by some sensible approach:

- Take k random points.
- Take a small random sample and cluster optimally.
- Take a sample; pick a random point, and then k–1 more points, each as far from the previously selected points as possible.

There are three classes of points. Discard set (DS), points close enough to a centroid to be summarized. Compression set (CS), groups of points that are close together but not close to any existing centroid. And these points are summarized, but not assigned to a cluster. Retained set (RS), isolated points waiting to be assigned to a compression set.

BFR: Picture


Points in the RS
Compressed sets. Their points are in the CS.
The centroid
A cluster. Its points are in the DS.

For each cluster, the discard set (DS) is summarized by the number of points, **N**, the vector SUM, whose $i^t h$ component is the sum of the coordinates of the points in the $i^t h$ dimension, and the vector SUMSQ is ith component = sum of squares of coordinates in $i^t h$.

let d equal to number of dimensions, and 2d + 1 values represent any size cluster. Average in each dimension (the centroid) can be calculated as SUM_i/N. We can get the variance of a cluster's discard set in dimension i, $(SUMSQ_i/N) - (SUM_i/N)^2$.

Step 2 to Step 5 Processing "Memory-load" of points. To step 3, find those points that are "sufficiently close" to a cluster centroid and add those points to that cluster and the DS. Then, to DS set, adjust statistics of the clusters to account for the new points, which includes Ns, SUMs and SUMSQs. These points are so close to the centroid that they can be summarized and then discarded. When we do step 4, use any in-memory clustering algorithm to cluster the remaining points and the old RS. Clusters go to the CS; outlying points to the RS. Consider merging compressed sets in the CS for step 5. Finally, if this is the last round, merge all compressed sets in the CS and all RS points into their nearest cluster.

There are two ways to decide whether to put a new point into a cluster (and discard). The Mahalanobis distance is less than a threshold, another way is high likelihood of the point belonging to currently nearest centroid. I want to go detail about Mahalanobis distance.

Mahalanobis distance

Mahalanobis distance is normalized Euclidean distance from centroid.

For point $(x_1, ..., x_d)$ and centroid $(c_1, ..., c_d)$,

$$d(x,c) = \sqrt(\sum_{i=1}^{n}(\frac{x_i - c_i}{\sigma_i})^2). \tag{1}$$

σ_i standard deviation of points in the cluster in the i^{th} dimension.

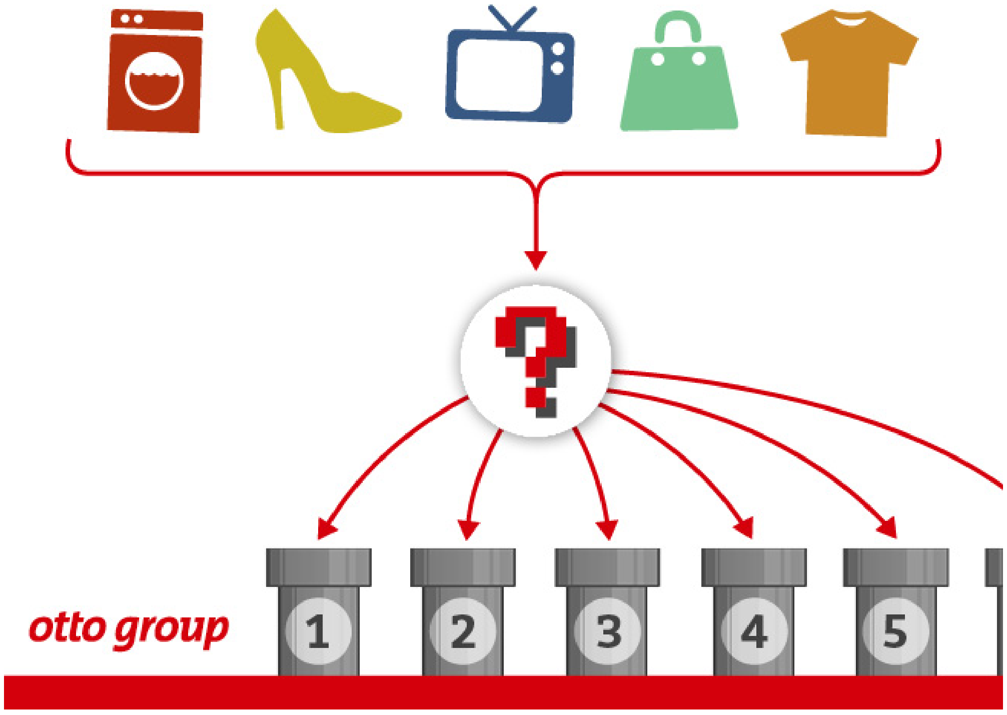
If clusters are normally distributed clusters are normally distributed in d dimensions, then after transformation, one standard deviation $= \sqrt(d)$. For example, 68% of the points of the cluster will have a Mahalanobis distance $< \sqrt(d)$. Accept a point for a cluster if its M.D. is $<$ some threshold, e.g. 2 standard deviations, normal distribution.

