Clustering

Eniko Szekely - University of Montpellier

March 29, 2012

1 R programming language

R is used for numerical computations in this TP. The following pages may be helpful:

- R: http://www.r-project.org/
- *mclust* package: http://www.stat.washington.edu/mclust/

2 Libraries for clustering

In the first step we will download some useful libraries for clustering:

- *mvtnorm*: Multivariate normal and t distributions
  http://cran.r-project.org/web/packages/mvtnorm/index.html
- *mclust*: Model-based clustering / Normal mixture modelling
  http://cran.r-project.org/web/packages/mclust/index.html

Now we will install the downloaded packages:

```
> install.packages("pathToMvtnorm/mvtnorm_0.9-9992.tar.gz")
> install.packages("pathToMclust/mclust_3.4.11.tar.gz ")
```

... and load them:

```
> library(mvtnorm)
> library(mclust)
> library(cluster)
```

Use *help* to get informations about the functions that you use:

```
> help(kmeans)
> help(hclust)
```

3 Generate data

We start by generating a mixture of two Gaussians (the normal distribution) $\mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, $c = \{1, 2\}$, $i = 1..N_c$ in two dimensions using the *rmvnorm* function. To generate matrix of R rows and C columns use the *matrix* function. An example of a matrix with 2 rows and 3 columns:

```
> M <- matrix(c(1,2,3,4,5,6),2,3,byrow=TRUE)
```

Exercise 1. Choose the parameters $N_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c$ and plot the generated distribution using the *plot* function.

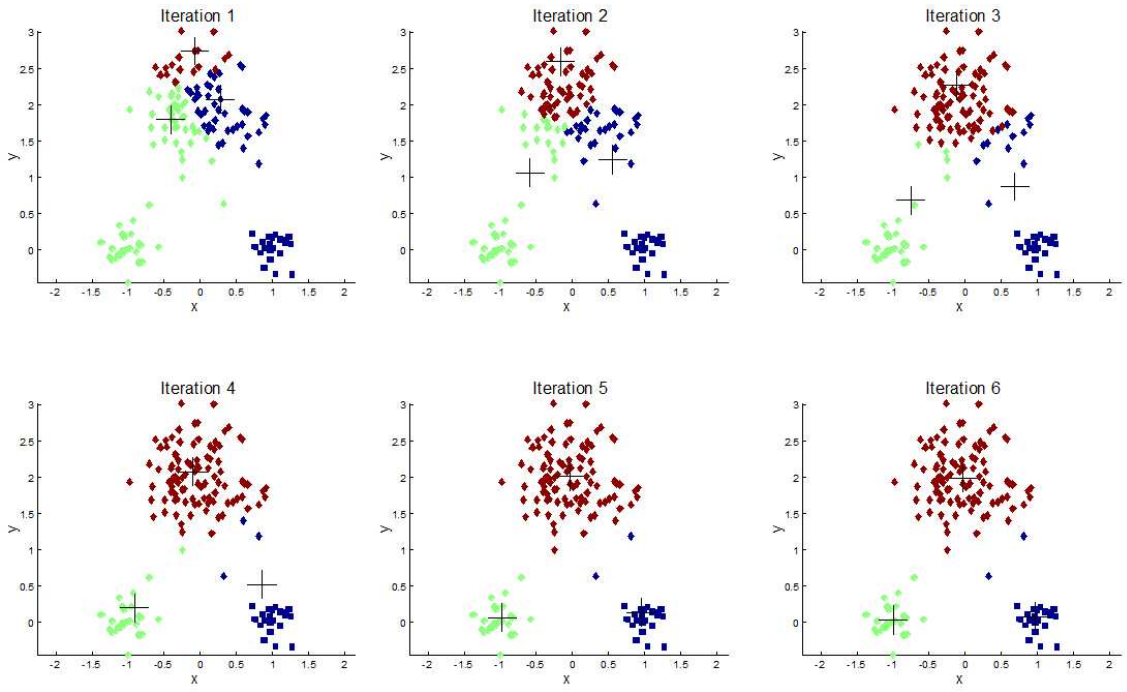

Figure 1: Kmeans.

```
> Y <- rbind(rmvnorm(N1, mean1, sigma=cov1), rmvnorm(N2, mean2, sigma=cov2))
> plot(Y, pch=20)
```

Exercise 2. Vary the three parameters of the Gaussians and list their impact on the final distribution. Choose the covariances as we saw in the course:

- Spherical:
  - $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \cdots = \mathbf{\Sigma}_K = \sigma^2\mathbf{I}$ (same spherical covariance matrix for all components)
  - $\mathbf{\Sigma}_c = \sigma_c^2\mathbf{I}$ (different spherical covariance matrices)
- Diagonal:
  - $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \cdots = \mathbf{\Sigma}_K$ (same diagonal covariance matrix for all components).
  - $\mathbf{\Sigma}_c$ (different diagonal covariance matrices)
- Full covariance.

