Prediction Assignment Writeup

Alexander Lifshitz

January 31, 2016

Summary

This works presents predictive analysis of Weight Lifting Exercises(WLE) Dataset (http://groupware.les.inf.puc-rio.br/har). The dataset includes data collected by wearable sensors of specific dumbbell exercise performed in five different fashions. Class A corresponds to the specified execution of the exercise, while the other 4 classes (B,C,D,E) correspond to common mistakes. In this work we construct a Random Forest classifier which is trained on a training set and then used to classify data given in the testing set. To estimate the performance of the classifier we use a separate validation set which was constructed by subsetting initial training set (and of course was not used for training). The constructed Random Forest classifier achieves extremely high overall accuracy of 99.64% (on a validation set) and correctly classifies all 20 samples of the test dataset.

Loading and cleaning data

First we shall remove columns that contain mostly NA values.

```
# Loading training data set and removing irrelevant columns
cwd<-getwd()
training.all <- read.csv(file.path(cwd, "pml-training.csv", fsep = .Platform$file.sep), n
a.strings = c("NA", "#DIV/0!"))
dim(training.all)
```

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

```
ind_col <- colSums(is.na(training.all))<19000
training.all <-training.all[,ind_col]
dim(training.all)
```

```
## [1] 19622    60
```

We confirm that the remaining dataset does not contain any NA values.

```
table(complete.cases(training.all))
```

```
##
##  TRUE
## 19622
```

Then the first 7 columns irrelevant for the classification purposes are removed as well.

```
training.all <-training.all[,-c(1:7)]
names(training.all)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

The remaing dataset includes 53 features corresponding to 13 measurements of 4 sensors (belt, arm, dumbbell, forearm) plus a coulmn ‘classe’ which includes data labels which we wil luse for training.

Now we repeat the same data cleaning process for the test dataset

```
# Loading the testing data set and removing irrelevant columns
testing <- read.csv(file.path(cwd, "pml-testing.csv", fsep = .Platform$file.sep),na.strin
gs = c("NA", "#DIV/0!"))
testing <- testing[,ind_col]
testing <-testing[,-c(1:7)]
```

Splitting a training set into training and validation subsets

In order to validate the prediction model we will split the training set into two unequal subsets: 70% will be used for the actual training and 30% for validation.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
# Spliting training set to 70% training and 30% validation sets
set.seed(3141)
inTrain <- createDataPartition(y=training.all$classe, p=0.7, list=FALSE)

training <- training.all[inTrain,]
validation <- training.all[-inTrain,]
```

Constructing a Random Forest Classifier

We suggest to construct Random Forest classifier which constructs multiple independent decision trees (we will use 300) and determines the classification output by using a majority rule.

```
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
fit_rf = randomForest(classe~., data=training, ntree=300,proximity=TRUE)
```

