Predicting the Manner of Activity

Araks Stepanyan

8/29/2017

Summary

The goal of this project is to predict the manner in which 6 individuals perform barbell lift. They did the lifts correctly and incorrectly in 5 different ways. We are going to use data from sensors in the users' glove, armband, lumbar belt and dumbbell. The data for this project come from this source(find paper in Reference): http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

For computational efficiency we use H2O package in R. It's free to use and instigates faster computation (see Reference).

We divided our data into training, validation and testing sets. We used training data to fit two random forest and two boosting models. Validation test was used to compare classification errors and select the best model. The second boosing model with a little parameter tuning was selected as the best model. Testing data was then used to predict and check the accuracy. The accuracy on testing data was pretty close to the accuracy on validation data, indicating that we didn't overfit the data.

Note. You can skip the first two parts and start viewing from the **Model Selection**. Before model selection we are just loading data and deleting the columns which were derived from the rest of the columns, as well as those columns which are id or time.

Download and Read Data

```
if(!file.exists("Machine_Learning_Progect")){dir.create("Mach_Learning_Project_Data")}
```

```
## Warning in dir.create("Mach_Learning_Project_Data"):
## 'Mach_Learning_Project_Data' already exists
```

```
trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(trainUrl,
              destfile = "./Mach_Learning_Project_Data/train_data.csv",
              method = "curl")

testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(testUrl,
              destfile = "./Mach_Learning_Project_Data/test_data.csv",
              method = "curl")
```

```
library("data.table")
options(datatable.fread.datatable=FALSE)
training <- fread("./Mach_Learning_Project_Data/train_data.csv", stringsAsFactors = T)
testing <- fread("./Mach_Learning_Project_Data/test_data.csv", stringsAsFactors = T)
```

Exploratory Analysis

Look into training data.

```
dim(training)
```

```
## [1] 19622   160
```

```
sum(is.na(training))
```

```
## [1] 1287472
```

We have 19622 training examples and a huge amount of missing values, 1287472.

Missing Values

By uncommenting the third line of code bellow, we see the names of the columns containing NA values. It turns out that those columns are all derived features. From the corresponding paper(in Reference), we know that there are 96 derived features*.

```
na_columns <- names(training[,colSums(is.na(training)) > 0])
length(na_columns)
```

```
## [1] 67
```

```
#na_columns
```

Looks like of 96 derived features, 67 contain NA values.
* The paper says, *"In each step of the sliding window approach we calculated features on the Euler angles (**roll**, **pitch** and **yaw**), as well as the raw **accelerometer**, **gyroscope** and **magnetometer** readings. For the Euler angles of each of the four sensors we calculated eight features: **mean, variance, standard deviation, max, min, amplitude, kurtosis and skewness**, generating in total 96 derived feature sets."*

Omitting variables

We will omit the 96 columns discussed above.

```
columns_to_remove_1 <-
        grep("avg|var|stddev|max|min|amplitude|kurtosis|skewness",
             names(training))
length(columns_to_remove_1)
```

```
## [1] 100
```

Notice that the number of columns that contain the keywords **"avg", "var", "stddev", "max", "min", "amplitude", "kurtosis", "skewness"** is actually 100 instead of expected 96. From these 100 columns let's have a look at those not containings words **roll**, **pitch** and **yaw**.

```
columns_to_remove_2 <- !grepl("roll|pitch|yaw",
                              names(training[,columns_to_remove_1]))
names(training[,columns_to_remove_1][,columns_to_remove_2])
```

```
##  [1] "kurtosis_picth_belt"     "max_picth_belt"
##  [3] "var_total_accel_belt"    "var_accel_arm"
##  [5] "kurtosis_picth_arm"      "max_picth_arm"
##  [7] "kurtosis_picth_dumbbell" "max_picth_dumbbell"
##  [9] "var_accel_dumbbell"      "kurtosis_picth_forearm"
## [11] "max_picth_forearm"       "var_accel_forearm"
```

We see
1. *"pitch"* is incorrectly entered as *"picth"* in 8 columns. We still want to remove those.

2. The columns not containing either of **roll, pitch** or **yaw** are again derived quantities. We will remove those as well.

We will also omit the id and time variables, as we don't want to predict the manner of the exercise based on who did it or the timestep.

```
training_1 <- training[,!names(training) %in% c("V1","user_name", "raw_timestamp_part_1", "raw_timestamp
dim(training_1)
```

```
## [1] 19622    53
```

Final check to see that we don't have any NAs.

```
#names(training_1)
sum(is.na(training_1))
```

```
## [1] 0
```

Model Selection

H2O

Install and start H2o.

```
training.h2o <- as.h2o(training_1)     # convert data frame into h2o Frame
splits <- h2o.splitFrame(training.h2o,
                         c(0.6, 0.2), seed = 1234 )     # split for cross validation
# assign a h2o id names to the splits
train <- h2o.assign( splits[[1]], "train" )
valid <- h2o.assign( splits[[2]], "valid" )
test  <- h2o.assign( splits[[3]], "test" )

dependant <- 53                          # index of variable classe
independant <- c(1:52)                   # indices of predictors
```

