adaBoost

Sarah Inman

March 11, 2016

```
library(gbm)
```

```
## Loading required package: survival
## Loading required package: lattice
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
```

```
library(ggplot2)
library(reshape2)
spam <- read.csv('/home/beeb/Documents/Data_Science/Advanced Computational Methods/Spam/spambase.data')

adaBoost <- function(formula, data, depth, noTrees, testdata = NA) {

  #Load in useful packages
  library(rpart)
  library(formula.tools)
  library(assertthat)

  # Check inputs
  assert_that(is.data.frame(data))
  assert_that(is.numeric(depth))
  assert_that(is.numeric(noTrees))

  # Initialise weights
  weightvector <- rep(1/nrow(data), nrow(data))
  all.pred <- matrix(NA, nrow = nrow(data), ncol = noTrees)

  if(length(testdata)>1) {
    all.pred.test <- matrix(NA, nrow = nrow(testdata), ncol = noTrees)
  }
  # Iterate
  for(m in 1:noTrees) {
      environment(formula) <- environment()
    tree <- rpart(as.formula(formula), data, weights = weightvector, maxdepth = depth,
                  method = 'class')
    predprobs <- round(predict(tree))
    predictions <- colnames(predprobs)[apply(predprobs, 1, FUN = function(x) {
      which(x == max(x))
      })]
    predictions <- as.numeric(predictions)
    indicator <- predictions != data[get.vars(lhs(formula))]
    error <- sum(weightvector * indicator) / sum(weightvector)
    alpha <- log((1 - error) / error)
    weightvector <- weightvector * exp(alpha * indicator)
    all.pred[, m] <- alpha * predictions
```

```
    if(length(testdata) > 1) {
      predprobs <- predict(tree, newdata = testdata)
      predictions <- colnames(predprobs)[apply(predprobs, 1, FUN = function(x) {
        which(x == max(x)) })]
      all.pred.test[,m] <- alpha * as.numeric(predictions)
    }
  }
  # For any of those finals that came out in between the two classes, we randomly pick
  # which it will be
  final <- sign(rowSums(all.pred))
  final[final == 0] <- sample(c(1, -1), 1)
  if(length(testdata) > 1) {
    final2 <- sign(rowSums(all.pred.test))
    final2[final2 == 0] <- sample(c(1, -1), 1)
    return(list(predLabels.test = final2, predLabels.train = final))
  }

  return(list(predLabels = final))
}

#test <- adaBoost(X1 ~ ., data = spam, 10, 10)

N <- 100
errors.test <- rep(NA, N)
errors.train <- rep(NA, N)
  trainingsample <- sample(1:nrow(spam), 4 * nrow(spam)/5)
  spam.train <- spam[trainingsample,]
  spam.test <- spam[setdiff(1:nrow(spam), trainingsample),]
for(i in 1:N) {
  ada <- adaBoost(X1 ~ ., data = spam.train, 1, noTrees =i, testdata = spam.test)
  errors.test[i] <- sum(ada$predLabels.test != spam.test$X1)/nrow(spam.test)
  errors.train[i] <- sum(ada$predLabels.train != spam.train$X1)/nrow(spam.train)
}
```

```
## Loading required package: operator.tools
```

```
errors.test
```

```
##   [1] 0.7836957 0.5891304 0.5945652 0.5934783 0.5891304 0.5891304 0.5891304
##   [8] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [15] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [22] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [29] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [36] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [43] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [50] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [57] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [64] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [71] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [78] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [85] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [92] 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304 0.5891304
##  [99] 0.5891304 0.5891304
```

```
errors.train
```

```
##   [1] 0.6103261 0.6103261 0.6119565 0.6103261 0.6103261 0.6103261 0.6103261
##   [8] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [15] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [22] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [29] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [36] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [43] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [50] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [57] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [64] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [71] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [78] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [85] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [92] 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261 0.6103261
##  [99] 0.6103261 0.6103261
```

```
r.ada <- gbm(X1 ~ ., data = spam.train)
```

