Ensemble Methods and Mixture Models

Maximilian Pfundstein (maxpf364)

27 November 2018

Contents

1 Ensemble Methods

Let's load the dataset and have a look at it. The dataset is truncated to only show the last 10 columns.

Table 1: spambase.csv

Word48	Char1	Char2	Char3	Char4	Char5	Char6	Capitalrun1	Capitalrun2	Capitalrun3	Spam
0	0.00	0.000	0	0.778	0.000	0.000	3.756	61	278	1
0	0.00	0.132	0	0.372	0.180	0.048	5.114	101	1028	1
0	0.01	0.143	0	0.276	0.184	0.010	9.821	485	2259	1
0	0.00	0.137	0	0.137	0.000	0.000	3.537	40	191	1
0	0.00	0.135	0	0.135	0.000	0.000	3.537	40	191	1
0	0.00	0.223	0	0.000	0.000	0.000	3.000	15	54	1

The following source code calls the Random Forest and AdaBoost implementation and uses the predict function of each for getting the error rates for the training and the validation data set. The functions are called for 10, 20, . . . , 100 trees.

```
# General Information
c_formula = Spam ~ .
tree_sizes = seq(from = 10, to = 100, by = 10)

# Random Forest
rf_errors = data.frame(n = numeric(), error_rate_training = numeric(),
                       error_rate_validation = numeric())

for (i in tree_sizes) {

  # Create the forest
  c_randomForest =
    randomForest(formula = c_formula, data = train_spambase, ntree = i)

  # Do the prediction on the validation dataset
  c_prediction_training =
    predict(object = c_randomForest, newdata = train_spambase)
  c_prediction_validation =
    predict(object = c_randomForest, newdata = val_spambase)

  # Get the error rate
  c_error_rate_training = 1 - sum(c_prediction_training ==
                                    train_spambase$Spam)/nrow(train_spambase)
  c_error_rate_validation = 1 - sum(c_prediction_validation ==
                                      val_spambase$Spam)/nrow(val_spambase)

  rf_errors = rbind(rf_errors,
                    list(n = i,
                         error_rate_training = c_error_rate_training,
                         error_rate_validation = c_error_rate_validation))
}

# AdaBoost
adb_errors = data.frame(n = numeric(), error_rate_training = numeric(),
                        error_rate_validation = numeric())

for (i in tree_sizes) {

  # Create the model
  c_adaBoost = blackboost(formula = c_formula,
                          data = train_spambase,
                          family = AdaExp(),
                          control=boost_control(mstop=i))

  # Do the prediction on the validation dataset
  c_prediction_training =
    predict(object = c_adaBoost, newdata = train_spambase, type = "class")
  c_prediction_validation =
    predict(object = c_adaBoost, newdata = val_spambase, type = "class")

  # Get the error rate
  c_error_rate_training = 1 - sum(c_prediction_training ==
                                    train_spambase$Spam)/nrow(train_spambase)
  c_error_rate_validation = 1 - sum(c_prediction_validation ==
                                      val_spambase$Spam)/nrow(val_spambase)

  adb_errors = rbind(adb_errors,
                    list(n = i, error_rate_training = c_error_rate_training,
                         error_rate_validation = c_error_rate_validation))
}
```

The following tables show the error rates for Random Forst and AdaBoost. The plot visualizes this data, the dashed lines represent the performance on the training data set.