CS subclusters combined

We should compute the variance of the combined subcluster. N, SUM and SUMSQ allow us to make that calculation quickly. Combine if the combined variance is below some threshold.

7. Example of Otto Group Product Classification

The dataset has 93 features for more than 200,000 products and 9 main product categories.

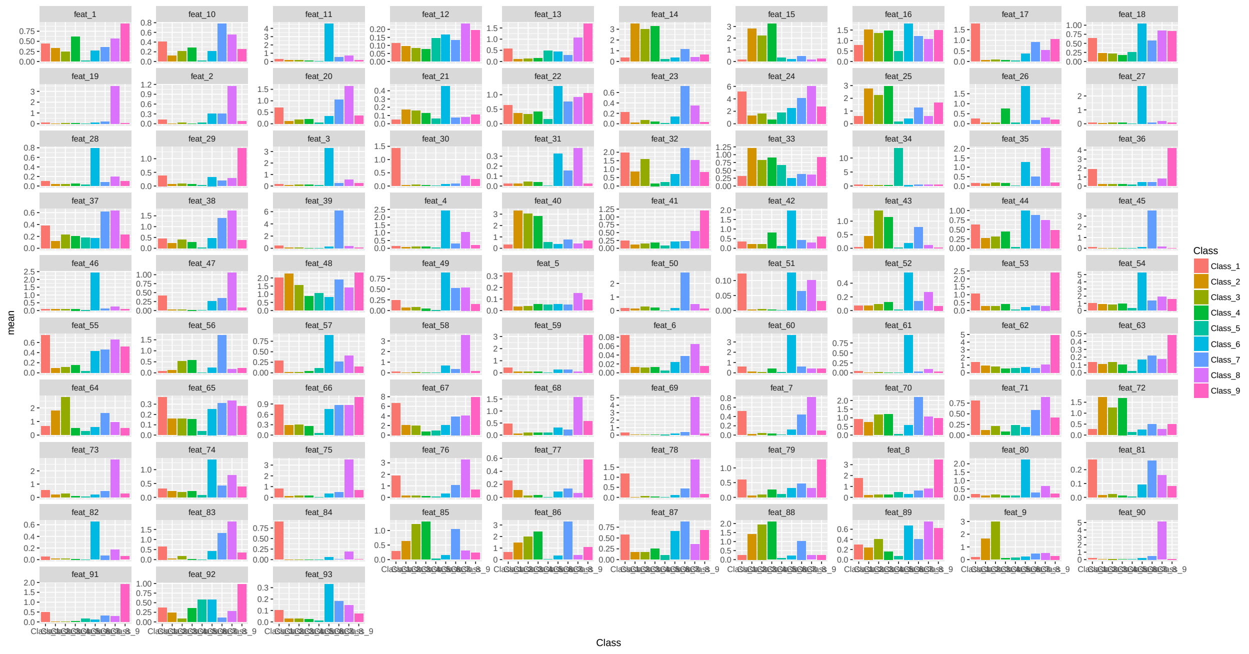


I want to do the data visualization firstly.

