STAT 3690 Lecture 23

zhiyanggeezhou.github.io

Zhiyang Zhou (zhiyang.zhou@umanitoba.ca)

Mar 25, 2022

Application of (sample) PCA

- Image compression: `mnist` is a list with two components: `train` and `test`. Each of these is a list with two components: images and labels.
  - The `images` component is a matrix with each row for one image consisting of 28*28 = 784 entries (pixels). Their value are integers between 0 and 255 representing grey scale.
  - The `labels` components is a vector representing the digit shown in the image.

```
library(tidyverse)
mnist <- dslabs::read_mnist()
dim(mnist$train$images)
dim(mnist$test$images)

# The i0-th image in the training set
i0 = 2022
matrix(mnist$train$images[i0,], ncol = 28) %>%
  image(col = gray.colors(12, rev = TRUE), axes = FALSE)

# The digit for the i0-th image
mnist$train$labels[i0]

# PCA for training images
decomp <- prcomp(mnist$train$images)

# Plot the first 9 loadings
par(mfrow = c(3, 3))
for (i in seq_len(9)) {
  matrix(decomp$rotation[,i], ncol = 28) %>%
    image(col = gray.colors(12, rev = TRUE), axes = FALSE, main = paste0("PC", i))
}

# Plot training images according to their 1st and 2nd PC scores
decomp$x[,1:2] %>%
  as.data.frame() %>%
  mutate(label = factor(mnist$train$labels)) %>%
  ggplot(aes(PC1, PC2, colour = label)) +
  geom_point(alpha = 0.5) +
  theme_minimal()

# Plot testing images according to their 1st and 2nd PC scores
```

```r
decomp %>%
  predict(newdata = mnist$test$images) %>%
  as.data.frame() %>%
  mutate(label = factor(mnist$test$labels)) %>%
  ggplot(aes(PC1, PC2, colour = label)) +
  geom_point(alpha = 0.5) +
  theme_minimal()

# Figure out the termination point of PCA
s = which(cumsum((decomp$sdev)^2)/sum((decomp$sdev)^2)>=.9)[1]

# Approximating the 2022nd image in the training set with s PC scores
x.bar.train = colMeans(mnist$train$images)
approx_mnist <- x.bar.train + decomp$rotation[, seq_len(s)] %*% decomp$x[i0, seq_len(s)]
par(mfrow = c(1, 2))
matrix(approx_mnist, ncol = 28) %>%
  image(col = gray.colors(12, rev = TRUE), axes = FALSE, main = "Approx")
matrix(mnist$train$images[i0,], ncol = 28) %>%
  image(col = gray.colors(12, rev = TRUE), axes = FALSE, main = "Original")

# Approximating the 2022nd image in the testing set with s PC scores
PCscores = t(mnist$test$images[i0,] - x.bar.train) %*% decomp$rotation
approx_mnist <- x.bar.train + decomp$rotation[, seq_len(s)] %*% PCscores[1:s]
par(mfrow = c(1, 2))
matrix(approx_mnist, ncol = 28) %>%
  image(col = gray.colors(12, rev = TRUE), axes = FALSE, main = "Approx")
matrix(mnist$test$images[i0,], ncol = 28) %>%
  image(col = gray.colors(12, rev = TRUE), axes = FALSE, main = "Original")
```

- PC regression (PCR): regression on PC scores
  1. Perform PCA on the observed data matrix of explanatory variables, usually centered
  2. Regress the outcome vector(s) on the selected PCs as covariates using linear regression to get a vector of estimated regression coefficients
  3. Transform this coefficient vector back to the scale of the actual covariates
- Identity: the prediction of PCR is identical to that of linear regression, when all the PCs are included.

- Example of PCR: dataset `Prostate` comes from a study that examined the correlation between the level of prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy; see Stamey et al, 1989, Journal of Urology 141(5), 1076–1083.
  - `lcavol`: log(cancer volume)
  - `lweight`: log(prostate weight)
  - `age`: patient age
  - `lbph`: log(benign prostatic hyperplasia amount)
  - `svi`: seminal vesicle invasion
  - `lcp`: log(capsular penetration)
  - `gleason`: Gleason score
  - `pgg45`: percentage Gleason scores 4 or 5
  - `lpsa`: log(prostate specific antigen)

```r
install.packages(c('lasso2'))
library(lasso2)
data(Prostate)
set.seed(1)
train.idx = sample(nrow(Prostate), 80)
train = Prostate[train.idx,]
train.c = data.frame(
  lpsa= train$lpsa,
  scale(subset(train, select = -lpsa), center = T, scale = F)
)

# linear regression
fit1 = lm(lpsa~., data=train.c)
(beta1 = coef(fit1))

# PCR
decomp <- prcomp(subset(train.c, select = -lpsa))
s = ncol(train.c)-1
# (s = which(cumsum((decomp$sdev)^2)/sum((decomp$sdev)^2)>=.99)[1])
fit2 = lm(train.c$lpsa~decomp$x[,1:s])
(beta2 = coef(fit2))
(beta2.transform = c(beta2[1], decomp$rotation[,1:s] %*% beta2[-1]))
```