Fitness move assessment - Practical Machine Learning

Jennifer Holtzman

July 9, 2017

Executive Summary

The goal of the assignment is to use fitness tracker data to predict whether weight lifting exercises were done well, i.e., using correct form. Separate training and test data sets were provided as described: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)

Exploration of the data led to the inclusion of 46 of the 160 variables. Columns were removed due to them containing no data, being indices and timestamps not relevant for prediction, or being highly correlated with other variables. A subset of 30 % of the remaining data set was used to fit a random forest model with 5-fold cross-validation. The calculated accuracy was very high (97 %), but so was the kappa statistic (96 %), so it can not be determined whether the classifier model performed any better than would be expected. Out-of-bounds error of the model was calculated at 1.83 %.

Applying the model to the testing data (20 samples) resulted in the following predictions: B A B A A E D B A A B C B A E E A B B B All were correct in the quiz.

Loading Data and Installing Packages

The task is to create a prediction model based on data collected with a fitness tracker in order to determine whether an exercise, barbell lifts was done correctly. The form of the exercise is captured by the variable "classe", which assumes one of 5 classes, A-E. Two data sets are provided, a training set (19622 observations over 160 variables, including the one to be predicted), and a testing set (20 observations over the same 160 variables)

```
setwd("C:/Users/Jenny/Documents/COURSERA/8 - Machine Learning")
training <- read.csv("pml-training.csv", na.strings=c("","NA"))
testing <- read.csv("pml-testing.csv", na.strings=c("","NA"))

set.seed(12547)

library(caret)
```

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

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

Brief discussion of data exploration

1. Checked for empty and NA values. Within the training set, there are 406 complete cases out of 19622 rows of observations. Do not want to throw out useful data. Instead, determine which columns (variables) are mostly empty or NA, i.e. contain no data, and can probably safely be omitted for purposes of prediction. Of 160 columns, 60 remain.

2. Removed variables that are not relevant to what is being predicted, such as indices and timestamps. Of the 60 columns retained after step 1, 53 remain.

3. Checked for near zero covariates, i.e., those that vary little and are therefore probably of limited or no value for prediction. There were no covariates with zero variance or non-zero variance (nzv). No columns removed.

4. Checked for highly correlated variables (numeric only), above 90 %, and removed these. Of the 53 columns, 46 remain.

```
# Exploration point 1
sum(complete.cases(training))
```

```
## [1] 406
```

```
fracn_empty <- sapply(training, function(x) sum(is.na(x))/19622) # proportion e
mpty or NA across the columns of training
sum(fracn_empty > 0.95) # returns 100 variables with > 95 % of observations emp
ty or NA
```

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

```
sum(fracn_empty > 0.97) # returns 100 variable with > 97 % of observations empt
y or NA
```

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

```
sum(fracn_empty > 0.98) # returns 0 variable with > 98 % of observations empty
or NA
```

```
## [1] 0
```

```
to_keep <- fracn_empty[which(fracn_empty < 0.95)] # a list of numerics
to_keep <- as.data.frame(to_keep, header = TRUE)

training_small1 <- training[, colnames(training) %in% rownames(to_keep)]

# Exploration point 2
to_remove = c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_
2', 'cvtd_timestamp', 'new_window', 'num_window')
training_small2 <- training_small1[, -which(names(training_small1) %in% to_remo
ve)]

# Exploration point 3
nsv <- nearZeroVar(training_small2, saveMetrics = TRUE)
nsv
```

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt             1.101904     6.7781062   FALSE FALSE
## pitch_belt            1.036082     9.3772296   FALSE FALSE
## yaw_belt              1.058480     9.9734991   FALSE FALSE
## total_accel_belt      1.063160     0.1477933   FALSE FALSE
## gyros_belt_x          1.058651     0.7134849   FALSE FALSE
## gyros_belt_y          1.144000     0.3516461   FALSE FALSE
## gyros_belt_z          1.066214     0.8612782   FALSE FALSE
## accel_belt_x          1.055412     0.8357966   FALSE FALSE
## accel_belt_y          1.113725     0.7287738   FALSE FALSE
## accel_belt_z          1.078767     1.5237998   FALSE FALSE
## magnet_belt_x         1.090141     1.6664968   FALSE FALSE
## magnet_belt_y         1.099688     1.5187035   FALSE FALSE
## magnet_belt_z         1.006369     2.3290184   FALSE FALSE
## roll_arm             52.338462    13.5256345   FALSE FALSE
## pitch_arm            87.256410    15.7323412   FALSE FALSE
## yaw_arm              33.029126    14.6570176   FALSE FALSE
## total_accel_arm       1.024526     0.3363572   FALSE FALSE
## gyros_arm_x           1.015504     3.2769341   FALSE FALSE
## gyros_arm_y           1.454369     1.9162165   FALSE FALSE
## gyros_arm_z           1.110687     1.2638875   FALSE FALSE
## accel_arm_x           1.017341     3.9598410   FALSE FALSE
## accel_arm_y           1.140187     2.7367241   FALSE FALSE
## accel_arm_z           1.128000     4.0362858   FALSE FALSE
## magnet_arm_x          1.000000     6.8239731   FALSE FALSE
## magnet_arm_y          1.056818     4.4439914   FALSE FALSE
## magnet_arm_z          1.036364     6.4468454   FALSE FALSE
## roll_dumbbell         1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell        2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell          1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell  1.072634     0.2191418   FALSE FALSE
## gyros_dumbbell_x      1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y      1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z      1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x      1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y      1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z      1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x     1.098266     5.7486495   FALSE FALSE
## magnet_dumbbell_y     1.197740     4.3012945   FALSE FALSE
## magnet_dumbbell_z     1.020833     3.4451126   FALSE FALSE
## roll_forearm         11.589286    11.0895933   FALSE FALSE
## pitch_forearm        65.983051    14.8557741   FALSE FALSE
## yaw_forearm          15.322835    10.1467740   FALSE FALSE
## total_accel_forearm   1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x       1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y       1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z       1.122917     1.5645704   FALSE FALSE
## accel_forearm_x       1.126437     4.0464784   FALSE FALSE
```

