Predict activity execution quality

By: Oleg Rozenshtain

This report is written as a solution to the "Practical Machine Learning" course project, as part of the "data science" specialization by Johns Hopkins University on Coursera.

Background

This human activity recognition research has traditionally focused on discriminating between different activities, i.e. to predict "which" activity was performed at a specific point in time or quantify "how much" of a particular activity they do. The approach that was proposed in the Weight Lifting Exercises dataset[1] is to investigate "how (well)" an activity was performed by the wearer. The "how (well)" investigation has only received little attention so far, even though it potentially provides useful information for a large variety of applications, such as sports training.

Data collection

For data recording four 9 degrees of freedom Razor inertial measurement units (IMU) were used, which provide three-axes acceleration, gyroscope and magnetometer data at a joint sampling rate of 45 Hz. The sensors were mounted in the users' glove, armband, lumbar belt and dumbbell. 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 corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. It was made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Feature extraction

For feature extraction a sliding window approach with different lengths from 0.5 second to 2.5 seconds, with 0.5 second overlap was used. In each step of the sliding window approach features on the Euler angles (roll, pitch and yaw) were calculated, as well as the raw accelerometer, gyroscope and magnetometer readings. For the Euler angles of each of the four sensors eight features were calculated: mean, variance, standard deviation, max, min, amplitude, kurtosis and skewness. So the data set has 160 parameters:

- Row number.
- user_name – subjects' name.
- raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp – 3 timestamp formats for sample.
- new_window – indicator if a window ends (values: yes/no).
- num_window – window number.
- roll/pitch/yaw_belt/arm/dumbbell/forearm - Euler angles for each sensor.
- avg/var/stddev/max/ min/amplitude/kurtosis/skewness_roll/pitch/yaw_belt/arm/ dumbbell/forearm – summary statistics in a specific window for each one of the Euler angles, for each sensor.

[1] Velloso, 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.

- gyros/accel/magnet_belt/arm/dumbbell/forearm _x/y/z - raw accelerometer, gyroscope and magnetometer readings, for each sensor.
- total_accel_belt/arm/dumbbell/forearm – total acceleration for each sensor.
- var_total_accel_belt/arm/dumbbell/forearm – variance of total acceleration in a specific window, for each sensor.
- classe – one of five ways to execute the activity (A, B, C, D, E). Prediction outcome.

Pre-Processing

Using sliding window approach we get number of records for each window (sampling rate of 45 Hz), but the summary statistics calculated for each window once. That leaves a lot of empty cells in the data frame. Further investigation of the summary statistics shows that even where there are values a lot of them are "#DIV/0!" errors, or values that just are incorrect. For example num_window = 12, 12 out of the summary statistics have "#DIV/0!" values, and the max_picth_belt value is 3 where all pitch_belt measurements are above 8. Therefore, the summary statistics are not good parameters to classify by, and I will remove them from the set. Furthermore the first 7 parameters hold general information about the records and are not good to predict by, so I will remove them as well.

```
# load activity execution quality training set
> executionQualityTainingData <- read.table(file = "pml-training.csv", sep = ",",
+                                           header = TRUE, comment.char = "",
+                                           na.strings = c("NA", "#DIV/0!"),
+                                           colClasses = c("integer", "factor",
+                                                          "integer", "integer",
+                                                          "factor", "factor",
+                                                          "integer",
+                                                          rep("numeric", 38*4),
+                                                          "factor"))
> # count NA values in each column
> naCount<-sapply(executionQualityTainingData,
+                                      function(y) sum(length(which(is.na(y)))))
> # remove columns with majority NA values
> removeColumns <- naCount < 19000
>  executionQualityTainingDataClean <-
+                                      executionQualityTainingData[ , removeColumns]
> # remove general information columns
> executionQualityTainingDataClean <- subset(executionQualityTainingDataClean,
+                                            select = -c(X, user_name,
+                                                        raw_timestamp_part_1,
+                                                        raw_timestamp_part_2,
+                                                        cvtd_timestamp,
+                                                        new_window,
+                                                        num_window))
> # make sure no more NA values exist
> sum(is.na(executionQualityTainingDataClean))
[1] 0
```

After this first classification parameters reduction to 52 parameters, further parameter reduction may be achieved by principal components analysis. It makes sense that measures from the four sensors recording body movement on number of different axis would be correlated.

```
> library(caret)
> preProc <- preProcess(executionQualityTainingDataClean[ , -53], method = "pca")
> # transform parameters to principal components
> executionQualityTainingDataCleanPC <-
+                         predict(preProc, executionQualityTainingDataClean[ , -53])
```

Now classification parameters reduced to 25, this will simplify model construction and selection.