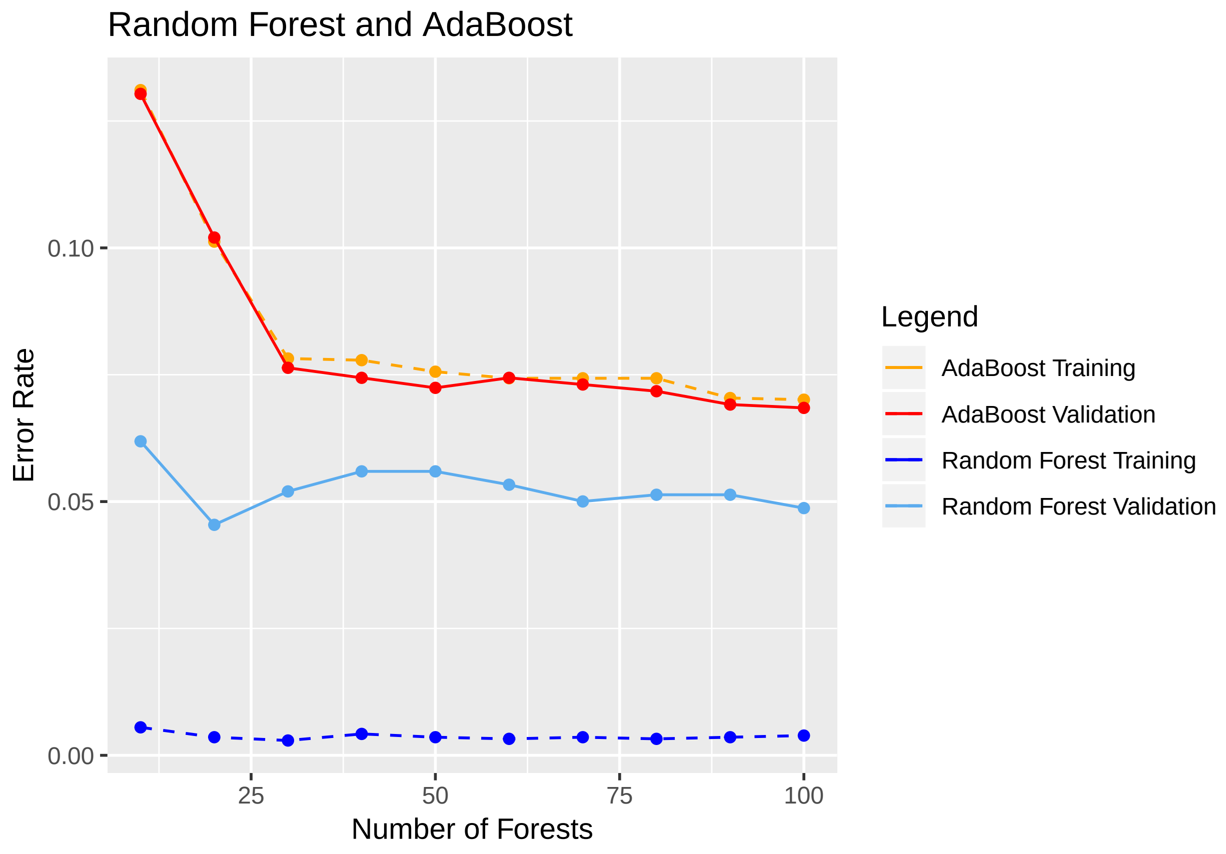
Table 2: Error rates for Random Forest

n	error_rate_training	error_rate_validation
10	0.0055159	0.0618828
20	0.0035691	0.0454246
30	0.0029202	0.0520079
40	0.0042180	0.0559579

n	error_rate_training	error_rate_validation
50	0.0035691	0.0559579
60	0.0032446	0.0533246
70	0.0035691	0.0500329
80	0.0032446	0.0513496
90	0.0035691	0.0513496
100	0.0038936	0.0487163

Table 3: Error rates for AdaBoost

n	error_rate_training	error_rate_validation
10	0.1310837	0.1303489
20	0.1012330	0.1020408
30	0.0781960	0.0763660
40	0.0778715	0.0743910
50	0.0756003	0.0724161
60	0.0743024	0.0743910
70	0.0743024	0.0730744
80	0.0743024	0.0717577
90	0.0704088	0.0691244
100	0.0700844	0.0684661



We can observe that the Random Forest is performing way better than AdaBoost. Still AdaBoost seems to perform way better on the training with respect to the validation data set than compared to the Random Forest which has a big gap between the training and validation data set. The only thing which seems to be weird is that Adaboost is actually performing better on the validation data set compared to the training data set. This behavior changes when the seed is changed, so it might be just an occasion.

