Sup. Learn

```
###-------------------------------------------------------------------------
###-------------------------------------------------------------------------
### Supervised learning: LDA (uni-bivariate), QDA (biv), KNN, Fisher's argument
###-------------------------------------------------------------------------
###-------------------------------------------------------------------------
load('mcshapiro.test.RData')
library(MASS)
library(class)
library(rgl)
library(mvtnorm)

### ----------------------------------------------------------------------------
### LDA, KNN
### (2 classes, univariate)   --   start
### ----------------------------------------------------------------------------
# Supponiamo il label come ultima colonna
data = read.table('cytokines.txt', header=T)
#data = data[,c(1,3)]
colnames(data) = c('X1', 'label')
data$label = as.factor(data$label)
head(data)

# ----------------------------------------------------------------------------
# Explore
# ----------------------------------------------------------------------------

livelli = levels(as.factor(data$label))
A = which(data$label == livelli[1])
B = which(data$label == livelli[2])
g = 2
p = 2

# ----------------------------------------------------------------------------
# LDA (univariate)
# ----------------------------------------------------------------------------
# Assumptions:
# 1) if L=i|X.i ~ N(mu.i, sigma.i^2), i=A,B
# 2) sigma.A = sigma.B
# 3) c(A|B)=c(B|A) (equal misclassification costs)

# 1)
shapiro.test(data[A,1])
shapiro.test(data[B,1])

# 2)  (H0: same variance)
var.test(data[A,1], data[B,1])
var.test(data[A,1], data[B,1])$p.value

nA = length(A)
nB = length(B)
n  = nA + nB

# Prior probabilities
PA = nA/n
PB = nB/n

# LDA
data.lda = lda(label ~ X1, data)
data.lda

# ----------------------------------------------------------------------------
# Posterior probability and classification for x=0
# ----------------------------------------------------------------------------
is(data)
x = data.frame(X1 = 0)

# prediction
predict(data.lda, x)$class

# posterior probabilities for the class
predict(data.lda, x)$posterior

# Fisher's discriminant score
predict(data.lda, x)$x

# ----------------------------------------------------------------------------
# Graphical representation
# ----------------------------------------------------------------------------

# Training + posterior probabilities for x
plot(data$X1[A], rep(0, length(A)), pch=16, col='blue', ylim=c(0,1),
     xlab='x', ylab='estimated posterior', main="LDA", xlim=range(data$X1))
points(data$X1[B], rep(0, length(B)), pch=16, col='red')
abline(v=0, col='grey')
points(c(0,0),c(predict(data.lda, data.frame(X1 = 0))$posterior),
       col=c('blue', 'red'), pch='*', cex=2.5)
```

