Analysis of Cancer Data with Boosting Algorithm for Nonconvex Loss

Zhu Wang
Connecticut Children's Medical Center
University of Connecticut School of Medicine
zwang@connecticutchildrens.org

September 13, 2016

This document presents analysis for the MAQC-II project, human breast cancer data set with boosting algorithms developed in Wang (2016a,b) and implemented in R package `bst`.

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=rhojvaiwkcsaihq&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")
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.

```
nrun <- 100
per <- c(0, 0.05, 0.1, 0.15)
learntype <- c("tree", "ls")[2]
tuning <- "error"
n.cores <- 5
plot.it <- TRUE
### robust tuning parameters used in bst/rbst function
s <- c(0.9, 1.01, 0.5, -0.2, 0.8, -0.5, -0.2)
nu <- c(0.01, 0.1, 0.01, rep(0.1, 4))
m <- 100  ### boosting iteration number
### whether to truncate the predicted values in each boosting
### iteration?
ctr.trun <- c(TRUE, rep(FALSE, 6))
### used in bst function
bsttype <- c("closs", "gloss", "qloss", "binom", "binom", "hinge",
    "expo")
### and corresponding labels
bsttype1 <- c("ClossBoost", "GlossBoost", "QlossBoost", "LogitBoost",
    "LogitBoost", "HingeBoost", "AdaBoost")
### used in rbst function
rbsttype <- c("closs", "gloss", "qloss", "tbinom", "binomd",
    "thinge", "texpo")
### and corresponding labels
rbsttype1 <- c("ClossBoostQM", "GlossBoostQM", "QlossBoostQM",
    "TLogitBoost", "DlogitBoost", "THingeBoost", "TAdaBoost")
```

