Classification of Cancer Patients with Penalized Robust Nonconvex Loss Functions

Zhu Wang
UT Health San Antonio
wangz1@uthscsa.edu

January 7, 2020

This document presents analysis for the MAQC-II project, human breast cancer data set with penalized classification algorithms developed in Wang (2019) and implemented in R package mpath.

Dataset comes from the MicroArray Quality Control (MAQC) II project and includes 278 breast cancer samples with 164 estrogen receptor (ER) positive cases. The data files GSE20194_series_matrix.txt.gz and GSE20194_MDACC_Sample_Info.xls can be downloaded from http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?token=rhojvaiwkcsalhq&acc=GSE20194. After reading the data, some unused variables are removed. From 22283 genes, the dataset is pre-screened to obtain 3000 genes with the largest absolute values of the two-sample t-statistics. The 3000 genes are standardized.

# The data files below were downloaded on June 1, 2016
require("gdata")
library("mpath")
bc <- t(read.delim("GSE20194_series_matrix.txt.gz", sep = ",",
header = FALSE, skip = 80))
colnames(bc) <- bc[1,]
bc <- bc[-1, -c(1, 2)]
### The last column is empty with variable name
### !series_matrix_table_end, thus omitted
bc <- bc[, -22284]
mode(bc) <- "numeric" ### convert character to numeric
dat1 <- read.xls("GSE20194_MDACC_Sample_Info.xls", sheet = 1,
header = TRUE)
y <- dat1$characteristics..ER_status
y <- ifelse(y == "P", 1, -1)
table(y)
res <- rep(NA, dim(bc)[2])
for (i in 1:dim(bc)[2]) res[i] <- abs(t.test(bc[, i] ~ y)$statistic)
### find 3000 largest absolute value of t-statistic
tmp <- order(res, decreasing = TRUE)[1:3000]
dat <- bc[, tmp]
### standardize variables
dat <- scale(dat)

Set up configuration parameters.

```r
### number of replicates
nrun <- 100
### penalty type
penalty <- c("enet", "snet", "mnet")
### Smallest value for lambda, as a fraction of lambda.max, the
### smallest value for which all coefficients are zero except
### the intercept
ratio <- 0.25
### Smallest value for lambda, as a fraction of lambda.max, the
### smallest value for which all coefficients are zero except
### the intercept
ratio <- 0.25
### The training data is contaminated by randomly switching
### response variable labels at varying pre-specified
### proportions
per <- c(0, 0.05, 0.1, 0.15)
### what quantity is minimized for tuning parameter selection
### robust nonconvex loss function, rfamily type and logistic
### type <- c("closs", "gloss", "qloss", "binomial")
### and corresponding labels
type1 <- c("Closs", "Gloss", "Qloss", "Logistic")
### and corresponding tuning parameter
s <- c(0.9, 1.01, 0.5)
mstop <- 50
plot.it <- TRUE
```