```
## accel_forearm_y       1.059406     5.1116094   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x      1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y      1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z      1.000000     8.5771073   FALSE FALSE
## classe                1.469581     0.0254816   FALSE FALSE
```

```
sum(nsv$nzv == TRUE)
```

```
## [1] 0
```

```
# Exploration point 4
corrMatrix <- cor(na.omit(training_small2[sapply(training_small2, is.numeri
c)]))
dim(corrMatrix)
```

```
## [1] 52 52
```

```
to_remove4 = findCorrelation(corrMatrix, cutoff = .90, verbose = TRUE)
```

```
## Compare row 10  and column  1 with corr  0.992
##   Means:  0.27 vs 0.168 so flagging column 10
## Compare row 1  and column  9 with corr  0.925
##   Means:  0.25 vs 0.164 so flagging column 1
## Compare row 9  and column  4 with corr  0.928
##   Means:  0.233 vs 0.161 so flagging column 9
## Compare row 8  and column  2 with corr  0.966
##   Means:  0.245 vs 0.157 so flagging column 8
## Compare row 19  and column  18 with corr  0.918
##   Means:  0.091 vs 0.158 so flagging column 18
## Compare row 46  and column  31 with corr  0.914
##   Means:  0.101 vs 0.161 so flagging column 31
## Compare row 46  and column  33 with corr  0.933
##   Means:  0.083 vs 0.164 so flagging column 33
## All correlations <= 0.9
```

```
training_small4 = training_small2[,-to_remove4]
dim(training_small4)
```

```
## [1] 19622    46
```

Building the prediction model with cross-

validation

A random forest (rf) is a non-linear method that resamples fram the data set samples (bootstrapping) as well as the variables. After growing a large number of trees, the outcomes are averaged to predict the outcome. For this data, the goal is to predict “classe”. This approach has the advantage of giving accurate results, but it can be slow and hard to intepret. Cross-validation is important to account for over-fitting.

In searching online for examples of using the correct parameters with the rf method, I found the following resource which appeared to use the same data set:
http://bigcomputing.blogspot.ca/2014/10/an-example-of-using-random-forest-in.html
(http://bigcomputing.blogspot.ca/2014/10/an-example-of-using-random-forest-in.html)

The example showed how to incorporate cross-validation and error estimation directly, and so I used this approach.

With the full training_small4 data set, it seems to take a very long time to run. Partition the data (0.3 of the samples) to speed things along. Note: the allowParallel parameter presumably enables parallel processing to speed up running the model. I was unable to find/install the package “doMC” which I believe is prerequisite for parallel processing, so this parameter is likely not doing anything when I run the model on my machine.

```
library(caret)
library(ggplot2)

InTrain<-createDataPartition(y=training_small4$classe,p=0.3,list=FALSE)
training_smaller<-training_small4[InTrain,]
modFit <- train(classe ~., data = training_smaller, method = "rf", trControl=tr
ainControl(method="cv",number=5), prox = TRUE, allowParallel=TRUE)
```

```
## 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
```

```
print(modFit)
```

```
## Random Forest
##
## 5889 samples
##   45 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4710, 4711, 4712, 4711, 4712
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9724901  0.9651880
##   23    0.9791138  0.9735785
##   45    0.9728310  0.9656288
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 23.
```

```
print(modFit$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE,      allowP
arallel = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 23
##
##         OOB estimate of  error rate: 1.85%
## Confusion matrix:
##      A    B   C   D    E class.error
## A 1665    4   3   0    2 0.005376344
## B   25 1095  16   4    0 0.039473684
## C    0   16 999  10    2 0.027263875
## D    0    1  14 947    3 0.018652850
## E    0    0   1   8 1074 0.008310249
```

```
# Variable importance analysis
varImp(modFit)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 45)
##
##                       Overall
## yaw_belt               100.00
## pitch_forearm           86.20
## pitch_belt              67.83
## magnet_dumbbell_z       67.02
## magnet_dumbbell_y       56.92
## roll_forearm            46.99
## magnet_belt_y           44.88
## magnet_belt_z           32.67
## roll_dumbbell           25.00
## magnet_dumbbell_x       24.54
## accel_dumbbell_y        24.17
## accel_forearm_x         22.18
## gyros_belt_z            21.00
## accel_dumbbell_z        18.84
## total_accel_belt        18.64
## accel_forearm_z         17.60
## total_accel_dumbbell    17.35
## magnet_belt_x           16.99
## magnet_forearm_z        16.62
## roll_arm                14.19
```

From the model information, the prediction produced by this model has a reported accuracy of 0.971 and a kappa of 0.964. The latter is a statistic for expected accuracy for a classifier prediction, and the high value does not in this case tell us much about the performance of the classifier model (https://stats.stackexchange.com/questions/82162/cohens-kappa-in-plain-english (https://stats.stackexchange.com/questions/82162/cohens-kappa-in-plain-english)).

The calculated OOB (out-of-bag) error for this model is 1.83 %.

Applying the model to 20 test cases

```
pred <- predict(modFit, testing)
pred
```

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