Data Mining with R

Clustering

Hugh Murrell

reference books

These slides are based on a book by Graham Williams:

Data Mining with Rattle and R,
The Art of Excavating Data for Knowledge Discovery.

for further background on decision trees try Andrew Moore's slides from: *http://www.autonlab.org/tutorials*

and as always, **wikipedia** is a useful source of information.

clustering

Clustering is one of the core tools that is used by the data miner.

Clustering gives us the opportunity to group observations in a generally unguided fashion according to how similar they are.

This is done on the basis of a measure of the distance between observations.

The aim of clustering is to identify groups of observations that are close together but as a group are quite separate from other groups.

k-means clustering

Given a set of observations, $(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k sets $(S_1, S_2, \ldots, S_k)$ so as to minimize the *within-cluster* sum of squares:

$$\sum_{i}^{k} \sum_{\vec{x}_j \in S_i} ||\vec{x}_j - \vec{\mu}_i||^2$$

where $\vec{\mu}_i$ is the mean of observations in S_i.

k-means algorithm

Given an initial set of k means, $\vec{m}_1, \ldots, \vec{m}_k$, the algorithm proceeds by alternating between two steps:

- **Assignment step:** Assign each observation to the cluster whose mean is closest to it.
- **Update step:** Calculate the new means to be the centroids of the observations in the new clusters.

The algorithm has converged when the assignments no longer change.

variants of k-means

As it stands the k-means algorithm gives different results depending on how the initial means are chosen. Thus there have been a number of attempts in the literature to address these problems.

The `cluster` package in R implements three variants of k-means.

- **pam**: partitioning around medoids
- **clara**: clustering large applications
- **fanny**: fuzzy analysis clustering

In the next slide, we outline the *k-medoids* algorithm which is implemented as the function `pam`.

partitioning around medoids

- **Initialize** by randomly selecting k of the n data points as the *medoids*.
- **Associate** each data point to the closest medoid.
- **For each medoid** m
  - **For each non-medoid** data point o
    - Swap m and o and compute the total cost of the configuration
  - Select the configuration with the lowest cost.
- **repeat** until there is no change in the medoid.

distance measures

There are a number of ways to measure *closest* when implementing the k-medoids algorithm.

- Euclidean distance $d(\vec{u}, \vec{v}) = (\sum_i (u_i - v_i)^2)^{\frac{1}{2}}$
- Manhattan distance $d(\vec{u}, \vec{v}) = (\sum_i |u_i - v_i|$
- Minkowski distance $d(\vec{u}, \vec{v}) = (\sum_i (u_i - v_i)^p)^{\frac{1}{p}}$

Note that Minkowski distance is a generalization of the other two distance measures with $p = 2$ giving Euclidian distance and $p = 1$ giving Manhatten (or taxi-cab) distance.

example data set

For purposes of demonstration we will again make use of the classic **iris** data set from R's datasets collection.



```
> summary(iris$Species)

    setosa versicolor  virginica
        50         50         50
```

Can we throw away the `Species` attribute and recover it through unsupervised learning?

partitioning the iris dataset

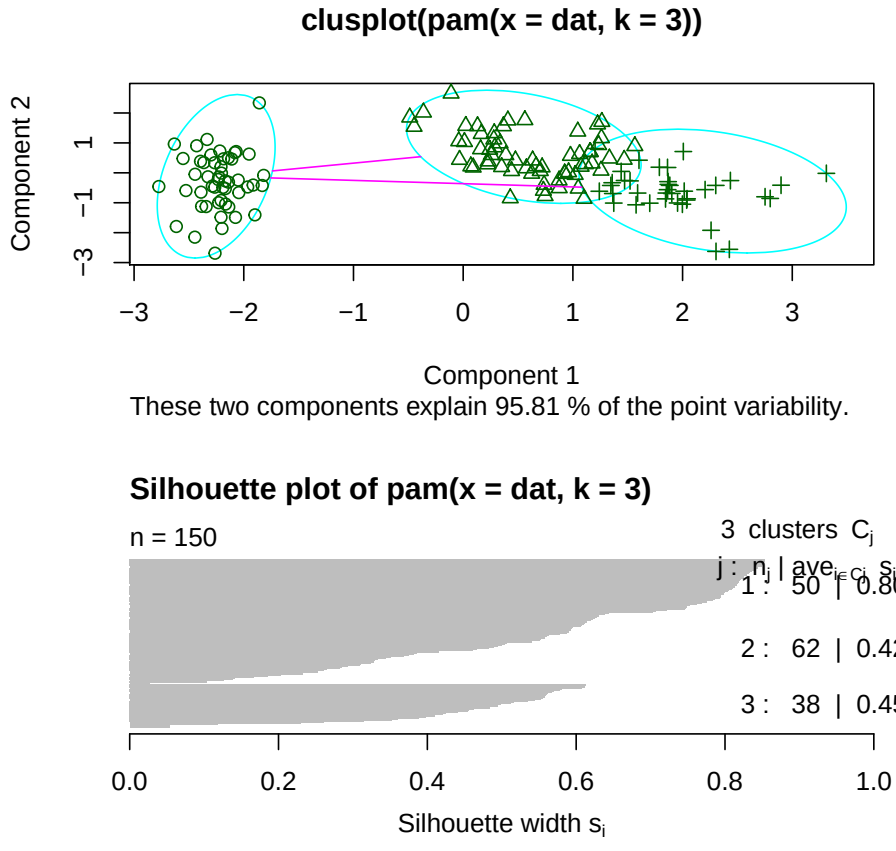
```
> library(cluster)            # load package
> dat <- iris[, -5]           # drop known Species
> pam.result <- pam(dat,3) # perform k-medoids
> pam.result$clustering    # print the clustering
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [18] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [35] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
 [52] 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [69] 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2
 [86] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2
[103] 3 3 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3
[120] 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3
[137] 3 3 2 3 3 3 2 3 3 3 2 3 3 2
```

