JHU Practical Machine Learning - course project

Harm Lammers

12 februari 2017

Recap the assignment

Instructions

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 dumbell of 6 participants.

Review criteria

What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with.

You should create a report describing
- how you built your model,
- how you used cross validation,
- what you think the expected out of sample error is, and
- why you made the choices you did.

You will also use your prediction model to predict 20 different test cases.

Peer Review Portion

Your submission for the Peer Review portion should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis.
Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).

Course Project Prediction Quiz Portion

Apply your machine learning algorithm to the 20 test cases available in the test data above and submit your predictions in appropriate format to the Course Project Prediction Quiz for automated grading.

Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

Prediction Assignment Writeup

Background

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 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).

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 data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

JHU Practical Machine Learning - course project

Introduction

Using data from a quantified self source (fitbits and alike) we practice different techniques in order to select a model for prediction purposes. In this case, data from devices like fitbits describe activitities but not how well an activity has been performed. The goal of this study is to determine wether the activity itself has been performed well based on the measured values.

Environment settings

```
library(ggplot2)
library(lattice)
library(caret)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Data

The data for this project come from : http://groupware.les.inf.puc-rio.br/har.

```
#Visual control of csv-data reveals #DIV/0! comments, to be treated as NA
TrainDat    = read.csv("C:/Users/harml/Desktop/pml-training.csv", na.strings=c("#DIV/0!"), row.names = 
TestDat     = read.csv("C:/Users/harml/Desktop/pml-testing.csv", na.strings=c("#DIV/0!"), row.names = 1)

#Explore training-data
# WorkDat    <- TrainDat
# names(WorkDat)
# summary(WorkDat)
# head(WorkDat)
# dim(WorkDat)

training       <- TrainDat

#Explore testing-data
# WorkDat        <- TestDat
# names(WorkDat)
# summary(WorkDat)
# head(WorkDat)
# dim(WorkDat)

testing        <- TestDat

#Remove non-measurements as predictor variables
training       <- training[, 6:dim(training)[2]]
# dim(training)

#Remove columns with almost no values (at least 95% NA or "")
treshold       <- 0.95 * dim(training)[1]
goodColumns    <- !apply(training, 2, function(x) sum(is.na(x)) > treshold  || sum(x=="") > treshold)
# goodColumns
training       <- training[, goodColumns]
# dim(training)

#Remove columns with almost no variance
badColumns     <- nearZeroVar(training, saveMetrics = TRUE)
# badColumns
training       <- training[, badColumns$nzv==FALSE]
# dim(training)

#Factorise dependent variable
training$classe = factor(training$classe)

# names(training)
# names(goodColumns)
# names(badColumns)
# names(testing)

#Prepare testing dataset like the training set
testing        <- testing[, 6:dim(testing)[2]]
testing        <- testing[, goodColumns]
```

```
#Add the dependent variable to the testset
testing$classe <- NA
testing        <- testing[, badColumns$nzv==FALSE]

# names(testing)
# names(training)
```

Building the training-, test- and validation sets

The original training dataset will be split into 3 subsets:
- 60% for training,
- 30% for cross-validation and
- 10% for test purposes.

```
set.seed(2305)

#Partition rows of the initial training dataset into 3 subsets (training, crossvalidation and test)
inTrain     <- createDataPartition(training$classe, p = 0.6)[[1]]
training    <- training[ inTrain,]
crossv      <- training[-inTrain,]

inTrain     <- createDataPartition(crossv$classe, p = 0.75)[[1]]
crossv      <- crossv[ inTrain,]

crosst      <- crossv[-inTrain,]
```

Training

Following the instruction video's and quizzes I try three modelling techniques; RF, GBM and LDA.

```
Sys.time()
```

```
## [1] "2017-02-18 19:28:35 CET"
```

```
set.seed(2323)
RFmod       <- train(classe ~ ., data=training, method="rf")                    # computational load i
Sys.time()
```

```
## [1] "2017-02-18 20:20:17 CET"
```

```
set.seed(2323)
GBMmod      <- train(classe ~ ., data=training, method="gbm", verbose = FALSE)   # computational load i
Sys.time()
```

```
## [1] "2017-02-18 20:41:33 CET"
```

```
set.seed(2323)
LDAmod      <- train(classe ~ ., data=training, method="lda")                    # computational load i
Sys.time()
```

```
## [1] "2017-02-18 20:41:40 CET"
```

```
RFpred      <- predict(RFmod, crossv)
GBMpred     <- predict(GBMmod, crossv)
LDApred     <- predict(LDAmod, crossv)
```