Random Forest 1

With default number of trees of 50 and default maximum depth of 20.

```
rf1 <- h2o.randomForest(
        training_frame = train,
        validation_frame = valid,
        x = independant,
        y = dependant,
        model_id = "rf_classe_v1",
        stopping_rounds = 2,
        score_each_iteration = T,
        seed = 1000000
)
```

Performance

```
## [1] 0.9889392
```

```
## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
##           A   B   C   D   E  Error         Rate
```

```
## A      1116   1   0   0   0 0.0009 =  1 / 1,117
## B         3 777   2   0   0 0.0064 =    5 / 782
## C         1   8 706   1   0 0.0140 =   10 / 716
## D         3   1  16 610   1 0.0333 =   21 / 631
## E         0   0   4   3 725 0.0096 =    7 / 732
## Totals 1123 787 728 614 726 0.0111 = 44 / 3,978
```

```
## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
##           A    B    C    D    E  Error           Rate
## A      3256   33   13   32    3 0.0243 =   81 / 3,337
## B        28 2117   49   41   19 0.0608 =  137 / 2,254
## C         4   61 1899   36    8 0.0543 =  109 / 2,008
## D         3   20   49 1866    8 0.0411 =   80 / 1,946
## E         2    3   12   15 2122 0.0149 =   32 / 2,154
## Totals 3293 2234 2022 1990 2160 0.0375 = 439 / 11,699
```

- **Accuracy** of the model on validation data is 0.9889.
- **Classification error** of validation data is 0.0111.

- **Out of bag error** is 0.0375.

These results are already quite good but we will improve them farther.
(For the rest of the models we will not print the whole Confusion matrix)

Random Forest 2

Increase number of trees and maximum depth.

```
rf2 <- h2o.randomForest(
    training_frame = train,
    validation_frame = valid,
    x = independant,
    y = dependant,
    model_id = "rf_classe_v2",
    ntrees = 200,
    max_depth = 30,
    stopping_rounds = 2,
    score_each_iteration = TRUE,
    seed = 3000000
)
```

Performance of the two random forest models

```
## [1] "Random Forest (model 1). Accuracy: 0.9889 , Classification Error: 0.0111 , Out of Bag Error: 0.0
```

```
## [1] "Random Forest (model 2). Accuracy: 0.9894 , Classification Error: 0.0106 , Out of Bag Error: 0.0
```

We see improvements. Let's go farther with boosting.

Gradiant Boosting 1

With defaults: ntrees = 50, max_depth = 5, learn_rate = 0.01.

```
gbm1 <- h2o.gbm(
    training_frame = train,
    validation_frame = valid,
```

```
    x = independant,
    y = dependant,
    model_id = "gbm_classe_v1",
    seed = 2000000)
```

Performance

```
## [1] "Reported on validation data"
```

```
## [1] "Accuracy: 0.9816"
```

```
## [1] "Classification Error: 0.0184"
```

With defaults boosting performs even worse than our first random forest model. Let's improve it.

Gradiant Boosting 2

1. learn_rate _ Increase the learning rate.

2. max_depth _ Adding depth makes each tree fit the data closer.

```
gbm2 <- h2o.gbm(
    training_frame = train,
    validation_frame = valid,
    x = independant,
    y = dependant,
    learn_rate = 0.3, # increase the learning rate
    max_depth = 10,
#   sample_rate = 0.7, # use a random 70% of the rows to fit each tree
#   col_sample_rate = 0.7, # use 70% of the columns to fit each tree
    stopping_rounds = 2,
    stopping_tolerance = 0.01,
    model_id = "gbm_classe_v2",
    seed = 2000000
)
```

Performance

```
## [1] "Reported on validation data"
```

```
## [1] "Accuracy: 0.9955"
```

```
## [1] "Classification Error: 0.0045"
```

This is an excellent result. We will stick to this second boosting model. (By uncommenting the two commented lines in above model, the results will improve even more. Those two lines are adding some of the nature of random forest into the GBM).

Prediction

We will predict the classe variable using the second boosting model.

```
gbm2_pred <- h2o.predict(gbm2, newdata = test)
paste("Test Set Accuracy:", round(mean(gbm2_pred$predict == test$classe),4))
```

```
## [1] "Test Set Accuracy: 0.9953"
```

Expected Out of Sample Error

Expected out of sample error is (1 - accuracy(test)). So we expect an error equal to 0.0047.

Prediction on Unseen Data

We also have testing set that we didn't touch so far (for this set we don't know the real classes).

```
testing.h2o <- as.h2o(testing)
gbm2_pred_final <- h2o.predict(gbm2, newdata = testing.h2o)
gbm2_pred_final$predict
```

```
##   predict
## 1       B
## 2       A
## 3       B
## 4       A
## 5       A
## 6       E
##
## [20 rows x 1 column]
```

Shut down H2o

```
#h2o.shutdown(prompt = FALSE)              # shut down h2o instance
```

Reference

1. Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013
2. H2o Ensemble Tree
3. Use H2O and data.table to build models on large data sets in R