```
#####################################
##第五回「正則化法と判別分析」
#####################################
#必要パッケージ: glmnet LiblineaR MASS mlbench
library(glmnet)
library(MASS)
library(mlbench)
library(LiblineaR)


#iris三値判別
data(iris) #フィッシャーのアヤメ(iris)データ
# Sepal.Length ガクの長さ（説明変数１）
# Sepal.Width   ガクの幅   （説明変数２）
# Petal.Length  花弁の長さ（説明変数３）
# Petal.Width   花弁の幅   （説明変数４）
# Species       花の種類   （これについて分類する）

# 訓練データとテストデータの作成
train.index <- sample(nrow(iris),nrow(iris)*0.7)
data.train <- iris[train.index,]
data.test <- iris[-train.index,]
indy = 5
dimx = 4

## LDAによる判別
iris.lda <- lda(Species ~., data=data.train, method ="moment")

iris.lda.predict <- predict(iris.lda,data.test)$class
(class_err_lda <- mean(iris.lda.predict != data.test[,indy]))
table(iris.lda.predict,data.test[,indy]) #混同行列. 行がテストデータ, 列が予測値
.
# ほぼ間違いなく判別できている.


## QDAによる判別
iris.qda <- qda(Species ~., data=data.train)
iris.qda.predict <- predict(iris.qda,data.test)$class
mean(iris.qda.predict != data.test[,indy]) #LDAとほぼ同じ性能
table(iris.qda.predict,data.test[,indy])


## L2正則化付きロジスティック回帰による判別
# glmnetを用いる
iris.glm <- glmnet(as.matrix(data.train[,-indy]),data.train[,indy], family="mult
inomial", alpha=0, standardize = FALSE) #alpha=1とするとL1正則化になる.
plot(iris.glm)  #正則化パス. lambdaを動かしてどう解が変わるか.

iris.glm <- glmnet(as.matrix(data.train[,-indy]),data.train[,indy], family="mult
inomial", lambda = 1, alpha=0, standardize = FALSE)  #lambdaを指定することも可能

mean(data.test[,indy] != predict(iris.glm, as.matrix(data.test[,-indy]), type="c
lass", standardize = FALSE))  #lambda=1はよくない

lamseq = exp(seq(-13,1,length.out=30))  #適切なlambdaを選ぼう
iris.glm.cv <- cv.glmnet(as.matrix(data.train[,-indy]),data.train[,indy], family
="multinomial", alpha=0, lam=lamseq, standardize = FALSE) #自動的にCVする. デフ
ォルトは10-fold CV.
plot(iris.glm.cv)  #lambdaに対してCVスコアをプロット
(class_err_glm_cv <- mean(data.test[,indy] !=
    predict(iris.glm.cv, as.matrix(data.test[,-indy]),
    s=iris.glm.cv$lambda.min, type="class", standardize = FALSE))) #最適なlambda
で判別

# 各lambdaごとの誤判別率を求める.
class_err_glm <- c(1:length(lamseq))
i <- 0
for(lam in lamseq){
    i <- i+1
    (class_err_glm[i] <- mean(data.test[,indy] != predict(iris.glm.cv, as.matrix
(data.test[,-indy]), s=lam, type="class", standardize = FALSE)))
}
```

```
class_err_glm
plot(class_err_glm)

##########################################
#もう少し難しいデータを使ってみよう.
#Vowelデータ. 多値判別
data(Vowel)
str(Vowel) #11次元, 990サンプル

train.index <- sample(nrow(Vowel),nrow(Vowel)*0.7)
data.train <- Vowel[train.index,]
data.test <- Vowel[-train.index,]
indy = 11; dimx = 10;

vowel.lda <- lda(Class ~., data=data.train)

vowel.lda.predict <- predict(vowel.lda,data.test)$class
mean(vowel.lda.predict != data.test[,indy])
table(vowel.lda.predict,data.test[,indy]) #そこそこ間違う。

# QDAを使ってみる.
vowel.qda <- qda(Class ~., data=data.train)
vowel.qda.predict <- predict(vowel.qda,data.test)$class
mean(vowel.qda.predict != data.test[,indy])
table(vowel.qda.predict,data.test[,indy]) #正答率が向上。

# データによってはldaの方が良い精度を出すことがある.
# qdaはモデルの自由度が上がっている分, 可適合しやすい.
# クロスバリデーションなどで適切な方法を選ぶべき.
# ロジスティック回帰は省略


#############################################
# MNIST手書き文字認識
# ロジスティック回帰
# サンプルが多くglmnetは動かない.
# 代わりにliblinearを使用：高速

data.train_origin <- read.csv("train_data.csv",header=FALSE) #訓練データの読み込
み
data.test <- read.csv("test_data.csv",header=FALSE) #テストデータの読み込み

train_size <- dim(data.train_origin)

ratio = 0.05;
(ntrain <- ceiling(train_size[1] * ratio)) #3000サンプルでやってみる.
permind <- sample(1:train_size[1],train_size[1])

data.train <- data.train_origin[permind[1:ntrain],] #3000サンプル取り出し
dimx <- train_size[2] - 1
yind <- train_size[2]

lamseq = seq(0,0.5,length.out =5)

numlam <- 10
lamseq = 10^(seq(-9,1,length.out =numlam)) #lambdaの候補
class_acc <- c(1:numlam)
i <- 0
for(lam in lamseq){
    i <- i+1;
    mnist.liblin <- LiblineaR(as.matrix(data.train[,1:dimx]),data.train[,yind],
type=0, cost=lam, epsilon = 0.01) #costは1/lambdaのようなもの.
    class_acc[i] <- mean(data.test[,yind] == predict(mnist.liblin, as.matrix(dat
a.test[,-yind]))$predictions)
}
class_acc #9割ほどの正答率. サンプル数を増やせばもっと精度は上がる.
plot(log(lamseq),class_acc,type="l",xlab="log(1/lambda)") #確かに正則化が効いて
いる.

bestlam <- lamseq[which.max(class_acc)]
mnist.bestliblin <- LiblineaR(as.matrix(data.train[,1:dimx]),data.train[,yind],
type=0, cost=bestlam, epsilon = 0.01)
```

```
table(data.test[,yind], predict(mnist.bestliblin, as.matrix(data.test[,-yind]))$
predictions)

# LDAは分散共分散行列が可逆にならない.
# 主成分分析などで次元を落としてから適用する.
```