Lista 09 - Fundamentos Estatísticos para Ciência dos Dados

Code

Nome: Vinícius de Oliveira Silva

Matrícula: 2013007820

Questão 2

a) Utilizando o código abaixo, podemos variar o valor da variável k para avaliar o desempenho do algoritmo utilizando diferentes quantidades de autovetores:

Hide

```
#Treinamento -> objetivo: encontrar os autovetores da matriz centralizada e os coefic
ientes médios para cada grupo de dígitos.
#-------------------------------------------------------------------------
# leitura dos dados e remoção do label de classificação.
trainingData <- read.table("zip.train")
unclassifiedTraining <- t(trainingData[,-1])
#calculo da média dos dígitos
avg_digit <- apply(unclassifiedTraining, MARGIN = 1, FUN=mean)
#centralizando a matriz em torno do digito médio
centered_training_matrix <- unclassifiedTraining - avg_digit
#calculo do PCA propriamente dito
pca <- prcomp(t(centered_training_matrix))
#salvamos os autovalores e os autovetores
eigenvalues <- (pca$sdev)^2
eigenvectors <- (pca$rot)
#escolha do numero de componentes principais - Variamos este valor para avaliarmos o
 modelo
k<-15
usedPCAs <- eigenvectors[,1:k]
#encontramos os coeficientes W<i,j> de todas as imagens disponíveis no treinamento
coefTraining <- t(usedPCAs) %*% centered_training_matrix
#devemos agora montar uma matriz contendo os coeficientes médios para cada grupo.
#A matriz deve conter 10 colunas e k linhas, uma coluna para cada dígito e uma linha
 para cada coeficiente que multiplicará um autovetor utilizado.
#Primeiro devemos pesquisar na tabela os índices de todas as imagens de cada dígito p
ara podermos computar a média de cada grupo:
trainingClassification <- trainingData[,1]
trainingClassificationIndexes <- list()
for(i in 0:9){
  trainingClassificationIndexes[[i+1]] <- which(trainingClassification==i)
}
#Computando a média, utilizando os índices que acabamos de encontrar
coefAvg <- matrix(ncol=10, nrow = k)
for (i in 0:9){
  coefAvg [, i+1]<- apply( coefTraining[, trainingClassificationIndexes[[i+1]] ], MAR
GIN=1, FUN=mean)
}
#                           Treinamento concluído
#-------------------------------------------------------------------------
#Teste -> objetivo: decompor cada imagem de teste com os autovetores encontrados, com
putar o conjunto de coeficientes e verificar qual dentre os conjuntos de coeficientes
 médios conhecidos mais se aproxima dele.
#-------------------------------------------------------------------------
#leitura dos dados e remoção dos labels de classificação
testData <- read.table("zip.test")
unclassifiedTest <- t(testData[,-1])
#centralização da matriz de teste
centered_test_matrix <- unclassifiedTest - avg_digit
#calculo dos coeficientes
coefTest <- t(usedPCAs) %*% centered_test_matrix
#extraímos os labels para verificarmos se acertamos ou não as nossas predições
testClassification <- testData[,1]
#Devemos agora procurar a coluna do vetor de coeficientes médios que mais se aproxima
m do coeficientes computados para esta amostra de teste. -> esta é a nossa predição
indproximo = numeric()
for(j in 1:ncol(coefTest)){
  indproximo[j] = which.min( apply((coefAvg - coefTest[,j])^2, 2, mean) ) - 1
```

```
}
#verificamos quantos acertos obtivemos dentre o nosso conjunto de teste
counterRight <- 0
for(i in 1:ncol(coefTest)){
  if(indproximo[i] == testClassification[i]){
    counterRight <- counterRight+1
  }
}
sprintf("Predição correta em %s%% dos casos.", format(round((counterRight/ncol(coefTe
st))*100, digits=2), nsmall = 2))
```

```
[1] "Predição correta em 79.22% dos casos."
```

b)

O código que gera a matriz de confusão é o seguinte:

Hide

```
confusion_matrix <- matrix(ncol = 10, nrow = 10)
for(i in 0:9){
  predictedI <- indproximo==i
  for(j in 0:9){
    confusion_matrix[i+1,j+1] <- sum(predictedI & (testClassification==j))
  }
}
prmatrix(confusion_matrix, rowlab=as.character(c(0:9)), collab=as.character(c(0:9)))
```

Variando o parâmetro K, temos as seguintes matrizes:

K=05:

