Practical Machine Learning - Classifying Correct Exercise Technique from Personal Device Data

S Carroll

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, the 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." [1]

"Six young healthy participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Class A is the correct execution, and the other classess correspond to common mistakes."[2]

Data

The dataset was provided by Ugulino, Cardaror, et. al.[3], and is available from this source.[4] The dataset contains 19622 observations in the training and 20 observations in the testing set. Columns majorly populated with data include 13 features for each of four devices worn on the belt, arm, dumbbell, and forearm (52 total). These features are measurements corresponding to roll, pitch, yaw, total acceleration, and triaxial device data from the gyros, accelerometers, and magnetometers. Other columns include information on subjectID, time/window, and the outcome. The outcome is 'classe'.

Summary

How the Model Was Built and Reasons for Choices

Three methods were used to fit the outcome (classe) to a set of 52 predictors: random forest, boosting with trees, and linear discriminant analysis, using the train function of the caret package. Default parameters were chosen, which included a 5-fold resampling. Accuracy was assessed from the prediction of the models individually and collectively, and the most accurate model (rf) was chosen. Additionally, the data did not appear to be linear, and the random forest method is robust for non-linear data.

Two potential sources of noise were evaluated in the initial examination of data. These include subjects not performing the same number of tests, and differences in the time windows. As the model fit was very good, these potential noise sources did not require further evaluation.

[1] Coursera Data Science Course, Practical Machine Learning, accessed 5/2018.

[2] http://groupware.les.inf.puc-rio.br/har

[3] UVelloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

[4] http://groupware.les.inf.puc-rio.br/har

Cross Validation and Expected Out of Sample Error

The training data was partitioned in a 60/40 split to allow for training and to predict testing results. In-sample variation was obtained from model fit data, and was cross-validated by using predicted values from the test portion of the training data. The expected out-of-sample error is defined as 1-accuracy of the prediction from the test portion of the data, and is less than 0.8%.

Code

Preliminaries

Libraries

```
library(data.table)
library(dplyr)
library(ggplot2)
library(caret)
library(doParallel)
library(parallel)
library(randomForest)
```

Due to the computational intensity, code for accessing n-1 cores is used to speed processing time.[5]

```
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
fitControl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)

# To invoke, add arg to train(... trControl = fitControl ).  De-register after modelling
```

Preprocess Data (Obtain, Examine, Clean, and Partition Data)

Obtain the data.

```
trainURL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
testURL <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

if(!file.exists("./data")){dir.create("./data")}
if(!file.exists("./data/training.csv")){download.file(trainURL,destfile="./data/training.csv")}
if(!file.exists("./data/testing.csv")){download.file(trainURL,destfile="./data/testing.csv")}

training <- as.data.table(fread('./data/training.csv', na.strings = c("NA","#DIV/0!","")))
testing <- as.data.table(fread('./data/testing.csv', na.strings = c("NA","#DIV/0!","")))
```

Examine the data.

```
rbind(c('training',dim(training)),c('testing',dim(testing)))
```

