Exercício 7 - PCA

Reconhecimento de Padrões - UFMG

Guilherme Capanema de Barros (25 de Outubro de 2017)

1 Introdução

O método PCA (*Principal Component Analysis*) possibilita a redução da dimensão do problema a ser analisado. Ele faz isso projetando os dados em eixos que seguem as direções de máxima variância, de forma a eliminar dimensões que não oferecem boa separação entre os dados.

Para este trabalho, o método PCA será aplicado em 3 bases de dados:

1. **BreastCancer**, da biblioteca *mlbanch*;
2. **USArrests**, nativa do R;
3. **Cancer**, da biblioteca *simone*.

Para os itens 1 e 3, a aplicação do PCA será seguida da classificação pelo método SVM.

2 PCA

O PCA foi implementado em uma função cujo retorno são os autovetores $\mathbf{u}_i$ e os autovalores λ_i da matriz de covariâncias do vetor de pontos transladado.

```
1 myPCA <- function(X) {
2       xm <- matrix(colMeans(X), nrow=nrow(X), ncol=ncol(X),
              byrow=T)
3       Xtrans <- X - xm
4       S <- cov(Xtrans)
5       eigenS <- eigen(S)
6       u <- eigenS$vectors
7       lambda <- eigenS$values
8       return(list(u=u, lambda=lambda))
9 }
```

3 Experimentos

3.1 BreastCancer

3.1.1 PCA

A base de dados *BreastCancer* possui 9 atributos. A Figura 1 mostra os autovalores por dimensão PCA e a Figura 2 mostra os plots bi e tridimensional dos dados.

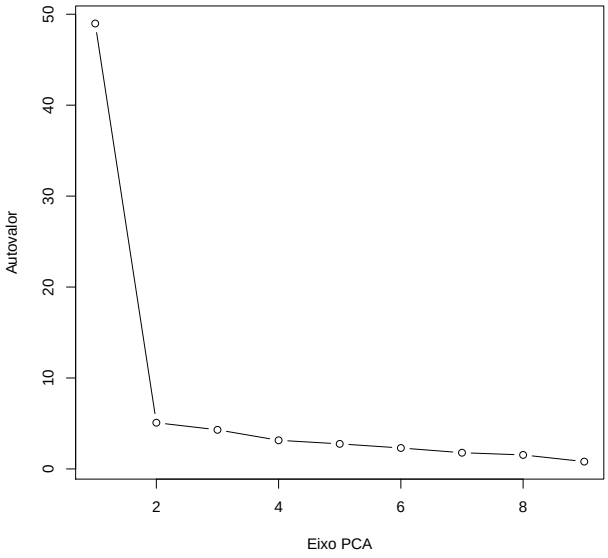

Figura 1: Autovalores por dimensão para a base de dados *BreastCancer*

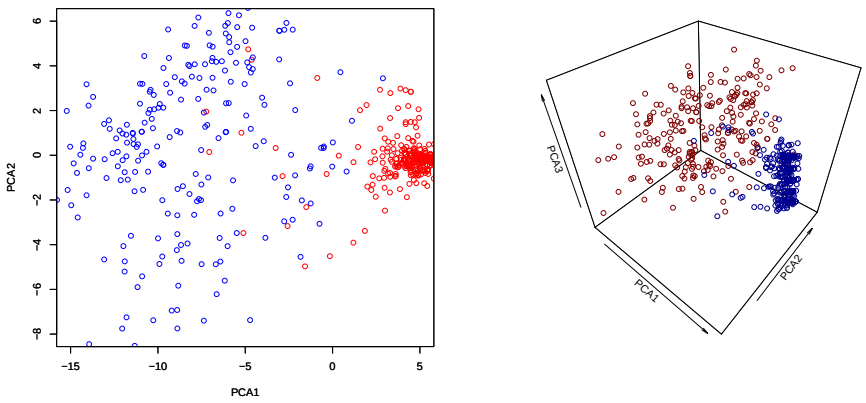

Figura 2: Plots bidimensional e tridimensional para a base de dados *BreastCancer*