	0	1	2	3	4	5	6	7	8	9
0	271	0	12	7	2	24	23	0	5	0
1	0	259	6	0	10	1	2	2	4	13
2	8	0	106	6	14	11	3	2	17	2
3	2	0	13	104	0	21	0	0	9	0
4	5	1	18	1	118	8	4	10	4	20
5	8	1	9	19	1	76	1	1	10	0
6	55	1	10	1	7	4	136	1	2	0
7	0	0	0	0	19	0	0	107	1	34
8	10	2	21	27	3	12	1	3	100	14
9	0	0	3	1	26	3	0	21	14	94

K=10:

	0	1	2	3	4	5	6	7	8	9
0	283	0	7	4	1	10	20	0	3	0
1	0	258	2	0	7	1	0	2	2	9
2	3	0	135	3	7	0	5	3	7	1
3	2	0	12	117	0	16	0	0	9	0
4	4	3	19	1	141	5	4	7	5	19
5	7	0	2	17	1	110	4	0	7	1
6	53	3	3	2	4	3	136	0	3	0
7	1	0	3	0	2	0	0	110	1	10
8	5	0	15	19	3	9	1	2	118	3
9	1	0	0	3	34	6	0	23	11	134

K=15:

	0	1	2	3	4	5	6	7	8	9
0	295	0	7	6	1	10	15	0	4	0
1	0	259	0	0	7	1	0	2	2	7
2	2	0	138	5	5	0	4	4	5	0
3	2	0	10	125	0	11	0	0	10	0
4	4	3	19	0	142	6	4	7	5	18
5	3	0	2	18	1	116	4	0	6	1
6	40	2	3	1	5	2	142	0	0	0
7	1	0	3	0	2	0	0	114	1	11
8	11	0	16	9	3	7	1	1	123	4
9	1	0	0	2	34	7	0	19	10	136

K=20:

	0	1	2	3	4	5	6	7	8	9
0	296	0	6	6	1	10	14	0	4	0
1	0	259	0	0	6	1	0	2	2	5
2	2	0	142	4	5	0	4	2	5	0
3	2	1	8	130	0	10	0	0	10	0
4	4	2	18	1	141	6	4	7	6	16
5	4	0	3	14	1	119	4	0	6	1
6	38	2	2	1	5	0	143	0	0	0
7	1	0	3	0	2	0	0	114	1	12
8	11	0	16	8	3	7	1	2	124	4
9	1	0	0	2	36	7	0	20	8	139

c)

Código para calcular a proporção:

Hide

```
diagonalSum <- sum(diag(confusion_matrix))
totalElements <- sum(confusion_matrix)
sprintf("A porcentagem de elementos na diagonal é: %s%% para k=%d", format(round((dia
gonalSum/totalElements)*100, digits=2), nsmall = 2), k)
```

A porcentagem de elementos na diagonal é: 68.31% para k=05;

A porcentagem de elementos na diagonal é: 76.83% para k=10;

A porcentagem de elementos na diagonal é: 79.22% para k=15;

A porcentagem de elementos na diagonal é: 80.07% para k=20;

Como esperado, a proporção é máxima para k=20.

d)

Código para computar a precisão:

Hide

```
rowSum <- apply(confusion_matrix, MARGIN = 1, FUN = sum)
precision <- round(diag(confusion_matrix)/rowSum, 2)
print(precision)
```

Código para computar o recall:

Hide

```
columnSum <- apply(confusion_matrix, MARGIN = 2, FUN=sum)
recall <- round(diag(confusion_matrix)/columnSum, 2)
print(recall)
```

Executando esses códigos variando o K, podemos montar a seguinte tabela:

K	Precision	Recall
5	0.79 0.87 0.63 0.70 0.62 0.60 0.63 0.66 0.52 0.58	0.75 0.98 0.54 0.63 0.59 0.48 0.80 0.73 0.60 0.53
10	0.86 0.92 0.82 0.75 0.68 0.74 0.66 0.87 0.67 0.63	0.79 0.98 0.68 0.70 0.70 0.69 0.80 0.75 0.71 0.76
15	0.87 0.93 0.85 0.79 0.68 0.77 0.73 0.86 0.70 0.65	0.82 0.98 0.70 0.75 0.71 0.72 0.84 0.78 0.74 0.77
20	0.88 0.94 0.87 0.81 0.69 0.78 0.75 0.86 0.70 0.65	0.82 0.98 0.72 0.78 0.70 0.74 0.84 0.78 0.75 0.79