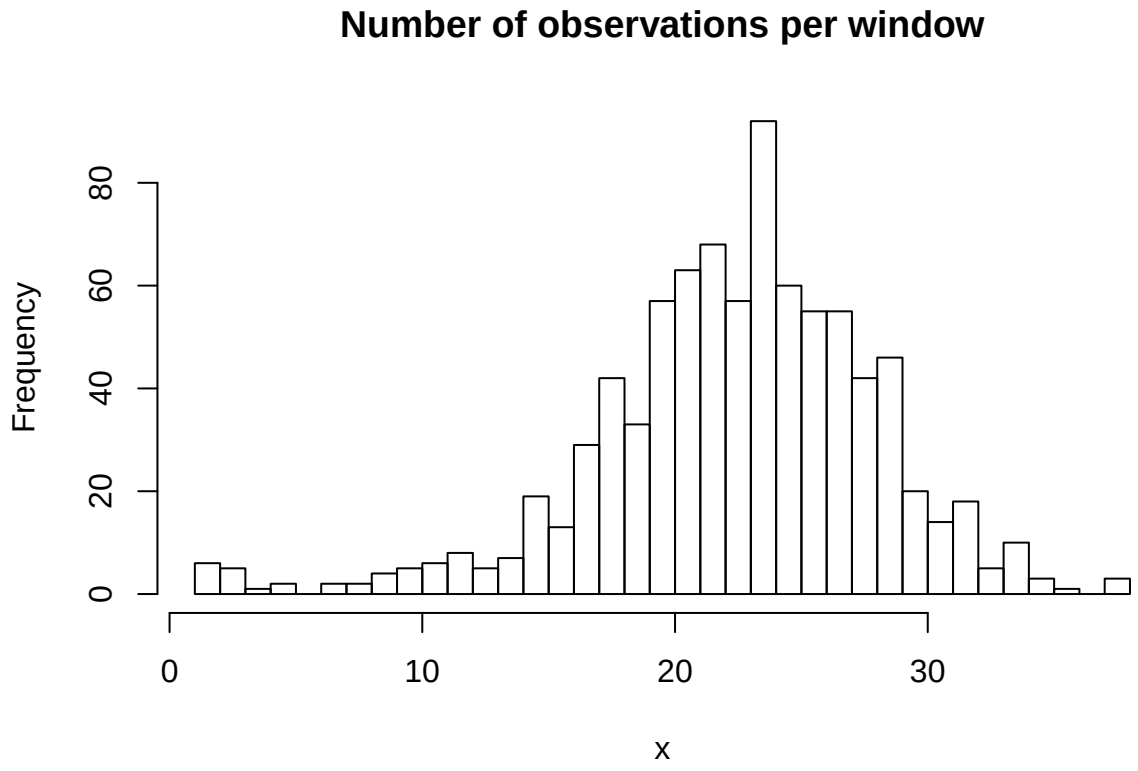
```
##      [,1]       [,2]    [,3]
## [1,] "training" "19622" "160"
## [2,] "testing"  "20"    "160"
```

```
# sampling is not balanced by user and exercise class
table(training$user_name,training$classe)
```

[5] Greski, L., https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md, accessed 5/2018.

```
##
##               A    B    C    D    E
##   adelmo   1165  776  750  515  686
##   carlitos  834  690  493  486  609
##   charles   899  745  539  642  711
##   eurico    865  592  489  582  542
##   jeremy   1177  489  652  522  562
##   pedro     640  505  499  469  497
```

```
# number of observations per window is not uniform
x <- table(training$num_window)
hist(x, main = "Number of observations per window", breaks = max(x))
```




```
# make classe variable is a factor
training$classe <-factor(training$classe)
```

Clean the data.

```
colNotKeep <- c(1:7)  # studyu, names, date/time

# columns with 90% or more missing data
thresh <- length(training)*0.90
badCol <- apply(training, 2, function(x) sum(is.na(x)) > thresh  || sum(x=="") > thresh)
sum(badCol) # 100 columns with missing data
```

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

```
# what are these
badColName <- names(training[,badCol, with = FALSE])
# what is kept
keptColName <- names(training[,!badCol, with = FALSE])

# columns with near zero variance
nzv <- nearZeroVar(training, saveMetrics = TRUE)   # already contained in badCol

# select kept columns and clean data
colNotKeep <- unique(as.numeric(c(colNotKeep, which(badCol),which(nzv$nzv==TRUE))))

training <- training[,!colNotKeep, with = FALSE]
testing <- testing[,!colNotKeep, with = FALSE]
```