Abaixo, o código usado para gerar essas visualizações:

```
source('myPCA.R')
source('mySVM.R')
source('projectPCA.R')
library(mlbench)
library(plot3D)

data(BreastCancer)

dataset <- na.omit(BreastCancer)
X <- data.matrix(dataset[,2:10])
benignIndexes <- (dataset$Class == 'benign')
malignantIndexes <- (dataset$Class == 'malignant')

# Apply PCA
pca <- myPCA(X)
plot(seq(from=1, to=ncol(X), by=1), pca$lambda, type='b',
     xlab='Eixo PCA', ylab='Autovalor')

# Plot 2D PCA space
proj2D <- projectPCA(X, 2)

xlim <- c(-15,5)
ylim <- c(-8, 6)
plot(proj2D[benignIndexes,1], proj2D[benignIndexes,2],
    xlim=xlim, ylim=ylim, xlab='PCA1', ylab='PCA2', col='
    red')
par(new=T)
plot(proj2D[malignantIndexes,1], proj2D[malignantIndexes
    ,2], xlim=xlim, ylim=ylim, xlab='PCA1', ylab='PCA2',
    col='blue')

# Plot 3D PCA space
proj3D <- projectPCA(X, 3)

colvar <- rep(1, nrow(X))
colvar[malignantIndexes] <- 4
points3D(proj3D[,1], proj3D[,2], proj3D[,3], colvar=
    colvar, colkey=FALSE, xlab='PCA1', ylab='PCA2', zlab='
    PCA3')
```

3.1.2 Classificação

Para fins de comparação, o algoritmo SVM desenvolvido em etapass anteriores foi aplicado na base de dados original, bidimensional e tridimensional. Os resultados obtidos estão apresentados na Tabela 1.

Tabela 1: Acurácias do SVM para a base *BreastCancer*

Original	2 dimensões	3 dimensões
96,08%	93,63%	93,63%

Abaixo, o código usado para gerar esses dados:

```
# Separating training, validation and test sets
swap <- sample(1:nrow(X))
attributes <- data.matrix(X[swap,])
classes <- dataset$Class[swap]

trainLim <- 0.5*nrow(attributes)
valLim <- 0.7*nrow(attributes)

# Calculating total SVM
training <- list(Attributes=attributes[1:trainLim,],
    Classes=classes[1:(trainLim)])
validation <- list(Attributes=attributes[(trainLim+1):
    valLim,], Classes=classes[(trainLim+1):valLim])
test <- list(Attributes=attributes[(valLim+1):nrow(
    attributes),], Classes=classes[(valLim+1):nrow(
    attributes)])

model <- mySVM(training, validation, c(0.1,2))
pred <- predict(model, test$Attributes)
err <- sum(pred != test$Classes)
acc <- 1 - err/length(pred)
print(acc)

# Calculating 2D SVM
attributes = proj2D[swap,]
training <- list(Attributes=attributes[1:trainLim,],
    Classes=classes[1:trainLim])
validation <- list(Attributes=attributes[(trainLim+1):
    valLim,], Classes=classes[(trainLim+1):valLim])
test <- list(Attributes=attributes[(valLim+1):nrow(
    attributes),], Classes=classes[(valLim+1):nrow(
    attributes)])

model2D <- mySVM(training, validation, c(0.1,2))
pred2D <- predict(model2D, test$Attributes)
err2D <- sum(pred2D != test$Classes)
acc2D <- 1 - err2D/length(pred2D)
print(acc2D)

# Calculating 3D SVM
attributes = proj3D[swap,]
training <- list(Attributes=attributes[1:trainLim,],
    Classes=classes[1:trainLim])
```

```
35 validation <- list(Attributes=attributes[(trainLim+1):
       valLim,], Classes=classes[(trainLim+1):valLim])
36 test <- list(Attributes=attributes[(valLim+1):nrow(
       attributes),], Classes=classes[(valLim+1):nrow(
       attributes)])
37
38 model3D <- mySVM(training, validation, c(0.1,2))
39 pred3D <- predict(model3D, test$Attributes)
40 err3D <- sum(pred3D != test$Classes)
41 acc3D <- 1 - err3D/length(pred3D)
42 print(acc3D)
```

