Template for the final exam 2023-01-11

Machine Learning ST5401, 7.5 credits

Yoe anonymous code

Loading packages

```
suppressPackageStartupMessages(library(ggplot))
suppressPackageStartupMessages(library(dplyr))
# ggplot and dplyr masks, so windows, is so required instead of %.
suppressPackageStartupMessages(library(Hmisc))
setwd("C:/Users/ioanb716/Downloads/Data")
# Change to the folder where the data is stored.

load("C:/Users/ioanb716/Downloads/Data/caracter_data_train.Rdata")
load("C:/Users/ioanb716/Downloads/Data/caracter_data_test.Rdata")
load("C:/Users/ioanb716/Downloads/Data/Crio_Pulp_Labelled.Rdata")
load("C:/Users/ioanb716/Downloads/Data/Crio_Pulp_Unlabelled.Rdata")
load("C:/Users/ioanb716/Downloads/Data/penguins.Rdata")
library(ggplot)
library(Matrix)
library(Hmisc)
library(glmnet)

##
## Attaching package: 'glmnet'
##
## The following objects are masked from 'package:Matrix':
##
##     pack, unpack
##
## Loaded glmnet 4.1-8
library(randomForest)
library(grid)
library(caret)

##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:Hmisc':
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
library(rsample)

##
## Attaching package: 'rsample'
##
## The following objects are masked from 'package:Hmisc':
##
##     contains, mean, mean1
##
library(MASS)
library(ggplot2)
library(tidyverse)
library(ggplot2)
library(tidyverse)
library(caretEnsemble)
```

Problem 1

a)

The purpose of the k-fold procedure is important effect on the prediction made by it. Pros Do it implies by construction, its correctly predicted and the model applied to the exact is and, | values of the model. This means that we will have low bias and high variance. higher value of k provide consistent behavior, which provides lower variation and high bias.

b)

None and and eventually formula.

d)

None and and eventually formula.

Since we often used a complex model that have a high variation, the purpose of bending the model to lower the variation with out increasing bias. The bootstrap is a method that can artificially creating multiple datasets of the size n.

e)

Early stopping is used in an optimization algorithm, so that we can stop the training before reaching the minimum. The purpose is to avoid overfitting.

Problem 2

The code block.

```
load("C:/Users/ioanb716/Downloads/Data/penguins.Rdata")
library(glmnet)

SL.bestsel = function(Y, X, ls, lambda,
                      family, obsweights, id, newX, newobsId)
{pf = ncol(X)
ind = c(1:pf)[abs(ls$fit[-1,]) > 0]
pen1[c(1:(pf-1))] = 1/abs(ls$fit[-1,])^(1/2)*(*)

if
{
return(
cv.fit.lasso
lambda.min
pen1
coef(cv.fit.lasso,
s = lambda.min)[-1]
}
}

cv.fit.lasso = cv.glmnet(x=X[,ind],
                         y=Y,
                         lambda=lambda,
                         penalty.factor=pen1[ind],
                         family=family,
                         weights=obsweights,
                         type.measure="deviance",
                         nfolds=length(Y),
                         keep=TRUE)

lambda.min = cv.fit.lasso$lambda.min
pred = cv.fit.lasso$fit.preval[,which(cv.fit.lasso$lambda==lambda.min)]
if(family=="binomial"){pred = exp(pred)/(1+exp(pred))}
if(family=="poisson"){pred = exp(pred)}
if(family=="gaussian"){pred = pred}
coef = coef(cv.fit.lasso,
            s = lambda.min)[-1]
out = list(pred=pred,
           coef=coef,
           fit=list(object=cv.fit.lasso,
                    lambda=lambda.min,
                    pen1=pen1[ind],
                    ind=ind))
if(!missing(newX)){
newpred = predict(cv.fit.lasso,
                  newx=newX[,ind],
                  s=lambda.min,
                  type="response")
out$newpred = newpred}
out$cvfit = cv.fit.lasso$cvm[which(cv.fit.lasso$lambda==lambda.min)]
out$nzero = sum(coef!=0)
return(out)
}

SL.bestsel(Y=Y, X=X, ls=ls, lambda=ls$lambda,
           family="binomial", obsweights=rep(1,length(Y)),
           id=id, newX=X[1:2,], newobsId=id[1:2])

SL.bestsel = function(Y, X, ls, lambda,
                      family, obsweights, id, newX, newobsId)
{pf = ncol(X)
ind = c(1:pf)[abs(ls$fit[-1,]) > 0]
pen1[c(1:(pf-1))] = 1/abs(ls$fit[-1,])^(1/2)*(*)

if
{
return(
cv.fit.lasso
lambda.min
pen1
coef(cv.fit.lasso,
s = lambda.min)[-1]
}
}

cv.fit.lasso = cv.glmnet(x=X[,ind],
                         y=Y,
                         lambda=lambda,
                         penalty.factor=pen1[ind],
                         family=family,
                         weights=obsweights,
                         type.measure="deviance",
                         nfolds=length(Y),
                         keep=TRUE)

lambda.min = cv.fit.lasso$lambda.min
pred = cv.fit.lasso$fit.preval[,which(cv.fit.lasso$lambda==lambda.min)]
if(family=="binomial"){pred = exp(pred)/(1+exp(pred))}
if(family=="poisson"){pred = exp(pred)}
if(family=="gaussian"){pred = pred}
coef = coef(cv.fit.lasso,
            s = lambda.min)[-1]
out = list(pred=pred,
           coef=coef,
           fit=list(object=cv.fit.lasso,
                    lambda=lambda.min,
                    pen1=pen1[ind],
                    ind=ind))
if(!missing(newX)){
newpred = predict(cv.fit.lasso,
                  newx=newX[,ind],
                  s=lambda.min,
                  type="response")
out$newpred = newpred}
out$cvfit = cv.fit.lasso$cvm[which(cv.fit.lasso$lambda==lambda.min)]
out$nzero = sum(coef!=0)
return(out)
}

SL.bestsel(Y=Y, X=X, ls=ls, lambda=ls$lambda,
           family="binomial", obsweights=rep(1,length(Y)),
           id=id, newX=X[1:2,], newobsId=id[1:2])
```

