Using Gower’s similarity in Ward’s clustering

Elsie Horne

16/01/2019

# Introduction

Ward’s clustering is a popular agglomerative hierarchical technique for clustering data. At each level of the hierarchy, Ward’s merges the two clusters that result in the smallest increase in error sum of squares between samples in the data and their corresponding cluster centres, i.e. the sum of squared euclidean distances between samples and corresponding cluster centres.

Some studies have been identified which use Ward’s algorithm, but start with a distance matrix calculated using the Gower distance. A limitation with this method is that, for speed, the function cluster::agnes uses properites of the Euclidean distance to avoid having to recalculate cluster centres at each level of the hierarchy. In the following example, we investigate whether using the Gower distance metric with cluster::agnes gives accurate results.

Note: typically the rationale for using the Gower distance metric is that the data contains both continuous and categorical features. However, the example we give here uses only continuous features. When dealing with categorical data Ward’s should be used with great caution, as the cluster centres will be calculated via the arithmetic mean, which is unlikely to be appropriate for categorical attributes.

# Example

The following line loads the functions that I’ve written to implement Ward’s clustering computing a new distance matrix for each level of the hierarchy, rather than the approach taken in cluster::agnes which relies on properties of the Euclidean distance. The code for these functions is included in the appendix at the end of this document.

source("functions\_wards.R")

## Loading required package: arrangements

The data for this example is the NCI60 microarray data, loaded from the package ISLR. This example is adapted from ‘10.6 Lab 3: NCI60 Data Example’ of ‘Introduction to Statistical Learining’ (ISLR).

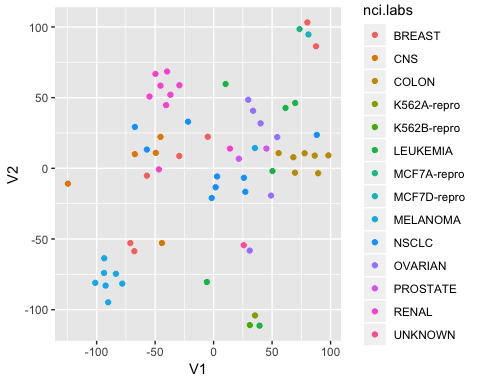
nci.labs <- NCI60$labs  
nci.data <- NCI60$data  
dim(nci.data)

## [1] 64 6830

## Exploratory data analysis

This data has 6830 features, so we use a projection technique called t-distributed stochastic neightbout embedding (t-SNE) tsne to visualise the data in 2 dimensions and get an initial feeling of whether there is some clustering corresponding to the nci.labs. The function Rtsne first reduces the data to 30 dimensions (initial\_dims = 30,) using principal components analysis (PCA) before applying t-SNE to learn a 2-dimensional embedding (dims = 2). The 2-dimensional embedding of the data can be visualised in a scatter plot. The colours in the plot below correspond to the type of cancer (nci.labs). It looks as though there is some grouping in the data which corresonds to the type of cancer. We now investigate this further with cluster analysis.

tmp <- data.frame(nci.data, scale = TRUE)  
# tsne\_30 <- Rtsne(tmp, dims = 2, initial\_dims = 30, perplexity = 5, theta = 0, pca = TRUE)  
# save(tsne\_30, file = "tsne\_30.RData")  
load("tsne\_30.RData")  
tmp <- as.data.frame(tsne\_30$Y)  
tmp <- cbind(tmp, nci.labs)  
tmp <- tmp %>% mutate\_at("nci.labs", as.factor)  
ggplot(data = tmp) + geom\_point(aes(x = V1, y = V2, colour = nci.labs))



## Dimensionality reduction

As the dataset is very wide we first use PCA to reduce dimensions. Here we standardise the data and reduce it to 5 dimensions, as in the ISLR example.

pr.out <- prcomp(nci.data, scale=TRUE)  
x <- pr.out$x[,1:5]  
rownames(x) <- 1:nrow(x)

## Cluster analysis

### Euclidean distance

First we compare results from cluster::agnes and my\_wards using the Euclidean distance to ensure that they are identical.

#### agnes

d\_euc <- daisy(x, metric = "euclidean", stand = FALSE)  
ag\_euc <- agnes(d\_euc, method = "ward", stand = FALSE)

#### my\_wards

This taks a few minutes to run so has been saved and loaded.