← Prior given by the proportion

LDA (2 classes, univariate)

```
# Estimate the posterior probabilities for a grid
# (grid of representation)
x = data.frame(X1=seq(-10, 35, 0.5))
data.LDA.A <- predict(data.lda, x)$posterior[,1] # posterior probability for class A
data.LDA.B <- predict(data.lda, x)$posterior[,2] # posterior probability for class B

# Add to the plot
lines(x[,1], data.LDA.A, type='l', col='blue', xlab='x',
      ylab='estimated posterior', main="LDA")
lines(x[,1], data.LDA.B, type='l', col='red')
abline(h = 0.5)
legend(-10, 0.9, legend=c('P(A|X=x)', 'P(B|X=x)'), fill=c('blue','red'), cex = 0.7)

# ------------------------------------------------------------------------------
# Different prior probabilities?
# ------------------------------------------------------------------------------
data.lda.1 <- lda(label ~ X1, data, prior=c(0.95,0.05))
data.lda.1

## What happens in graphical representation?
# (the grey are the old probabilities (with priors not fixed by us))
x = data.frame(X1=seq(-10, 35, 0.5))
data.LDA.A.1 = predict(data.lda.1, x)$posterior[,1] # posterior probability for class A
data.LDA.B.1 = predict(data.lda.1, x)$posterior[,2] # posterior probability for class B

plot  (x[,1], data.LDA.A.1, type='l', col='blue', xlab='x',
       ylab='estimated posterior', main="LDA", ylim=c(0,1))
points(x[,1], data.LDA.B.1, type='l', col='red')
abline(h = 0.5)
legend(-10, 0.9, legend=c('P(A|X=x)', 'P(B|X=x)'), fill=c('blue','red'), cex = 0.7)
points(data$X1[A], rep(0, length(A)), pch=16, col='blue')
points(data$X1[B], rep(0, length(B)), pch=16, col='red')
points(x[,1], data.LDA.A, type='l', col='grey')
points(x[,1], data.LDA.B, type='l', col='grey')

# ------------------------------------------------------------------------------
# univariate LDA vs. KNN
# ------------------------------------------------------------------------------
data.knn       = knn(train = data$X1, test = x, cl = data$label, k = 3, prob=T)
data.knn.class = (data.knn == 'B')+0
data.knn.B     = ifelse(data.knn.class==1,
                        attributes(data.knn)$prob,
                        1 - attributes(data.knn)$prob)

plot(x[,1], data.LDA.B, type='l', col='red', lty=2, xlab='x', ylab='estimated posterior')
points(x[,1], data.knn.B, type='l', col='black', lty=1)
abline(h = 0.5)
legend(-10, 0.75, legend=c('LDA','knn'), lty=c(2,1), col=c('red','black'))


# let's change k
par(mfrow=c(3,4))
for(k in 1:12){
  data.knn       = knn(train = data$X1, test = x, cl = data$label, k = k, prob=T)
  data.knn.class = (data.knn == 'B')+0
  data.knn.B <- ifelse(data.knn.class==1,
                       attributes(data.knn)$prob,
                       1 - attributes(data.knn)$prob)

  plot(x[,1], data.LDA.B, type='l', col='red', lty=2, xlab='x',
       ylab='estimated posterior', main=k)
  points(x[,1], data.knn.B, type='l', col='black', lty=1, lwd=2)
  abline(h = 0.5)
}
dev.off()

### ------------------------------------------------------------------------------
### (2 classes, univariate)   --   end
### ------------------------------------------------------------------------------

### ------------------------------------------------------------------------------
### LDA, QDA, KNN
### (3 classes, bivariate)   --   start
### (p = 2, g = 3)
### ------------------------------------------------------------------------------

data = iris[,c(1,2,5)]
colnames(data) = c('X1', 'X2', 'label')
head(data)

attach(data)

labels = as.factor(data$label)
tipi   = levels(data$label)
g      = length(tipi)
```

LDA (2 classes, univariate)

LDA vs. KNN

LDA (p=2, g=3)

LDA (p=2, g=3)

```
i1 <- which(label==tipi[1])
i2 <- which(label==tipi[2])
i3 <- which(label==tipi[3])

n1 <- length(i1)
n2 <- length(i2)
n3 <- length(i3)
n  <- n1+n2+n3

detach(data)

plot(data[,1:2], main='Title', xlab='X1', ylab='Sepal.Width', pch=19)
points(data[i1,], col='red', pch=19)
points(data[i2,], col='green', pch=19)
points(data[i3,], col='blue', pch=19)
legend("topright", legend=levels(data$label), fill=c('red','green','blue'))

data = data[,1:2]
m  <- colMeans(data)
m1 <- colMeans(data[i1,])
m2 <- colMeans(data[i2,])
m3 <- colMeans(data[i3,])

S1 <- cov(data[i1,])
S2 <- cov(data[i2,])
S3 <- cov(data[i3,])
Sp  <- ((n1-1)*S1+(n2-1)*S2+(n3-1)*S3)/(n-g)

# We check che i labels producano effettivamente effetto sulle variabili
# One-way MANOVA
fit <- manova(as.matrix(data) ~ labels)
summary.manova(fit,test="Wilks")

# ------------------------------------------------------------------------------------
# LDA (multivariate)
# ------------------------------------------------------------------------------------

lda.data <- lda(data, labels)
lda.data

# "coefficients of linear discriminants" and "proportion of trace":
# Fisher discriminant analysis.
# In particular:
# - coefficients of linear discriminants: versors of the canonical directions
#   [to be read column-wise]
# - proportion of trace: proportion of variance explained by the corresponding
#   canonical direction

# ------------------------------------------------------------------------------------
# ESTIMATE OF AER
# 1. APER
# 2. AER by cross-validation
# ------------------------------------------------------------------------------------

# 1)
Lda.data <- predict(lda.data, data)
names(Lda.data)

table(class.true=labels, class.assigned=Lda.data$class)

errors = (Lda.data$class != labels)
sum(errors)
length(labels)

APER = sum(errors)/length(labels)
APER

#### NOTE: if there are priors given:
# prior <- c(1/3,1/3,1/3)
# G <- 3
# misc <- table(class.true=labels, class.assigned=Lda.data$class)
# APER <- 0
# for(g in 1:G)
#   APER <- APER + sum(misc[g,-g])/sum(misc[g,]) * prior[g]

# 2)
LdaCV.data <- lda(data, labels, CV=TRUE)  # specify the argument CV
table(class.true=labels, class.assignedCV=LdaCV.data$class)
errorsCV <- (LdaCV.data$class != labels)
AERCV   <- sum(errorsCV)/length(labels)
AERCV

# ------------------------------------------------------------------------------------
# plot the partition induced by LDA
# ------------------------------------------------------------------------------------
plot(data, main='Iris Sepal', xlab='X1', ylab='X1', pch=20)
points(data[i1,], col='red', pch=20)
```