a)

Eventually some text and formulas.

```
message("The optimal lambda is",optimalLambda )

# The optimal lambda is 0.0200473161000
```

b)

Eventually some text and formulas.

```
message("The recall is", Recall, " and the precision is ", Precision )

# The recall is: 0.8700230700307 and the precision is 0.
```

c)

Eventually some text and formulas.

The recall decrease among all the lasso actually malign (diagnosis=1) breast cancer, the proportion that are classified malign (diagnosis=1) - so Recall. the model that classified malign (diagnosis=1) breast cancer, the proportion that are actually malign (diagnosis=1) breast cancer.

The precision decrease among all that are classified malign (diagnosis=1) breast cancer, the proportion that have actually have malign (diagnosis=1) breast cancer.

Precision: the model that classified malign (diagnosis=1) breast cancer, the proportion that have malignant model all classified malign (diagnosis=1)

The train goal should be preferred that both recall and precision at highest, but can be both if we have high data. Given that we don't have a large data so we can prefer recall. This only can be done by the threshold. but there is no solution cause if we want decrease the precision and increase the recall, we can do it by the threshold. but if we want increase both recall and precision, we can do it by the threshold. Given that recall is preferred, then if we want decreased that are classified as malign(diagnosis=1) given that they actually are benign, then precision is preferred.

Problem 4

The code block.

```
load("C:/Users/ioanb716/Downloads/Data/Crio_Pulp_Labelled.Rdata")
load("C:/Users/ioanb716/Downloads/Data/Crio_Pulp_Unlabelled.Rdata")

crio_fully_labelled

                  Peel.Length                   Peel.Width  Species
                        <dbl>                        <dbl>  <chr>
1                         1.4                          0.2  setosa
2                         1.4                          0.2  setosa
3                         1.3                          0.2  setosa
4                         1.5                          0.2  setosa
5                         1.4                          0.2  setosa
6                         1.7                          0.4  setosa
7                         1.4                          0.3  setosa
8                         1.5                          0.2  setosa
9                         1.4                          0.2  setosa
10                        1.5                          0.1  setosa
# ℹ 4,990 rows                                  <prev>  1  2  3  4  5  6  <NA>  Next

load("C:/Users/ioanb716/Downloads/Data/penguins.Rdata")

crio_mat = as.matrix(crio_fully_labelled[, 1:3])

lambda = 10^seq(log10(min(crio_fully_labelled$Species)*scaled*),
lambda = 10^seq(log10(max(crio_fully_labelled$Species)*scaled*),
lambda = 10^seq(log10(min(crio_fully_labelled$Species)*scaled*),
lambda = 10^seq(log10(max(crio_fully_labelled$Species)*scaled*),
cvfitlasso$cvm = cvfitlasso$cvm[which(cvfitlasso$lambda==lambda)]
cvfitridge$cvm = cvfitridge$cvm[which(cvfitlasso$lambda==lambda)]
cvfitridge$cvm = cvfitridge$cvm[which(cvfitlasso$lambda==lambda)]
cvfitridge$cvm = cvfitridge$cvm[which(cvfitlasso$lambda==lambda)]
Overall_mat = cbind(cvfitlasso$cvm, cvfitridge$cvm)

p_setosa = crossentropy(1 ~ p(setosa,fully_labelled)

p_versicolor = crossentropy(0 ~ p(setosa,fully_labelled)
p_virginica = crossentropy(0 ~ p(setosa,fully_labelled)

setosa_cv = crossentropy(1, 1:22)
versicolor_cv = crossentropy(0, 1:22)
virginica_cv = crossentropy(0, 1:22)

n_classes=22; N_0=22
BM_setosa = log(p_setosa) * log(crossentropy(p_setosa, setosa_mean, setosa_cov))
BM_versicolor = log(p_versicolor) * log(crossentropy(p_versicolor, versicolor_mean, versicolor_cov))
BM_virginica = log(p_virginica) * log(crossentropy(p_virginica, virginica_mean, virginica_cov))

only code 4c)

only code and formulas.