#my\_euc <- my\_wards(x, dist = "euclidean")  
#save(my\_euc, file = "my\_euc.RData")  
load("my\_euc.RData")

Check all the distances between merges are the same.

all(near(sort(ag\_euc$height), unname(unlist(my\_euc))))

## [1] TRUE

The are, so when using the Euclidean matrix the two functions give the same solution.

### Gower distance

Now we try with the Gower distance.

#### agnes

d\_gow <- daisy(x, metric = "gower", stand = FALSE)  
ag\_gow <- agnes(d\_gow, method = "ward", stand = FALSE)

#### my\_wards

#my\_gow <- my\_wards(x, dist = "gower")  
#save(my\_gow, file = "my\_gow.RData")  
load("my\_gow.RData")

Check if the distances afe all the same:

near(sort(ag\_gow$height), unname(unlist(my\_gow)))

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## [12] FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE  
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

sort(ag\_gow$height)/unname(unlist(my\_gow))

## [1] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000  
## [8] 1.0000000 1.0000000 1.0000000 1.0000000 0.9777971 0.9775991 1.0032989  
## [15] 1.0264446 1.0231045 1.0089887 1.0000000 1.0000000 1.0000000 1.0000000  
## [22] 1.0000000 0.9925793 0.9536054 0.9513463 1.0217912 1.0410276 1.0447580  
## [29] 1.0354007 1.0429300 1.0889797 0.9886133 0.9969919 0.9769313 0.9913864  
## [36] 0.9711469 0.9806035 0.9939123 0.9754309 1.0338999 0.9964393 1.0406513  
## [43] 1.0043601 0.9670595 0.9426034 0.9985131 1.0367316 1.0345506 1.0202894  
## [50] 1.0323883 1.0603133 1.0207567 1.0760676 0.9738197 0.9791576 0.9966992  
## [57] 0.9306807 1.0513793 1.1491615 0.9680913 0.9685066 1.0739396 0.9583270

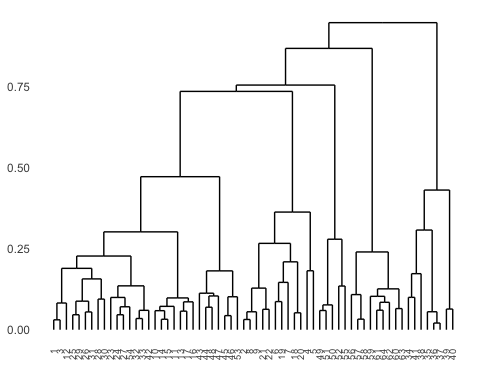
This time some od the distances are different. The differences are not huge though, so take a look to see how it affects the custering.

Use the dendrogram from the agnes clustering to select a k to compare solutions.

ggdendrogram(ag\_gow) +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1, size = rel(0.8)))

## Warning in if (dataClass %in% c("dendrogram", "hclust")) {: the condition  
## has length > 1 and only the first element will be used

## Warning in if (dataClass %in% c("dendrogram", "hclust")) {: the condition  
## has length > 1 and only the first element will be used



From the dendrogram of the agnes clustering, 5 clusters seems reasonable. Tabulate the 5 cluster solutions obtained from both functions.

ag\_5 <- cutree(ag\_gow, k = 5)  
my\_5 <- my\_clusters(my\_gow, k = 5)  
  
table(ag\_5, my\_5)

## my\_5  
## ag\_5 1 2 3 4 5  
## 1 0 0 3 0 27  
## 2 0 0 12 0 0  
## 3 0 0 0 8 0  
## 4 0 5 0 0 0  
## 5 9 0 0 0 0

Only 3 samples have been reallocated. Take a look at how this solution corresponds to the types of cancer.

table(nci.labs, ag\_5)

## ag\_5  
## nci.labs 1 2 3 4 5  
## BREAST 0 3 0 2 2  
## CNS 2 3 0 0 0  
## COLON 7 0 0 0 0  
## K562A-repro 0 0 1 0 0  
## K562B-repro 0 0 1 0 0  
## LEUKEMIA 0 0 6 0 0  
## MCF7A-repro 0 0 0 1 0  
## MCF7D-repro 0 0 0 1 0  
## MELANOMA 1 0 0 0 7  
## NSCLC 6 2 0 1 0  
## OVARIAN 5 1 0 0 0  
## PROSTATE 2 0 0 0 0  
## RENAL 7 2 0 0 0  
## UNKNOWN 0 1 0 0 0

table(nci.labs, my\_5)