given priors!

(2D)

```
points(data[i2,], col='green', pch=20)
points(data[i3,], col='blue', pch=20)
legend("topright", legend=levels(labels), fill=c('red','green','blue'), cex=.7)
points(lda.data$means, pch=4,col=c('red','green','blue') , lwd=2, cex=1.5)

x  <- seq(min(data[,1]), max(iris[,1]), length=200)
y  <- seq(min(data[,2]), max(iris[,2]), length=200)
xy <- expand.grid(X1=x, X2=y)

z  <- predict(lda.data, xy)$post  # these are P_i*f_i(x,y)
z1 <- z[,1] - pmax(z[,2], z[,3])  # P_1*f_1(x,y)-max{P_j*f_j(x,y)}
z2 <- z[,2] - pmax(z[,1], z[,3])  # P_2*f_2(x,y)-max{P_j*f_j(x,y)}
z3 <- z[,3] - pmax(z[,1], z[,2])  # P_3*f_3(x,y)-max{P_j*f_j(x,y)}

# Plot the contour line of level (levels=0) of z1, z2, z3:
# P_i*f_i(x,y)-max{P_j*f_j(x,y)}=0 i.e., boundary between R.i and R.j
# where j realizes the max.
contour(x, y, matrix(z1, 200), levels=0, drawlabels=F, add=T)
contour(x, y, matrix(z2, 200), levels=0, drawlabels=F, add=T)
contour(x, y, matrix(z3, 200), levels=0, drawlabels=F, add=T)

# ------------------------------------------------------------------------------
# 3D plot of the partition induced by LDA
# ------------------------------------------------------------------------------

open3d()
points3d(data[i1,1], data[i1,2], 0, col='red', pch=15)
points3d(data[i2,1], data[i3,2], 0, col='green', pch=15)
points3d(data[i3,1], data[i2,2], 0, col='blue', pch=15)
surface3d(x,y,matrix(dmvnorm(xy, m1, Sp) / 3, 50), alpha=0.4, color='red')
surface3d(x,y,matrix(dmvnorm(xy, m2, Sp) / 3, 50), alpha=0.4, color='green', add=T)
surface3d(x,y,matrix(dmvnorm(xy, m3, Sp) / 3, 50), alpha=0.4, color='blue', add=T)
box3d()

# ------------------------------------------------------------------------------
# QDA (multivariate)
# ------------------------------------------------------------------------------

qda.data <- qda(data, labels)
qda.data

Qda.data <- predict(qda.data, data)

# APER
# Remark: correct only if we estimate the priors through the sample frequencies!
table(class.true=labels, class.assigned=Qda.data$class)
errorsq    <- (Qda.data$class != labels)
APERq      <- sum(errorsq)/length(labels)

# AER
QdaCV.data <- qda(data, labels, CV=T)
errorsqCV  <- (QdaCV.data$class != labels)
AERqCV     <- sum(errorsqCV)/length(labels)

# ------------------------------------------------------------------------------
# plot the partition induced by QDA
# ------------------------------------------------------------------------------
plot(data, main='Main Title', xlab='X1', ylab='X2', pch=20)
points(data[i1,], col='red', pch=20)
points(data[i2,], col='green', pch=20)
points(data[i3,], col='blue', pch=20)
legend("topright", legend=levels(labels), fill=c('red','green','blue'))
points(qda.data$means, col=c('red','green','blue'), pch=4, lwd=2, cex=1.5)

x  <- seq(min(iris[,1]), max(data[,1]), length=200)
y  <- seq(min(iris[,2]), max(data[,2]), length=200)
xy <- expand.grid(X1=x, X2=y)

z  <- predict(qda.data, xy)$post
z1 <- z[,1] - pmax(z[,2], z[,3])
z2 <- z[,2] - pmax(z[,1], z[,3])
z3 <- z[,3] - pmax(z[,1], z[,2])

contour(x, y, matrix(z1, 200), levels=0, drawlabels=F, add=T)
contour(x, y, matrix(z2, 200), levels=0, drawlabels=F, add=T)
contour(x, y, matrix(z3, 200), levels=0, drawlabels=F, add=T)

# ------------------------------------------------------------------------------
# 3D plot of the partition induced by LDA
# ------------------------------------------------------------------------------

open3d()
points3d(data[i1,1], iris2[i1,2], 0, col='red', pch=15)
points3d(data[i2,1], iris2[i3,2], 0, col='green', pch=15)
points3d(data[i3,1], iris2[i2,2], 0, col='blue', pch=15)
surface3d(x,y,matrix(dmvnorm(xy, m1, S1) / 3, 50), alpha=0.4, color='red')
surface3d(x,y,matrix(dmvnorm(xy, m2, S2) / 3, 50), alpha=0.4, color='green', add=T)
```

```
surface3d(x,y,matrix(dmvnorm(xy, m3, S3) / 3, 50), alpha=0.4, color='blue', add=T)
box3d()
```