The training data contains randomly selected 50 samples with positive estrogen receptor status and 50 samples with negative estrogen receptor status, and the rest were designated as the test data. The training data is contaminated by randomly switching response variable labels at varying pre-specified proportions per = 0, 0.05, 0.1, 0.15. This process is repeated nrun = 100 times. Robust non-convex loss functions include C-loss, G-loss and Q-loss, each with penalty LASSO, SCAD and MCP. The initial values are derived using the boosting package bst with mstop = 50 and nu provided below depending on loss function type. For SCAD and MCP penalty, a penalty tuning parameter gam is provided below. To select optimal penalization tuning parameters, we run five-fold cross-validation averaging classification errors. The classification errors and number of selected variables are tabularized and plotted with plot.it = TRUE. With type.path = "nonactive", the prediction accuracy can be slightly improved in some scenarios, for instance, with contamination rate 15%. However, the computing times can be doubled. Finally, this script also contains results with penalized logistic regression using glmreg.

```r
summary7 <- function(x) c(summary(x), sd = sd(x))
ptm <- proc.time()
for (k in (1:4)) {
```
### k controls family argument rfamily type (see above)
if (type[k] == "gloss")
  nu <- 0.1 else nu <- 0.01
for (j in (1:3)) {
  ### j controls argument penalty type (see above)
  gam <- ifelse(penalty[j] == "snet", 3.7, 12)
  err.m1 <- nvar.m1 <- errbest.m1 <- lambest.m1 <- matrix(NA,
    ncol = 4, nrow = nrun)
  nvarbest.m1 <- mstopcv.m1 <- matrix(NA, ncol = 4, nrow = nrun)
  colnames(err.m1) <- c("cont-0\%", "cont-5\%", "cont-10\%",
    "cont-15\%")
  colnames(nvar.m1) <- colnames(nvarbest.m1) <- colnames(err.m1)
  colnames(mstop.m1) <- colnames(err.m1)
  colnames(lambest.m1) <- colnames(err.m1)
  for (ii in 1:nrun) {
    set.seed(1000 + ii)
    trid <- c(sample(which(y == 1))[1:50], sample(which(y ==
      -1))[1:50])
    dtr <- dat[trid, ]
    dte <- dat[-trid, ]
    ytrold <- y[trid]
    yte <- y[-trid]
    ### number of patients/no. variables in training and test data
    dim(dtr)
    dim(dte)
    ### randomly contaminate data
    ntr <- length(trid)
    set.seed(1000 + ii)
    con <- sample(ntr)
    for (i in (1:4)) {
      ### i controls how many percentage of data contaminated, see
      ### argument per above
      ytr <- ytrold
      percon <- per[i]
      ### randomly flip labels of the samples in training set
      ### according to pre-defined contamination level
      if (percon > 0) {
        ji <- con[1:(percon * ntr)]
        ytr[ji] <- -ytrold[ji]
      }
      ### fit a model with nclreg for nonconvex loss or glmreg for
      ### logistic loss, and use cross-validation to select best
      ### penalization parameter
      if (type[k] %in% c("closs", "gloss", "qloss")) {
        dat.m1 <- nclreg(x = dtr, y = ytr, s = s[k],
          iter = 100, rfamily = type[k], penalty = penalty[j],
          lambda.min.ratio = ratio, gamma = gam, mstop.init = mstop,
          nu.init = nu, type.path = type.path, decreasing = FALSE,
        } else {
          glmreg(x = dtr, y = ytr, s = s[k],
            iter = 100, rfamily = type[k], penalty = penalty[j],
            lambda.min.ratio = ratio, gamma = gam, mstop.init = mstop,
            nu.init = nu, type.path = type.path, decreasing = FALSE,
          )
        }
      }
    }
  }
}
type.init = "bst")
lambda <- dat.m1$lambda[1:nlam]
set.seed(1000 + ii)
cvm1 <- cv.nclreg(x = dtr, y = ytr, nfolds = 5,
n.cores = n.cores, s = s[k], lambda = lambda,
rfamily = type[k], penalty = penalty[j],
gamma = gam, type = tuning, plot.it = FALSE,
type.init = dat.m1$type.init, mstop.init = dat.m1$mstop.init,
nu.init = dat.m1$nu.init, type.path = type.path,
decreasing = dat.m1$decreasing)
err1 <- predict(dat.m1, newdata = dte, newy = yte,
type = "error")
}
else {
dat.m1 <- glmreg(x = dtr, y = (ytr + 1)/2,
family = type[k], penalty = penalty[j], lambda.min.ratio = ratio,
gamma = gam)
set.seed(1000 + ii)
cvm1 <- cv.glmreg(x = dtr, y = (ytr + 1)/2,
nfolds = 5, n.cores = n.cores, lambda = dat.m1$lambda,
family = type[k], penalty = penalty[j], gamma = gam,
plot.it = FALSE)
err1 <- apply((yte > -1) != predict(dat.m1,
newx = dte, type = "class"), 2, mean)
}
optmstop <- cvm1$lambda.which
err.m1[ii, i] <- err1[optmstop]
nvar.m1[ii, i] <- length(predict(dat.m1, which = optmstop,
type = "nonzero"))
errbest.m1[ii, i] <- min(err1, na.rm = TRUE)
lambest.m1[ii, i] <- which.min(err1)
nvarbest.m1[ii, i] <- length(predict(dat.m1,
which = which.min(err1), type = "nonzero"))
}
if (ii%%nrun == 0) {
if (type[k] %in% c("closs", "gloss", "qloss"))
cat(paste("\nfamily ", type1[k],", s=", s[k],
sep = ""), "\n") else cat(paste("\nfamily ", type1[k], sep = ""), "\n")
pentype <- switch(penalty[j], enet = "LASSO",
mnet = "MCP", snet = "SCAD")
cat("penalty="), pentype, "\n")
if (penalty[j] %in% c("snet", "mnet"))
cat("gamma="), gam, "\n")
cat("best misclassification error\n")
print(round(apply(errbest.m1, 2, summary7), 4))
cat("which lambda has best error\n")
print(round(apply(lambest.m1, 2, summary7), 1))
cat("number of variables selected with best error\n")
print(round(apply(nvarbest.m1, 2, summary7), 4)
1))
cat("CV based misclassification error\n")
print(round(apply(err.m1, 2, summary7), 4))
cat("number of variables selected by CV\n")
print(round(apply(nvar.m1, 2, summary7), 1))
if (plot.it) {
  par(mfrow = c(2, 1))
  boxplot(err.m1, main = "Misclassification error",
          subset = "", sub = paste(type1[k], "-", pentype,
          sep = ""))
  boxplot(nvar.m1, main = "No. variables", subset = "",
          sub = paste(type1[k], "-", pentype, sep = ""))
}
References