HW2_P3

Qiuying Li UNI ql2280

2/15/2017

1. LDA on the original 256 dimensional space.

```
setwd("~/Desktop/2017 spring/GR 5241/HW/hw2")
library(MASS)
train_3 <- read.table("train_3.txt",sep=",",head = F)
train_5 <- read.table("train_5.txt",sep=",",head = F)
train_8 <- read.table("train_8.txt",sep=",", head = F)
#making the matrix of of training data
training_x = rbind(as.matrix(train_3),as.matrix(train_5),as.matrix(train_8))
training_y = rep(c(3,5,8),c(nrow(train_3),nrow(train_5),nrow(train_8)))
test_data = as.matrix(read.table("zip_test.txt",head = F))
#making the test data, y is the first column of the test data
test_y = test_data[,1]
test_x = test_data[(test_y==3|test_y==5|test_y==8), -1]
test_y = test_y[ test_y==3 | test_y==5 | test_y==8]

pred_errors= matrix(0,nrow = 4, ncol = 2)
library(MASS)
m1 = lda(training_x,training_y)
pred_errors[1,1] = sum(predict(m1)$class!=training_y)/length(training_y) ;pred_errors[1,1]
```

```
## [1] 0.01594533
```

```
pred_errors[1,2] = sum( predict(m1,test_x)$class != test_y) / length(test_y); pred_errors[1,2]
```

```
## [1] 0.08739837
```

(b) LDA on the leading 49 principal components

```
train_x_center = apply(training_x,2,mean)
train_xx = scale ( training_x , center = train_x_center  , scale=F)
test_xx = scale ( test_x , center = train_x_center, scale=F)
pc = svd(train_xx)$v[ ,1:49]
pctrain = train_xx %*% pc
pctest = test_xx%*% pc
m2=lda (pctrain , training_y)
pred_errors [2 ,1] = sum(predict(m2)$class !=training_y)/length(training_y) ; pred_errors [2 ,1]
```

```
## [1] 0.04384966
```

```
pred_errors[2,2] = sum(predict(m2,pctest)$class!=test_y)/length(test_y);pred_errors[2,2]
```

```
## [1] 0.08536585
```

3. LDA when you filter the data as follows. Each non-overlapping 2×2 pixel block is replaced by its average.

```
filterdata = function(x){
x = matrix(x,16,16)
filter = rep(1:2,8)
x = x[filter==1,]+x[filter==2,]
x = x[,filter==1]+x [,filter==2]
```

```
as.vector(x)/4
}
xtrain = t(apply(training_x,1,filterdata))
xtest = t(apply(test_x,1,filterdata))
m3 = lda(xtrain,training_y)
pred_errors[3,1] = sum(predict(m3)$class!=training_y)/length(training_y);pred_errors[3,1]
```

```
## [1] 0.03359909
```

```
pred_errors[3,2] = sum(predict(m3,xtest)$class!=test_y)/length(test_y);pred_errors[3,2]
```

```
## [1] 0.07520325
```

4. Multiple linear logistic regression using the same filtered data as in the previous question. [Use the multinomial family in the R package glmnet; use the solution at the end of the path].

```
library (glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
library(nnet)
a1 = kronecker(diag(8), cbind(c(.5, .5)))
a2 <- kronecker(a1, a1)
dim(a2)
```

```
## [1] 256  64
```

```
dim(training_x)
```

```
## [1] 1756  256
```

```
training2 <- data.frame(training_y,(training_x %*% a2) )
test2 <- data.frame(test_y, test_x %*% a2 )
m_logist <- multinom(training_y ~ ., training2)
```

```
## # weights:  198 (130 variable)
## initial  value 1929.163179
## iter  10 value 245.164450
## iter  20 value 77.741202
## iter  30 value 50.356671
## iter  40 value 41.704361
## iter  50 value 37.611158
## iter  60 value 26.884207
## iter  70 value 11.438433
## iter  80 value 0.519605
## iter  90 value 0.011950
## final  value 0.000063
## converged
```

```
p_train <- predict(m_logist , training2)
p_test <- predict(m_logist , test2)
pred_errors[4,1] = sum(p_train!=training_y)/length(training_y)
pred_errors[4,2] = sum(p_test!=test_y)/length(test_y)
print(pred_errors)
```

```
##            [,1]       [,2]
```

```
## [1,] 0.01594533 0.08739837
## [2,] 0.04384966 0.08536585
## [3,] 0.03359909 0.07520325
## [4,] 0.00000000 0.09552846
```