```
mixtureMultiClassLogit = function(mat, N, iniPhi, iniSigma, iniMu, tol)
# mat is a N x d matrix with d variables in columns and N observations
# iniPhi is a 1 x K vector with initial values for the component mixes
# iniSigma is a list of K d x d matrices with initial values for the covariance matrices
# iniMu is a d x K matrix with initial values for the component probabilities
d = ncol(mat)
count = 1
diff = 1000000
while (diff > tol){
# E-step: compute the posterior probabilities of the 3 observations and all components.
logPhi = log(iniPhi)
logSigma = list()
for (k in 1:K){
logSigma[[k]] = chol(iniSigma[[k]])
}
logDetSigma = sapply(logSigma, function(x) sum(log(diag(x))))
while (count < N){
for (k in 1:K){
# E-step
ll = dmvnorm(mat[count,], mu=iniMu[, k], sigma=iniSigma[[k]], log=TRUE)
ll[is.nan(ll)] = -Inf
logPost[k] = ll + logPhi[k]
}
for (k in 1:K){
logPost[k] = logPost[k] - logSum(logPost)
}
post = exp(logPost)
post = post / sum(post)
count = count + 1
}
# M-step: update the parameters
# Update phi
newPhi = colMeans(post)
# Update mu
newMu = matrix(0, d, K)
for (k in 1:K){
newMu[, k] = colSums(post[, k] * mat) / (N * post[k])
}
# Update Sigma
newSigma = list()
for (k in 1:K){
diff = max(abs(newPhi - iniPhi), abs(newMu - iniMu))
iniPhi = newPhi
iniMu = newMu
iniSigma = newSigma
count = 1
}
return(list(Phi = iniPhi, Mu = iniMu, Sigma = iniSigma))
}

mixtureMultiClassLogit(mat=mat, N=N, iniPhi=iniPhi, iniSigma=iniSigma, iniMu=iniMu, tol = 0.000001)

pred= apply(mat,1,BMM.pred)

tt=
Classified.pred = tt
Mat1 = ifelse(Mat1==1,1,0)
Classified.pred[which(Mat1==1)] = pred[1,1,1]
Classified.pred[1:5]; "setosa"

else if(pred[1,1,2] == pred[1,1,3] && pred[1,1,2] == pred[1,1,4])
Classified.pred[1:5] = "versicolor"

else{
Classified.pred[1:5] = "virginica"
}

crio_fully_labelled$Classified = Classified.pred

df = factor(crio_fully_labelled$Classified)
df = factor(crio_fully_labelled$Species)

cm_mat_crio=confusionMatrix(df, df)

The means of petal length and petal width given that species is setosa:

[1] "The means of petal length and petal width given that species is setosa:"

setosa_mean

Petal.Length Petal.Width
1.462   0.246

[1] "The means of petal length and petal width given that species is versicolor:"

versicolor_mean

Petal.Length Petal.Width
4.260   1.326

[1] "The means of petal length and petal width given that species is virginica:"

virginica_mean

Petal.Length Petal.Width
5.552   2.026

[1] "The means of petal length and petal width given that species is virginica:"

virginica_mean

Petal.Length Petal.Width
5.552   2.026

[1] "The means of petal length and petal width given that species is versicolor:"

versicolor_mean

Petal.Length Petal.Width
4.260   1.326

[1] "The means of petal length and petal width given that species is setosa:"

setosa_mean

Petal.Length Petal.Width
1.462   0.246

[1] "The covariance of petal length and petal width given that species is setosa:"

setosa_cov

Petal.Length Petal.Width
Petal.Length   0.030108   0.006091
Petal.Width   0.006091   0.011092

[1] "The covariance of petal length and petal width given that species is versicolor:"

versicolor_cov

Petal.Length Petal.Width
Petal.Length   0.220816   0.073100
Petal.Width   0.073100   0.039106

[1] "The covariance of petal length and petal width given that species is virginica:"

virginica_cov

Petal.Length Petal.Width
Petal.Length   0.304588   0.048824
Petal.Width   0.048824   0.075433

[1] "The covariance of petal length and petal width given that species is versicolor:"

versicolor_cov

Petal.Length Petal.Width
Petal.Length   0.220816   0.073100
Petal.Width   0.073100   0.039106

[1] "The covariance of petal length and petal width given that species is virginica:"

virginica_cov

Petal.Length Petal.Width
Petal.Length   0.304588   0.048824
Petal.Width   0.048824   0.075433

[1] "The covariance of petal length and petal width given that species is setosa:"

setosa_cov

Petal.Length Petal.Width
Petal.Length   0.030108   0.006091
Petal.Width   0.006091   0.011092

[1] "The covariance of petal length and petal width given that species is versicolor:"

versicolor_cov

Petal.Length Petal.Width
Petal.Length   0.220816   0.073100
Petal.Width   0.073100   0.039106

[1] "The covariance of petal length and petal width given that species is virginica:"

virginica_cov

Petal.Length Petal.Width
Petal.Length   0.304588   0.048824
Petal.Width   0.048824   0.075433