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. The base learner is `learntype`=ls. To select optimal boosting iteration from maximum value of `m`=100, we run five-fold cross-validation averaging classification errors. In cross-validation, we set the number of cores for parallel computing by `n.cores`=5. Selected results can be plotted if `plot.it=TRUE`. Gradient based boosting includes ClossBoost, GlossBoost, QlossBoost, LogitBoost, HingeBoost and AdaBoost. Robust boosting using `rbst` contains ClossBoostQM, GlossBoostQM, QlossBoostQM, TLogitBoost, DlogitBoost, THingeBoost and TAdaBoost.

```
summary7 <- function(x) c(summary(x), sd = sd(x))
ptm <- proc.time()
for (k in 1:7) {
    ### k controls which family in bst, and rfamily in rbst
    err.m1 <- err.m2 <- nvar.m1 <- nvar.m2 <- errbest.m1 <- errbest.m2 <- matrix(NA,
        ncol = 4, nrow = nrun)
    mstopbest.m1 <- mstopbest.m2 <- mstopcv.m1 <- mstopcv.m2 <- matrix(NA,
        ncol = 4, nrow = nrun)
```

```
colnames(err.m1) <- colnames(err.m2) <- c("cont-0%", "cont-5%",
    "cont-10%", "cont-15%")
colnames(mstopcv.m1) <- colnames(mstopcv.m2) <- colnames(err.m1)
colnames(nvar.m1) <- colnames(nvar.m2) <- colnames(err.m1)
colnames(errbest.m1) <- colnames(errbest.m2) <- colnames(err.m1)
colnames(mstopbest.m1) <- colnames(mstopbest.m2) <- 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 (j in 1) {
        ### controls learntype i controls how many percentage of data
        ### contaminated
        for (i in 1:4) {
            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]
            }
            dat.m1 <- bst(x = dtr, y = ytr, ctrl = bst_control(mstop = m,
              center = FALSE, trace = FALSE, nu = nu[k],
              s = s[k], trun = ctr.trun[k]), family = bsttype[k],
              learner = learntype[j])
            err1 <- predict(dat.m1, newdata = dte, newy = yte,
              type = "error")
            err1tr <- predict(dat.m1, newdata = dtr, newy = ytr,
              type = "loss")
            ### cross-validation to select best boosting iteration
            set.seed(1000 + ii)
            cvm1 <- cv.bst(x = dtr, y = ytr, K = 5, n.cores = n.cores,
              ctrl = bst_control(mstop = m, center = FALSE,
                trace = FALSE, nu = nu[k], s = s[k], trun = ctr.trun[k]),
              family = bsttype[k], learner = learntype[j],
              main = bsttype[k], type = tuning, plot.it = FALSE)
            optmstop <- max(10, which.min(cvm1$cv))
            err.m1[ii, i] <- err1[optmstop]
            nvar.m1[ii, i] <- nsel(dat.m1, optmstop)[optmstop]
            errbest.m1[ii, i] <- min(err1)
            mstopbest.m1[ii, i] <- which.min(err1)
            mstopcv.m1[ii, i] <- optmstop
            dat.m2 <- rbst(x = dtr, y = ytr, ctrl = bst_control(mstop = m,
              iter = 100, nu = nu[k], s = s[k], trun = ctr.trun[k],
              center = FALSE, trace = FALSE), rfamily = rbsttype[k],
              learner = learntype[j])
            err2 <- predict(dat.m2, newdata = dte, newy = yte,
              type = "error")
            err2tr <- predict(dat.m2, newdata = dtr, newy = ytr,
              type = "loss")
            ### cross-validation to select best boosting iteration
            set.seed(1000 + ii)
            cvm2 <- cv.rbst(x = dtr, y = ytr, K = 5, n.cores = n.cores,
              ctrl = bst_control(mstop = m, iter = 100, nu = nu[k],
                s = s[k], trun = ctr.trun[k], center = FALSE,
                trace = FALSE), rfamily = rbsttype[k], learner = learntype[j],
              main = rbsttype[k], type = tuning, plot.it = FALSE)
            optmstop <- max(10, which.min(cvm2$cv))
            err.m2[ii, i] <- err2[optmstop]
            nvar.m2[ii, i] <- nsel(dat.m2, optmstop)[optmstop]
            errbest.m2[ii, i] <- min(err2)
            mstopbest.m2[ii, i] <- which.min(err2)
            mstopcv.m2[ii, i] <- optmstop
        }
    }
    if (ii%%nrun == 0) {
        if (bsttype[k] %in% c("closs", "gloss", "qloss"))
            cat(paste("\nbst family ", bsttype1[k], ", s=",
              s[k], ", nu=", nu[k], sep = ""), "\n")
        if (bsttype[k] %in% c("binom", "hinge", "expo"))
            cat(paste("\nbst family ", bsttype1[k], ", nu=",
              nu[k], sep = ""), "\n")
        cat("best misclassification error from bst\n")
        print(round(apply(errbest.m1, 2, summary7), 4))
        cat("CV based misclassification error from bst\n")
        print(round(apply(err.m1, 2, summary7), 4))
        cat("best mstop with best misclassification error from bst\n")
        print(round(apply(mstopbest.m1, 2, summary7), 0))
        cat("best mstop with CV from bst\n")
        print(round(apply(mstopcv.m1, 2, summary7), 0))
        cat("nvar from bst\n")
        print(round(apply(nvar.m1, 2, summary7), 1))

        cat(paste("\nrbst family ", rbsttype1[k], ", s=",
            s[k], ", nu=", nu[k], sep = ""), "\n")
        cat("\nbest misclassification error from rbst\n")
```

```
print(round(apply(errbest.m2, 2, summary7), 4))
cat("CV based misclassification error from rbst\n")
print(round(apply(err.m2, 2, summary7), 4))
cat("best mstop with best misclassification error from rbst\n")
print(round(apply(mstopbest.m2, 2, summary7), 0))
cat("best mstop with CV from rbst\n")
print(round(apply(mstopcv.m2, 2, summary7), 0))
cat("nvar from rbst\n")
print(round(apply(nvar.m2, 2, summary7), 1))
res <- list(err.m1 = err.m1, nvar.m1 = nvar.m1, errbest.m1 = errbest.m1,
    mstopbest.m1 = mstopbest.m1, mstopcv.m1 = mstopcv.m1,
    err.m2 = err.m2, nvar.m2 = nvar.m2, errbest.m2 = errbest.m2,
    mstopbest.m2 = mstopbest.m2, mstopcv.m2 = mstopcv.m2,
    s = s[k], nu = nu[k], trun = ctr.trun[k], family = bsttype[k],
    rfamily = rbsttype[k])
if (plot.it) {
    par(mfrow = c(2, 1))
    boxplot(err.m1, main = "Misclassification error",
      subset = "", sub = bsttype1[k])
    boxplot(err.m2, main = "Misclassification error",
      subset = "", sub = rbsttype1[k])
    boxplot(nvar.m1, main = "No. variables", subset = "",
      sub = bsttype1[k])
    boxplot(nvar.m2, main = "No. variables", subset = "",
      sub = rbsttype1[k])
}
check <- FALSE
if (check) {
    par(mfrow = c(3, 1))
    title <- paste("percentage of contamination ",
      percon, sep = "")
    plot(err2tr, main = title, ylab = "Loss value",
      xlab = "Iteration", type = "l", lty = "dashed",
      col = "red")
    points(err1tr, type = "l", lty = "solid", col = "black")
    legend("topright", c(bsttype1[k], rbsttype1[k]),
      lty = c("solid", "dashed"), col = c("black",
        "red"))
    plot(err2, main = title, ylab = "Misclassification error",
      xlab = "Iteration", type = "l", lty = "dashed",
      col = "red")
    points(err1, type = "l")
    legend("bottomright", c(bsttype1[k], rbsttype1[k]),
      lty = c("solid", "dashed"), col = c("black",
        "red"))
    plot(nsel(dat.m2, m), main = title, ylab = "No. variables",
      xlab = "Iteration", lty = "dashed", col = "red",
      type = "l")
    points(nsel(dat.m1, m), ylab = "No. variables",
                  xlab = "Iteration", lty = "solid", type = "l",
                  col = "black")
                legend("bottomright", c(bsttype1[k], rbsttype1[k]),
                  lty = c("solid", "dashed"), col = c("black",
                    "red"))
            }
        }
    }
}
print(proc.time() - ptm)
```

```
sessionInfo()
## R version 3.3.1 (2016-06-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.3 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets
## [6] methods   base
##
## other attached packages:
## [1] knitr_1.11
##
## loaded via a namespace (and not attached):
##  [1] magrittr_1.5      formatR_1.2.1     tools_3.3.1
##  [4] R.rsp_0.20.0      stringi_0.4-1     R.methodsS3_1.7.0
##  [7] stringr_1.0.0     R.cache_0.10.0    R.utils_1.34.0
## [10] evaluate_0.8      R.oo_1.18.0
```

References

Zhu Wang. Robust boosting with truncated loss functions. 2016a. manuscript.

Zhu Wang. Quadratic majorization for nonconvex loss with applications to boosting algorithm. 2016b. manuscript.