Exercise: cluster analysis of gene expression data

1 Introduction (recap)

Cluster analysis may also be referred as *unsupervised learning*, *unsupervised class discovery* or simply *class discovery*. In cluster analysis the classes (clusters) are unknown from the beginning and need to be identified from data. **Three main steps in clustering** are defining the distance between objects, defining the distance between clusters and selecting the clustering method/algorithm. These steps depend on data and questions under interest.

The most common goal with clustering microarray data is to identify subsets of genes that have similar expression patterns (gene-based clustering, clustering by rows). That is, we would like to divide genes in such a way that genes with similar expression pattern fall into the same cluster. Gene-based clustering is often used for identifying groups of co-regulated genes, finding experimental artifacts or simply for nicer display of expression data.

Depending on a task, one can instead cluster arrays/samples (sample-based clustering, clustering by columns) or both genes and samples (two-way clustering). Clustering by samples is useful for identifying new classes of biological samples, but can also be used for detecting the artifacts or displaying the expression data.

Two-way clustering is used for finding groups of genes that behave similarly across groups of samples, or the opposite - finding groups of samples that behave similarly across groups of genes.

Similarity between objects

To decide which genes are similar in behaviour, we need to define a distance (or dissimilarity) measure which captures the degree of similarity for each gene pair. Eucledian distance and Pearson correlation coefficient are the most common measures used. These and some other typical choices of distance measures can be calculated:

- *The Pearson correlation* captures how similar the shapes of the expression curves of two items (genes or samples) are. That is, if they go to the same

direction (positive cor), opposite direction (negative cor) or are completely different (cor=0). Of course, using 1 - Pearson correlation gives the dissimilarity measure

$$d_{ij} = 1 - r_{ij}$$

where

$$r_{ij} = \frac{\sum_c (e_{ic} - e_i)(e_{jc} - e_j)}{\sqrt{\sum_c (e_{ic} - e_i)^2 \sum_c (e_{jc} - e_j)^2}}$$

- *The uncentered correlation* is basically the same as Pearson correlation, except that it assumes that item's expression mean is 0. The difference is that, if two expression curves have identical shape, but are offset relative to each other by a fixed value (linear shift), they will have a standard Pearson correlation of 1 but will not have an uncentered correlation of 1. Similarly, 1 - uncentered correlation measures distance:

  $$d_{ij} = 1 - r_{ij}$$

  where

  $$r_{ij} = \frac{\sum_c (e_{ic})(e_{jc})}{\sqrt{\sum_c (e_{ic})^2 \sum_c (e_{jc})^2}}$$

- *Absolute correlation* considers two items to be similar even if they have opposite expression patterns. Again, 1 - absolute correlation is used for distance:

  $$d_{ij} = 1 - |r_{ij}|$$

  .

- *Euclidean distance* is the usual distance (L2 norm) between two points:

  $$\sqrt{\sum_c (e_{ic} - e_{jc})^2}$$

  .

Clustering methods

Clustering methods can be classified by their searching strategies. Two main classes of clustering methods are hierarchical clustering and partitioning.

Hierarchical methods subdivide each cluster into smaller clusters, thus forming a tree-shaped structure. The results are displayed in the form of a dendrogram. Hierarchical clustering methods can further be divided according the direction of tree-building:

- *agglomerative methods* start with each observation forming a separate group. Observations close to each other are successively merged.
- *divisive methods* start with one initial cluster containing the whole dataset. This is successively split into smaller clusters until each cluster contains exactly one object.

In R you can for example use method `hclust` in package `stat` or several methods in package `cluster`.

Hiererchical clustering is based on an assumption of hierarchical data - genes can be ranked in terms of their similarity. If this is not the case, or if genes are not correlated across all samples/conditions, hiererchical clustering will not perform well. Partitioning methods, like *k-means* and *SOM* do not assume the hierarchy in data, but require the estimate of the number of clusters in the data.

2 Data

We will use the same dataset as in PCA exercises. Open **R** and download the tab-separated data file "leukemia.txt" from the course homepage.

```
>download.file("http://www.math.chalmers.se/~sarv/mve130/labs/leukemia.txt",
"leukemia.txt")
>x<-read.table("leukemia.txt",sep="\t",header=T)
>data<-log2(x)
```

As before, you can use GEO (http://www.ncbi.nlm.nih.gov/geo/) accession number GSE4392 to get information about the data set. Samples (columns) 1-8 represent the patients with good leukemia prognosis, samples 9-16 represent the patients with poor leukemia prognosis.

You can check your working directoty in R with `getwd()`. Setting is done with `setwd()`.

