pset-5-Katia-Williams

Katia Williams

4/13/2018

Note: Got some help and did some collaboration with Alaa Elshahawi

1) LDA

1.1) lda_fit()

```
#Create LDA fit
lda_fit <- function(Xdf, ydf) {
  X <- as.matrix(Xdf)
  y <- as.matrix(ydf)
  n <- nrow(X)
  p <- ncol(X)
  groups <- unique(y)
  K <- length(groups)

  nkvec <- c()
  mukmat <- matrix(rep(0,p), nrow=1, ncol=p)
  sigmamat <- matrix(rep(0,p**2), nrow=p, ncol=p)


  for (i in c(1:K)) {
    k <- groups[i]
    nk <- length(which(y == k))
    nkvec <- c(nkvec, nk)
    xk <- X[c(which(y==k)),]

    muk <- apply(xk, MARGIN=2, FUN=sum)/nk
    mukmat <- rbind(mukmat, muk)

    sigterm <- sweep(xk, MARGIN=2, STATS=muk, FUN="-")
    sigterm2 <- t(sigterm) %*% (sigterm)
    sigmamat <- sigmamat + sigterm2
  }
  pi_hat <- nkvec/n
  muhat <- mukmat[1:i+1,]

  rownames(muhat) <- groups

  return(list(pi_hat, muhat, sigmamat/(n-K)))
}
```

1.3.1)

```r
#Testing LDA fit
trainingX <- iris[c(1:47, 51:97, 101:146), 1:4]
trainingY <- iris[c(1:47, 51:97, 101:146), 5]

fit <- lda_fit(as.matrix(trainingX), as.matrix(trainingY))
fit
```

```
## [[1]]
## [1] 0.3357143 0.3357143 0.3285714
##
## [[2]]
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         5.008511    3.429787     1.463830   0.2489362
## versicolor     5.953191    2.772340     4.289362   1.3319149
## virginica      6.619565    2.973913     5.584783   2.0282609
##
## [[3]]
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.27084678  0.09672053   0.16682306  0.03908908
## Sepal.Width    0.09672053  0.11913165   0.05524487  0.03340797
## Petal.Length   0.16682306  0.05524487   0.18140540  0.04254695
## Petal.Width    0.03908908  0.03340797   0.04254695  0.04345135
```

1.2) lda_predict()

```r
#Create lda_predict
lda_predict <- function(fit, newdata){
  pihat <- fit[[1]]
  muhat <- fit[[2]]
  sigmahat <- fit[[3]]
  groups <- rownames(muhat)


  K <- length(pihat)
  m <- nrow(newdata)

  preds <- c()
  deltas <- c()

  for (rowind in c(1:m)) {
    obs <- unlist(newdata[rowind,])
    obs_pred <- c()
    for (k in c(1:K)){
      muhatk <- muhat[k,]
      obs_pred <- c(obs_pred, log(pihat[k])
                    - .5*(t(muhatk) %*% solve(sigmahat) %*% muhatk)
                    +t(muhatk) %*% solve(sigmahat) %*% obs)
    }

    preds <- c(preds, groups[match(max(obs_pred), obs_pred)])
    deltas <- c(deltas, obs_pred)
  }
```

```
  deltamat <- matrix(deltas, ncol=K, byrow = TRUE)
  num <- exp(deltamat)
  denom <- rowSums(num)
  posts <- sweep(num, 1, STATS=denom, FUN="/")
  colnames(posts) <- groups

  return(list(preds, posts))
  }
```

1.3.2)

```
#Test lda_train
testing <- iris[c(48:50, 98:100, 147:150),][1:4]

lda_predict(fit, testing)
```

```
## [[1]]
##  [1] "setosa"     "setosa"     "setosa"     "versicolor" "versicolor"
##  [6] "versicolor" "virginica"  "virginica"  "virginica"  "virginica"
##
## [[2]]
##              setosa   versicolor    virginica
##  [1,] 1.000000e+00 1.902934e-18 2.066054e-37
##  [2,] 1.000000e+00 2.710046e-23 1.385488e-43
##  [3,] 1.000000e+00 1.655456e-20 3.601589e-40
##  [4,] 2.098963e-18 9.999561e-01 4.392786e-05
##  [5,] 2.560763e-10 1.000000e+00 1.273872e-08
##  [6,] 7.377643e-19 9.999373e-01 6.272708e-05
##  [7,] 2.150458e-35 9.801042e-03 9.901990e-01
##  [8,] 1.630159e-34 4.840281e-03 9.951597e-01
##  [9,] 5.482963e-40 2.258174e-05 9.999774e-01
## [10,] 5.273542e-33 2.312620e-02 9.768738e-01
```

