PCA: applicazioni

Contents

Dati Wine

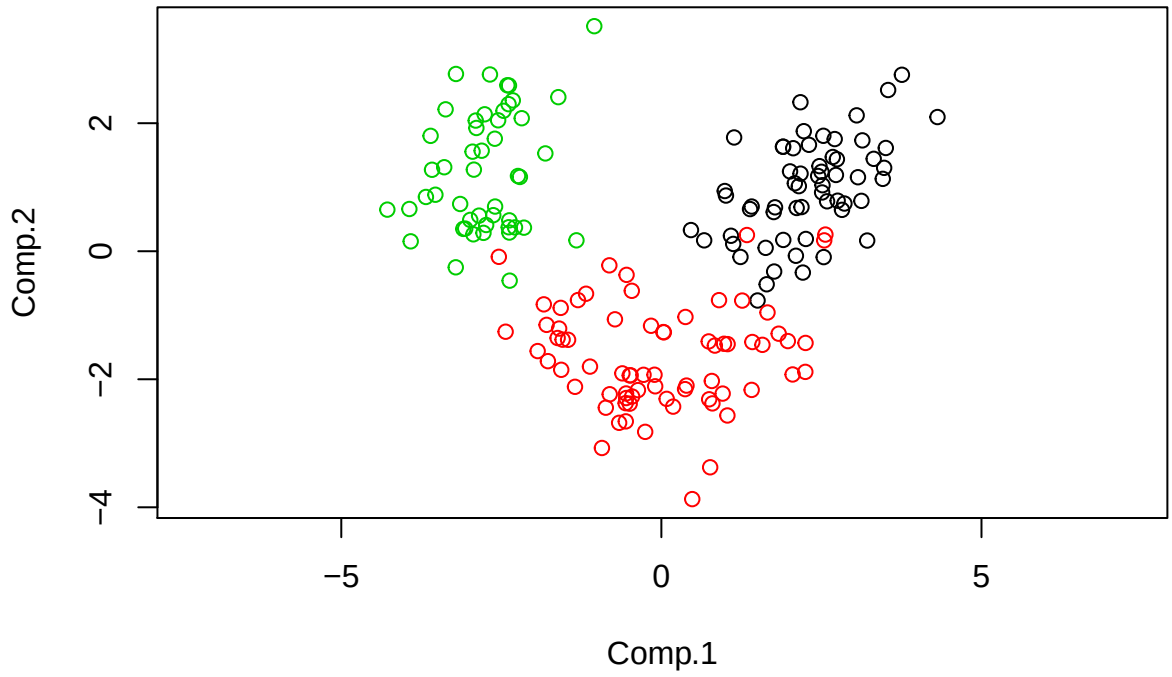
1. Importare i dati ed effettuare l'analisi delle componenti principali dei dati standardizzati $\underset{n\times p}{Z}$ (escludendo la variabile `type`), calcolando la matrice dei punteggi $\underset{n\times p}{Y} = \underset{n\times p}{X}\,\underset{p\times p}{Z}$ e la matrice dei pesi $\underset{p\times p}{V}$. Calcolare la percentuale di varianza spiegata dalle prime q componenti principali, per $q = 1, \dots, 10$.

```
rm(list=ls())
wine <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data", header  = F
colnames(wine)<-c("Type","Alcohol",
                  "Malic","Ash",
                  "Alcalinity","Magnesium",
                  "Phenols","Flavanoids",
                  "Nonflavanoids","Proanthocyanins",
                  "Color_int","Hue","Dilution","Proline")
X = as.matrix(wine[,-1])
n = nrow(X)
p = ncol(X)

# PCA
pca = princomp(X, cor=T)

V = pca$loadings
Y = pca$scores

# prop. di varianza spiegata dalle prime q cp
q <- 10
cumsum(pca$sdev^2/sum(pca$sdev^2))[1:q]
```

```
   Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7    Comp.8    Comp.9   Comp.10
0.3619885 0.5540634 0.6652997 0.7359900 0.8016229 0.8509812 0.8933680 0.9201754 0.9423970 0.9616972
```

2. Costruire il diagramma di dispersione dei punteggi delle prime due componenti principali, colorando i punti con colori diversi a seconda della tipologia di vino (variabile `type`).

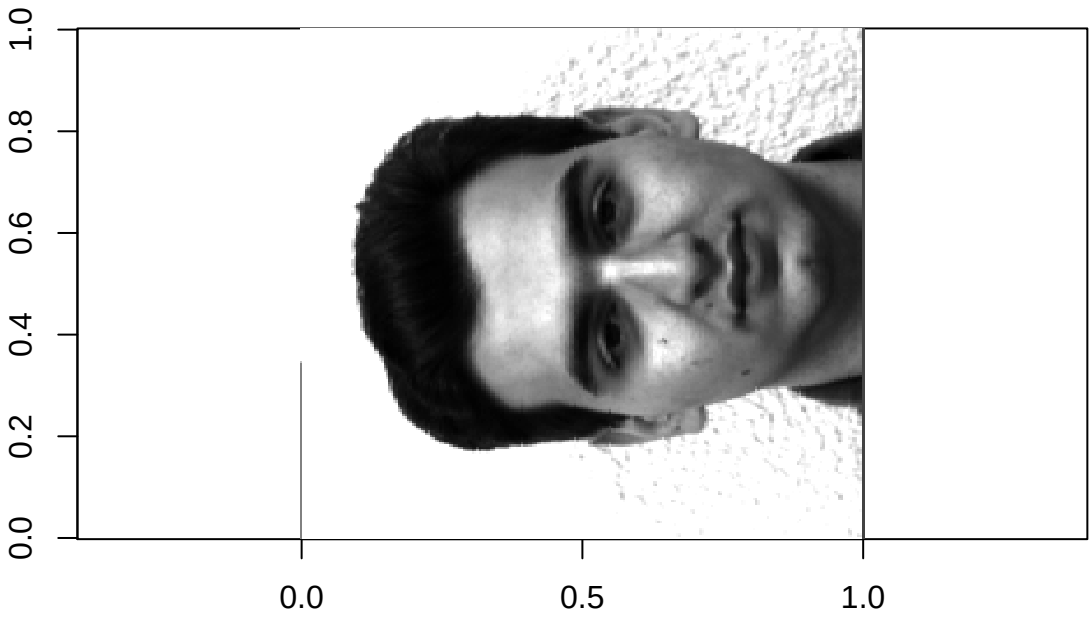
```r
plot(Y[,1:2], col=as.numeric(wine$Type), asp=1)
```



