Naive Bayes_Numerical and Categorical Features

Zhoutao Pei

Read in data and preprocessing

```r
rawdata <- read.table("processed.cleveland.data", sep=',')
rawdata$V14[rawdata$V14 != 0] <- 1

### missing data imputation
rawdata$V12 <- as.character(rawdata$V12)
rawdata$V13 <- as.character(rawdata$V13)
rawdata$V12[rawdata$V12 == '?'] <- '0.0'
rawdata$V13[rawdata$V13 == '?'] <- '3.0'
rawdata$V12 <- as.numeric(rawdata$V12)
rawdata$V13 <- as.numeric(rawdata$V13)

X = rawdata[, -14]
y = rawdata[, 14]
```

Function to train a Naiva Bayes classifier with both numerical and categorical features.

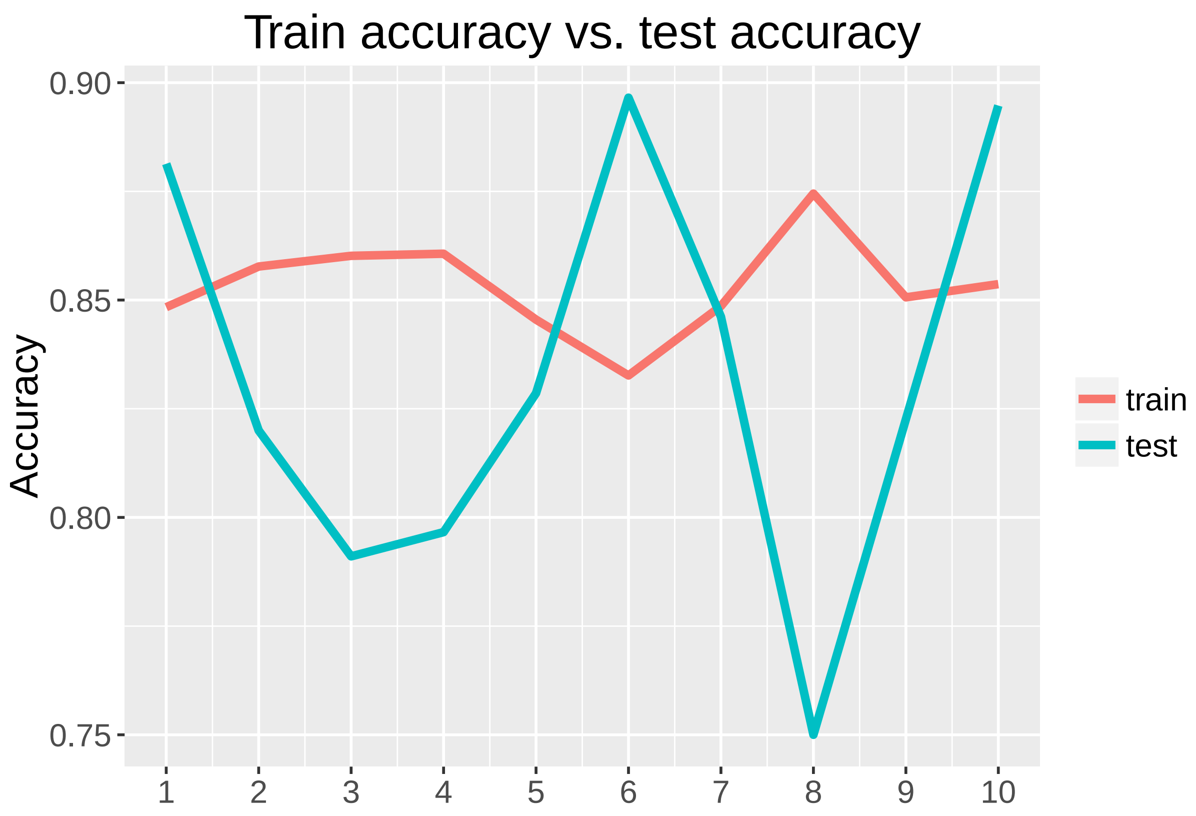
```r
NBTrain <- function(trainX, trainy, cateV, numeV) {
  # cateV and numeV are the column names for categorical variables and
  # numerical variables in data frame trainX
  trainX.pos <- trainX[trainy == 1,]
  trainX.neg <- trainX[trainy == 0,]
  p.pos <- nrow(trainX.pos) / nrow(trainX)
  p.neg <- 1 - p.pos
  cateVp.pos <- list()
  cateVp.neg <- list()
  for(c in cateV){
    cateVp.pos[[c]] <- summary(as.factor(trainX.pos[, c])) / nrow(trainX.pos)
    cateVp.pos[[c]]['Absent'] <- 0.5 / nrow(trainX.pos)
    cateVp.neg[[c]] <- summary(as.factor(trainX.neg[, c])) / nrow(trainX.neg)
    cateVp.neg[[c]]['Absent'] <- 0.5 / nrow(trainX.neg)
  }
  numeVmean.pos <- colMeans(trainX.pos[, numeV], na.rm=T)
  numeVmean.neg <- colMeans(trainX.neg[, numeV], na.rm=T)
  numeVsd.pos <- apply(trainX.pos[, numeV], 2, sd, na.rm=T)
  numeVsd.neg <- apply(trainX.neg[, numeV], 2, sd, na.rm=T)
  return(list("cateV" = cateV,
              "numeV" = numeV,
              "pPos" = p.pos,
              "pNeg" = p.neg,
              "cateVpPos" = cateVp.pos,
              "cateVpNeg" = cateVp.neg,
              "numeVmeanPos" = numeVmean.pos,
              "numeVmeanNeg" = numeVmean.neg,
              "numeVsdPos" = numeVsd.pos,
              "numeVsdNeg" = numeVsd.neg))
}
```