4 Kmeans

Exercise 3. We reconsider the example that we saw in the course (Figure 1).

```
> C1 <- rmvnorm(100, c(0,2), sigma=matrix(c(0.1,0,0,0.1),2,2))
> C2 <- rmvnorm(20, c(-1,0), sigma=matrix(c(0.05,0,0,0.05),2,2))
> C3 <- rmvnorm(20, c(1,0), sigma=matrix(c(0.05,0,0,0.05),2,2))
> Y <- rbind(C1, C2, C3)
> plot(Y, pch=20)
> mean1 <- matrix(c(-0.4,1.75,0,2.75,0.3,2), 3, 2, byrow=TRUE)
```

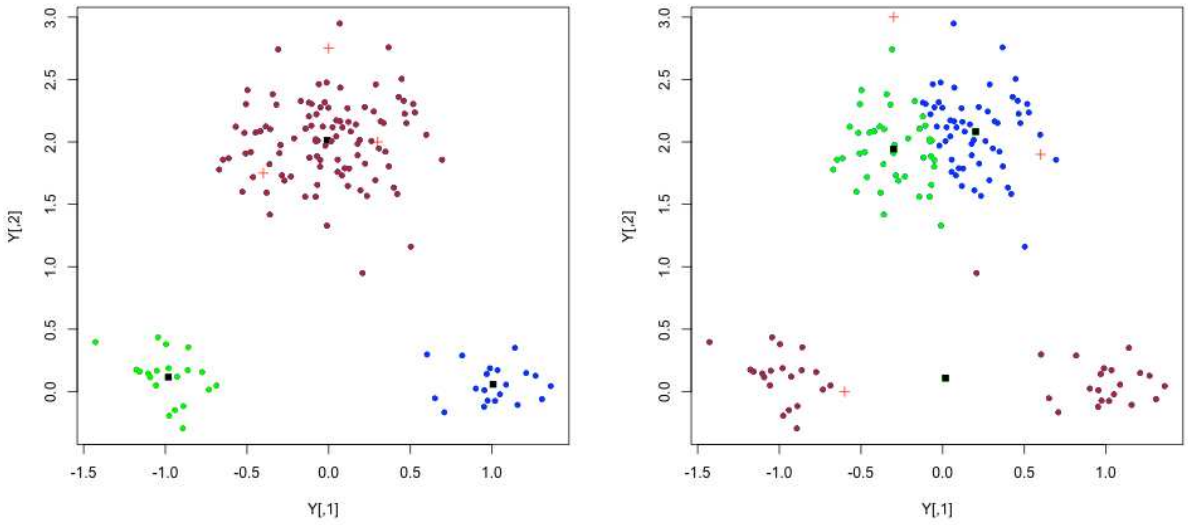

Figure 2: Kmeans with R. Initial centers are marked with a cross and final centers with a square. (a) Good center initialization (Exercise 3). (b) Bad center initialization (Exercise 4).

```
> mean1
> points(mean1, pch=3, col='red')
> Ykmeans1 <- kmeans(Y, mean1)
> Ykmeans1
> Ykmeans1[[2]]
> points(Ykmeans1[[2]], pch=15, col='black')
> points(Y[which(Ykmeans1[[1]]==1),],pch=20,col='green')
> points(Y[which(Ykmeans1[[1]]==2),],pch=20,col='brown')
> points(Y[which(Ykmeans1[[1]]==3),],pch=20,col='blue')
```

Exercise 4. We change now the center initialization:

```
> plot(Y, pch=20)
> mean2 <- matrix(c(-0.6,0,-0.3,3,0.6,1.9), 3, 2, byrow=TRUE)
> mean2
> points(mean2, pch=3, col='red')
> Ykmeans2 <- kmeans(Y, mean2)
> Ykmeans2
> Ykmeans2[[2]]
> points(Ykmeans2[[2]], pch=15, col='black')
> points(Y[which(Ykmeans2[[1]]==1),],pch=20,col='green')
> points(Y[which(Ykmeans2[[1]]==2),],pch=20,col='brown')
> points(Y[which(Ykmeans2[[1]]==3),],pch=20,col='blue')
```

The figures that you should obtain in the two cases are presented in Figure 2.