3.2 USArrests

A base de dados *USArrests* possui 4 atributos. A Figura 3 mostra os autovalores por dimensão PCA e a Figura 4 mostra os plots bi e tridimensional dos dados.

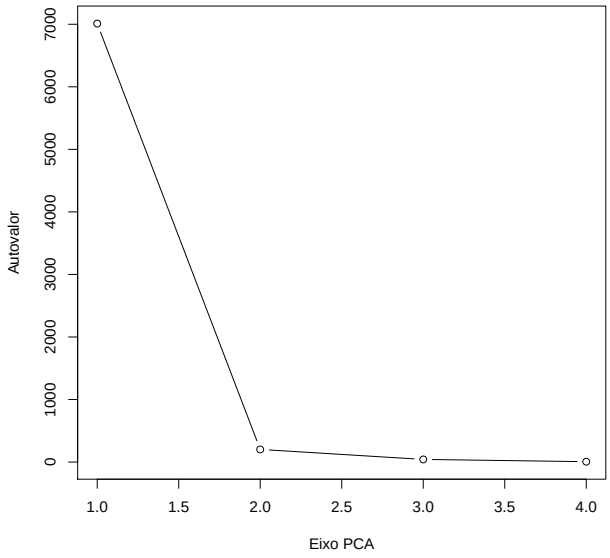

Figura 3: Autovalores por dimensão para a base de dados *USArrests*

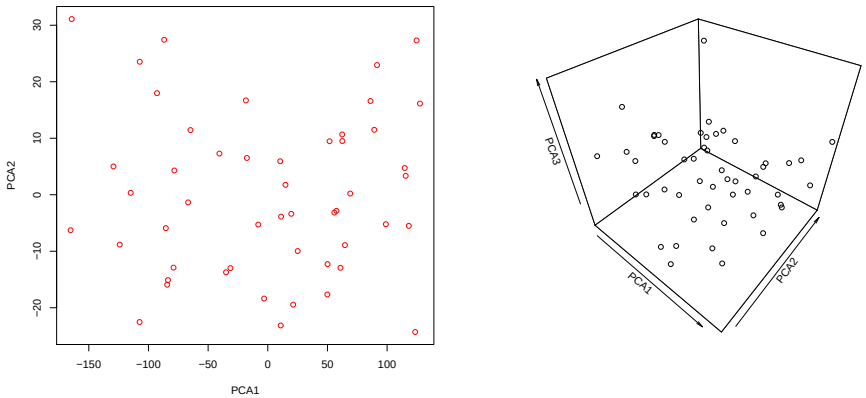

Figura 4: Plots bidimensional e tridimensional para a base de dados *USArrests*

Abaixo, o código usado para gerar essas visualizações:

```
1 source('myPCA.R')
2 source('mySVM.R')
```

```
3  source('projectPCA.R')
4  library(mlbench)
5  library(plot3D)
6
7  data(USArrests)
8
9  X <- data.matrix(USArrests)
10
11 # Apply PCA
12 pca <- myPCA(X)
13 plot(seq(from=1, to=ncol(X), by=1), pca$lambda, type='b',
       xlab='Eixo PCA', ylab='Autovalor')
14
15 # Plot 2D PCA space
16 proj2D <- projectPCA(X, 2)
17
18 xlim <- c(-15,5)
19 ylim <- c(-8, 6)
20 plot(proj2D[,1], proj2D[,2], xlab='PCA1', ylab='PCA2',
     col='red')
21
22 # Plot 3D PCA space
23 proj3D <- projectPCA(X, 3)
24
25 points3D(proj3D[,1], proj3D[,2], proj3D[,3], colvar=NULL,
       colkey=FALSE, xlab='PCA1', ylab='PCA2', zlab='PCA3')
```

3.3 Cancer

3.3.1 PCA

A base de dados *Cancer* possui 26 atributos. A Figura 5 mostra os autovalores por dimensão PCA e a Figura 6 mostra os plots bi e tridimensional dos dados.