success rate

```
> # how many does it get wrong
> #
> sum(pam.result$clustering != as.numeric(iris$Species
[1] 16
> #
> # plot the clusters and produce a cluster silhouette
> par(mfrow=c(2,1))
> plot(pam.result)
```

In the silhouette, a large s_i (almost 1) suggests that the observations are very well clustered, a small s_i (around 0) means that the observation lies between two clusters. Observations with a negative s_i are probably in the wrong cluster.

cluster plot



hierarchical clustering

In hierarchical clustering, each object is assigned to its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster.

At each stage distances between clusters are recomputed by a dissimilarity formula according to the particular clustering method being used.

hierarchical clustering of iris dataset

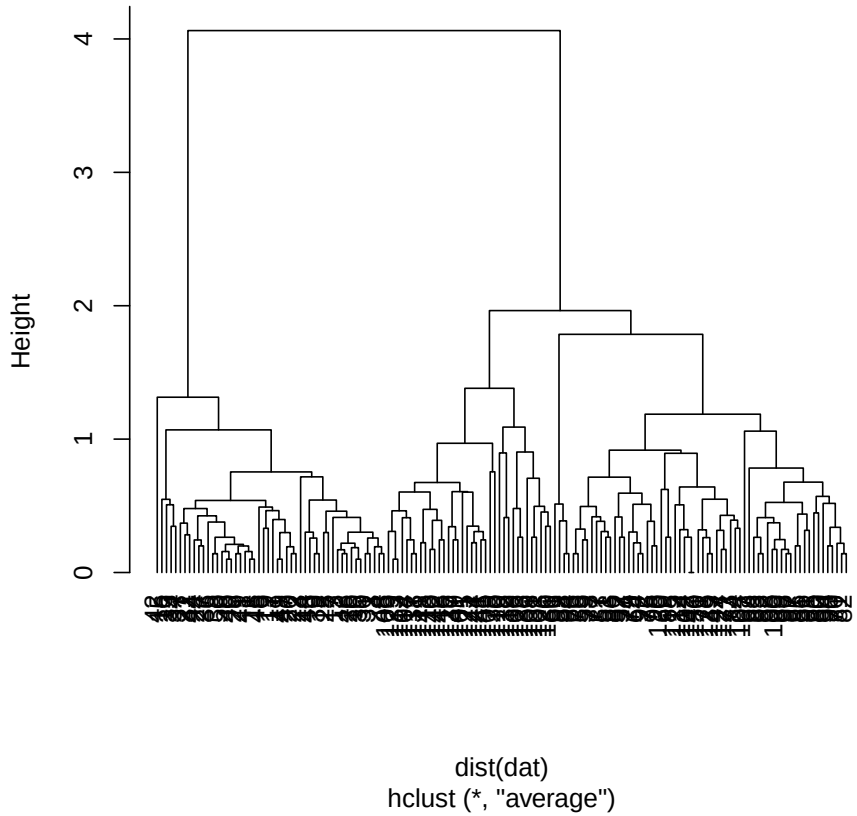
The `cluster` package in R implements two variants of hierarchical clustering.

- **agnes**: AGglomerative NESting
- **diana**: DIvisive ANAlysis Clustering

However, R has a built-in hierarchical clustering routine called `hclust` (equivalent to agnes) which we will use to cluster the iris data set.

```
> dat <- iris[, -5]
> # perform hierarchical clustering
> hc <- hclust(dist(dat),"ave")
> # plot the dendogram
> plclust(hc,hang=-2)
```

cluster plot



Similar to the k-means clustering, `hclust` shows that cluster *setosa* can be easily separated from the other two clusters, and that clusters *versicolor* and *virginica* are to a small degree overlapped with each other.

success rate

```
> # how many does it get wrong
> #
> clusGroup <- cutree(hc, k=3)
> sum(clusGroup != as.numeric(iris$Species))
```

```
[1] 14
```

exercises

By invitation only:

Revisit the *wine* dataset from my website. This time discard the `Cultivar` variable.

Use the `pam` routine from the `Cluster` package to derive 3 clusters for the wine dataset. Plot the clusters in a 2D plane and compute and report on the success rate of your chosen method.

Also perform a hierarchical clustering of the wine dataset and measure its performance at the 3-cluster level.

email your wine clustering script to me by Monday the 9^{th} May, 06h00.