Exercício 2

Flávio Barros - RA016120

21 de março de 2013

1 Leitura dos Dados

Os dados do problema são imagens do tipo PGM com 64 x 64 pixels por imagem, onde cada pixel tem valor 1 ou 0. Cada imagem tem um nome no formato X_ yyy.BMP.inv.pgm, onde o X é o dígito desenhado na imagem.

Assim a primeira parte do Exercício 2 é efetuar a leitura dos dados. Para isso me utilizarei do pacote `pixmap` com o qual é possível ler e manipular imagens PGM.

```
library(pixmap)
```

Após isso será feita a leitura dos arquivos no conjunto de treino e do conjunto de teste. Esta consiste na criação de dois `data.frames` chamados treino e teste, e também da criação de um vetor do tipo `factor` para armazenar as classes, obtidas dos nomes dos arquivos de cada imagem.

Inicialmente é definido o local onde estão os arquivos.

```
path_treino <- "/home/ra016120/Dropbox/MO444/Exercicio2/treino/"
setwd(path_treino)
```

Em seguinda os nomes dos arquivos são armazenados na variável `files` e as classes são extraídas dos nomes dos arquivos.

```
files <- dir()
classes <- as.factor(substring(files, first = 1, last = 1))
```

Por fim é realizada a leitura dos arquivos para o `data.frame` treino. É importante observar que a matriz 64 x 64, dos dados da imagem, foi armazenada em `x@grey` e depois transformada em vetor, tal que na matriz final cada linha é a representação de uma das imagens. Como são 1949 imagens, a matriz tem 1949 linhas.

```
treino <- as.data.frame(matrix(rep(0, length(files) * 64 * 64), nrow = length(files)))
for (i in 1:length(files)) {
    x <- read.pnm(files[i])
    treino[i, ] <- as.vector(x@grey, mode = "integer")
}
```

O mesmo procedimento foi realizado para os dados de teste. Somente vale ressaltar que as classes foram armazenadas na variável `predic`.

```
path_teste <- "/home/ra016120/Dropbox/MO444/Exercicio2/teste/"
setwd(path_teste)
files <- dir()
predic <- as.factor(substring(files, first = 1, last = 1))
teste <- as.data.frame(matrix(rep(0, length(files) * 64 * 64), nrow = length(files)))
for (i in 1:length(files)) {
    x <- read.pnm(files[i], )
    teste[i, ] <- as.vector(x@grey, mode = "integer")
}
```

2 Aprendizado com k-nn

Nesta etapa será realizado o aprendizado com o algoritmo k-nn sem nenhum tratamento dos dados.

```
predito <- knn(train = treino, test = teste, cl = classes, k = 1, prob = T)
result <- data.frame(cbind(predic, predito, acerto = predic == predito))
sum(result$acerto)/nrow(result)
```

Como pode-se verificar, a taxa de acerto com `k = 1` foi de 78%. Para comparação foi realizado o procedimento do k-nn para todos os números ímpares de 1 a 81 e como pode-ser ver na Figura (1), o melhor `k` foi `k=1`. Vale ressaltar também, que dos valores de k entre $1, 3, 5, 11, 17, 21$ a menor taxa de acerto está destacada no gráfico e foi de 28%.

Assim, pelo menos nos dados brutos sem nenhum tipo de tratamento a melhor opção é utilizar o `k = 1`.

3 Aplicação do PCA

Para o procedimento de aplicação do PCA utilizei a função `prcomp` que utiliza um procedimento de cálculo das componentes principais baseado na decomposição SVD.

Como temos que inferir as componentes principais do PCA a partir do conjunto de treino, mas temos que utilizar essas mesmas componentes principais no conjunto de teste, temos que fazer a rotação do conjunto de dados no novo sistema de coordenas, utilizando os resultados da saída de `prcomp`.