Exercise 5. Choice of the number of centers: for the same data as in Exercises 3 and 4 choose different number centers, $k = \{2, 4, 5\}$. Plot the centers and optionally color differently the different clusters.

```
> Ykmeans <- kmeans(Y, k)
```

identifier	Model	HC	EM	Distribution	Volume	Shape	Orientation
E		•	•	(univariate)	equal		
V		•	•	(univariate)	variable		
EII	λI	•	•	Spherical	equal	equal	NA
VII	$\lambda_k I$	•	•	Spherical	variable	equal	NA
EEI	λA		•	Diagonal	equal	equal	coordinate axes
VEI	$\lambda_k A$		•	Diagonal	variable	equal	coordinate axes
EVI	λA_k		•	Diagonal	equal	variable	coordinate axes
VVI	$\lambda_k A_k$		•	Diagonal	variable	variable	coordinate axes
EEE	$\lambda D A D^T$	•	•	Ellipsoidal	equal	equal	equal
EEV	$\lambda D_k A D_k^T$		•	Ellipsoidal	equal	equal	variable
VEV	$\lambda_k D_k A D_k^T$		•	Ellipsoidal	variable	equal	variable
VVV	$\lambda_k D_k A_k D_k^T$	•	•	Ellipsoidal	variable	variable	variable

Figure 3: The different models used in *mclust*.

```
> points(Ykmeans[[2]], pch=15, col='black')
```

5 Gaussian Mixture Models

We will use the *mclust* package to analyze the mixtures of gaussians. *Mclust* function chooses the best model by comparing BIC (model selection criteria that penalizes complexity) values. The different models are presented in Figure 5. The *plot* function shows 4 plots: BIC values, clustering, uncertainty and density contour.

```
> C1 <- rmvnorm(2000, c(4,6), sigma=matrix(c(1,0.7,0.7,1),2,2))
> C2 <- rmvnorm(1000, c(0,0), sigma=matrix(c(0.7,-0.5,-0.5,0.7),2,2))
> C3 <- rmvnorm(50, c(7,2), sigma=matrix(c(0.3,0,0,0.3),2,2))
> Y <- rbind(C1, C2, C3)
> plot(Y, pch=20)
> gmm <- Mclust(Y)
> gmm
> plot(gmm, data=Y)
```

Exercise 6. Test the model using a different number of mixture components. Try to explain what happens.

6 Hierarchical clustering

We generate a dataset Y with 8 points and cluster it using hierarchical clustering.

```
> H1 <- rmvnorm(5, c(7,2), sigma=matrix(c(0.3,0,0,0.3),2,2))
> H2 <- rmvnorm(3, c(3,3), sigma=matrix(c(0.3,0,0,0.3),2,2))
> Y <- rbind(H1, H2)
> plot(Y, pch=20)
> help(hclust)
> d <- dist(Y)
> HC <- hclust(d, method="single")
> plclust(HC)
```

Exercise 7. Choose a different linkage criteria and compare the results. Use the *cutree* function to cut the tree either at a given height or as to obtain a given number of clusters.

7 DBSCAN

Download, install and load the *kernlab* package and load the *spirals* dataset. To use the *dbscan* function from the *fpc* package you need to download, install and load the following packages: *multcomp*, *modeltools*, *flexmix* and *fpc*.

```
> data(spirals)
> C1 <- rmvnorm(200, c(4,4), sigma=matrix(c(0.2,0.1,0.1,0.2),2,2))
> Y <- rbind(spirals, C1)
> plot(Y, pch=20)
> Ydbscan <- dbscan(Y, 0.2, method="raw")
> plot.dbscan(Ydbscan, Y)
```

Exercise 8. Use the *help* function to see the parameters passed to dbscan and test the effect of the *eps* and *minPts* parameters on the clustering.

8 Spectral clustering

The spectral clustering function *specc* can be found in the package *kernlab* (loaded previously).

```
> sc <- specc(spirals, centers=2)
> sc
> plot(spirals,col=sc)
```

Exercise 9. Find the best values for *eps* and *minPts* in dbscan to discriminate between the two spirals, and compare the results with spectral clustering.

9 Clustering evaluation

We will use the 3-class *iris* dataset.

```
> summary(iris)
> pairs(iris[1:4], main = "Edgar Anderson's Iris Data", pch = 21, bg = c("red", "green3", "blue")
 [unclass(iris$Species)])
```

Exercise 10. Use a clustering method of your choice and write a function that estimates the purity of the clusters as discussed in the cours:

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{c} \max_{m} |\omega_c \cap c_m| \tag{1}$$

where Ω = the set of classes and $\mathbb{C}$ is the set of clusters. Each cluster c_m is assigned the label of the most frequent class ω_c in that cluster and the accuracy is measured by counting the number of elements that are assigned to the correct class.