To prepare for the exercises, find 50 most differentially expressed genes between the two types of leukemia.

```
>library(limma)
>leuk<-as.factor(c(rep("good",8),rep("poor",8)))
>design<-model.matrix(~-1+leuk)
>fit.lm<-lmFit(data,design)
>contrasts<-makeContrasts(leukgood-leukpoor,levels=design)
>fit.con<-contrasts.fit(fit.lm,contrasts)
>y<-eBayes(fit.con)
```

```
>top<-topTable(y,n=50,genelist=rownames(data))
>top50<-as.matrix(data[top$ID,]
```

While doing the exercises, it is recommended that you save your code or more complicated parts of it in a separate text (.r) file. This makes tracking your work easier and you can make use of old code parts when writing new commands. If needed, the whole file can be run in R by `source("myverysmartcode.r")`.

3 Exercises

Exercise 1: Calculate and view distance matrices

Distance matrix is often calculated within the clustering algorithm, but it can also be calculated and displayed separately. Try calculating and displaying Euclidean distance and 1 - Pearson correlation. Heatmaps are useful for revealing the structure in data matrix and are frequently used for displaying clustering results. R has several functions for heatmaps, but in this section we use just `levelplot`.

```
>library(lattice)
```

Calculate Eucledian distance between samples:
First, use the whole dataset

```
>edist1<-dist(t(data),method="euclidean",diag=T,upper=T)
>levelplot(as.matrix(edist1),col.regions=rev(heat.colors(50)),
scales=list(x=list(rot=90))
```

Then, only filtered data

```
>edist2<-dist(t(top50),method="euclidean",diag=T,upper=T)
>levelplot(as.matrix(edist2),col.regions=rev(heat.colors(50)),
scales=list(x=list(rot=90))
```

Compare and comment the results.

Now, calculate Pearson correlation distance between samples:

```
>pdist<-(1-cor(top50))
>levelplot(pdist, col.regions=rev(heat.colors(50)),scales =
list(x=list(rot=90)))
```

For top 50 genes, compare Pearson corelation distance with Eucledian. (Hint: use `x11()` for opening a new graphical window.) Are the two pictures identical? What could be the reason?

FYI: Images could also be saved into a postscript file (see functions `postscript` and `dev.off`) with any user-given paper size.

Exercise 2: Hierarchical clustering

Gene-based clustering

Let us start with 1 - Pearson correlation as a distance measure. For now, we will use average intercluster distance and agglomerative clustering method. Compute

```
>dist1<-as.dist(1-cor(t(top50)))
>hc1.gene<-hclust(dist1,method="average")
```

View the hierarchical cluster tree

```
>plot(hc1.gene)
```

For divisive clustering you could use

```
>hdiv<-diana(dist1,diss=T)
>plot(hdiv, which.plot=2)
```

Now, dendrogram is good for viewing the structure, but one would also like to have information about the way the genes are similar to each other. Try viewing the tree together with the heatmap of expression values

```
>heatmap(top50, Rowv=as.dendrogram(hc1.gene), Colv=NA, scale="none")
```

Same, but expression values standardized gene-wise, might reveal more easily visible clustering patterns

```
>heatmap(top50, Rowv=as.dendrogram(hc1.gene), Colv=NA, scale="row")
```

Randomize samples gene by gene, to see if you still find patterns

```
>top50.rand<-t(apply(top50,1,sample))
```

Recalculate the distance measure and hierarchical tree as above, display results.

One way of reducing noise in gene-wise clustering would be to average across replicate samples. You can try to view the results, but remember that with leukemia data it is not really appropriate to average across arrays since we have patient-based biological variation. With yeast arrays the averaging could be appropriate.

FYI: For making more advanced plots with color keys, method `heatplot.2` in package `gplots` works well. This package is currently not available among standard packages installed at Chalmers, but could be installed locally to your account. More instructions can be found after the last exercise. However, you can complete your lab exrecises without installing the `gplots` package.

Sample-based clustering

Sample-based clustering is analogous to gene-based clustering. Try out hierarchical clustering by samples.

```
>dist2<-as.dist(1-cor(top50))
>hc2<-hclust(dist2,method="average")
>heatmap(top50, Rowv=NA,Colv=as.dendrogram(hc2))
```

What do you see? Do the different leukemia types cluster together?

Two-way clustering

Cluster both by genes and samples to find co-expressed subgroups:

```
>d1<-as.dist(1-cor(t(top50)))
>d2<-as.dist(1-cor(top50))
>h1<-hclust(d1,method="average")
>h2<-hclust(d2,method="average")
>heatmap(top50, Rowv=as.dendrogram(h1), Colv=as.dendrogram(h2))
```

Comment the results.

You can also try out different cluster similarity methods (see `?hclust`).