Partition the training data to allow cross-validation

```
set.seed(05072)
inTrain <- createDataPartition(training$classe, p = 0.6, list = FALSE)
myTrain <- training[inTrain,]
myTest <- training[-inTrain,]

rbind(c('myTrain',dim(myTrain)),c('myTest',dim(myTest)))
```

```
##      [,1]      [,2]    [,3]
## [1,] "myTrain" "11776" "53"
## [2,] "myTest"  "7846"  "53"
```

Model, Prediction and Figures of Merit

Build models from the train portion of the training set. The data below show that the random forest method yields the highest accuracy. The top 5 most influential predictors are shown.

```
set.seed(62433)

# train models
modRF <- train(classe ~ ., method="rf",data=myTrain,verbose=FALSE,trControl = fitControl)
modGBM <- train(classe ~ ., method="gbm",data=myTrain,verbose=FALSE,trControl = fitControl)
modLDA <- train(classe ~ ., method="lda",data=myTrain,verbose=FALSE,trControl = fitControl)

# examine  fit metrics from modelling provided by train()
modRF
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9420, 9420, 9422, 9421, 9421
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9883656  0.9852814
```

```
##   27    0.9894698  0.9866778
##   52    0.9823363  0.9776514
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
modGBM
```

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9421, 9421, 9420, 9421, 9421
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7562853  0.6910854
##   1                  100      0.8210779  0.7735942
##   1                  150      0.8569139  0.8189065
##   2                   50      0.8509694  0.8111807
##   2                  100      0.9047216  0.8793971
##   2                  150      0.9329995  0.9152206
##   3                   50      0.8961451  0.8685527
##   3                  100      0.9426803  0.9274770
##   3                  150      0.9616172  0.9514419
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
modLDA
```

```
## Linear Discriminant Analysis
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9422, 9421, 9421, 9420, 9420
## Resampling results:
##
##   Accuracy   Kappa
##   0.6979451  0.617739
```

```
# examine importance of predictors for random forest method
impRF <- modRF$finalModel$importance[order(modRF$finalModel$importance[,1], decreasing = TRUE),]
impRF[1:5]
```

```
##         roll_belt     pitch_forearm          yaw_belt magnet_dumbbell_z
##         1212.3360          721.5317          625.1312          546.9352
## magnet_dumbbell_y
##          531.8200
```

Predict in-sample error. These results also indicate that the random forest method is best. It is not improved by combining with other models.

```
# predict using individual and combined models
predRFi <- predict(modRF, myTrain)
predGBMi <- predict(modGBM, myTrain)
predLDAi <- predict(modLDA, myTrain)

predDfi <- data.frame(predRFi, predGBMi,predLDAi, classe = myTrain$classe)
predDfi2 <- data.frame(predRFi, predGBMi, classe = myTrain$classe)

combModelFiti <- train(classe ~ ., method = 'rf', data = predDfi,trControl = fitControl)
combModelFiti2 <- train(classe ~ ., method = 'rf', data = predDfi2,trControl = fitControl)

predCombi <- predict(combModelFiti, predDfi)
predCombi2 <- predict(combModelFiti2, predDfi2)

confuMati <- function(pred) {confusionMatrix(pred, myTrain$classe)[2:3]}

confuMati(predRFi)
```

```
## $table
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9996868      1.0000000      0.2843071
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
confuMati(predGBMi)
```

```
## $table
##           Reference
## Prediction    A    B    C    D    E
##          A 3317   42    0    1    3
##          B   21 2197   43    4   10
##          C    6   36 1997   43   16
##          D    4    3   11 1880   19
##          E    0    1    3    2 2117
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.772418e-01   9.712112e-01   9.743849e-01   9.798593e-01   2.843071e-01
## AccuracyPValue  McnemarPValue
```

```
##   0.000000e+00   1.316731e-10
```

```
confuMati(predLDAi)
```

```
## $table
##           Reference
## Prediction    A    B    C    D    E
##          A 2736  354  211  102   77
##          B   74 1462  191   80  376
##          C  276  267 1363  236  187
##          D  248   89  240 1428  219
##          E   14  107   49   84 1306
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   7.043988e-01   6.259646e-01   6.960649e-01   7.126315e-01   2.843071e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00  1.631442e-122
```