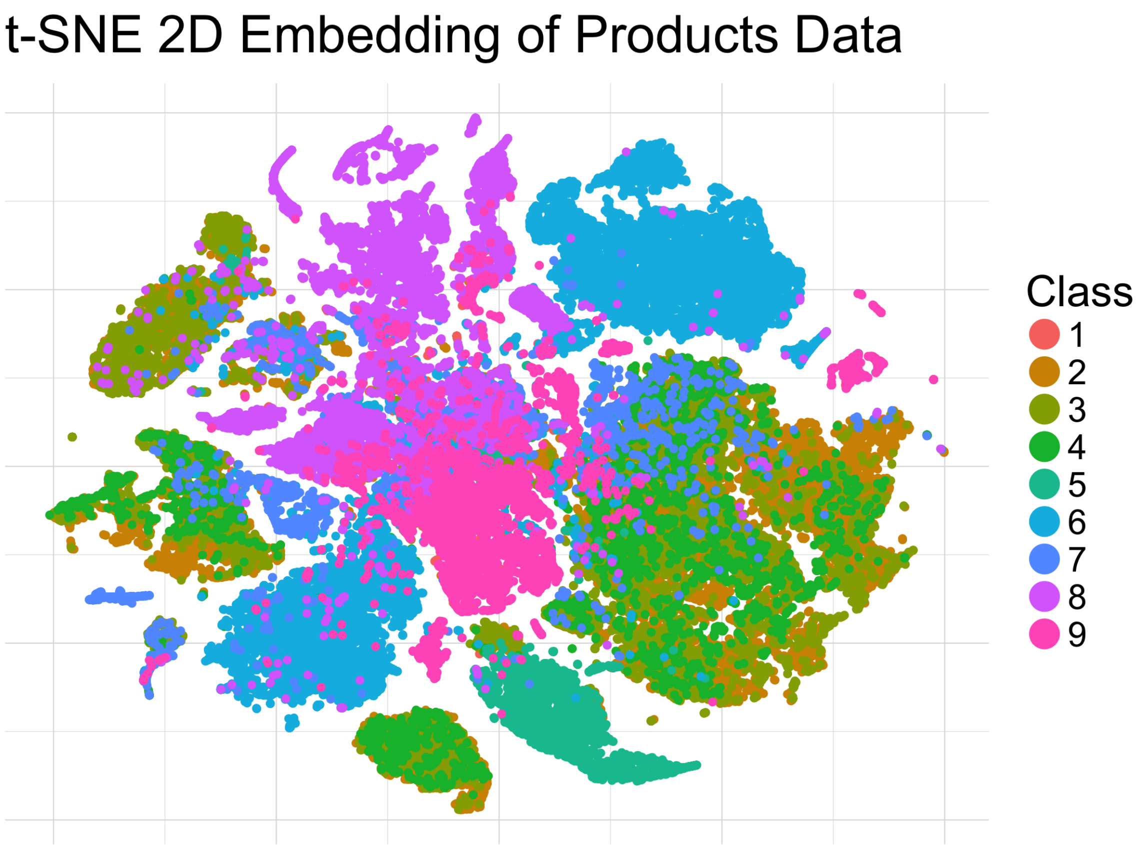
```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```



From the output, Mean of 93 features by 9 class. Then i want to use t-SNE to do the visualization of Data. This can help me to see the 9 classes directly.



Because of the large data set, I try to use map-reduced method to do K-means cluster.

```
## Warning: S3 methods 'gorder.default', 'gorder.factor', 'gorder.data.frame',
## 'gorder.matrix', 'gorder.raw' were declared in NAMESPACE but not found

## Please review your hadoop settings. See help(hadoop.settings)

##
## Attaching package: 'rmr2'

## The following object is masked from 'package:tidyr':
##
##     gather

##          nearest
##              1    2    3    4    5    6    7    8    9
##   Class_1    2  434    0    0 1468    0    0    6   19
##   Class_2    5   74 1798    0 5725  105 1546    0 6869
##   Class_3    0   22 1519    0 2877   44  536    0 3006
##   Class_4    1    7   11    0 1248   11  320    0 1093
##   Class_5    0    0   17    0  607 2105    0    0   10
##   Class_6 5213  450   69   25 8345    1    0    9   23
##   Class_7   15  320   75  208 2028    3    0   10  180
##   Class_8  370  852   97    0 6014    1    0 1127    3
##   Class_9    6 1343    1    0 3567    1    0    0   37
```

From the table, we can see there are 9 classes which is as the requirement.

Reference

@misc{Michael Marshall:2013 , author = “R.A. Fisher”, year = “2013”, title = “{UCI} Machine Learning Repository”, url = “http://archive.ics.uci.edu/ml”, institution = “University of California, Irvine, School of Information and Computer Sciences” }

https://www.kaggle.com/c/otto-group-product-classification-challenge