Peer-graded Assignment: Prediction Assignment Writeup

Vicente Castro

18 de febrero de 2017

Introduction

The goal of this project is to predict the manner in which they did the exercise. I've created a report describing how I built the model, how I used cross validation, what I think the expected out of sample error is, and why I made the choices I did.

Data preparation

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")

dim(training)
```

```
## [1] 19622   160
```

Reduce the number of predictors by removing variables with nearly zero variance, NA.

```
# remove variables with nearly zero variance

nzv <- nearZeroVar(training)

subTraining <- training[, - nzv]
dim(subTraining)
```

```
## [1] 19622   100
```

```
# remove variables that are almost always NA
mostNA <- sapply(subTraining, function(x) mean(is.na(x)))>0.9
subTraining <- subTraining[, mostNA==F]
dim(subTraining)
```

```
## [1] 19622    59
```

```
#remove variables that don't make intuitive sense for prdiction (V1 seems to be a serial number and cvt

subTraining <- subTraining[,-1]
subTraining <- subTraining[, c(1:3, 5:58)]

dim(subTraining)
```

```
## [1] 19622    57
```

Creating cross validation data

How I will demostrate at the end pof this report, the best model is Rain forrest, and according with the lectures and http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#overview, there is no need to create a separate set of validation; however, as it is one of the project evaluation criteria, there is no problem to create a cross validation dataste to compare the model created by the training subset.

```
#Divide trainign set for validation

inTrain <- createDataPartition(subTraining$classe, p=0.7, list=FALSE)

subTrainingModel <- subTraining[inTrain,]
subValidation <- subTraining[-inTrain,]

dim(subTrainingModel)
```

```
## [1] 13737    57
```

```
dim(subValidation)
```

```
## [1] 5885   57
```

Creating prediction model

Using the first set of data, to create the prediction model (using random forest).

According with some forums and other resources , first, I setup to run in parallel, using all the CPU cores available.

```
model <- "modelFit.RData"
set.seed(2017)
if (!file.exists(model)){
    require(parallel)
    require(doParallel)
    cl <- makeCluster(detectCores() - 1)
    registerDoParallel(cl)

    fitModelRF <- train(subTrainingModel$classe ~ ., method = "rf", data = subTrainingModel)
    save(fitModelRF, file = "modelFit.RData")

    stopCluster(cl)

}else{

  load(file="modelFit.RData", verbose = TRUE)


}
```

```
## Loading objects:
##   fitModelRF
```

Measure the Accuracy and sample error

```
predictTrain <- predict(fitModelRF, subTrainingModel)
```

```
## Loading required package: 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
```

```
confusionMatrix(predictTrain, subTrainingModel$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2657    2    0    0
##          C    0    1 2394    1    0
##          D    0    0    0 2250    0
##          E    0    0    0    1 2525
##
## Overall Statistics
##
##                Accuracy : 0.9996
##                  95% CI : (0.9992, 0.9999)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9996   0.9992   0.9991   1.0000
## Specificity            1.0000   0.9998   0.9998   1.0000   0.9999
## Pos Pred Value         1.0000   0.9992   0.9992   1.0000   0.9996
## Neg Pred Value         1.0000   0.9999   0.9998   0.9998   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1934   0.1743   0.1638   0.1838
## Detection Prevalence   0.2843   0.1936   0.1744   0.1638   0.1839
## Balanced Accuracy      1.0000   0.9997   0.9995   0.9996   1.0000
```

Now, I'm going to use the validation subset and creat a prediction.

```
predictValidation <- predict(fitModelRF, subValidation)

confusionMatrix(predictValidation, subValidation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    1    0    0
##          C    0    0 1025    0    0
##          D    0    0    0  964    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9995
##                  95% CI : (0.9985, 0.9999)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9994
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9990   1.0000   0.9991
## Specificity            0.9998   0.9998   1.0000   0.9998   1.0000
## Pos Pred Value         0.9994   0.9991   1.0000   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9998   1.0000   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1742   0.1638   0.1837
## Detection Prevalence   0.2846   0.1935   0.1742   0.1640   0.1837
## Balanced Accuracy      0.9999   0.9995   0.9995   0.9999   0.9995
```

From the validation subset, the accuracy is still hig, above 99.9%.

Given the high level of accuracy, I think there is no need to build another prediction model for better accuracy. These will only complicate the exercise - making it hard to explain, and takes too long a time to run another training process.

List of important predictor in the model

```
varImp(fitModelRF)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 60)
##
##                      Overall
## raw_timestamp_part_1 100.000
## num_window            54.747
## roll_belt             45.207
## pitch_forearm         28.991
## magnet_dumbbell_z     21.320
## yaw_belt              17.262
## magnet_dumbbell_y     16.672
```

```
## pitch_belt            16.502
## roll_forearm          12.446
## accel_dumbbell_y       7.680
## roll_dumbbell          7.429
## magnet_dumbbell_x      7.266
## accel_belt_z           6.760
## total_accel_dumbbell   6.372
## accel_forearm_x        5.956
## magnet_belt_y          5.556
## accel_dumbbell_z       5.060
## magnet_belt_z          4.940
## yaw_dumbbell           3.072
## accel_dumbbell_x       2.747
```

Final Model

```
fitModelRF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 31
##
##         OOB estimate of  error rate: 0.09%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    1 2655    1    1    0 0.001128668
## C    0    6 2389    1    0 0.002921536
## D    0    0    2 2249    1 0.001332149
## E    0    0    0    0 2525 0.000000000
```

The reported OOB estimated error is at 0.09%, the prediction model should be applied to the final testing set, and predict th classe in the 20 test cases.

Apply the prediction Model

```
predictTesting <- predict(fitModelRF, testing)
predictTesting
```

```
##  [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
```