Dati Face

1. Importare i dati e visualizzare l'immagine con il comando `image`. Effettuare l'analisi delle componenti principali dei dati centrati $\underset{n\times p}{\tilde{X}}$, calcolando la matrice dei punteggi $\underset{n\times p}{Y} = \underset{n\times p}{\tilde{X}}\,\underset{p\times p}{V}$ e la matrice dei pesi $\underset{p\times p}{V}$.

```r
rm(list=ls())
face <- read.table("https://raw.githubusercontent.com/aldosolari/AE/master/dati/face.txt", header=FALSE)

X = as.matrix(face)
n = nrow(face)
p = ncol(face)

# visualizza immagine
image(X, col=gray(0:255/255), asp=p/n)
```



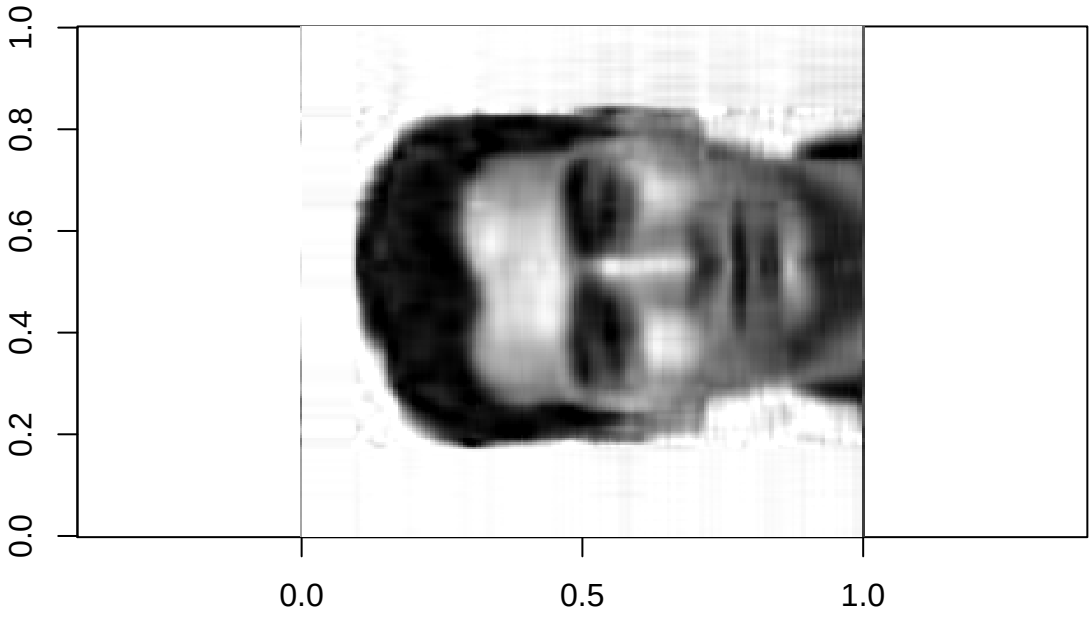

```
# PCA
pca = princomp(X, cor=F)
V = pca$loadings
Y = pca$scores
xbar = matrix(pca$center, ncol=1)
```

2. Ottenere la migliore approssimazione per $\underset{n\times p}{\tilde{X}}$ di rango q, $\underset{n\times p}{A_q} = \underset{n\times q}{Y_q}\ \underset{q\times p}{V_q'}$, con $q = 10$. Costruire l'immagine compressa $\underset{n\times p}{C} = \underset{n\times p}{A_q} + \underset{n\times 1}{1}\ \underset{1\times p}{\bar{x}}$, assicurandosi che tutti gli elementi di $\underset{n\times p}{C}$ siano compresi tra 0 e 1.

```
q = 10
Yq = Y[,1:q]
Vq = V[,1:q]

# migliore approssimazione di rango q
Aq = Yq %*% t(Vq)

# compressione immagine
one.n = matrix(rep(1,n), ncol=1)
face2 = Aq + one.n %*% t(xbar)

# forzo i valori tra 0 e 1
face2 <- pmax(pmin(face2, 1), 0)
```

3. Visualizzare l'immagine compressa e confrontarla con l'immagine originale calcolando il fattore di riduzione in termini di pixels e bytes (utilizzando il comando `object.size`)

```
# visualizza immagine compressa
image(face2, col=gray(0:255/255), asp=p/n)
```




```
# salve immagine compressa
# library(png)
# writePNG(face,"face.png")

# confronta pixels utilizzati

pixels = prod(dim(face))
pixels2 = prod(dim(Yq)) + prod(dim(Vq)) + prod(dim(xbar))
round(pixels/pixels2, 2) # fattore di riduzione
```

```
[1] 11.02
```

```
# confronta memoria utilizzata
size = object.size(face)
size2 = object.size(Yq) + object.size(Vq) + object.size(xbar)
round( size/size2, 2) # fattore di riduzione
```

```
8.2 bytes
```