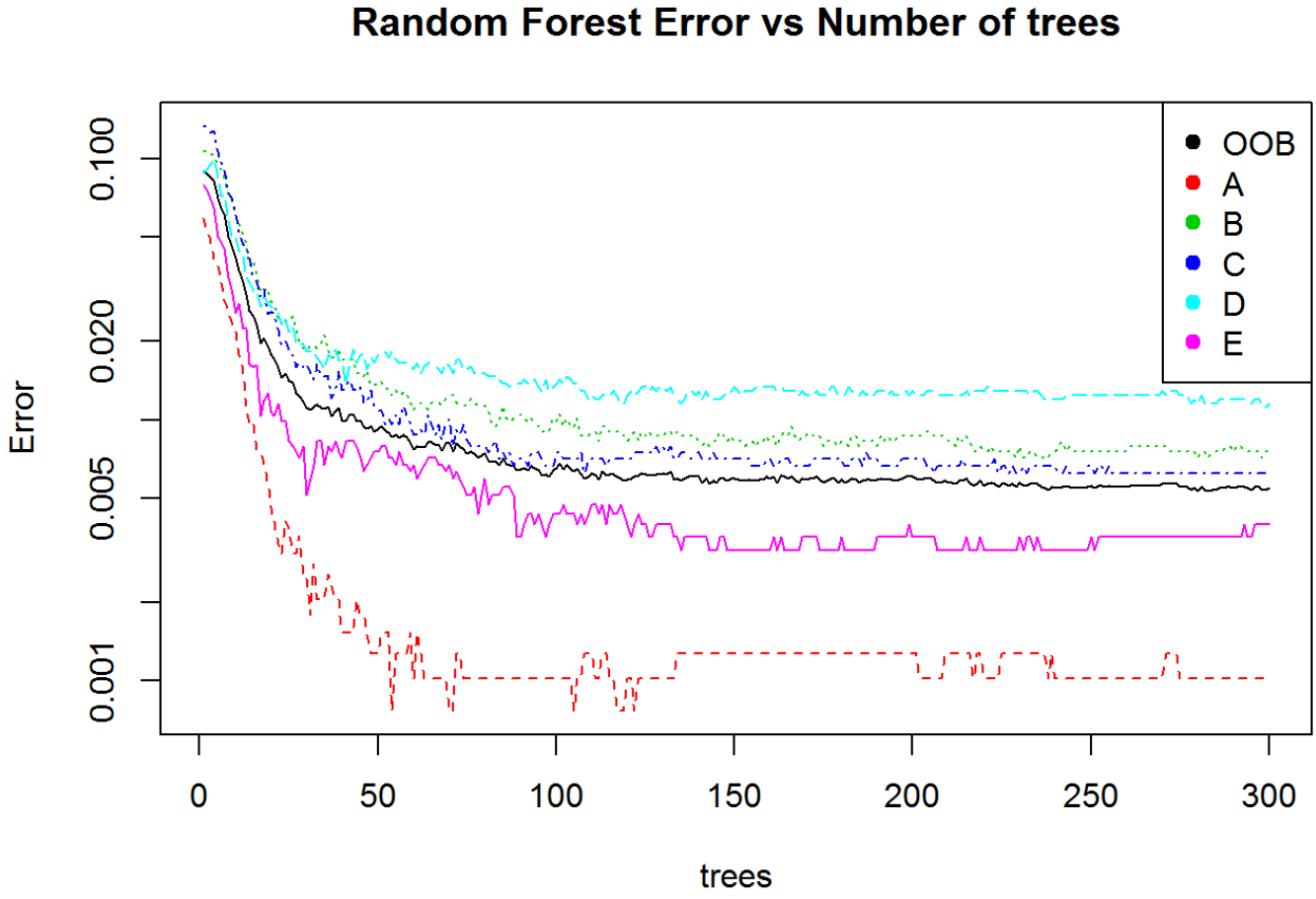
For the training set the total Out Of Bag (OOB) error is about 0.55%, which implies very high accuracy.

```
tail(fit_rf$err.rate,1)
```

```
##                OOB           A           B           C          D
## [300,] 0.005459707 0.001024066 0.007524454 0.006260434 0.01154529
##                   E
## [300,] 0.003960396
```

We can also plot model error rate vs number of trees.

```
#Plotting model error vs number of trees
plot(fit_rf, log="y",main = "Random Forest Error vs Number of trees")
legend("topright", legend=c("OOB", as.character(unique(training$classe))), col=1:6, pch=1
9)
```



Even though we used 300 trees in our model, from the error graph above we can conclude that the algorithm achieves optimal performance after growing about 100-150 trees.

Let us now test our model on a validation set.

```
p_rf <- predict(fit_rf,validation)
confusionMatrix(p_rf,validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1135    6    0    0
##          C    0    0 1020   10    0
##          D    0    0    0  953    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9964
##                  95% CI : (0.9946, 0.9978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9942   0.9886   1.0000
## Specificity            0.9991   0.9987   0.9979   1.0000   0.9998
## Pos Pred Value         0.9976   0.9947   0.9903   1.0000   0.9991
## Neg Pred Value         1.0000   0.9992   0.9988   0.9978   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1733   0.1619   0.1839
## Detection Prevalence   0.2851   0.1939   0.1750   0.1619   0.1840
## Balanced Accuracy      0.9995   0.9976   0.9960   0.9943   0.9999
```

The confusion matrix indicates that the obtained classifier performs extremely well on a validation set with overall accuracy of 99.64%.

Predicting outcomes for test dataset

Applying the classifier to the 20 samples of the test set yields

```
predict_testing<-predict(fit_rf,testing)
print(predict_testing)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

These results are confirmed as 100% correct (via Project Prediction Quiz).