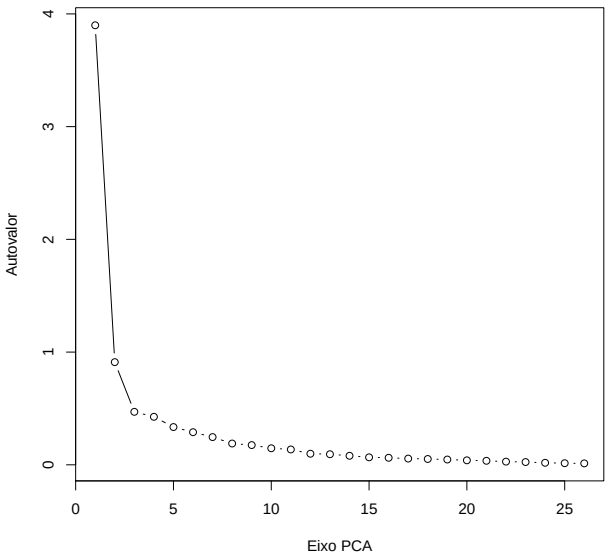

Figura 5: Autovalores por dimensão para a base de dados *Cancer*

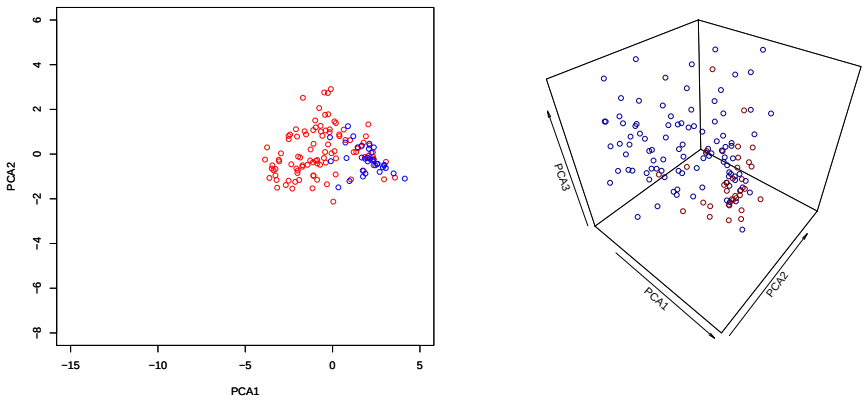

Figura 6: Plots bidimensional e tridimensional para a base de dados *Cancer*

Abaixo, o código usado para gerar essas visualizações:

```
1 source('myPCA.R')
```

```
2  source('mySVM.R')
3  source('projectPCA.R')
4  library(simone)
5  library(plot3D)
6
7  data(cancer)
8
9  X <- data.matrix(cancer$expr)
10 notIndexes <- (cancer$status == 'not')
11 pcrIndexes <- (cancer$status == 'pcr')
12
13 # Apply PCA
14 pca <- myPCA(X)
15 plot(seq(from=1, to=ncol(X), by=1), pca$lambda, type='b',
       xlab='Eixo PCA', ylab='Autovalor')
16
17 # Plot 2D PCA space
18 proj2D <- projectPCA(X, 2)
19
20 xlim <- c(-15,5)
21 ylim <- c(-8, 6)
22 plot(proj2D[notIndexes,1], proj2D[notIndexes,2], xlim=
     xlim, ylim=ylim, xlab='PCA1', ylab='PCA2', col='red')
23 par(new=T)
24 plot(proj2D[pcrIndexes,1], proj2D[pcrIndexes,2], xlim=
     xlim, ylim=ylim, xlab='PCA1', ylab='PCA2', col='blue')
25
26 # Plot 3D PCA space
27 proj3D <- projectPCA(X, 3)
28
29 colvar <- rep(1, nrow(X))
30 colvar[pcrIndexes] <- 4
31 points3D(proj3D[,1], proj3D[,2], proj3D[,3], colvar=
     colvar, colkey=FALSE, xlab='PCA1', ylab='PCA2', zlab='
     PCA3')
```

