Machine Learning Project

Carlos Gutierrez Sanchez del Rio

18 de julio de 2016

Summary

The goal of this project is to predict the manner in which a set of participants performed a barbell lifting exercise (correctly and incorrectly in 5 different ways) using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The "classe" variable is the outcome to predict:

- A: according to specification
- B: elbows to the front
- C: lifting halfway
- D: lowering halfway
- E: hips to the front

Given that this is a classification problem we will try different tree models to find the best fit (using rpart, random forest...). The test data file doesn't have a classe column but a problem_id column that is to be predicted according to the model fitted using the training data.

Download and Clean the Data

We will first download and clean the data. Missing values are coded as "NA", "#DIV/0!" or "".

```
trainraw <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", na.strings=

testraw <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", na.strings=c
```

Several columns are full of missing values (not imputable) and others are non-relevant for prediction (e.g. user_name, raw_timestamp_part_1, raw_timestamp_part_,2 cvtd_timestamp, new_window, and num_window).

By eliminating these columns we reduce the variables from 160 to 53

No columns were identified as having Near Zero Variance by using the **nearZeroVar(train)** function.

```
# Delete columns with missing values
train<-trainraw[,colSums(is.na(trainraw)) == 0]
testfinal <-testraw[,colSums(is.na(testraw)) == 0]
# Delete first 7 columns
train   <-train[,-c(1:7)]
testfinal <-testfinal[,-c(1:7)]
```

Separating the train data into train and validation

We separate the train data into two groups. The training group will be used to fit the models and the validation group to estimate out-of-sample error:

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
set.seed(9)
inTrain <- createDataPartition(train$classe,p=0.7,list=FALSE)
training <- train[inTrain,]
validation <- train[-inTrain,]
dim(training)
```

```
## [1] 13737    53
```

```
dim(validation)
```

```
## [1] 5885   53
```

Fitting classification tree models

We will try fitting two popular classification tree models and compare the accuracy of each one. Even though the number of variables is significant (53) we we have 13737 observations to fit the models (warning: it takes a while!!).

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
# Fit randomForest (avoid using the train function for rf, takes too long)
modelRF <- randomForest(classe ~ ., data = training, importance = TRUE, ntrees = 20)

# Fit classification tree
modelRpart <- train(classe ~.,data=training,method="rpart")
```

We can now compare the results on the training data and the validation data:

```
RF_training <- predict(modelRF, training)
confusionMatrix(RF_training, training$classe)$overall['Accuracy']
```

```
## Accuracy
##        1
```

```
Rpart_training <- predict(modelRpart, training)
confusionMatrix(Rpart_training, training$classe)$overall['Accuracy']
```

```
##  Accuracy
## 0.4953774
```

We see that the Random Forest model, even limiting the number of trees to try, has almost perfect accuracy (under 0.6% error rate), while the classification tree has a low accuracy. We could try further models (Bayes LogitBoost...) but the RandomForest seems to do a good job (no need for a combined model either)

By looking at the confusion matrix we see an almost perfect classification:

```
modelRF
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, importance = TRUE,      ntrees = 20)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.56%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    4    0    0    0 0.001024066
## B   15 2638    5    0    0 0.007524454
## C    0   17 2376    3    0 0.008347245
## D    0    0   22 2230    0 0.009769094
## E    0    0    2    9 2514 0.004356436
```

Estimate out-of-sample error

Using the validation data we can estimate the out of sample error (under 0.5%)

```
RF_validation <- predict(modelRF, validation)
confusionMatrix(RF_validation, validation$classe)$overall['Accuracy']
```

```
##  Accuracy
## 0.9959218
```