```
########################################
##第七回「L1正則化法：高次元データ解析」
########################################
## 必要なライブラリ: caret glmnet LibLinear
library(caret)
library(LiblineaR)
library(glmnet)

set.seed(1)  #乱数種初期化

##############################
## glmnetによる判別
## バイオデータ
data(dhfr)
dim(dhfr) # 228次元, サンプル数325:高次元小標本データ
names(dhfr)

# 訓練データとテストデータの作成
train.index <- sample(nrow(dhfr),nrow(dhfr)*0.75)
data.train <- dhfr[train.index,]
data.test <- dhfr[-train.index,]
indy = 1

dhfr.glm <- glmnet(as.matrix(data.train[,-indy]),data.train[,indy], family="bino
mial") #デフォルトでalpha=1.つまりL1正則化.
plot.new()
plot(dhfr.glm) #正則化パスのプロット


lamseq = dhfr.glm$lambda  #正則化パラメータの列
dhfr.glm.cv <- cv.glmnet(as.matrix(data.train[,-indy]),data.train[,indy], family
="binomial",lam=lamseq)  #CVスコアの計算
par(mfrow=c(2,1))
plot(dhfr.glm.cv)  #CVスコアのプロット
plot(log(dhfr.glm.cv$lambda),dhfr.glm.cv$nzero,xlab="log(Lambda)") #228変数のう
ち使われているのは約50変数以下:スパース！
par(mfrow=c(1,1))
(dhfr.glm.cv$nzero[which(dhfr.glm.cv$lambda == dhfr.glm.cv$lambda.min)]) #CVで選
ばれた変数は17個
(class_err_glm_cv <- mean(data.test[,indy] != predict(dhfr.glm.cv, as.matrix(dat
a.test[,-indy]), s=dhfr.glm.cv$lambda.min, type="class"))) #誤判別率


# 各lambdaごとの誤判別率およびテストデータでの負の対数尤度を求める.
class_err_glm <- c(1:length(lamseq))
pred_acc_glm <- c(1:length(lamseq))
i <- 0
for(lam in lamseq){
    i <- i+1
    (class_err_glm[i] <- mean(data.test[,indy] != predict(dhfr.glm.cv, as.matrix
(data.test[,-indy]), s=lam, type="class"))) #誤判別率
    pred_link <- predict(dhfr.glm.cv, as.matrix(data.test[,-indy]), s=lam,type="
link") #beta^T xの値.
    pred_prob <- predict(dhfr.glm.cv, as.matrix(data.test[,-indy]), s=lam,type="
response") #ラベル=inactiveとなる確率.
    activeyind = which(data.test[,indy]=="active") #テストデータでラベル=active
となっているインデックス.
    (pred_acc_glm[i] <- (sum(log(1+exp(-pred_link[-activeyind])))+sum(log(1+exp(
pred_link[activeyind])))/length(data.test[,indy])) #テストデータでの負の対数尤
度
}

plot.new()
par(mfrow=c(3,1))
plot(lamseq,class_err_glm,log="x",ylab="判別誤差")
plot(lamseq,dhfr.glm.cv$cvm,log="x",ylab="CVスコア")
plot(lamseq,pred_acc_glm,log="x",ylab="予測精度")
par(mfrow=c(1,1)) # CVスコアと判別誤差・予測精度との関係に注目


####################################
## LiblineaRを用いたL1正則化判別分析
```

```
## スパムメール判別

spam <- read.csv("./spambase/spambase.data",header=FALSE)
train.index <- sample(nrow(spam),nrow(spam)*0.5)
data.train <- spam[train.index,]
data.test <- spam[-train.index,]
yind <- 58
dimx <- 57

numlam <- 20
lamseq = 10^(seq(-4,4,length.out =numlam)) #lambdaの候補


class_acc <- c(1:numlam)
i <- 0
for(lam in lamseq){
    i <- i+1;
    spam.liblin <- LiblineaR(as.matrix(data.train[,1:dimx]),data.train[,yind], t
ype=6, cost=lam, epsilon = 0.001) #costは1/lambdaのようなもの.
    class_acc[i] <- mean(data.test[,yind] == predict(spam.liblin, as.matrix(data
.test[,-yind]))$predictions)
}
max(class_acc) #最適な判別精度
plot(log(lamseq),class_acc,type="l",xlab="log(1/lambda)")

bestlam <- lamseq[which.max(class_acc)] #最適なlambda
spam.bestliblin <- LiblineaR(as.matrix(data.train[,1:dimx]),data.train[,yind], t
ype=6, cost=bestlam, epsilon = 0.001) #最適なlambdaでの判別

which(spam.bestliblin$W==0)
table(data.test[,yind], predict(spam.bestliblin, as.matrix(data.test[,-yind]))$p
redictions) #よく判別できている.


#続いてCVスコアと実際の判別誤差の関係をプロットしてみる.
CV_SCORE <- c(1:numlam)
i <- 0
for(lam in lamseq){
    i <- i+1;
    CV_SCORE[i] <- LiblineaR(as.matrix(data.train[,1:dimx]),data.train[,yind], t
ype=6, cost=lam, epsilon = 0.001, cross = 10)
}



plot(class_acc,type='l',col='red',ylim=c(0.7,0.95))
lines(CV_SCORE,col='black')

# デモのスパムメール判別では低次元の問題を扱ったが, 興味のある人は,
# 次のリンク先にあるデータで試してみると良いだろう.
# ECML-PKDD 2006 Discovery Challenge
# http://www.ecmlpkdd2006.org/challenge.html
# Download of Data Setsからデータは手に入る.
# livsvmフォーマットでデータが記述されているのでデータの整形が必要.
```