Assignment Practical Machine Learning

Long Nguyen Hoang

26/07/2018

Executive summary

In this assignment I look at the Weight Lifting Exercises Dataset (http://groupware.les.inf.puc-rio.br/har) of Velloso et al. (2013). I will use this data to train a model to predict the exercise performance. First, the dataset is explored and cleaned. The dataset is then splitted. One part is used to train a model using random forest method. The smaller part is used to test and estimate the out of sample error. The constructed model is quite good with expected out of error sample only about 4%.

Exploratory analyse and cleaning data

First, the training data is loaded and explored.

```
pmlTrain <- read.csv("pml-training.csv")
dim(pmlTrain)
```

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

There are 19622 records of 160 variables of the data. The variables include:
- X = index
- Username = names of 6 males participants
- raw_timestamp (part 1 and part 2) and cvtd_timestamp
- new_window and num_window

```
head(pmlTrain[ , c(1:7)], 4)
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 1 1  carlitos           1323084231               788290 05/12/2011 11:23
## 2 2  carlitos           1323084231               808298 05/12/2011 11:23
## 3 3  carlitos           1323084231               820366 05/12/2011 11:23
## 4 4  carlitos           1323084232               120339 05/12/2011 11:23
##   new_window num_window
## 1         no         11
## 2         no         11
## 3         no         11
## 4         no         12
```

- 38 measurements x 4 sensors on belt, arm, dumbbell, forearm (glove). The measurements are:
  - 3 records at 3 Euler angles (roll, pitch, yaw)
  - 2 records of total and variance of acceleration
  - 24 records of 8 features (average, standard deviation, variance, kurtosis, skewness, max, min, amplitude) x 3 Euler angles (roll, pitch, yaw)
  - 9 records of 3 directions (x, y, z) x 3 raw readings of acceleration, gyroscope and magnetometer. Example of variables from arm sensors:

```
grep("_arm", names(pmlTrain), value=T)
```

```
##  [1] "roll_arm"            "pitch_arm"           "yaw_arm"
##  [4] "total_accel_arm"     "var_accel_arm"       "avg_roll_arm"
##  [7] "stddev_roll_arm"     "var_roll_arm"        "avg_pitch_arm"
## [10] "stddev_pitch_arm"    "var_pitch_arm"       "avg_yaw_arm"
## [13] "stddev_yaw_arm"      "var_yaw_arm"         "gyros_arm_x"
## [16] "gyros_arm_y"         "gyros_arm_z"         "accel_arm_x"
## [19] "accel_arm_y"         "accel_arm_z"         "magnet_arm_x"
## [22] "magnet_arm_y"        "magnet_arm_z"        "kurtosis_roll_arm"
## [25] "kurtosis_picth_arm"  "kurtosis_yaw_arm"    "skewness_roll_arm"
## [28] "skewness_pitch_arm"  "skewness_yaw_arm"    "max_roll_arm"
## [31] "max_picth_arm"       "max_yaw_arm"         "min_roll_arm"
## [34] "min_pitch_arm"       "min_yaw_arm"         "amplitude_roll_arm"
## [37] "amplitude_pitch_arm" "amplitude_yaw_arm"
```

- classe = how well they did the exercise (A = exactly according to the specification, B = throwing the elbows to the front, C = lifting the dumbbell only halfway, D = lowering the dumbbell only halfway, E = throwing the hips to the front).

The classe will be predicted by the measurement data only. First, variables having little variance are identified for removal since they likely are not good predictors:

```
library(caret)
```