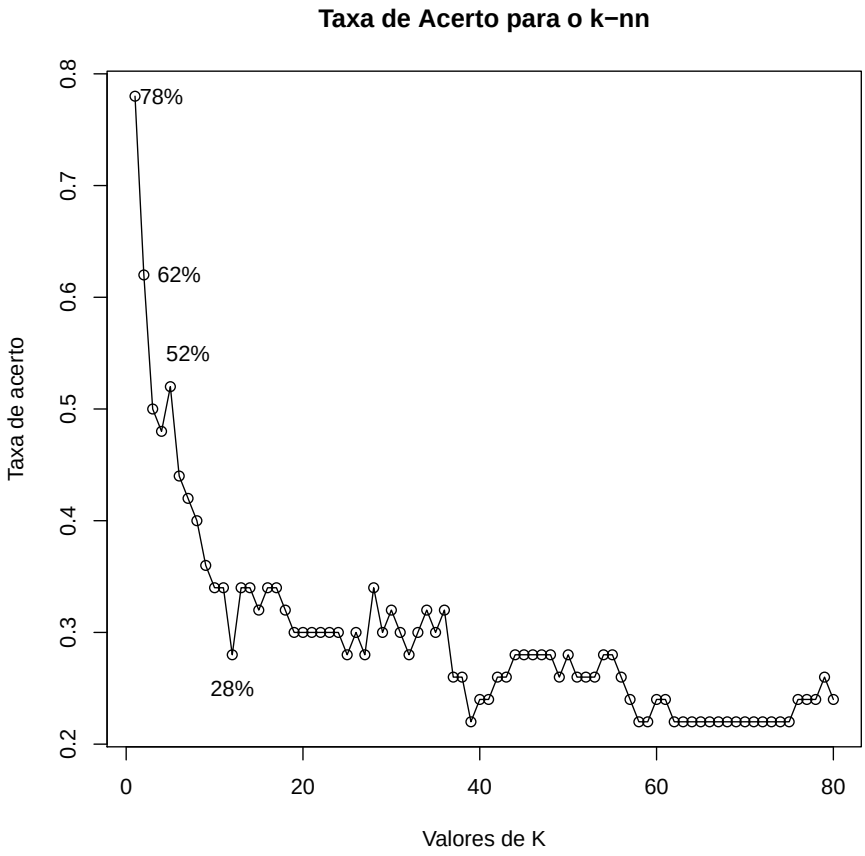

Figura 1: Variação da taxa de acerto com a variação dos k's

3.1 Explicação da Rotação

Para ilustrar o procedimento de rotação do conjunto de teste, vamos visualizar o efeito das componente principais em um conjunto de testes menor. O conjunto escolhido foi retirado do livro "A User's Guide to Principal Components", específicamente da página 5. Foram feitas 15 observações, com dois métodos diferentes

```
chemic <- data.frame(m1 = c(10, 10.4, 9.7, 9.7, 11.7, 11, 8.7, 9.5, 10.1, 9.6,
    10.5, 9.2, 11.3, 10.1, 8.5), m2 = c(10.7, 9.8, 10, 10.1, 11.5, 10.8, 8.8,
    9.3, 9.4, 9.6, 10.4, 9, 11.6, 9.8, 9.2))
chemic

##      m1   m2
```

```
## 1  10.0 10.7
## 2  10.4  9.8
## 3   9.7 10.0
## 4   9.7 10.1
## 5  11.7 11.5
## 6  11.0 10.8
## 7   8.7  8.8
## 8   9.5  9.3
## 9  10.1  9.4
## 10  9.6  9.6
## 11 10.5 10.4
## 12  9.2  9.0
## 13 11.3 11.6
## 14 10.1  9.8
## 15  8.5  9.2
```

Agora vamos calcular as componentes principais e armazenar a matriz de rotação:

```
pc <- prcomp(chemic)
rotacao <- pc$rotation
rotacao

##        PC1     PC2
## m1 -0.7236 -0.6902
## m2 -0.6902  0.7236
```

Vamos visualizar os dados rotacionados:

```
pc$x

##             PC1      PC2
##  [1,] -0.48314  0.50654
##  [2,] -0.15141 -0.42080
##  [3,]  0.21709  0.20706
##  [4,]  0.14807  0.27942
##  [5,] -2.26545 -0.08789
##  [6,] -1.27578 -0.11129
##  [7,]  1.76894  0.02890
##  [8,]  0.84495 -0.16144
##  [9,]  0.34175 -0.50319
## [10,]  0.56553 -0.01337
## [11,] -0.63789 -0.05565
## [12,]  1.26909 -0.17147
## [13,] -2.04502  0.26055
## [14,]  0.06568 -0.21374
## [15,]  1.63759  0.45639
```

Cada componente z_i, que em PCA são os scores, ou as observações no sistema rotacionado, podem ser calculadas diretamente utilizando a matriz de rotação, o vetor das médias e as observações. Assim para uma dada linha da tabela, seu score seria calculado por ':