KNN

```
# ------------------------------------------------------------------------------
# KNN
# ------------------------------------------------------------------------------

# Plot the partition induced by knn

k <- 7

plot(data, main='Iris.Sepal', xlab='X1', ylab='X2', pch=20)
points(data[i1,], col=2, pch=20)
points(data[i3,], col=4, pch=20)
points(data[i2,], col=3, pch=20)
legend("topright", legend=levels(labels), fill=c(2,3,4))

x  <- seq(min(data[,1]), max(data[,1]), length=200)
y  <- seq(min(data[,2]), max(data[,2]), length=200)
xy <- expand.grid(Sepal.Length=x, Sepal.Width=y)

data.knn <- knn(train = data, test = xy, cl = labels, k = k)

z  <- as.numeric(data.knn)

contour(x, y, matrix(z, 200), levels=c(1.5, 2.5), drawlabels=F, add=T)
```

FISHER's argument

```
# ------------------------------------------------------------------------------
# Fisher's argument          --   start
# Let's look at the directions that highlight the discrimination among groups:
# => canonical directions
# ------------------------------------------------------------------------------
# covariance between groups (estimate)
B <- 1/g*(cbind(m1 - m) %*% rbind(m1 - m) +
            cbind(m2 - m) %*% rbind(m2 - m) +
            cbind(m3 - m) %*% rbind(m3 - m))
B

# covariance within groups (estimate)
Sp

s <- min(g-1,p)

val.Sp <- eigen(Sp)$val
vec.Sp <- eigen(Sp)$vec
invSp.2 <- 1/sqrt(val.Sp[1])*vec.Sp[,1]%*%t(vec.Sp[,1]) +
  1/sqrt(val.Sp[2])*vec.Sp[,2]%*%t(vec.Sp[,2])
invSp.2

# spectral decomposition of Sp^(-1/2) B Sp^(-1/2)
spec.dec <- eigen(invSp.2 %*% B %*% invSp.2)

# first canonical coordinate
a1 <- invSp.2 %*% spec.dec$vec[,1]
a1

# second canonical coordinate
a2 <- invSp.2 %*% spec.dec$vec[,2]
a2

# compare with the output of lda():
lda.data

# ------------------------------------------------------------------------------
# How are the data classified?
# ------------------------------------------------------------------------------
# Compute the canonical coordinates of the data

cc1.data <- as.matrix(data)%*%a1
cc2.data <- as.matrix(data)%*%a2

coord.cc <- cbind(cc1.data,cc2.data)

# Compute the coordinates of the mean within groups along the canonical directions
cc.m1 <- c(m1%*%a1, m1%*%a2)
cc.m2 <- c(m2%*%a1, m2%*%a2)
cc.m3 <- c(m3%*%a1, m3%*%a2)

# Assign data to groups
f.class=rep(0, n)
for(i in 1:n) # for each datum
{
  # Compute the Euclidean distance of the i-th datum from mean within the groups
  dist.m=c(d1=sqrt(sum((coord.cc[i,]-cc.m1)^2)),
           d2=sqrt(sum((coord.cc[i,]-cc.m2)^2)),
           d3=sqrt(sum((coord.cc[i,]-cc.m3)^2)))
  # Assign the datum to the group whose mean is the nearest
```