```r
confusionMatrix(crossv$classe, RFpred)  # accuracy 1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1008    0    0    0    0
##          B    0  681    0    0    0
##          C    0    0  624    0    0
##          D    0    0    0  562    0
##          E    0    0    0    0  666
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2847   0.1923   0.1762   0.1587   0.1881
## Detection Rate         0.2847   0.1923   0.1762   0.1587   0.1881
## Detection Prevalence   0.2847   0.1923   0.1762   0.1587   0.1881
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```r
confusionMatrix(crossv$classe, GBMpred) # accuracy 0.9941
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1008    0    0    0    0
##          B    0  677    4    0    0
##          C    0    2  621    1    0
##          D    0    1    7  554    0
##          E    0    2    0    4  660
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9909, 0.9963)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9927   0.9826   0.9911   1.0000
## Specificity            1.0000   0.9986   0.9990   0.9973   0.9979
## Pos Pred Value         1.0000   0.9941   0.9952   0.9858   0.9910
## Neg Pred Value         1.0000   0.9983   0.9962   0.9983   1.0000
## Prevalence             0.2847   0.1926   0.1785   0.1579   0.1864
## Detection Rate         0.2847   0.1912   0.1754   0.1565   0.1864
## Detection Prevalence   0.2847   0.1923   0.1762   0.1587   0.1881
## Balanced Accuracy      1.0000   0.9956   0.9908   0.9942   0.9990
```

```
confusionMatrix(crossv$classe, LDApred) # accuracy 0.7074
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 831  23  68  82   4
##          B  92 421 107  26  35
##          C  61  50 419  75  19
##          D  23  21  84 415  19
##          E  19  93  71  64 419
##
## Overall Statistics
##
##                Accuracy : 0.7074
##                  95% CI : (0.6921, 0.7224)
##     No Information Rate : 0.2897
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6302
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8099   0.6924   0.5594   0.6269   0.8448
## Specificity            0.9296   0.9114   0.9266   0.9489   0.9189
## Pos Pred Value         0.8244   0.6182   0.6715   0.7384   0.6291
## Neg Pred Value         0.9230   0.9346   0.8869   0.9171   0.9732
## Prevalence             0.2897   0.1717   0.2115   0.1870   0.1401
## Detection Rate         0.2347   0.1189   0.1183   0.1172   0.1183
## Detection Prevalence   0.2847   0.1923   0.1762   0.1587   0.1881
## Balanced Accuracy      0.8698   0.8019   0.7430   0.7879   0.8818
```

```
#Combine the models
# predDF      <- data.frame(RFpred, GBMpred, LDApred, crossv$classe)
# COMBmod     <- train(crossv$classe ~., data=predDF, method="gam")
# COMBpred    <- predict(COMBmod, predDF)
#In-sample error
# confusionMatrix(COMBpred,crossv$classe)
# confusionMatrix(COMBpred,crossv$classe)$overall['Accuracy'] #
```

```
#Out-sample error
# COMBpred    <- predict(COMBmod, crosst)
# confusionMatrix(COMBpred,crosst$classe)
# confusionMatrix(COMBpred,crosst$classe)$overall['Accuracy'] #
```

Based on the results we see a perfect fit on the cross-validation-set with the resulting Random Forest model based on the trainig set. GBM comes close to this result but I choose the model based on the random forest technique as the resulting model.

In order to predict the out-of-sample error I use the 3rd subset of the trainingdata as a testset. Though the results have been calculated for the other two models, I will not report them in the presentation.

```
RFpredt     <- predict(RFmod, crosst)
#GBMpredt    <- predict(GBMmod, crosst)
#LDApredt    <- predict(LDAmod, crosst)

confusionMatrix(crosst$classe, RFpredt)  # accuracy 1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 257   0   0   0   0
##          B   0 168   0   0   0
##          C   0   0 149   0   0
##          D   0   0   0 146   0
##          E   0   0   0   0 171
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9959, 1)
##     No Information Rate : 0.2884
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2884   0.1886   0.1672   0.1639   0.1919
## Detection Rate         0.2884   0.1886   0.1672   0.1639   0.1919
## Detection Prevalence   0.2884   0.1886   0.1672   0.1639   0.1919
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
#confusionMatrix(crosst$classe, GBMpredt) # accuracy 0.9898
#confusionMatrix(crosst$classe, LDApredt) # accuracy 0.713

# CM <- confusionMatrix(crosst$classe, RFpredt)
# CM
```

Even in the test-set we see a 100% prediction-score with a 95% confidence interval for Acccuracy: (0.9958, 1).

The Random Forest - model will be used for the final predictions.

The theoretical logic behind using the random forest method as the predictor rather than other methods or a combination of various methods is:
- Random forest - method can handle a large number of inputs
- Random forest - method can handle data when the interactions between the variables are unknown.
- Random forest - method's built in cross-validation function gives an unbiased estimate of the out-of-sample (or bag / OOB) error rate.
- Random forest - method can handle unscaled variables and categorical variables.

Predicting the dependent variable 'classe' for the testset

The predictions are based on the resulting Random Forest model and use the given testdata as a resource. Afterwards I prepare the predictions for upload by writing each prediction into a separate CSV-file.

```
# Predict classe based on test-data
RFpredict     <- predict(RFmod, newdata=testing)
RFpredict
```

```
##  [1] 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
```

```
#Prepare the predictions for upload by writing each prediction into a separate CSV-file
PML_write_files = function(x)
   {
      n = length(x)
      for(i in 1:n)
         {
            filename = paste0("prediction_id_",i,".txt")
            write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
         }
   }

PML_write_files(RFpredict)
```