2) QDA

```
#Create qda_fit()

qda_fit <- function(Xdf,ydf) {
  X <- as.matrix(Xdf)
  y <- as.matrix(ydf)
  n <- nrow(X)
  p <- ncol(X)
  groups <- unique(y)
  K <- length(groups)

  nkvec <- c()
  mukmat <- matrix(0, nrow=1, ncol=p)


  sigmamat <- array(0, dim=c(p,p,K))
```

```
  for (i in c(1:K)) {
    k <- groups[i]
    nk <- length(which(y == k))
    nkvec <- c(nkvec, nk)
    xk <- X[c(which(y==k)),]

    muk <- apply(xk, MARGIN=2, FUN=sum)/nk
    mukmat <- rbind(mukmat, muk)

    sigterm <- sweep(xk, MARGIN=2, STATS=muk, FUN="-")
    sigterm2 <- t(sigterm) %*% (sigterm)
    sigmamat[,,i] <- sigterm2/(nk-1)
  }
  pi_hat <- nkvec/n
  muhat <- mukmat[1:i+1,]

  rownames(muhat) <- groups

  rownames(sigmamat) <- colnames(muhat)
  colnames(sigmamat) <- colnames(muhat)

  return(list(pi_hat, muhat, sigmamat))
}
```

2.3.1)

```
#test qda_fit()

trainingX <- iris[c(1:47, 51:97, 101:146),][1:4]
trainingY <- iris[c(1:47, 51:97, 101:146),][5]

fitqda <- qda_fit(as.matrix(trainingX), as.matrix(trainingY))
fitqda
```

```
## [[1]]
## [1] 0.3357143 0.3357143 0.3285714
##
## [[2]]
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         5.008511    3.429787     1.463830   0.2489362
## versicolor     5.953191    2.772340     4.289362   1.3319149
## virginica      6.619565    2.973913     5.584783   2.0282609
##
## [[3]]
## , , 1
##
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.12688252 0.101914894  0.016618871 0.010878816
## Sepal.Width    0.10191489 0.149962997  0.011753006 0.009814986
## Petal.Length   0.01661887 0.011753006  0.031924144 0.006373728
## Petal.Width    0.01087882 0.009814986  0.006373728 0.011683626
##
```

```
## , , 2
##
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.26558742  0.08519889   0.17036078  0.05522202
## Sepal.Width    0.08519889  0.10291397   0.08056892  0.04264107
## Petal.Length   0.17036078  0.08056892   0.19923219  0.07143386
## Petal.Width    0.05522202  0.04264107   0.07143386  0.04048104
##
## , , 3
##
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.42338647  0.10318841   0.31674879  0.05143478
## Sepal.Width    0.10318841  0.10419324   0.07381643  0.04808696
## Petal.Length   0.31674879  0.07381643   0.31598551  0.04999517
## Petal.Width    0.05143478  0.04808696   0.04999517  0.07896135
```

2.2) Function predict_qda()

```
predict_qda <- function(fit, newdata) {
  pihat <- fit[[1]]
  muhat <- fit[[2]]
  sigmahat <- fit[[3]]
  groups <- rownames(muhat)


  K <- length(pihat)
  m <- nrow(newdata)

  preds <- c()
  deltas <- c()

  for (rowind in c(1:m)) {
    obs <- unlist(newdata[rowind,])
    obs_pred <- c()
    for (k in c(1:K)){
      muhatk <- muhat[k,]
      sigmahatk <- sigmahat[,,k]
      obs_pred <- c(obs_pred, log(pihat[k])
                    - .5*(t(muhatk) %*% solve(sigmahatk) %*% muhatk)
                    +t(muhatk) %*% solve(sigmahatk) %*% obs)
    }

    preds <- c(preds, groups[match(max(obs_pred), obs_pred)])
    deltas <- c(deltas, obs_pred)
  }

  deltamat <- matrix(deltas, ncol=K, byrow = TRUE)
  num <- exp(deltamat)
  denom <- rowSums(num)
  posts <- sweep(num, 1, STATS=denom, FUN="/")
  colnames(posts) <- groups

  return(list(preds, posts))
```

```
  }
```

2.3.2)

```
testingX <- iris[c(48:50, 98:100, 147:150),][1:4]
predict_qda(fitqda, testingX)
```

```
## [[1]]
##  [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
##  [8] "setosa" "setosa" "setosa"
##
## [[2]]
##       setosa   versicolor    virginica
##  [1,]      1 1.588208e-19 1.678035e-38
##  [2,]      1 4.581566e-24 1.284568e-46
##  [3,]      1 2.100140e-23 3.254825e-44
##  [4,]      1 7.969363e-66 1.125114e-79
##  [5,]      1 1.455095e-44 1.155252e-54
##  [6,]      1 8.655949e-59 3.600870e-70
##  [7,]      1 1.846387e-75 4.598869e-82
##  [8,]      1 1.702569e-74 3.035675e-81
##  [9,]      1 1.172907e-67 1.345525e-69
## [10,]      1 5.329863e-67 1.049842e-73
```