FISHER's argument

```
  f.class[i]=which.min(dist.m)
}
f.class

table(class.true=labels, class.assigned=f.class)

errors <- n - sum(diag(table(class.true=labels, class.assigned=f.class)))
APERf   <- errors/length(labels)

# ------------------------------------------------------------------------------
# How do I classify a new observation?
# ------------------------------------------------------------------------------
x.new <- c(5.85, 2.90)
# compute the canonical coordinates
cc.new <- c(x.new%*%a1, x.new%*%a2)
# compute the distance from the means
dist.m <- c(d1=sqrt(sum((cc.new-cc.m1)^2)),
            d2=sqrt(sum((cc.new-cc.m2)^2)),
            d3=sqrt(sum((cc.new-cc.m3)^2)))
# assign to the nearest mean
which.min(dist.m)

# ------------------------------------------------------------------------------
# visually
# ------------------------------------------------------------------------------
color.species=rep(c('red','green','blue'), each=50)
iris2 = data
species.name = labels
par(mfrow=c(1,2))
plot(iris2[,1], iris2[,2], main='Plane of original coordinates',
     xlab='Sepal.Length', ylab='Sepal.Width', pch=20, col=as.character(color.species))
legend("topleft", legend=levels(species.name), fill=c('red','green','blue'), cex=.7)
points(x.new[1], x.new[2], col='gold', pch=19)
points(m1[1], m1[2], pch=4,col='red' , lwd=2, cex=1.5)
points(m2[1], m2[2], pch=4,col='green' , lwd=2, cex=1.5)
points(m3[1], m3[2], pch=4,col='blue' , lwd=2, cex=1.5)
plot(cc1.data, cc2.data, main='Plane of canonical coordinates',
     xlab='first canonical coordinate', ylab='second canonical coordinate',
     pch=20, col=as.character(color.species))
legend("topleft", legend=levels(species.name), fill=c('red','green','blue'), cex=.7)
points(cc.m1[1], cc.m1[2], pch=4,col='red' , lwd=2, cex=1.5)
points(cc.m2[1], cc.m2[2], pch=4,col='green' , lwd=2, cex=1.5)
points(cc.m3[1], cc.m3[2], pch=4,col='blue' , lwd=2, cex=1.5)
points(cc.new[1], cc.new[2], col='gold', pch=19)
segments(cc.m1[1], cc.m1[2], cc.new[1], cc.new[2])
segments(cc.m2[1], cc.m2[2], cc.new[1], cc.new[2])
segments(cc.m3[1], cc.m3[2], cc.new[1], cc.new[2])
dev.off()

# ------------------------------------------------------------------------------
# We plot the partition generated by the canonical coordinates
# ------------------------------------------------------------------------------
cc1.iris = cc1.data
cc2.iris = cc2.data
color.species <- species.name
levels(color.species) <- c('red','green','blue')
plot(cc1.iris, cc2.iris, main='Fisher discriminant analysis',
     xlab='first canonical coordinate', ylab='second canonical coordinate',
     pch=20, col=as.character(color.species))
legend("topleft", legend=levels(species.name), fill=c('red','green','blue'), cex=.7)
points(cc.m1[1], cc.m1[2], pch=4,col='red' , lwd=2, cex=1.5)
points(cc.m2[1], cc.m2[2], pch=4,col='green' , lwd=2, cex=1.5)
points(cc.m3[1], cc.m3[2], pch=4,col='blue' , lwd=2, cex=1.5)
x.cc  <- seq(min(cc1.iris),max(cc1.iris),len=200)
y.cc  <- seq(min(cc2.iris),max(cc2.iris),len=200)
xy.cc <- expand.grid(cc1=x.cc, cc2=y.cc)
z  <- cbind( sqrt(rowSums(scale(xy.cc,cc.m1,scale=FALSE)^2)),
             sqrt(rowSums(scale(xy.cc,cc.m2,scale=FALSE)^2)),
             sqrt(rowSums(scale(xy.cc,cc.m3,scale=FALSE)^2)))
z1.cc <- z[,1] - pmin(z[,2], z[,3])
z2.cc <- z[,2] - pmin(z[,1], z[,3])
z3.cc <- z[,3] - pmin(z[,1], z[,2])
contour(x.cc, y.cc, matrix(z1.cc, 200), levels=0, drawlabels=F, add=T)
contour(x.cc, y.cc, matrix(z2.cc, 200), levels=0, drawlabels=F, add=T)
contour(x.cc, y.cc, matrix(z3.cc, 200), levels=0, drawlabels=F, add=T)

# ------------------------------------------------------------------------------
# Plot LDA
# ------------------------------------------------------------------------------
lda.iris = lda.data
plot(iris2, main='Iris Sepal', xlab='Sepal.Length', ylab='Sepal.Width', pch=20)
points(iris2[i1,], col='red', pch=20)
points(iris2[i2,], col='green', pch=20)
points(iris2[i3,], col='blue', pch=20)
legend("topright", legend=levels(species.name), fill=c('red','green','blue'), cex=.7)
points(rbind(m1,m2,m3), pch=4,col=c('red','green','blue') , lwd=2, cex=1.5)
x  <- seq(min(iris[,1]), max(iris[,1]), length=200)
```