```
confuMati(predCombi)
```

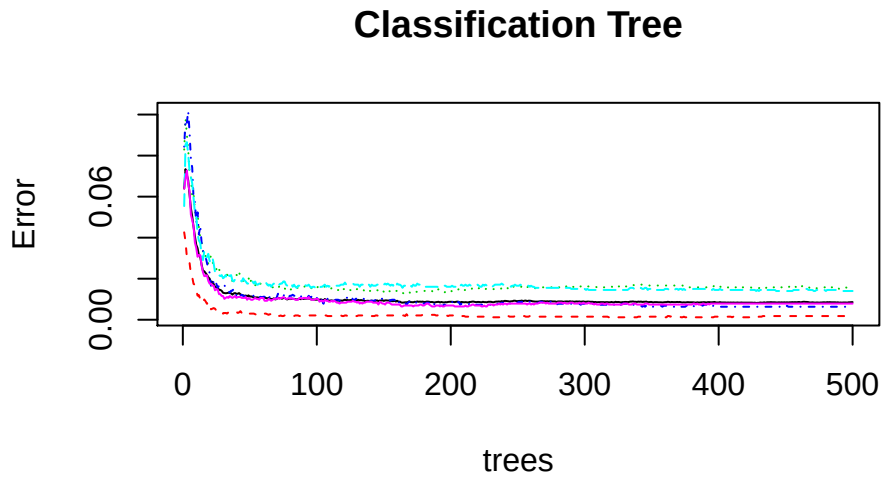
```
## $table
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9996868      1.0000000      0.2843071
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
confuMati(predCombi2)
```

```
## $table
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9996868      1.0000000      0.2843071
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

For the final model using random forest, 500 trees were used. The error as a function of the number of trees is shown below. The accuracy increases with additional trees, but the error is still small for a smaller number of trees (~100). This could simplify the interpretability of predictors.

```
plot(modRF$finalModel, main = "Classification Tree")
```



Predict the out-of-sample error using the reserved data from the training set. The accuracy is highest for the random forest method.

```
predRF <- predict(modRF, myTest)
predGBM <- predict(modGBM, myTest)
predLDA <- predict(modLDA, myTest)

confuMat <- function(pred) {confusionMatrix(pred, myTest$classe)$overall[1]}
# accuracy
rbind(c('predRF',confuMat(predRF)),c('predGBM',confuMat(predGBM)),c('predLDA',confuMat(predLDA) ))
```

```
##                 Accuracy
## [1,] "predRF"  "0.991715523833801"
## [2,] "predGBM" "0.957940351771603"
## [3,] "predLDA" "0.704435381085904"
```

```
# results for best
confusionMatrix(predRF, myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230   19    0    0    0
##          B    2 1494    6    0    1
##          C    0    4 1355   17    3
##          D    0    1    7 1268    4
##          E    0    0    0    1 1434
##
## Overall Statistics
##
##                Accuracy : 0.9917
##                  95% CI : (0.9895, 0.9936)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9895
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9842   0.9905   0.9860   0.9945
## Specificity            0.9966   0.9986   0.9963   0.9982   0.9998
## Pos Pred Value         0.9916   0.9940   0.9826   0.9906   0.9993
## Neg Pred Value         0.9996   0.9962   0.9980   0.9973   0.9988
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1904   0.1727   0.1616   0.1828
## Detection Prevalence   0.2866   0.1916   0.1758   0.1631   0.1829
## Balanced Accuracy      0.9979   0.9914   0.9934   0.9921   0.9971
```

The models are applied to the testing data.

```
x <- testing
predict(modRF, newdata=x)
```

```
##  [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
```

Cleanup

```
# de-register parallel processing cluster and return to single threaded processing
stopCluster(cluster)
registerDoSEQ()
```

References