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

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

```
pmlTrain <- pmlTrain[ , -c(1:7)]
nzv <- nearZeroVar(pmlTrain[ , -153])
names(pmlTrain[ , nzv])
```

```
##  [1] "kurtosis_roll_belt"      "kurtosis_picth_belt"
##  [3] "kurtosis_yaw_belt"       "skewness_roll_belt"
##  [5] "skewness_roll_belt.1"    "skewness_yaw_belt"
##  [7] "max_yaw_belt"            "min_yaw_belt"
##  [9] "amplitude_yaw_belt"      "avg_roll_arm"
## [11] "stddev_roll_arm"         "var_roll_arm"
## [13] "avg_pitch_arm"           "stddev_pitch_arm"
## [15] "var_pitch_arm"           "avg_yaw_arm"
## [17] "stddev_yaw_arm"          "var_yaw_arm"
## [19] "kurtosis_roll_arm"       "kurtosis_picth_arm"
## [21] "kurtosis_yaw_arm"        "skewness_roll_arm"
## [23] "skewness_pitch_arm"      "skewness_yaw_arm"
## [25] "max_roll_arm"            "min_roll_arm"
## [27] "min_pitch_arm"           "amplitude_roll_arm"
## [29] "amplitude_pitch_arm"     "kurtosis_roll_dumbbell"
## [31] "kurtosis_picth_dumbbell" "kurtosis_yaw_dumbbell"
## [33] "skewness_roll_dumbbell"  "skewness_pitch_dumbbell"
## [35] "skewness_yaw_dumbbell"   "max_yaw_dumbbell"
## [37] "min_yaw_dumbbell"        "amplitude_yaw_dumbbell"
## [39] "kurtosis_roll_forearm"   "kurtosis_picth_forearm"
## [41] "kurtosis_yaw_forearm"    "skewness_roll_forearm"
## [43] "skewness_pitch_forearm"  "skewness_yaw_forearm"
## [45] "max_roll_forearm"        "max_yaw_forearm"
## [47] "min_roll_forearm"        "min_yaw_forearm"
## [49] "amplitude_roll_forearm"  "amplitude_yaw_forearm"
## [51] "avg_roll_forearm"        "stddev_roll_forearm"
```

```
## [53] "var_roll_forearm"        "avg_pitch_forearm"
## [55] "stddev_pitch_forearm"    "var_pitch_forearm"
## [57] "avg_yaw_forearm"         "stddev_yaw_forearm"
## [59] "var_yaw_forearm"
```

```
pmlTrain <- pmlTrain[ , -nzv]
```

59 near-zero-variance variables have been removed. In the 94 variables left, there are many variables containing mostly NA values (~19216 records).

```
naVar <- apply(pmlTrain, MARGIN=2, FUN=function(x) sum(is.na(x)))
naVar[naVar > 0]
```

```
##            max_roll_belt           max_picth_belt            min_roll_belt
##                    19216                    19216                    19216
##           min_pitch_belt      amplitude_roll_belt     amplitude_pitch_belt
##                    19216                    19216                    19216
##     var_total_accel_belt            avg_roll_belt         stddev_roll_belt
##                    19216                    19216                    19216
##            var_roll_belt           avg_pitch_belt        stddev_pitch_belt
##                    19216                    19216                    19216
##           var_pitch_belt             avg_yaw_belt          stddev_yaw_belt
##                    19216                    19216                    19216
##             var_yaw_belt            var_accel_arm            max_picth_arm
##                    19216                    19216                    19216
##              max_yaw_arm              min_yaw_arm        amplitude_yaw_arm
##                    19216                    19216                    19216
##        max_roll_dumbbell       max_picth_dumbbell        min_roll_dumbbell
##                    19216                    19216                    19216
##       min_pitch_dumbbell  amplitude_roll_dumbbell amplitude_pitch_dumbbell
##                    19216                    19216                    19216
##       var_accel_dumbbell        avg_roll_dumbbell     stddev_roll_dumbbell
##                    19216                    19216                    19216
##        var_roll_dumbbell       avg_pitch_dumbbell    stddev_pitch_dumbbell
##                    19216                    19216                    19216
##       var_pitch_dumbbell         avg_yaw_dumbbell      stddev_yaw_dumbbell
##                    19216                    19216                    19216
##         var_yaw_dumbbell        max_picth_forearm        min_pitch_forearm
##                    19216                    19216                    19216
##  amplitude_pitch_forearm        var_accel_forearm
##                    19216                    19216
```

These variables are also need to be removed

```
pmlTrain <- pmlTrain[ , -which(naVar > 0)]
dim(pmlTrain)
```

```
## [1] 19622    53
```

The tidy data now have 19622 records of 53 variables.

Building model

Random forest will be used to predict the classe of performance because it has high accuracy and it is also my favourite. Cross validaton will be performed by bootstrapping with 10 times resampling. However, bootrap is random sampling with replacement so it tend to underestimate the out of sample error. Therefore, the train data is splited into 2 parts, one for training and one is kept for estimating the out of sample error.