FISHER'S argument

```
y  <- seq(min(iris[,2]), max(iris[,2]), length=200)
xy <- expand.grid(Sepal.Length=x, Sepal.Width=y)
z  <- predict(lda.iris, xy)$post  # these are P_i*f_i(x,y)
z1 <- z[,1] - pmax(z[,2], z[,3])  # P_1*f_1(x,y)-max{P_j*f_j(x,y)}
z2 <- z[,2] - pmax(z[,1], z[,3])  # P_2*f_2(x,y)-max{P_j*f_j(x,y)}
z3 <- z[,3] - pmax(z[,1], z[,2])  # P_3*f_3(x,y)-max{P_j*f_j(x,y)}

# Plot the contour line of level (levels=0) of z1, z2, z3:
# P_i*f_i(x,y)-max{P_j*f_j(x,y)}=0 i.e., boundary between R.i and R.j
# where j realizes the max.
contour(x, y, matrix(z1, 200), levels=0, drawlabels=F, add=T)
contour(x, y, matrix(z2, 200), levels=0, drawlabels=F, add=T)
contour(x, y, matrix(z3, 200), levels=0, drawlabels=F, add=T)

# ------------------------------------------------------------------------------
# Plot of the projections on the canonical directions (not orthogonal!)
# ------------------------------------------------------------------------------
plot(iris2, main='Projection on the canonical directions',
     xlab='Sepal.Length', ylab='Sepal.Width', pch=20, xlim=c(-3,8), ylim=c(-3,7))
points(iris2[i1,], col='red', pch=20)
points(iris2[i2,], col='green', pch=20)
points(iris2[i3,], col='blue', pch=20)
legend('topleft', legend=levels(species.name), fill=c('red','green','blue'), cex=.7)
points(rbind(m1,m2,m3), pch=4,col=c('red','green','blue') , lwd=2, cex=1.5)
abline(h=0,v=0, col='grey35')
arrows(x0=0, y0=0, x1=a1[1], y1=a1[2], length=.1)
arrows(x0=0, y0=0, x1=a2[1], y1=a2[2], length=.1)
text(a1[1], a1[2], 'a1',pos=3)
text(a2[1], a2[2], 'a2',pos=2)
abline(coef=c(0,(a1[2]/a1[1])), col='grey55',lty=2)
abline(coef=c(0,(a2[2]/a2[1])), col='grey55',lty=2)
points(cc1.iris*a1[1]/(sum(a1^2)),cc1.iris*a1[2]/(sum(a1^2)),
       col=as.character(color.species))
points(cc2.iris*a2[1]/(sum(a2^2)),cc2.iris*a2[2]/(sum(a2^2)),
       col=as.character(color.species))

plot(cc1.iris, cc2.iris, main='Coordinate system of the canonical coordinates',
     xlab='first canonical coordinate', ylab='second canonical coordinate',
     pch=20, col=as.character(color.species))
legend('topleft', legend=levels(species.name), fill=c('red','green','blue'), cex=.7)
cc.m1 <- c(m1%*%a1, m1%*%a2)
cc.m2 <- c(m2%*%a1, m2%*%a2)
cc.m3 <- c(m3%*%a1, m3%*%a2)
points(cc.m1[1], cc.m1[2], pch=4,col='red' , lwd=2, cex=1.5)
points(cc.m2[1], cc.m2[2], pch=4,col='green' , lwd=2, cex=1.5)
points(cc.m3[1], cc.m3[2], pch=4,col='blue' , lwd=2, cex=1.5)

# ------------------------------------------------------------------------------
# Fisher's argument          --   end
# ------------------------------------------------------------------------------

### ------------------------------------------------------------------------------
### (3 classes, bivariate)   --   end
### (p = 2, g = 3)
### ------------------------------------------------------------------------------
```