$$z_i = \mathbf{u}_i^T[\mathbf{x} - \bar{\mathbf{x}}] \tag{1}$$

Assim, para a primeira observação:

```
L <- as.matrix(pc$center)
rotacionado <- matrix(rep(0, 2 * 15), nrow = 15)
for (i in 1:nrow(rotacionado)) {
    rotacionado[i, ] <- t(rotacao %*% (t(chemic[i, ]) - L))
}
as.matrix(rotacionado)
```

```
##           [,1]     [,2]
##  [1,] -0.48314  0.50654
##  [2,] -0.15141 -0.42080
##  [3,]  0.21709  0.20706
##  [4,]  0.14807  0.27942
##  [5,] -2.26545 -0.08789
##  [6,] -1.27578 -0.11129
##  [7,]  1.76894  0.02890
##  [8,]  0.84495 -0.16144
##  [9,]  0.34175 -0.50319
## [10,]  0.56553 -0.01337
## [11,] -0.63789 -0.05565
## [12,]  1.26909 -0.17147
## [13,] -2.04502  0.26055
## [14,]  0.06568 -0.21374
## [15,]  1.63759  0.45639
```

```
pc$x
```

```
##            PC1      PC2
##  [1,] -0.48314  0.50654
##  [2,] -0.15141 -0.42080
##  [3,]  0.21709  0.20706
##  [4,]  0.14807  0.27942
##  [5,] -2.26545 -0.08789
##  [6,] -1.27578 -0.11129
##  [7,]  1.76894  0.02890
##  [8,]  0.84495 -0.16144
##  [9,]  0.34175 -0.50319
## [10,]  0.56553 -0.01337
## [11,] -0.63789 -0.05565
```

```
## [12,]  1.26909 -0.17147
## [13,] -2.04502  0.26055
## [14,]  0.06568 -0.21374
## [15,]  1.63759  0.45639
```

Como pode-se ver é o mesmo resultado utilizando o sistema rotacionado pelo `prcomp`

3.2 Rotação nas Imagens

Seguindo o procedimento descrito anteriormente, ou utilizando o método `predict` do objeto `prcomp` é possível rotacionar o conjunto de dados. Tanto usando o loop, ou utilizando o método `predict`.

```
pc <- prcomp(treino)
treino_x <- pc$x
test.p <- predict(pc, newdata = teste)
predito <- knn(train = pc$x[, 1:400], test = test.p[, 1:400], cl = classes,
    k = 1, prob = T)
result <- data.frame(cbind(predic, predito, acerto = predic == predito))
sum(result$acerto)/nrow(result)
```

Lembrando que as observações rotacionadas podem ser calculadas linha a linha com o procedimento descrito anteriormente. Com as 400 componentes principais a taxa de acerto, com `k=1`, foi de 82%, mostrando uma melhora. De forma a explorar a variação de componentes, foi construído o gráfico da Figura (2). A melhor taxa de acerto foi alcançada com 70 dimensões.

Fazendo a variação dos k's com 70 dimensões para vários valores de k, vemos que, apesar da melhora nas taxas para k = {1,3,5,11,17,21} que passaram de {0.78, 0.50, 0.53, 0.34, 0.34, 0.30} para {0.84, 0.58, 0.52, 0.50, 0.62, 0.58} o `k=1` ainda é a melhor opção.

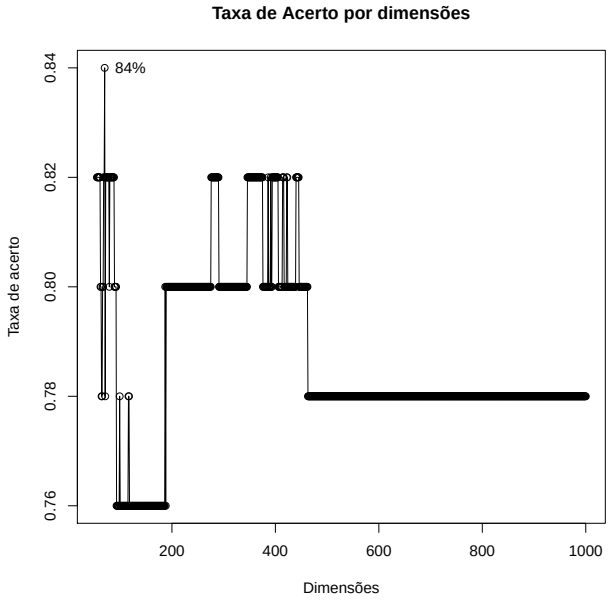

Figura 2: Variação da taxa de acerto com o número de dimensões.

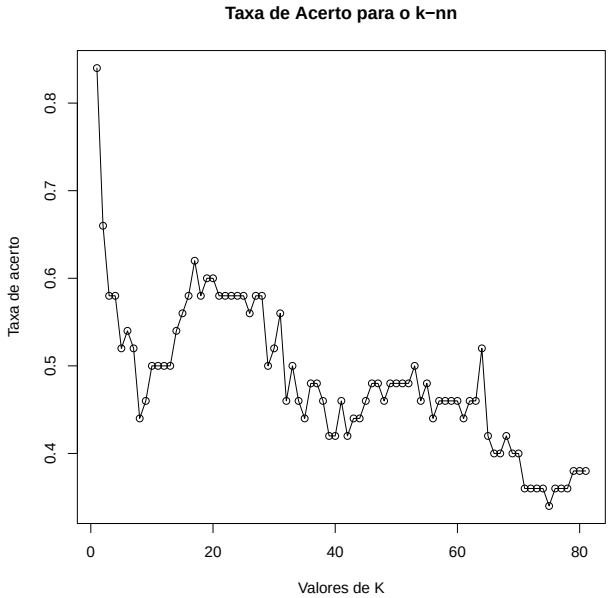

Figura 3: Variação da taxa de acerto com o número de dimensões.