Model

Next step is choosing the best classification model. Try to fit several models, and choose the most accurate one.

```
> set.seed(1209)
> # Random Forest
> rfModelFit <- train(executionQualityTainingDataClean$classe ~ ., method = "rf",
+                         data = executionQualityTainingDataCleanPC)

> set.seed(1209)
> # Stochastic Gradient Boosting (boosting with trees)
> gbmModelFit <- train(executionQualityTainingDataClean$classe ~ ., method = "gbm",
+                          trControl = trainControl(method = "cv"),
+                          data = executionQualityTainingDataCleanPC, verbose = FALSE)

> set.seed(1209)
> # Linear Discriminant Analysis
> ldaModelFit <- train(executionQualityTainingDataClean$classe ~ ., method = "lda",
+                         trControl = trainControl(method = "cv"),
+                         data = executionQualityTainingDataCleanPC)

> set.seed(1209)
> # Quadratic Discriminant Analysis
> qdaModelFit <- train(executionQualityTainingDataClean$classe ~ ., method = "qda",
+                          trControl = trainControl(method = "cv"),
+                          data = executionQualityTainingDataCleanPC)

> set.seed(1209)
> # Naive Bayes
> nbModelFit <- train(executionQualityTainingDataClean$classe ~ ., method = "nb",
+                         trControl = trainControl(method = "cv"),
+                        data = executionQualityTainingDataCleanPC)
```

For parameter tuning, and for performance assessment of out of sample errors, I used cross-validation with default 10-fold method in all models except of the random forest model (the error can be assessed using out of bagging error, no need for cross-validation).
For choosing the most accurate method let's look at the accuracy of all models:

```
> max(rfModelFit$results$Accuracy)
[1] 0.9748373
> max(gbmModelFit$results$Accuracy)
[1] 0.8236695
> max(ldaModelFit$results$Accuracy)
[1] 0.5265528
> max(qdaModelFit$results$Accuracy)
[1] 0.7386618
> max(nbModelFit$results$Accuracy)
[1] 0.651822
```

It is prominent that linear models such as linear discriminant analysis or naive bayes don't fit as prediction models. Quadratic discriminant analysis and boosting with trees are better to predict, but the best model is random forest with a very high 0.97 accuracy. Let's further explore the chosen model

```
> rfModelFit$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: 2


      OOB estimate of error rate: 1.67%
Confusion matrix:
     A    B    C    D    E  class.error
A 5554    7   12    4    3 0.004659498
B   52 3713   27    1    4 0.022122728
C    3   33 3357   26    3 0.018994740
D    2    1  105 3103    5 0.035136816
E    0    9   18   12 3568 0.010812309
```

The chosen model consist out of 500 trees forest (default model value), and in each split 2 variables are chosen. The out of bagging error is 1.67% (which we can expect in the testing set and other out of sample data) is very low and satisfies the goal of this project.

Prediction

The final part of the project is to predict the activity execution quality of the test set. First need to pre-process the test set in the same manner as the training set. That is, remove unnecessary columns and apply to the remaining parameters the same principal component

transformation as was applied to the training set. Then, use the chosen random forest model to classify the data records.

```
> # load activity execution quality testing set
> executionQualityTestingData <- read.table(file = "pml-testing.csv", sep = ",",
+                                           header = TRUE, comment.char = "",
+                                           na.strings = c("NA", "#DIV/0!"),
+                                           colClasses = c("integer", "factor",
+                                                          "integer", "integer",
+                                                          "factor", "factor",
+                                                          "integer",
+                                                          rep("numeric", 38*4),
+                                                          "integer"))
> # remove columns with majority NA values
> executionQualityTestingDataClean <-
+                                    executionQualityTestingData[ , removeColumns]
> # remove general information columns
> executionQualityTestingDataClean <- subset(executionQualityTestingDataClean,
+                                            select = -c(X, user_name,
+                                                        raw_timestamp_part_1,
+                                                        raw_timestamp_part_2,
+                                                        cvtd_timestamp,
+                                                        new_window,
+                                                        num_window))
> executionQualityTestingDataCleanPC <-
+                                 predict(preProc, executionQualityTestingDataClean[ , -53])
>
>    data.frame(problem_id = executionQualityTestingDataClean$problem_id,
+               classe =  predict(rfModelFit, executionQualityTestingDataCleanPC))
```