Unsup learn

Hierarchical

```
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
### Unsupervised: Hierarchical, K-means clustering
###-------------------------------------------------------------------------------
###-------------------------------------------------------------------------------
library(mvtnorm)
library(rgl)
library(car)
load("mcshapiro.test.RData")

### ----------------------------------------------------------------------------
### HIERARCHICAL CLUSTERING
### (p=4, g=3, ma va bene con tutto, non c'è da cambiare)
### ----------------------------------------------------------------------------

data_grezzi = iris

n = dim(data_grezzi)[1]
p = dim(data_grezzi)[2]-1

# data without labels
data = data_grezzi[,1:4]

# data'slabels
species.name = as.factor(data_grezzi[,5])
g = length(levels(species.names))

pairs(data)

## Se non c'è il label
## n = dim(data)[1]
## p = dim(data)[2]

# ----------------------------------------------------------------------------
# Compute dissimilarities
# methods: "euclidean", "manhattan", "canberra"
# ----------------------------------------------------------------------------
iris.e = dist(data, method="euclidean")
iris.m = dist(data, method="manhattan")
iris.c = dist(data, method="canberra")

par(mfrow=c(1,3))
image(1:n,1:n,as.matrix(iris.e), main='metrics: Euclidean', asp=1, xlab='i', ylab='j' )
image(1:n,1:n,as.matrix(iris.c), main='metrics: Canberra',  asp=1, xlab='i', ylab='j' )
image(1:n,1:n,as.matrix(iris.m), main='metrics: Manhattan', asp=1, xlab='i', ylab='j' )
dev.off()

# Comment
# light colors = small values
# dark  colors = large values

# ----------------------------------------------------------------------------
# Hierarchical clustering
# distance: "euclidean"
# linkages: "single", "average", "complete", "ward"
# ----------------------------------------------------------------------------
iris.es <- hclust(iris.e, method='single')
iris.ea <- hclust(iris.e, method='average')
iris.ec <- hclust(iris.e, method='complete')
iris.wa <- hclust(iris.e, method="ward.D2")

# order of aggregation
iris.es$merge

# Comment
# [128,] -119   80    (esempio)
# vuol dire che allo step 128 l'unità 119 è stata aggregata al cluster prodotto
# allo step 80. Se entrambi sono positivi, es.[140,] 90 95, allora si stanno aggregando
# i cluster prodotti allo step 90 e 95. Se sono entrambi negativi allora si sta creando
# un nuovo cluster

# distance at which we have aggregations
iris.es$height

# ordering that allows to avoid intersection in the dendrogram
iris.es$order

# ----------------------------------------------------------------------------
# Plot of the dendrograms
# ----------------------------------------------------------------------------
par(mfrow=c(2,2))
plot(iris.es, main='euclidean-single',   hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
plot(iris.ec, main='euclidean-complete', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
plot(iris.ea, main='euclidean-average',  hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
plot(iris.wa, main='euclidean-ward',     hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
```