3.3.2 Classificação

Para fins de comparação, o algoritmo SVM desenvolvido em etapass anteriores foi aplicado na base de dados original, bidimensional e tridimensional. Os resultados obtidos estão apresentados na Tabela 2.

Tabela 2: Acurácias do SVM para a base *Cancer*

Original	2 dimensões	3 dimensões
84,62%	76,92%	72,92%

Abaixo, o código usado para gerar esses dados:

```
1 # Separating training, validation and test sets
```

```
2  swap <- sample(1:nrow(X))
3  attributes <- data.matrix(X[swap,])
4  classes <- cancer$status[swap]
5
6  trainLim <- 0.5*nrow(attributes)
7  valLim <- 0.7*nrow(attributes)
8
9  # Calculating total SVM
10 training <- list(Attributes=attributes[1:trainLim,],
       Classes=classes[1:(trainLim)])
11 validation <- list(Attributes=attributes[(trainLim+1):
       valLim,], Classes=classes[(trainLim+1):valLim])
12 test <- list(Attributes=attributes[(valLim+1):nrow(
       attributes),], Classes=classes[(valLim+1):nrow(
       attributes)])
13
14 model <- mySVM(training, validation, c(0.1,2))
15 pred <- predict(model, test$Attributes)
16 err <- sum(pred != test$Classes)
17 acc <- 1 - err/length(pred)
18 print(acc)
19
20 # Calculating 2D SVM
21 attributes = proj2D[swap,]
22 training <- list(Attributes=attributes[1:trainLim,],
       Classes=classes[1:trainLim])
23 validation <- list(Attributes=attributes[(trainLim+1):
       valLim,], Classes=classes[(trainLim+1):valLim])
24 test <- list(Attributes=attributes[(valLim+1):nrow(
       attributes),], Classes=classes[(valLim+1):nrow(
       attributes)])
25
26 model2D <- mySVM(training, validation, c(0.1,2))
27 pred2D <- predict(model2D, test$Attributes)
28 err2D <- sum(pred2D != test$Classes)
29 acc2D <- 1 - err2D/length(pred2D)
30 print(acc2D)
31
32 # Calculating 3D SVM
33 attributes = proj3D[swap,]
34 training <- list(Attributes=attributes[1:trainLim,],
       Classes=classes[1:trainLim])
35 validation <- list(Attributes=attributes[(trainLim+1):
       valLim,], Classes=classes[(trainLim+1):valLim])
36 test <- list(Attributes=attributes[(valLim+1):nrow(
       attributes),], Classes=classes[(valLim+1):nrow(
       attributes)])
37
38 model3D <- mySVM(training, validation, c(0.1,2))
39 pred3D <- predict(model3D, test$Attributes)
```

```
40 err3D <- sum(pred3D != test$Classes)
41 acc3D <- 1 - err3D/length(pred3D)
42 print(acc3D)
```

4 Conclusão

Fica claro que, para as bases de dados analisadas, a aplicação do PCA para redução das dimensões tem baixo impacto na acurácia final dos classificadores. A simplificação dos problemas pode resultar, portanto, em modelos mais rápidos e simples, com baixa perda de acurácia.