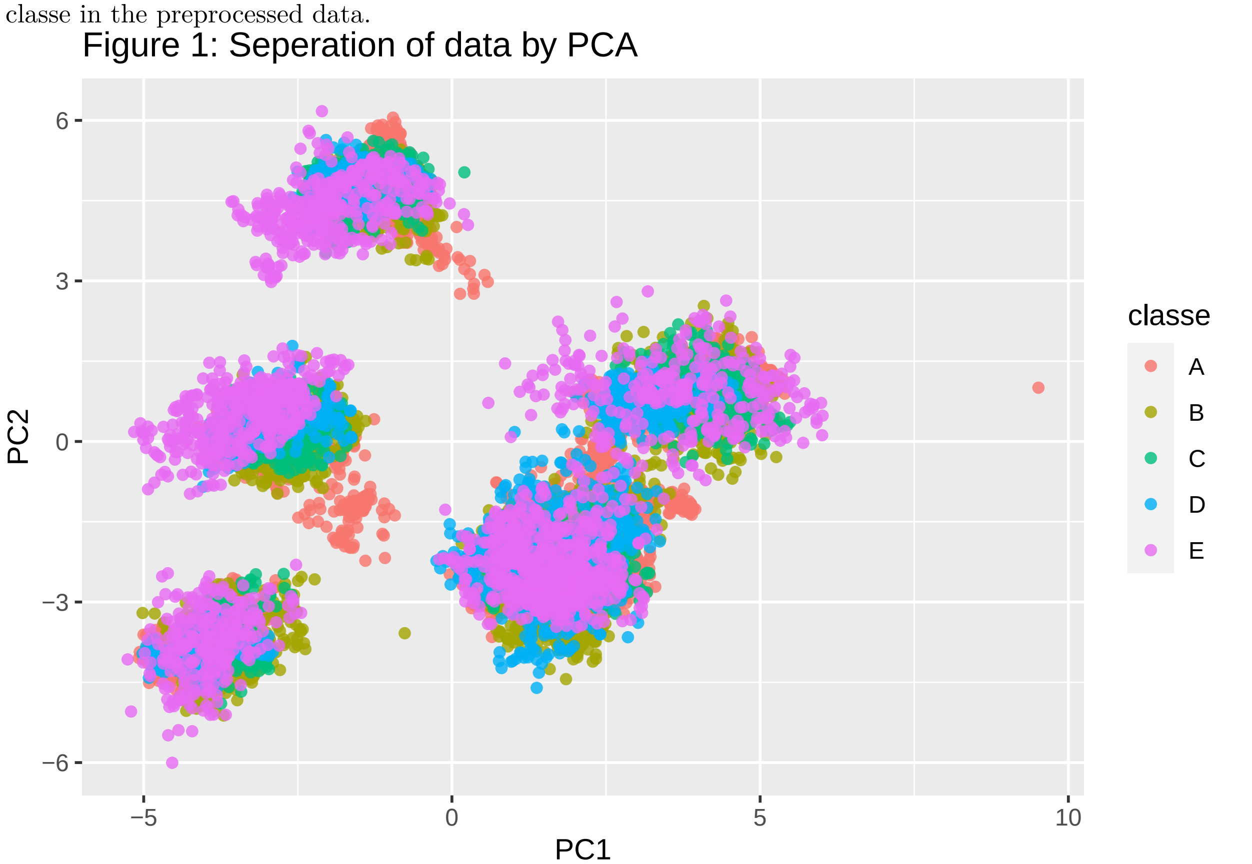
```r
set.seed=50
inTrain <- createDataPartition(y=pmlTrain$classe, p=0.6, list=F)
training <- pmlTrain[inTrain, ]
testing <- pmlTrain[-inTrain, ]
```

The data is first preprocessed with Principle Component Analysis. This helps to reduce the number of predictors by removing unecessary highly correlated predictors, and therefore reduce the noise and complexity.

```r
preProc <- preProcess(training[ ,-53], method="pca", thresh=0.90)
trainPC <- predict(preProc, newdata=training[ ,-53])
trainPC$classe <- training$classe
dim(trainPC)
```

```
## [1] 11776    19
```

With a 90% cutoff for the cumulative percent of variance to be retained, there are only 18 predictors and classe in the preprocessed data.