Exercise 3: Various distance measures

In this exercise you are about to compare the four distance measures described in introduction section. Let us turn back to the agglomerative hierarchical clustering with average distance between clusters. Compute the distance measures for both genes and samples using the top 50 genes in leukemia dataset. Display heatmaps with dendrograms.

Pearson correlation distance

```
>g.pd<-as.dist(1-cor(t(top50)))
>s.pd<-as.dist(1-cor(top50))
>ghc.pd<-hclust(g.pd,method="average")
>shc.pd<-hclust(s.pd,method="average")
>x11()
>heatmap(top50, main="Pearson correlation", Rowv=as.dendrogram(ghc.pd),
Colv=as.dendrogram(shc.pd), scale="none")
```

The rest is analogous. Repeat the procedure for the following distances:

Uncentered correlation distance

```
>m1<-top50/sqrt(apply(top50**2,1,sum))
>g.ud<-as.dist(1-m1%*%t(m1))
>m2<-t(top50)/sqrt(apply(t(top50)**2,1,sum))
>s.ud<-as.dist(1-m2%*%t(m2))
...
```

Absolute correlation distance

```
>g.ad<-as.dist(1-abs(cor(t(top50))))
>s.ad<-as.dist(1-abs(cor(top50)))
...
```

Eucledian distance

```
>g.ed<-dist(top50,method="euclidean")
>s.ed<-dist(t(top50),method="euclidean")
...
```

Do you notice similarities or differences? Can you motivate why?

Clustering patterns might be more visible with appropriately scaled data. For example, standardise expression values gene-wise (`scale="row"`) when looking at clusters obtained by the Pearson correlation distance.

```
>x11()
>heatmap(top50, main="Pearson correlation and gene-wise standardised
values", Rowv=as.dendrogram(ghc.pd), Colv=as.dendrogram(shc.pd), scale="row")
```

For uncentered correlation you might divide the expression values with gene-wise standard deviation assuming mean 0

```
>data2<-top50/sqrt(apply(top50**2,1,sum))
>x11()
>heatmap(data2, main="Uncentered correlation and gene-wise scaled values",
Rowv=as.dendrogram(ghc.ud), Colv=as.dendrogram(shc.ud), scale="none")
```

Exercise 4: Partitioning

Perform k-means clustering and mark the clusters with different colors

```
>km1<-kmeans(top50,4)
>km2<-kmeans(t(top50),2)
>table(km1$cluster)
>mycol <- palette()[2:5]
>col1 <- mycol[as.vector(km1$cluster)]
>genecol <- col1[order(km1$cluster)]
>col2 <- mycol[as.vector(km2$cluster)]
>sampcol <- col2[order(km2$cluster)]
```

Compare with hierarchical clustering

```
>heatmap(top50, Rowv=as.dendrogram(ghc.pd), Colv=as.dendrogram(shc.pd),
scale="row", RowSideColors=genecol, ColSideColors=sampcol)
```

Try out different number of clusters if you have time.

For real-data analysis you could consider running k-means with different random seeds for centroids, since the initial centroid positions may give different clustering results.

You can get information about clusters if you search annotations for respective genes (see GO-annotation possibilities on the web for example).

*Exercise: Evaluation in short

Evaluation of clustering results is a complicated task, due to unknown classes. Mathematically good clusters are not neccessarily biologically good clusters. You can search for publications about evaluation (e.g. Datta S. and Datta S. (2006)) to have some idea.

Silhouette plot in package `pam` might be useful for deciding the number of clusters in partitioning arund medioids.

`hclust` has a `coef` function for displaying the agglomerative (AC) or divisive coefficient (DC). They are varying between 0 and 1. Value close to 1 indicates that a very clear structuring has been found. Value close to 0 indicates that the algorithm has not found a natural structure. In other words, the data consists of only one big cluster.

*Exercise: HCE

Hierarchical Clustering Explorer (`http://www.cs.umd.edu/hcil/hce/`) is one of the many freewares available with a graphical interface. You can install it to your personal computer and try out same and other methods that you have used in this lab.

Extra material: loading and installing R-packages

You can check the list of installed packages in R environment by typing
`>library()` If you have packages already installed, just run
`>library(RpackageName)`
for example
`>library(cluster)`
and all the functions in the package `cluster` will be loaded into R.
To display the infromation about the package (version, function list, etc.), simply type
`>library(help=cluster)`
For checking which packages and versions are loaded, type
`>sessionInfo()`.
With no administrative rights for the computer, one can also install pacakages locally by giving local directories as parameters

```
>install.packages("packagename",destdir="/mydir/myRpackages",
lib="/mydir/myRpackages")
>library(packagename,lib.loc="/mydir/myRpackages")
```
otherwise only giving the package name is enough.