Hierarchical

```
# ------------------------------------------------------------------------------
# Cutting the dendrogram (k=2 clusters)
# ------------------------------------------------------------------------------
par(mfrow=c(2,2))
plot(iris.es, main='euclidean-single',   hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.es, k=2)
plot(iris.ec, main='euclidean-complete', hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.ec, k=2)
plot(iris.ea, main='euclidean-average',  hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.ea, k=2)
plot(iris.wa, main='euclidean-ward',     hang=-0.1, xlab='', labels=F, cex=0.6, sub='')
rect.hclust(iris.wa, k=2)

# How to cut a dendrogram? (k=2)
# We generate vectors of labels through the command cutree()
cluster.ec <- cutree(iris.ec, k=2)
cluster.es <- cutree(iris.es, k=2)
cluster.ea <- cutree(iris.ea, k=2)
cluster.wa <- cutree(iris.wa, k=2)

# ------------------------------------------------------------------------------
# How good is the clustering?
# ------------------------------------------------------------------------------
# Did it aggregate coherently with the dissimilarity matrix or not?

# Cophenetic Matrices
coph.es <- cophenetic(iris.es)
coph.ec <- cophenetic(iris.ec)
coph.ea <- cophenetic(iris.ea)
coph.wa <- cophenetic(iris.wa)

# Compare with dissimilarity matrix (Euclidean distance)
#layout(rbind(c(0,1,0),c(2,3,4,5)))
#image(as.matrix(iris.e),  main='Euclidean', asp=1 )
#image(as.matrix(coph.es), main='Single', asp=1 )
#image(as.matrix(coph.ec), main='Complete', asp=1 )
#image(as.matrix(coph.ea), main='Average', asp=1 )
#image(as.matrix(coph.wa), main='Ward', asp=1)

# Cophenetic Coefficients
es <- cor(iris.e, coph.es)
ec <- cor(iris.e, coph.ec)
ea <- cor(iris.e, coph.ea)
ew <- cor(iris.e, coph.wa)
c("Eucl-Single"=es,"Eucl-Compl."=ec,"Eucl-Ave."=ea, "Eucl-Ward"=ew)

# interpret the clusters (SOLO se abbiamo i true labels)
table(label.true = species.name, label.cluster = cluster.es)
table(label.true = species.name, label.cluster = cluster.ec)
table(label.true = species.name, label.cluster = cluster.ea)
table(label.true = species.name, label.cluster = cluster.wa)

# Plot
plot(data, col=ifelse(cluster.es==1,'red','blue'), pch=19)
plot(data, col=ifelse(cluster.ec==1,'red','blue'), pch=19)
plot(data, col=ifelse(cluster.ea==1,'red','blue'), pch=19)
plot(data, col=ifelse(cluster.wa==1,'red','blue'), pch=19)

# ------------------------------------------------------------------------------
# Se p=3 => plot 3d
# ------------------------------------------------------------------------------
# Data
plot3d(data, size=3, col='orange', aspect=F)

# Single linkage
plot3d(data, size=3, col=cluster.es+1, aspect=F)

# Average linkage
plot3d(data, size=3, col=cluster.ea+1, aspect=F)

# Complete linkage
plot3d(data, size=3, col=cluster.ec+1, aspect=F)

# Ward linkage
plot3d(data, size=3, col=cluster.wa+1, aspect=F)
```

K-mean

```
### ------------------------------------------------------------------------------
### K-MEAN CLUSTERING
### (p=4, g=3)
### ------------------------------------------------------------------------------

data_grezzi = iris

n = dim(data_grezzi)[1]
p = dim(data_grezzi)[2]-1

# data without labels
```

K-means

```
data = data_grezzi[,1:4]

# data'slabels
species.name = as.factor(data_grezzi[,5])
g = length(levels(species.names))

pairs(data)

## Se non c'è il label
## n = dim(data)[1]
## p = dim(data)[2]

# ------------------------------------------------------------------------------
# K-means
# ------------------------------------------------------------------------------

# We fix the number of centers
result.k = kmeans(data, centers=2)

# Labels of cluster
result.k$cluster

# Centers of clusters
result.k$centers

# Tot sum of squares
result.k$totss

# Sum of squares within clusters
result.k$withinss

# sum(sum of squares nei clusters)
result.k$tot.withinss

# sum of squares between clusters
result.k$betweenss

# dimension of the clusters
result.k$size

# Plot
plot(data, col = result.k$cluster+1)
dev.off()

# Plot 3d (se p=3)
plot3d(data, size=3, col=result.k$cluster+1, aspect = F)
points3d(result.k$centers,size=10)

# ------------------------------------------------------------------------------
# Come scegliere k
# ------------------------------------------------------------------------------
# Evaluate the variability between the groups with respect to the variability
# withing the groups

b <- NULL
w <- NULL
for(k in 1:10){
  result.k <- kmeans(data, k)
  w <- c(w, sum(result.k$wit))
  b <- c(b, result.k$bet)
}
matplot(1:10, w/(w+b), pch='', xlab='clusters', ylab='within/tot',
        main='Choice of k', ylim=c(0,1))
lines(1:10, w/(w+b), type='b', lwd=2)
```