Function to make predictions given a trained NB model

```
NBPredict <- function(NBModel, testX) {
  pMat.pos <- matrix(NA, nrow(testX), ncol(testX))
  pMat.neg <- matrix(NA, nrow(testX), ncol(testX))
  colnames(pMat.pos) <- c(NBModel$cateV, NBModel$numeV)
  colnames(pMat.neg) <- c(NBModel$cateV, NBModel$numeV)
  testX.numeV.normed.pos <- t((t(testX[, NBModel$numeV]) - NBModel$numeVmeanPos)/NBModel$numeVsdPos)
  pMat.pos[, NBModel$numeV] <- t(t(-(1/2)*testX.numeV.normed.pos^2) - log(NBModel$numeVsdPos))
  testX.numeV.normed.neg <- t((t(testX[, NBModel$numeV]) - NBModel$numeVmeanNeg)/NBModel$numeVsdNeg)
  pMat.neg[, NBModel$numeV] <- t(t(-(1/2)*testX.numeV.normed.neg^2) - log(NBModel$numeVsdNeg))
  for (c in NBModel$cateV) {
    cateP.pos <- NBModel$cateVpPos[[c]][as.character(testX[, c])]
    cateP.pos[is.na(cateP.pos)] <- NBModel$cateVpPos[[c]]['Absent']
    pMat.pos[, c] <-log(cateP.pos)
    cateP.neg <- NBModel$cateVpNeg[[c]][as.character(testX[, c])]
    cateP.neg[is.na(cateP.neg)] <- NBModel$cateVpNeg[[c]]['Absent']
    pMat.neg[, c] <- log(cateP.neg)
  }
  loglik.pos <- rowSums(pMat.pos) + log(NBModel$pPos)
  loglik.neg <- rowSums(pMat.neg) + log(NBModel$pNeg)
  pred <- as.numeric(loglik.pos > loglik.neg)
  return(pred)
}
```

Run model 10 times with different train data/test data splits

```
categoricalV <- paste('V', c(2,3,6,7,9,11,12,13), sep='')
numericalV <- paste('V', c(1,4,5,8,10), sep='')
train.acc <- array(dim=10)
test.acc <- array(dim=10)
for (i in 1:10) {
  set.seed(i*100)
  trainIdx <- runif(nrow(X)) < 0.8
  trainX <- X[trainIdx,]
  trainy <- y[trainIdx]
  testX <- X[!trainIdx,]
  testy <- y[!trainIdx]
  NBModel <- NBTrain(trainX, trainy, categoricalV, numericalV)
  # training accuracy
  pred <- NBPredict(NBModel, trainX)
  train.acc[i] <- sum(pred == trainy) / length(trainy)
  # test accuracy
  pred <- NBPredict(NBModel, testX)
  test.acc[i] <- sum(pred == testy) / length(testy)
}
```


Train accuracy vs. test accuracy
Accuracy
0.90
0.85
0.80
0.75
1
2
3
4
5
6
7
8
9
10
train
test