The figure 1 shows seperation between classe in 3 data cloud on the left. This illustrates how effective the preprocessing with PCA is. Now the model is trained with the new training dataset.

```r
set.seed=100
modFitPC <- train(classe ~., method="rf", data=trainPC,
                trControl=trainControl(method="boot", number=10))
modFitPC
```

```
## Random Forest
##
## 11776 samples
##    18 predictor
```

```
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (10 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9532256  0.9408188
##   10    0.9409773  0.9253390
##   18    0.9244653  0.9044478
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

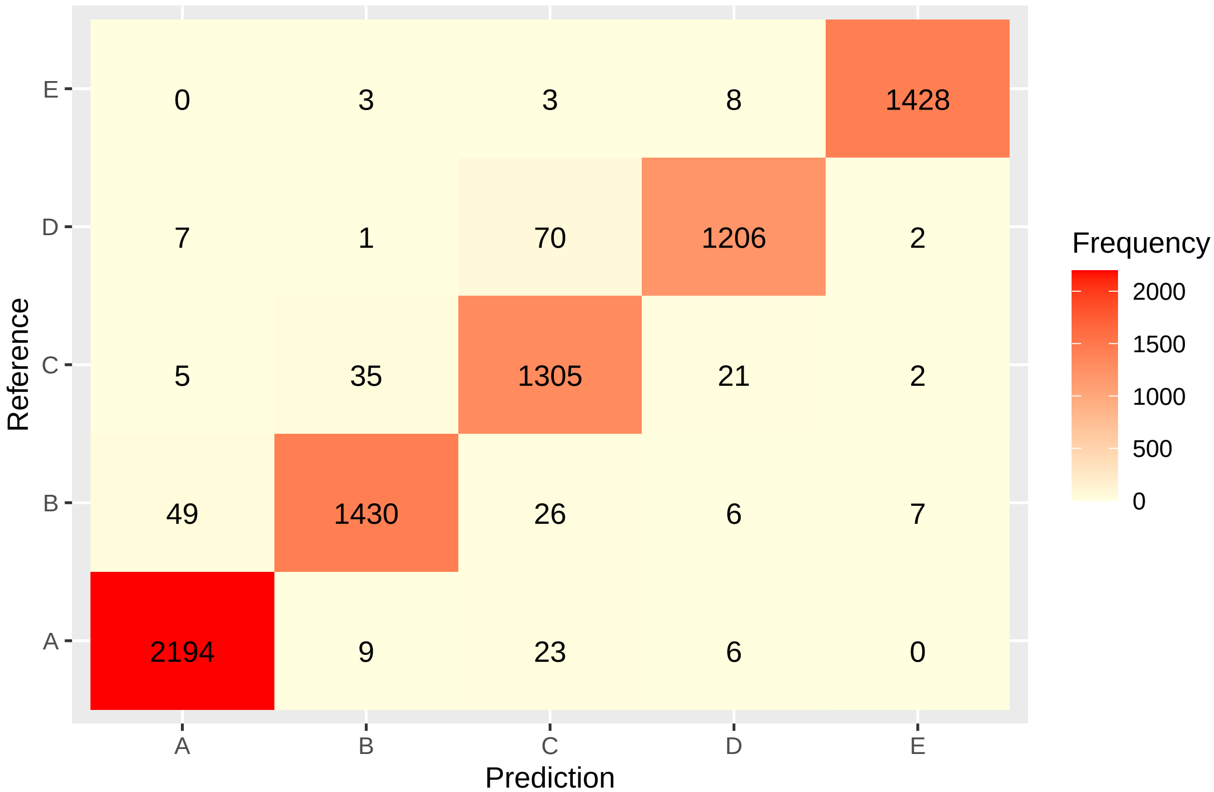
The accuracy was in range of 92% to 95%. The out of sample error is estimated with the testing data.

```r
testPC <- predict(preProc, newdata=testing[ ,-53])
testPC$classe <- testing$classe
predPC <- predict(modFitPC, newdata=testPC)
confusionMatrix(predPC,as.factor(testPC$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2194   49    5    7    0
##          B    9 1430   35    1    3
##          C   23   26 1305   70    3
##          D    6    6   21 1206    8
##          E    0    7    2    2 1428
##
## Overall Statistics
##
##                Accuracy : 0.9639
##                  95% CI : (0.9596, 0.9679)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9544
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9830   0.9420   0.9539   0.9378   0.9903
## Specificity            0.9891   0.9924   0.9812   0.9938   0.9983
## Pos Pred Value         0.9729   0.9675   0.9145   0.9671   0.9924
## Neg Pred Value         0.9932   0.9862   0.9902   0.9879   0.9978
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2796   0.1823   0.1663   0.1537   0.1820
## Detection Prevalence   0.2874   0.1884   0.1819   0.1589   0.1834
## Balanced Accuracy      0.9861   0.9672   0.9676   0.9658   0.9943
```

The out of sample error is expected to be 1 - Accuracy = 0.044 or 4%. This model is therefore quite good.

Figure 2: Summary confusion matrix



The testing data is now loaded to predict the classe of performance.

```r
pmlTest <- read.csv("pml-testing.csv")
predictPC <- predict(preProc, newdata=pmlTest)
pred <- predict(modFitPC, newdata=predictPC)
problem <- data.frame(problem_id=predictPC$problem_id, predicton=pred)
problem
```

```
##    problem_id predicton
## 1           1         B
## 2           2         A
## 3           3         A
## 4           4         A
## 5           5         A
## 6           6         E
## 7           7         D
## 8           8         B
## 9           9         A
## 10         10         A
## 11         11         B
## 12         12         C
## 13         13         B
## 14         14         A
## 15         15         E
## 16         16         E
## 17         17         A
## 18         18         B
## 19         19         B
```

```
## 20              20              B
```

Conclusion

After cleaning and preprocessing the Weight Lifting Exercises Dataset, I fit a random forest model with 18 predictors to predict the exercise performance. The constructed model is quite good with expected out of error sample only about 4%.

Reference

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.