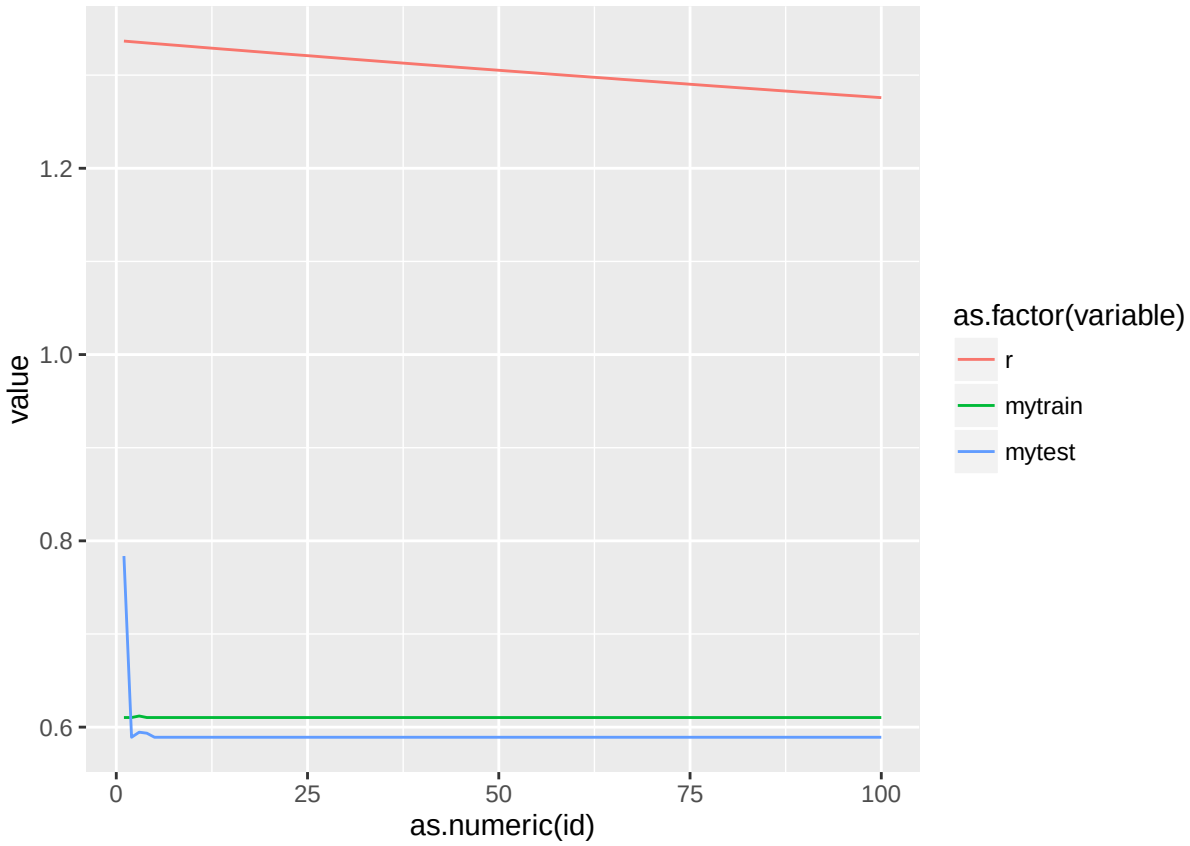
```
## Distribution not specified, assuming bernoulli ...
```

```
forplot <- data.frame(r = r.ada$train.error, id = as.factor(1:100),
                      mytrain = errors.train, mytest = errors.test)

forplot <- melt(forplot)
```

```
## Using id as id variables
```

```
ggplot(data = forplot,
       aes(x = as.numeric(id), y = value, colour = as.factor(variable))) + geom_line()
```


1.2
1.0
0.8
0.6
value
0
25
50
75
100
as.numeric(id)
as.factor(variable)
r
mytrain
mytest