Automatic Qualitative Recognition of Physical Exercises

Renato P. dos Santos

August 9, 2016

Introduction

It is now possible to collect a large amount of data about personal activity relatively inexpensively using monitoring devices such as a *Fitbit*, *Nike Fuelband*, or *Jawbone Up*. These type of devices are part of the *quantified self* movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. There are opportunities here for Big Data scientists to develop new models to support QS data collection, integration, and analysis (Swan, 2013).

However, these data remain under-utilized both because the raw data are usually not freely shared, and there is still some uncertainty about the best statistical methods and software for processing and interpreting the data. Furthermore, even if people regularly quantify **how much** of a particular activity they do, they rarely quantify **how well** they do it.

Velloso et al. (2013) investigated the automatic and robust qualitative recognition of execution mistakes, of “how well” an activity was performed by the wearer. That investigation potentially provides useful information for a large variety of applications, including digital assistants for physical exercises.

The goal of this project was to apply Machine Learning to the *Weight Lifting Exercises Dataset* to predict whether participants performed barbell lifts correctly or incorrectly.

Methods

Six male healthy participants aged between 20-28 years, with little weight lifting experience, were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: Class A, corresponding exactly to the specified execution of the exercise, and 4 other classes corresponding to the common mistakes of throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D), and throwing the hips to the front (Class E).

Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate.

Data was collected from accelerometers on the belt, forearm, arm, and dumbbells of 6 participants

More information is available on the Human Activity Recognition Project webpage (http://groupware.les.inf.puc-rio.br /har#weight_lifting_exercises) and in the paper of Velloso et al. (2013).

The data for this project already partitioned as training and testing datasets are available here (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) and here (https://d396qusza40orc.cloudfront.net /predmachlearn/pml-testing.csv), respectively.

This work was done with the resources of the R statistical data analysis language (R Core Team, 2016) R version 3.3.1 (2016-06-21) on Windows 7 (build 7600). The needed packages were loaded as below:

```
suppressMessages(library(caret, quietly = TRUE, verbose = FALSE))
suppressMessages(library(randomForest, quietly = TRUE, verbose = FALSE))
suppressMessages(library(kernlab, quietly = TRUE, verbose = FALSE))
suppressMessages(library(gbm, quietly = TRUE, verbose = FALSE))
suppressMessages(library(klaR, quietly = TRUE, verbose = FALSE))
suppressMessages(library(knitr, quietly = TRUE, verbose = FALSE))
```

Data reading and preparation

```
training <- read.csv("pml-training.csv",
                     na.strings=c("NA","#DIV/0!",""))
testing <- read.csv("pml-testing.csv",
                    na.strings=c("NA","#DIV/0!",""))
save.image(file = ".RData")
```

The training and testing datasets contain 160 variables each; the training dataset contains 19622 observations, while the testing one contains only 20.

As the data was provided already partitioned in training and testing datasets, it is advisable to do some checking.

Firstly, let us check if the variable names are the same in both datasets.

```
nameMatch <- all.equal(colnames(testing)[1:ncol(testing)],
                       colnames(training)[1:ncol(training)])
if (is.character(nameMatch)) {
        mismatch <- colnames(testing)[1:ncol(testing)] != colnames(training)[1:ncol(training
)]
        print(paste0("There is a mismatch at variable #", which(mismatch), ": '", names(test
ing)[mismatch], "' (testing) vs. '", names(training)[mismatch], "' (training)."))
} else print("All variable names are the same.")
```

```
## [1] "There is a mismatch at variable #160: 'problem_id' (testing) vs. 'classe' (training)
."
```

Notice, however, that the *classe* variable in the training set is the one that classifies the manner in which the participants did the exercise, containing the values A, B, C, D, or E, as explained above. Consequently, we will take special care of it, during the analysis.

Before any further processing, it is advisable to check if the dataset needs some data cleaning, which is considered an essential part of the statistical analysis (de Jonge; van der Loo, 2013).

To begin with, variables related to data acquisition, such as observation id, timestamps, and participants' names, which are not relevant for prediction, were removed.

```
training <- training[, -(1:6)]
testing <- testing[, -(1:6)]
```

Now, it is known that missing values are a problem that plagues any data analysis, their presence introducing bias into some calculations or summaries of the data (Peng, 2016, p. 134). Therefore, variables with more than 50% missing values (coded as *NA*) were also removed.

```
NAIndex <- sapply(colnames(training),
                  function(x) sum(is.na(training[, x])) > 0.5*nrow(training))
training <- training[, !NAIndex]
testing <- testing[, !NAIndex]
```

Next, we checked for variables with near-zero variances.

```
NZVIndex <- nearZeroVar(training, saveMetrics = FALSE,
                        names = FALSE,
                        foreach = TRUE,
                        allowParallel = TRUE)
if (sum(NZVIndex) > 0) {
        training <- training[, !NZVIndex]
        testing <- testing[, !NZVIndex]
}
```

However, this test resulted in 0 near-zero variances and, therefore, no variables were further removed.