2 Mixture Models

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
library(mboost)
library(randomForest)
library(ggplot2)
library(knitr)
set.seed(1234567890)
spambase = read.csv("spambase.csv", sep=";", dec = ",")
spambase$Spam = as.factor(spambase$Spam)

n = dim(spambase)[1]
id = sample(1:n, floor(n*0.67))
train_spambase = spambase[id,]
val_spambase = spambase[-id,]

kable(head(spambase[,48:58]), caption = "spambase.csv")

# General Information
c_formula = Spam ~ .
tree_sizes = seq(from = 10, to = 100, by = 10)

# Random Forest
rf_errors = data.frame(n = numeric(), error_rate_training = numeric(),
                       error_rate_validation = numeric())

for (i in tree_sizes) {

  # Create the forest
  c_randomForest =
    randomForest(formula = c_formula, data = train_spambase, ntree = i)

  # Do the prediction on the validation dataset
  c_prediction_training =
    predict(object = c_randomForest, newdata = train_spambase)
  c_prediction_validation =
    predict(object = c_randomForest, newdata = val_spambase)

  # Get the error rate
  c_error_rate_training = 1 - sum(c_prediction_training ==
                                    train_spambase$Spam)/nrow(train_spambase)
  c_error_rate_validation = 1 - sum(c_prediction_validation ==
                                      val_spambase$Spam)/nrow(val_spambase)
  rf_errors = rbind(rf_errors,
                    list(n = i,
                         error_rate_training = c_error_rate_training,
                         error_rate_validation = c_error_rate_validation))
}

# AdaBoost
adb_errors = data.frame(n = numeric(), error_rate_training = numeric(),
                       error_rate_validation = numeric())

for (i in tree_sizes) {

  # Create the model
  c_adaBoost = blackboost(formula = c_formula,
                          data = train_spambase,
                          family = AdaExp(),
                          control=boost_control(mstop=i))

  # Do the prediction on the validation dataset
  c_prediction_training =
    predict(object = c_adaBoost, newdata = train_spambase, type = "class")
  c_prediction_validation =
    predict(object = c_adaBoost, newdata = val_spambase, type = "class")

  # Get the error rate
  c_error_rate_training = 1 - sum(c_prediction_training ==
                                    train_spambase$Spam)/nrow(train_spambase)
  c_error_rate_validation = 1 - sum(c_prediction_validation ==
                                      val_spambase$Spam)/nrow(val_spambase)

  adb_errors = rbind(adb_errors,
                    list(n = i, error_rate_training = c_error_rate_training,
                         error_rate_validation = c_error_rate_validation))
}


kable(rf_errors, caption = "Error rates for Random Forest")
kable(adb_errors, caption = "Error rates for AdaBoost")

ggplot(adb_errors) +
  geom_line(aes(x = n, y = error_rate_training,
                colour = "AdaBoost Training"), linetype = "dashed") +
  geom_point(aes(x = n, y = error_rate_training), colour = "orange") +

  geom_line(aes(x = n, y = error_rate_validation,
                colour = "AdaBoost Validation")) +
  geom_point(aes(x = n, y = error_rate_validation), colour = "red") +

  geom_line(aes(x = n, y = error_rate_training,
                colour = "Random Forest Training"),
            data = rf_errors, linetype = "dashed") +
  geom_point(aes(x = n, y = error_rate_training),
             colour = "blue", data = rf_errors) +
  geom_line(aes(x = n, y = error_rate_validation,
                colour = "Random Forest Validation"), data = rf_errors) +
  geom_point(aes(x = n, y = error_rate_validation),
             colour = "steelblue2", data = rf_errors) +
  labs(title = "Random Forest and AdaBoost", y = "Error Rate",
       x = "Number of Forests", color = "Legend") +
  scale_color_manual(values = c("orange", "red", "blue", "steelblue2"))
```

Bibliography