## my\_5  
## nci.labs 1 2 3 4 5  
## BREAST 2 2 3 0 0  
## CNS 0 0 5 0 0  
## COLON 0 0 0 0 7  
## K562A-repro 0 0 0 1 0  
## K562B-repro 0 0 0 1 0  
## LEUKEMIA 0 0 0 6 0  
## MCF7A-repro 0 1 0 0 0  
## MCF7D-repro 0 1 0 0 0  
## MELANOMA 7 0 0 0 1  
## NSCLC 0 1 2 0 6  
## OVARIAN 0 0 1 0 5  
## PROSTATE 0 0 0 0 2  
## RENAL 0 0 3 0 6  
## UNKNOWN 0 0 1 0 0

This has resulted in very minor changes - my\_wards has grouped all 5 CNS cancers into one cluster while agnes has split them across two clusters (2|3). The split for RENAL from my\_wards is 7|2 and 6|3 for agnes. Other than this the solutions are identical.

# Conclusions

The differences betweent he solutions demonstrate that agnes does calucate the exact gower distances. In this example with 5 features and 64 samples the differences were fairly minor. However, caution should be taken when working with larger datasets.

(I will complete another example, but my\_wards is slow, so I think that datasets bigger than ~100 samples may be infeasible.)

# Appendix

Below is the code for each of the functions.

my\_wards

## function (x, dist)   
## {  
## ess\_direct <- function(C) {  
## C <- str\_c(C, collapse = ",")  
## C\_ind <- unlist(str\_split(C, ","))  
## if (length(C\_ind) == 1)   
## return(0)  
## else {  
## d <- d\_current[C, C\_ind]  
## return(sum(d \* d))  
## }  
## }  
## change\_ess\_direct <- function(L) {  
## ess\_direct(c(L[1], L[2])) - ess\_direct(L[1]) - ess\_direct(L[2])  
## }  
## my\_mean <- function(combo) {  
## combo <- unlist(str\_split(combo, ","))  
## x\_mean <- colMeans(x\_current[combo, ])  
## }  
## levs <- nrow(x) - 1  
## merges <- vector(mode = "list", length = levs)  
## names <- vector(mode = "list", length = levs)  
## clusters <- as.character(1:nrow(x))  
## x\_rows <- as.character(1:nrow(x))  
## x\_current <- x  
## for (i in 1:levs) {  
## combos <- as.data.frame(t(combinations(x = clusters,   
## k = 2))) %>% mutate\_all(as.character)  
## names(combos) <- unname(apply(as.matrix(combos), 2, function(x) str\_c(x,   
## collapse = ",")))  
## means <- sapply(combos, my\_mean)  
## means <- as.data.frame(t(means))  
## x\_current <- rbind(x\_current, means)  
## d\_current <- daisy(x\_current, metric = dist, stand = FALSE)  
## d\_current <- as.matrix(d\_current)  
## d\_combos <- lapply(combos, change\_ess\_direct)  
## names(d\_combos) <- unname(apply(as.matrix(combos), 2,   
## function(x) str\_c(x, collapse = " and ")))  
## d\_combos <- unlist(d\_combos)  
## d\_combos <- (2 \* d\_combos)^0.5  
## d\_min <- min(d\_combos)  
## c\_rem <- combos[d\_combos == d\_min]  
## merges[i] <- d\_min  
## c\_rem <- as.character(unlist(c\_rem))  
## c\_new <- str\_c(unlist(c\_rem), collapse = ",")  
## clusters <- clusters[!(clusters %in% c\_rem)]  
## clusters <- c(clusters, c\_new)  
## x\_rows <- c(x\_rows, c\_new)  
## x\_current <- x\_current[x\_rows, ]  
## names[i] <- str\_c(clusters, collapse = "--")  
## }  
## names(merges) <- names  
## return(merges)  
## }

my\_clusters

## function (my\_wards\_out, k)   
## {  
## r <- names(my\_wards\_out)  
## r <- r[[nrow(x) - k]][[1]]  
## r <- str\_split(r, "--")[[1]]  
## r <- sapply(r, function(x) str\_split(x, ","))  
## r <- sapply(r, as.numeric)  
## x\_r <- data.frame()  
## for (i in 1:length(r)) {  
## m\_r <- matrix(data = c(r[[i]], rep(i, length = length(r[[i]]))),   
## ncol = 2)  
## m\_r <- as.data.frame(m\_r)  
## x\_r <- rbind(x\_r, m\_r)  
## }  
## x\_r <- unname(unlist(arrange(x\_r, V1)[2]))  
## return(x\_r)  
## }  
## <bytecode: 0x7fee5f3e7e18>