Finally, we searched for highly-correlated variables. The usual PCA (Principal Components) approach could make it harder to interpret the predictors and, therefore, we chose instead to simply remove variables which were correlated by more than 80%.

```
corIndex <- findCorrelation(cor(training[,-54]), cutoff=0.8)
training <- training[, -corIndex]
testing <- testing[, -corIndex]
save.image(file = ".RData")
```

With all this cleaning, training and testing datasets were reduced to 41 variables. Now, we can proceed to analysis.

Analysis

When coming to the choice of the machine learning algorithm to use, the *train* function of the *caret* package allows for any of 227 different classification or regression models.

The following three models were chosen for no other reason as they having been mentioned in the Course: (radial) Support Vector Machine, Random Forest (rf), and Linear Discriminant Analysis (lda).

In order to assure reproducibility, an overall pseudo-random number generator seed was set for all code.

```
set.seed(42)
```

For cross-validation, the training data was further partitioned by random sampling within the levels of the *classe* variable, in an attempt to balance the class distributions within the splits, into two data subsets, one (70%) for training the candidate models and one (30%) for later evaluating (pre-testing) the performance of the selected model.

```
splIndex <- createDataPartition(y = training$classe, p = 0.7, list = FALSE)
myTraining <- training[splIndex,]
myTesting <- training[-splIndex,]
```

In order to avoid overfitting and to reduce out of sample errors, the *TrainControl* function was used to perform a 7-fold cross validation.

```
trCont <- trainControl(method = "cv", number = 7, verboseIter=FALSE)
```

Then, we proceed to apply the selected models to the *myTraining* dataset.

```
svmr_model <- train(classe ~ ., data = myTraining, method = "svmRadial", trControl= trCont,
verbose=FALSE)
rf_model <- train(classe ~ ., data = myTraining, method = "rf", trControl= trCont, verbose=F
ALSE)
lda_model <- train(classe ~ ., data = myTraining, method = "lda", trControl= trCont, verbose
=FALSE)
```

```
Model <- c("SVMR", "RF", "LDA", "NB")
Accuracy <- c(max(svmr_model$results$Accuracy),
              max(rf_model$results$Accuracy),
              max(lda_model$results$Accuracy))

Kappa <- c(max(svmr_model$results$Kappa),
           max(rf_model$results$Kappa),
           max(lda_model$results$Kappa))

comparison <- cbind(Model,Accuracy,Kappa)
```

```
## Warning in cbind(Model, Accuracy, Kappa): number of rows of result is not a
## multiple of vector length (arg 2)
```

```
kable(comparison, caption = "Table 1 - Comparison between models")
```

Table 1 - Comparison between models

Model	Accuracy	Kappa
SVMR	0.924219097770943	0.904017399608112
RF	0.996505987782484	0.995580389942636
LDA	0.656984018263926	0.566092595661081
NB	0.924219097770943	0.904017399608112

From Table 1, we see that Random Forest (RF) provided the best results and, therefore, we chose it for the predictions on the *testing* dataset.

Before that, however, we use the *confusion matrix* to pre-test its prediction performance on the *myTesting* subset.

```
rf_pred <- predict(rf_model, myTesting)
confMx <- confusionMatrix(rf_pred, myTesting$classe)
unname(confMx[2])
```

```
## [[1]]
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    3    0    0    0
##          B    1 1135    0    0    0
##          C    0    1 1025    5    0
##          D    0    0    1  959    0
##          E    0    0    0    0 1082
```

The confusion matrix is clearly diagonal, and the performance measures below are also impressive.

```
unname(confMx[3])
```

```
## [[1]]
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9981308      0.9976357      0.9966580      0.9990666      0.2844520
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

More importantly, the accuracy obtained in the *myTesting* subset (0.9981308) is comparable with the one obtained with the

myTraining dataset (0.996506), what suggests that an overfitting is not happening and, therefore, that the out-of-the-sample error is small, being the in-sample error even smaller.

With such good results, we proceed to generate the file with the predictions to the *testing* dataset to submit for the assignment.

```
rf_pred2 <- predict(rf_model, testing)
pml_write_files = function(x){
        n = length(x)
        for(i in 1:n){
                filename = paste0("problem_id_",i,".txt")
                write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
        }
}

pml_write_files(rf_pred2)
```

Conclusions

The Random Forest model surpassed the other candidate models and exhibited an outstanding accuracy, providing predictions for the *myTesting* subset correct in 100% of the cases. It is not surprising that it is one of the most common models that win Kaggle and other prediction contests.

References

- de Jonge, E.; van der Loo, M. (2013). *An introduction to data cleaning with R*. The Hague: Statistics Netherlands.
- Peng, R. D. (2016). *Exploratory Data Analysis with R*. Victoria, CA-BC: Leanpub.
- R Core Team (2016). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. Available at R site: https://www.R-project.org/ (https://www.R-project.org/).
- Swan, M. (2013). The Quantified Self: Fundamental Disruption in Big Data Science and Biological Discovery. *Big Data*, *1*(2): 85-99. http://doi.org/10.1089/big.2012.0002 (http://doi.org/10.1089/big.2012.0002)
- Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. *Proceedings of 4th International Conference on Cooperation with SIGCHI* (Augmented Human '13). Stuttgart, Germany: ACM SIGCHI, 2013.