Practical Machine Learning - Prediction Assignment

Martina Hoever

August 7th - 2016

Predicting exercise 'classe' using machine learning

1. Overview

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. 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. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

This report focuses on creating a prediction model to predict the manner or 'classe' in which our subjects did the exercise.

2. Exploratory data analysis

2.1 Load the data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv.

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv.

The test data set will be used for validation of the chosen model, and the training set will be split up into training and test sets. To make cross validation possible, 3 quarters of the training data will be used for training of the model, and 1 quarter of the data will wbe used for testing.Using the caret package in R we can use cross validation for our predictions.

```
## Load the caret package
library(caret)

## Load the training and testing data
training_url <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testing_url <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(training_url))
validation <- read.csv(url(testing_url))

## To make sure the results are reproducible, a seed is set
set.seed(12345)

## The training data is split up in a training set and a test set
inTrain <- createDataPartition(training$classe, p=3/4, list=F)
training <- training[inTrain,]
testing <- training[-inTrain,]
```

2.2 Clean the data

We will remove all the variables that have to many NA values. We will remove all variables that have more than 10% NA values. Next to this, all variables with near zero variance will be removed from the training set. Then we will check for all the numeric variables if the variables are highly correlated with each other, because then it might not be useful to include all these variables in the model.

```
## Load the caret package
training <- training[, -which(colMeans(is.na(training)) > 0.1)]

## Remove the columns with near zero variance
training <- training[, -nearZeroVar(training)]

## Check the correlation between all numeric variables
nums <- training[sapply(training,is.numeric)]
cm <- abs(cor(nums))
## Count how many variables have a correlation higher than 0.9
nrow(which(cm> 0.9, arr.ind = T))
```

```
## [1] 78
```

We see that there are 78 variables highly correlated. Therefore we will use principle component analysis (PCA) when we build our prediction models. This will be done by setting preprocess to "pca". To see if this is beneficial we will create one prediction model with and without PCA.

3. Prediction model

We will use three different methods to create a prediction model:

- **Rpart** without principle component analysis
- **Rpart** with principle component analysis
- **Rf** with principle component analysis
- **Gbm** with principle component analysis

```
## For performance reasons we allow parallel processing
library(parallel)
library(doParallel)
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

fitControl <- trainControl(method = "cv", number = 10, allowParallel = TRUE)

## Train the four models we have chosen
mod_rp_nopca <- train(
    classe ~ ., data = training,
    method = "rpart", trControl = fitControl)
mod_rp <- train(
    classe ~ ., data = training, preProcess= "pca",
    method = "rpart", trControl = fitControl)
mod_rf <- train(
    classe ~ ., data = training, preProcess= "pca",
    method = "rf", trControl = fitControl)
mod_gbm <- train(
```

```
    classe ~ ., data = training, preProcess= "pca",
    method = "gbm", trControl = fitControl)

pred_rp_nopca <- predict(mod_rp_nopca, testing)
pred_rp <- predict(mod_rp, testing)
pred_rf <- predict(mod_rf, testing)
pred_gbm <- predict(mod_gbm, testing)
```

We would also like to see if combining these models would result in a better prediction model. Therefore we create a new prediction model that uses the previously created models.

```
## Train the combined model
predDF <- data.frame(pred_rp_nopca, pred_rp, pred_rf,pred_gbm,classe=testing$classe)
combModFit <- train(classe ~., method = "rf", data =predDF, trControl = fitControl)
combPred <- predict(combModFit,predDF)

## Stop the parallel processing
stopCluster(cluster)
```

4. Model assessment

To evaluate the models that we have created we will look at the confusion matrices and the accuracy of the different models.

```
confusionMatrix(pred_rp_nopca, testing$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1029    0    0    0    0
##          B    0  717    0    0    0
##          C    0    0    0    0    0
##          D    0    0    0    0    0
##          E    0    0  652  594  691
```

```
confusionMatrix(pred_rp, testing$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1029  717  652  594  691
##          B    0    0    0    0    0
##          C    0    0    0    0    0
##          D    0    0    0    0    0
##          E    0    0    0    0    0
```

```
confusionMatrix(pred_rf, testing$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1029    0    0    0    0
##          B    0  717    0    0    0
```

```
##          C    0    0  652    0    0
##          D    0    0    0  594    0
##          E    0    0    0    0  691
```

```
confusionMatrix(pred_gbm, testing$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1014   20    0    0    0
##          B   10  670   32    2    0
##          C    5   26  615   22    2
##          D    0    1    4  567    7
##          E    0    0    1    3  682
```

```
confusionMatrix(combPred, testing$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1029    0    0    0    0
##          B    0  717    0    0    0
##          C    0    0  652    0    0
##          D    0    0    0  594    0
##          E    0    0    0    0  691
```

```
AccuracyResults <- data.frame(
  Model = c('Rpart_nopca', 'Rpart',  'RF', 'GBM', 'Combined'),
  Accuracy = rbind(
    confusionMatrix(pred_rp_nopca, testing$classe)$overall[1],
    confusionMatrix(pred_rp, testing$classe)$overall[1],
    confusionMatrix(pred_rf, testing$classe)$overall[1],
    confusionMatrix(pred_gbm, testing$classe)$overall[1],
    confusionMatrix(combPred, testing$classe)$overall[1]
    )
)
print(AccuracyResults)
```

```
##         Model  Accuracy
## 1 Rpart_nopca 0.6616888
## 2       Rpart 0.2793918
## 3          RF 1.0000000
## 4         GBM 0.9633451
## 5    Combined 1.0000000
```

We see that random forest and the combined prediction model both have 100% accuracy. If we look at all the statistics that the confusion matrix provides we see that combining the models does not add any value. We will therefor use the random forest as our prediction model.

A strange result is the difference in accuracy between the model with and the model without PCA. We would have expected the model with PCA to show a much better result, but this was not the case at all. The model with PCA seems to just assign all values to class A, while the model without PCA only incorrectly classifies classes C and D. This would indicate that PCA actually decreases the accuracy in our models. But since we have the result of 100% accuracy, an out of sample error of 0%, leaving out PCA cannot improve the model any way. We will use the random forest with PCA as our prediction model.

5. Prediction

We will now apply our prediction model on the validation data.

```
prediction <- predict(mod_rf,validation)
prediction
```

```
##  [1] B A A A A E D B A A B A B A E E A B B B
## Levels: A B C D E
```

These results will be submitted as the answers to our prediction problem.