Untitled

2023-04-19

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

summary(cars)

## speed dist   
## Min. : 4.0 Min. : 2.00   
## 1st Qu.:12.0 1st Qu.: 26.00   
## Median :15.0 Median : 36.00   
## Mean :15.4 Mean : 42.98   
## 3rd Qu.:19.0 3rd Qu.: 56.00   
## Max. :25.0 Max. :120.00

## Including Plots

You can also embed plots, for example:

library(neuralnet)  
  
file <- "TCGA\_breast\_cancer\_ERstatus\_allGenes.txt"  
nfold <- 5  
sd\_threshold <- 1  
#sd\_threshold <- 4  
#top\_num <- 10  
#top\_num <- 50  
#top\_num <- 100  
  
header <- scan(file, nlines = 1, sep="\t", what = character())  
data <- read.table(file, skip = 2, header = FALSE, sep = "\t", quote = "", check.names=FALSE)  
names(data) <- header  
  
header2 <- scan(file, skip = 1, nlines = 1, sep="\t", what = character())  
  
# cleanup - remove genes with sd < 1  
# compute sd for each gene  
data\_sd<-sapply(seq(nrow(data)), function(x) { as.numeric(sd(data[x,-1])) })  
  
# add gene names to the sd list  
data\_sd\_names<-cbind(data.frame(data\_sd),data[,1])  
  
# create an "include" list of all those genes where sd > threashold  
include\_list <- data\_sd\_names[data\_sd\_names[,1]>sd\_threshold,2]  
  
Positive <- data[data$id %in% include\_list,header2=='Positive']  
Negative <- data[data$id %in% include\_list,header2=='Negative']  
  
# define function cross\_valid so we can rerun the cross validataion with various parameters  
cross\_validation <- function (nfold, top\_num, alg) {  
   
 Positive\_groups <- split(sample(colnames(Positive)), 1+(seq\_along(colnames(Positive)) %% nfold))  
 Negative\_groups <- split(sample(colnames(Negative)), 1+(seq\_along(colnames(Negative)) %% nfold))  
   
 result <- array()  
   
 for (test\_group in 1:nfold) {  
   
 testA <- Positive[,colnames(Positive) %in% unlist(Positive\_groups[test\_group])]  
 testB <- Negative[,colnames(Negative) %in% unlist(Negative\_groups[test\_group])]  
   
 trainingA <- Positive[,!(colnames(Positive) %in% unlist(Positive\_groups[test\_group]))]  
 trainingB <- Negative[,!(colnames(Negative) %in% unlist(Negative\_groups[test\_group]))]  
   
 # Feature selection --   
   
 # compute t-statistic for each row  
 training\_t\_stat<-data.frame(sapply(seq(nrow(trainingA)), function(x) { abs(as.numeric(t.test(trainingA[x,], trainingB[x,])$statistic)) }))  
   
 # add gene id column  
 training\_t\_stat\_geneid<-cbind(training\_t\_stat,rownames(trainingA))  
 colnames(training\_t\_stat\_geneid) <- c('t','id')  
   
 # pick top 50 based on t-statistic  
 selected\_genes <- head(training\_t\_stat\_geneid[order(-training\_t\_stat\_geneid$t),],n=top\_num)[,2]  
   
 # narrow down the list of genes based on t-statistic  
 testA <- testA[rownames(testA) %in% selected\_genes,]  
 testB <- testB[rownames(testB) %in% selected\_genes,]  
 trainingA <- trainingA[rownames(trainingA) %in% selected\_genes,]  
 trainingB <- trainingB[rownames(trainingB) %in% selected\_genes,]  
   
 if(alg == "centroid") {  
   
 centroidA <- rowMeans(trainingA)  
 centroidB <- rowMeans(trainingB)  
   
 misclassifiedA <- sum(sapply(testA, function(x) { sqrt(sum((x-centroidA)^2))-sqrt(sum((x-centroidB)^2))>0 }))  
 misclassifiedB <- sum(sapply(testB, function(x) { sqrt(sum((x-centroidA)^2))-sqrt(sum((x-centroidB)^2))<0 }))  
 }  
   
 if(alg == "glm") {  
 trainingCombined <- rbind(cbind(data.frame(t(trainingA)),cancer=0),cbind(data.frame(t(trainingB)),cancer=1))  
 testA0 <- data.frame(t(testA))  
 testB0 <- data.frame(t(testB))  
   
 model <- glm(cancer ~ ., data=trainingCombined, family=binomial, control = list(maxit=50))  
 pA <- predict(model, newdata= testA0, type="response")  
 pB <- predict(model, newdata= testB0, type="response")  
   
 misclassifiedA <- sum(ifelse(pA<0.5,0,1))  
 misclassifiedB <- sum(ifelse(pB>0.5,0,1))  
 }  
   
 if(alg == "nn") {  
 trainingCombined <- rbind(cbind(data.frame(t(trainingA)),cancer=0),cbind(data.frame(t(trainingB)),cancer=1))  
 testA0 <- data.frame(t(testA))  
 testB0 <- data.frame(t(testB))  
   
 ## 1-layer network with 2 neurons, startweights - random, linear.output = False for classification  
 model <- neuralnet(cancer ~., data=trainingCombined, hidden=c(2), startweights = NULL, linear.output =F)  
 pA <- predict(model, testA0, rep = 1, all.units = FALSE)  
 pB <- predict(model, testB0, rep = 1, all.units = FALSE)  
   
 misclassifiedA <- sum(ifelse(pA<0.5,0,1))  
 misclassifiedB <- sum(ifelse(pB>0.5,0,1))  
 }  
 ## Plot the Neural Net model - Black lines = connnections between each layer and weights, Blue lines = bias  
 plot(model)  
 result[test\_group] <- (misclassifiedA+misclassifiedB)/(ncol(testA)+ncol(testB))  
 }  
   
 paste0(round(mean(result),4)," sd=(",round(sd(result),4),")")  
}  
  
centroid\_50\_all <- cross\_validation(nfold=5, top\_num = 50, alg= "centroid")  
centroid\_100\_all <- cross\_validation(nfold=5, top\_num = 100, alg= "centroid")  
  
glm\_50\_all <- cross\_validation(nfold=5, top\_num = 50, alg = "glm")  
glm\_100\_all <- cross\_validation(nfold=5, top\_num = 100, alg = "glm")  
  
nn\_50\_all <- cross\_validation(nfold=5, top\_num = 50, alg = "nn")  
nn\_100\_all <- cross\_validation(nfold=5, top\_num = 100, alg = "nn")  
  
x<-data.frame("Centroid"=c(centroid\_50\_all),"GLM"=c(glm\_50\_all))  
rownames(x) <- c("50 genes")  
kable(x) # Requires knitr package  
  
x1<-data.frame("Centroid"=c(centroid\_50\_all,centroid\_100\_all),"GLM"=c(glm\_50\_all,glm\_100\_all),"NeuralNet"=c(nn\_50\_all,nn\_100\_all))  
rownames(x1) <- c("50 genes","100 genes")  
  
  
x3<-data.frame("Centroid"=c(centroid\_50\_all),"GLM"=c(glm\_50\_all), "NeuralNet"=c(nn\_50\_all))  
rownames(x3) <- c("50 genes")  
kable(x3)

Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.