Practical Machine Learning Course Project

Martin Schulz

Executive Summary

We use data from accelerometers on the belt, forearm, arm, and dumbell 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).

We use a training data set (19,622 observations, 160 variables) to build a machine learning classifier to predict the manner in which they did the exercise (`classe` variable in training data set). We then test the model on a test set of 20 observations, for which we predict 19 correctly (95% accuracy).

Data preparation

First we load the training and test data sets and required libraries.

```
# Download and import data sets

if(!file.exists("pml-training.csv"))
        download.file(
        "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                "pml-training.csv")
if(!file.exists("pml-testing.csv"))
        download.file(
        "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                "pml-testing.csv")
training <- read.csv("pml-training.csv", na.strings = c("","NA","#DIV/0!"))
testing <- read.csv("pml-testing.csv", na.strings = c("","NA","#DIV/0!"))

# Load relevant libraries

library(caret, quietly = TRUE)
library(parallel, quietly = TRUE)
library(doParallel, quietly = TRUE)
```

We then remove the first 7 variables since they are IDs, labels and timestamps, none of which are useful predictors. We further reduce the data set by removing sparse variables, i.e. variables with mostly missing values, which have low predictive value. This results in a reduced data set with 53 variables, including the outcome variable `classe`.

```
# Remove ID, label, timestamp, etc variables

training.red1 <- training[,-c(1:7)]

# Remove sparse variables

nas <- sort(sapply(training.red1, function(x) sum(is.na(x))), decreasing = TRUE)
unique(nas) # either >=19216 (98% of records) or 0 missing values by variable
```

```
##  [1] 19622 19301 19300 19299 19296 19294 19293 19248 19227 19226 19225
## [12] 19221 19220 19218 19217 19216     0
```

```
training.red2 <- training.red1[,names(nas[nas == 0])] # cull sparse variables
```

```
# Applying final column selection to test data set

cols <- colnames(training.red2)[-53] # excluding "classe" from column names
testing.red2 <- testing[,cols]
```

Model building

We intend to build a random forest classifier. To reduce dimensionality we first pre-process the data by performing PCA requiring 97.5% of variance to be retained. This results in a data set of 30 predictors (principal components). We apply the same process to the test data set and re-append the outcome variable to the training data set.

```
# Preprocessing: apply PCA to further compress data (training and test)

preproc <- preProcess(training.red2[,-53], method = "pca", thresh = 0.975)
training.pca <- predict(preproc, training.red2[,-53])
training.pca <- cbind(classe = training.red2$classe, training.pca)

testing.pca <- predict(preproc, testing.red2)
```

Next, to reduce the runtime of the training algorithm, we set up and register a multi-core cluster to enable parallel processing and configure the training parameters accordingly. We also choose automatic 5-fold cross-validation to be performed upon completion of the model build.

```
# Configure parallel processing cluster

cluster <- makeCluster(detectCores()-1) # convention to leave 1 core for OS
registerDoParallel(cluster)

# Configure model fitting: parallel processing cluster and cross-validation

fitControl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)
```

Finally, we build the random forest classifier and measure its runtime.

```
# Fit random forest model (with measuring running time)

start_time <- Sys.time()
fit <- train(classe ~ ., method="rf", data = training.pca,
             trControl = fitControl)
end_time <- Sys.time()
end_time-start_time
```

```
## Time difference of 7.310593 mins
```

```
# De-register parallel processing cluster

stopCluster(cluster)
registerDoSEQ()
```

Model evaluation and application to test cases

Evaluating the model shows an average cross-validation accuracy of 97.9% / average cross-validation error rate of 2.1%, which appears to be satisfactory.

```
# Evaluate model
```

```
fit
```

```
## Random Forest
##
## 19622 samples
##    30 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15698, 15698, 15698, 15697, 15697
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9800735  0.9747882
##   16    0.9757416  0.9693116
##   30    0.9681993  0.9597735
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
confusionMatrix.train(fit)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.3  0.3  0.0  0.0  0.0
##          B  0.1 18.8  0.2  0.0  0.0
##          C  0.1  0.2 17.1  0.7  0.1
##          D  0.0  0.0  0.1 15.6  0.1
##          E  0.0  0.0  0.0  0.1 18.2
##
##  Accuracy (average) : 0.9801
```

Applying the random forest classifier to the 20 test cases results in 19 correct predictions which is equivalent to an accuracy of 95%. (Verification of predictions was performed externally and is not described here.)

```
# Predict on 20 test cases

prediction <- predict(fit, testing.pca)
data.frame(problem_id = testing$problem_id, prediction)
```

```
##    problem_id prediction
## 1           1          B
## 2           2          A
## 3           3          A
## 4           4          A
## 5           5          A
## 6           6          E
## 7           7          D
## 8           8          B
## 9           9          A
## 10         10          A
```

```
## 11          11          B
## 12          12          C
## 13          13          B
## 14          14          A
## 15          15          E
## 16          16          E
## 17          17          A
## 18          18          B
## 19          19          B
## 20          20          B
```