lab9

shichenh

10/29/2017

```
knitr::opts_chunk$set(message = F)
```

```
library(MASS)
library(mvtnorm)
library(caret)
library(e1071)
```

```
iris <- iris
```

```
X <- as.matrix(dplyr::select(iris[1:140,], -Species))
y <- iris$Species[1:140]
X.test <- as.matrix(dplyr::select(iris[141:150,], -Species))
```

LDA

```
my_lda <- function(X, y) {
  # Arguements:
  #   X: the predictor matrix, which is an n x p matrix
  #   y: ther response vector, a factor vector of length n
  # Return:
  #   pi_hat: the prior probability vector
  #   mu_hat: a K x p matrix in which each row contains the mean of the group
  #   sigma_hat: the p x p covariance matrix of the predictors

  #need to make sure y is a factor vector
  pi_hat <- table(y)/length(y)
  mu_hat <- aggregate(X ~ y, FUN=mean)
  sigma_hat <- cov(X) * (nrow(X) - 1) / (nrow(X) - nrow(mu_hat))
  return (list("pi_hat"=pi_hat, "mu_hat"=as.matrix(mu_hat[,-c(1)]), "sigma_hat"=sigma_hat))
}

predict_my_lda <- function(fit, newdata) {
  # Arguements:
  #   fit: the output from my_lda
  #   newdata: a m x p matrix of new observations
  # Returns:
  #   class: a length-m factor vector; each of its elements indicate the
  #predicted class of an observation
  #   posterior: a m x K matrix of posterior probabilities

  # finding the likelihood for each data point
  f <- vector()
  for (i in 1:nrow(fit$mu_hat)) {
    f <- cbind(f, dmvnorm(newdata, mean=fit$mu_hat[i,], sigma=fit$sigma_hat))
  }
```

```
  # finding p * f(x), taking prior into account
  fpi <- f %*% diag(fit$pi_hat[1:3])
  # scaling the posterior so that the probability add up to 1
  posterior <- t(apply(fpi, 1, function(i) i/sum(i)))
  # making classification based on maximum posterior
  label <- factor(apply(posterior, 1, function(x) names(fit$pi_hat)[which.max(x)]))

  return(list("class"=label, "posterior"=posterior))
}
```

```
lda.out <- lda(Species ~ ., iris[1:140,])
lda.my <- my_lda(X, y)

lda.pred <- predict(lda.out, newdata=data.frame(X.test))
lda.my.pred <- predict_my_lda(lda.my, newdata=X.test)

as.character(lda.pred$class) == as.character(lda.my.pred$class)
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

QDA

```
my_qda <- function(X, y) {
  # Arguements:
  #   X: the predictor matrix, which is an n x p matrix
  #   y: ther response vector, a factor vector of length n
  # Return:
  #   pi_hat: the prior probability vector
  #   mu_hat: a K x p matrix in which each row contains the mean of the group
  #   sigma_hat: the p x p x k covariance matrix of the predictors

  #need to make sure y is a factor vector
  pi_hat <- table(y)/length(y)
  mu_hat <- aggregate(X ~ y, FUN=mean)

  p <- ncol(X)
  k <- length(pi_hat)
  types <- names(pi_hat)

  sigma_hat <- sigma_hat <- array(rep(-1, p * p * k), dim=c(p, p, k))
  for (i in 1:k) {
    sigma_hat[,,i] <- cov(X[y==types[i],])
  }

  return (list("pi_hat"=pi_hat, "mu_hat"=as.matrix(mu_hat[,-c(1)]), "sigma_hat"=sigma_hat))
}

predict_my_qda <- function(fit, newdata) {
  # Arguements:
  #   fit: the output from my_lda
  #   newdata: a m x p matrix of new observations
  # Returns:
```

```
  #   class: a length-m factor vector; each of its elements indicate the
  # predicted class of an observation
  #   posterior: a m x K matrix of posterior probabilities

  # finding the likelihood for each data point
  f <- vector()
  for (i in 1:nrow(fit$mu_hat)) {
    f <- cbind(f, dmvnorm(newdata, mean=fit$mu_hat[i,], sigma=fit$sigma_hat[,,i]))
  }
  # finding p * f(x), taking prior into account
  fpi <- f %*% diag(fit$pi_hat[1:length(fit$pi_hat)])
  # scaling the posterior so that the probability add up to 1
  posterior <- t(apply(fpi, 1, function(i) i/sum(i)))
  # making classification based on maximum posterior
  label <- factor(apply(posterior, 1, function(x) names(fit$pi_hat)[which.max(x)]))

  return(list("class"=label, "posterior"=posterior))
}
```

```
qda.out <- qda(Species ~ ., iris[1:140,])
qda.my <- my_qda(X, y)

qda.pred <- predict(qda.out, newdata=data.frame(X.test))
qda.my.pred <- predict_my_qda(qda.my, newdata=X.test)

as.character(qda.pred$class) == as.character(qda.my.pred$class)
```

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

Confusion Matrix

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

X <- as.matrix(dplyr::select(iris[train_idx,], -Species))
y <- iris$Species[train_idx]
X.test <- as.matrix(dplyr::select(iris[-train_idx,], -Species))
```

```
lda.pred <- predict_my_lda(my_lda(X, y), newdata=data.frame(X.test))
qda.pred <- predict_my_qda(my_qda(X, y), newdata=data.frame(X.test))

table(lda.pred$class, test_set$Species)
```

```
##
##              setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         16         5
##   virginica       0          1        14
```

```
confusionMatrix(lda.pred$class, test_set$Species)$table
```

```
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         16         5
##   virginica       0          1        14
```

```
table(qda.pred$class, test_set$Species)
```

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

```
confusionMatrix(qda.pred$class, test_set$Species)$table
```

```
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
```

Multi-nomial Logistic Regression

```
find_multinom_coef <- function(X, y) {
  # Arguments
  #   X: a n x p matrix of predictors
  #   y: a factor vector of length n with at least 2 categories
  # Return
  #   param: a (p+1) * (K-1) matrix
  k <- length(levels(y))
  p <- ncol(X)
  beta <- rep(0, (p+1)*(k-1))
  dum.raw <- spatstat::dummify(y)
  k0 <- colnames(dum.raw)[1]
  dum <- dum.raw[,2:ncol(dum.raw)]
  model.m <- cbind(1, X)

  cost <- function(para) {
  # takes in para and returns the multinomial logistic regression cost
    beta.m <- matrix(para, ncol=k-1)
    print(beta.m)
    print(dim(model.m))
    inter <- model.m %*% beta.m
    right <- sum(apply(inter, 1, function(x) sum(log(1 + sum(exp(x))))))
    left <- sum(inter*dum)
    -(left - right)
  }
  para <- optim(beta, cost, method="BFGS")
  return(
    matrix(para$par,
           nrow = p+1,
           dimnames = list(
```

```
            c("(Intercept)", colnames(X)),
            colnames(dum)
          ))
  )
}
```

```
param <- find_multinom_coef(as.matrix(iris[1:140,-c(5)]), iris$Species[1:140])
```

```
param
```

```
##              versicolor virginica
## (Intercept)  17.7254642 -24.63122
## Sepal.Length -6.7005419  -9.10777
## Sepal.Width  -6.2433328 -12.86990
## Petal.Length 13.7900511  23.11828
## Petal.Width  -0.5066345  17.59611
```