3) k-Nearest Neighbords

3.1) Function knn_fit()

```
#Create knn_fit()

knn_predict <- function(Xtrain, Xtest, ytrain, k) {
  n <- nrow(Xtrain)
  p <- ncol(Xtrain)
  groups <- unique(ytrain)
  K <- length(groups)


  predictions <- c()

  for(i in c(1:nrow(Xtest))){
    obs <- as.matrix(Xtest)[i,]
    dists <- apply(as.matrix(Xtrain), 1, function(vec) as.numeric(vec-obs))
    dists <- t(dists)
    dists <- dists**2
    dists <- apply(dists, 1, sum)
    labeled_dists <- data.frame("D" = dists, "class" = ytrain)
    labeled_dists <- labeled_dists[order(labeled_dists$D),]
    k_nearest <- labeled_dists[1:k, 2]

    conditionals <- c()
    for (j in c(1:K)){
```

```
      conditionals <- c(conditionals, sum(k_nearest == groups[j]))
    }
    conditionals <- conditionals/k
    predictions <- c(predictions, strtoi(as.character(match(max(conditionals), conditionals))))
  }
  return(groups[predictions])
}
```

3.2) Classification with k-NN

```
training <- c(1:47, 51:97, 101:146)
testing <- c(48:50, 98:100, 147:150)
train_set <- iris[training,]
test_set <- iris[testing,]

pred_knn <- knn_predict(train_set[,-5], test_set[, -5], train_set$Species, k=1)
pred_knn
```

```
##  [1] setosa     setosa     setosa     versicolor versicolor versicolor
##  [7] virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

3.3) k-NN CV

```
find_kcv <- function(Xtrain, ytrain, k=c(1:10), nfold) {
  df <- cbind.data.frame(Xtrain, ytrain)
  df <- df[sample(nrow(df)),]
  p <- ncol(df)
  X <- df[,-p]
  y <- df[,p]
  folds <- cut(seq(1,nrow(X)), breaks=nfold,labels=FALSE)
  accuracy <- c()

  for (h in k) {
    testIndexes <- which(folds==h,arr.ind=TRUE)
    testData <- X[testIndexes, ]
    trainData <- X[-testIndexes, ]
    trainClasses <- y[-testIndexes]
    testClasses <- y[testIndexes]

    pred <- knn_predict(trainData, testData,trainClasses, h)
    accuracy <- c(accuracy, sum(pred == testClasses))
  }
  goodk <- which(accuracy == max(accuracy))
  if (length(goodk) >1) {
    goodk <- sample(goodk, 1)
  }
  return(goodk)
}
```

```
find_kcv(train_set[, -5], train_set[, 5], nfold=10)
```

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

4) Confusion Matrix

```
set.seed(100)
train_idx <- sample(nrow(iris), 90)
train_set <- iris[train_idx,]
test_set <- iris[-train_idx,]
```

```
#TrainLDA, Generate Predictions
fitlda <- lda_fit(train_set[, -5], train_set[, 5])
ldapreds <- lda_predict(fitlda, test_set[, -5])

#TrainQDA< Generate Predictions
fitqda <- qda_fit(train_set[, -5], train_set[, 5])
qdapreds <- predict_qda(fitqda, test_set[, -5])

#Find k, Predict using KNN
bestk <- find_kcv(train_set[, -5], train_set[, 5], nfold=10)
knnpreds <- knn_predict(train_set[, -5], test_set[, -5], train_set[, 5], bestk)
```

```
#Put all predictions, and real values, into one table
Predictions <- data.frame("LDA" = ldapreds[[1]], 'QDA' = qdapreds[[1]], 'KNN'=knnpreds,
                          'TRUEy' = test_set[, 5])
```

```
#LDA confusion matrix
LDApred <- Predictions[, c(1,4)]
LDAconfusion <- table(LDApred)
LDAconfusion
```

```
##             TRUEy
## LDA          setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
```

```
#QDA confusion matrix
QDApred <- Predictions[, c(2,4)]
QDAconfusion <- table(QDApred)

QDAconfusion <- rbind(QDAconfusion, rep(0,3), rep(0,3))
rownames(QDAconfusion) <- c('setosa', 'versicolor', 'virginica')
QDAconfusion
```

```
##            setosa versicolor virginica
## setosa         24         17        19
## versicolor      0          0         0
## virginica       0          0         0
```

```
#KNN Confusion Matrix
KNNpred <- Predictions[, c(3,4)]
KNNconfusion <- table(KNNpred)
KNNconfusion
```

```
##             TRUEy
## KNN          setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         16         2
##   virginica       0          1        17
```

```
#LDA test error rate
1 - sum(diag(LDAconfusion))/sum(LDAconfusion)
```

```
## [1] 0.01666667
```

```
#QDA test error rate
1 - sum(diag(QDAconfusion))/sum(QDAconfusion)
```

```
## [1] 0.6
```

```
#KNN test error rate
1 - sum(diag(KNNconfusion))/sum(KNNconfusion)
```

```
## [1] 0.05
```

LDA did the best, KNN the second best (very closely second best), and QDA did the worst.

LDA operates under the assumption that each of the species' multivariate distribution has the same sigma, whereas QDA does not. I suppose that accounting for different possible sigma made it easier to fit the model to noise in the data. Sometimes, it is better to go with a simpler model :)