Practical machine learning

2023-10-02

Overview

The goal is 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 (classe A to E). "Six young health 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)." (Read more: http:/groupware.les.inf.puc-rio.br/har#ixzz4TphcNzin)

In section "Loading and cleaning" we load and pre-process the data, to remove the "NA" and variables "trivially correlated" such as the index ("X"), ot the time (See Fig. 1, in the Appendix). In the section "Model" we train our model, using a random forest (without principal component analysis). We used 80% of the observations for training and the remaining 20% for testing. In section "Prediction" we predict the values for the "testing data" provided in this exercise.

The cross validation and the estimation of the out of the sample error are shown in the summary. We performed (k-folds, k=10, as this is assumed to be a good balance between data variability and bias). We analysed the accuracy of 4 different training models: multivariate, regression trees ("rpart"), boosting ("gbm") and random forest "rf". We also evaluated the difference between pre-processing with and without principal component analysis "pca". We show that the random forest without "pca" is the model that performs better. Preliminary analysis of the data is shown in the Apendix 2.

Loading and cleaning

```
data <- read.csv("/home/paula/Documents/DATA_SCIENCE/practical_ML/project/pml-training.csv")
dim(data)
```

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

```
data$classe<-as.factor(data$classe)

set.seed(23457)
inTrain<-createDataPartition(y=data$classe,p=0.8,list=FALSE)
training<-data[inTrain,]
testing<-data[-inTrain,]
```

```
accur<-function(values,prediction){confusionMatrix(values$classe, prediction)}
```

pre-processing:removing NA and non numerical values (" "). (See Appendix 2.)

```
Y<- c()
for (i in c(1:length(names(training)))) {
    X<-(is.na(as.numeric(training[,i])))
    Y[i]<-(dim(training[X,])[1]==0)
}
    Clean_training<-training[,Y]
```

```r
Final_training<-Clean_training[,c(5:dim(Clean_training)[2])]  #removing un-meaningful variables
dim(Final_training)
```

```
## [1] 15699    53
```

```r
names(Final_training)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

repeating the pre-processing for the testing set

```r
Clean_testing<-testing[,Y]
Final_testing<-Clean_testing[,c(5:dim(Clean_training)[2])]
```

Training the full model:

```r
FIT_final <-train(classe~. , data=Final_training,   method="rf")
```

```r
Predict_final    <-predict(FIT_final , Final_testing)
```

```r
accur(Final_testing,Predict_final  )$overall[1]
```

```
##  Accuracy
## 0.9959215
```

Conclusions: Predicting values in the test set provided:

```r
test_data <- read.csv("/home/paula/Documents/DATA_SCIENCE/practical_ML/project/pml-testing.csv")
predict_values<-predict(FIT_final, test_data)
predict_values
```

```
##  [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
```

Appendix

Cross-reference and model selection:

We will use 4 different models with and without principal component "pca" pre-processing. We use cross validation with k-fold (with 10 folds) We will test the Multinomial Log-linear Models "multinom", regression trees "rpart", boosting "gbm" and random forest "rf".

```
DATA<-Final_training
train_control<- trainControl(method="cv", number=10, savePredictions = TRUE)

#insmall<-createDataPartition(y=Final_training$classe,p=0.2,list=FALSE)
#small_test<-Final_training[insmall,]

#DATA<-small_test
```

With "pca"

```
FIT_multinorm<-train(classe~. , data=DATA,  preProcess="pca", method="multinom", trControl=train_control
```

```
FIT_rpart<-train(classe~. , data=DATA,  preProcess="pca", method="rpart", trControl=train_control)
```

```
FIT_boosting<-train(classe~. , data=DATA,  preProcess="pca", method="gbm", trControl=train_control,verb
```

```
FIT_rf<-train(classe~. , data=DATA,  preProcess="pca", method="rf", trControl=train_control)
```

Without "pca"

```
FIT_multinorm2<-train(classe~. , data=DATA,   method="multinom", trControl=train_control, verbose=FALSE
```

```
FIT_rpart2<-train(classe~. , data=DATA,   method="rpart", trControl=train_control)
```

```
FIT_boosting2<-train(classe~. , data=DATA,   method="gbm", trControl=train_control,verbose=FALSE)
```

```
FIT_rf2<-train(classe~. , data=DATA,   method="rf", trControl=train_control)
```

```
results_training<-data.frame(model=character(), out_the_sample_Error=double())
results_training<-rbind(results_training, data.frame(model="multinorm/pca"
                        ,out_the_sample_Error=mean((FIT_multinorm$resample)[1]$Accuracy) ))
results_training<-rbind(results_training, data.frame(model="rpart/pca"
                        ,out_the_sample_Error=mean((FIT_rpart$resample)[1]$Accuracy)     ))
results_training<-rbind(results_training, data.frame(model="gbm/pca"
                        ,out_the_sample_Error=mean((FIT_boosting$resample)[1]$Accuracy)  ))
results_training<-rbind(results_training, data.frame(model="rf/pca"
                        ,out_the_sample_Error=mean((FIT_rf$resample)[1]$Accuracy)        ))
results_training<-rbind(results_training, data.frame(model="multinorm"
                        ,out_the_sample_Error=mean((FIT_multinorm2$resample)[1]$Accuracy)    ))
results_training<-rbind(results_training, data.frame(model="rpart"
                        ,out_the_sample_Error=mean((FIT_rpart2$resample)[1]$Accuracy)        ))
results_training<-rbind(results_training, data.frame(model="gbm"
                        ,out_the_sample_Error=mean((FIT_boosting2$resample)[1]$Accuracy)     ))
results_training<-rbind(results_training, data.frame(model="rf"
                        ,out_the_sample_Error=mean((FIT_rf2$resample)[1]$Accuracy)           ))
results_training
```

```
##           model out_the_sample_Error
## 1 multinorm/pca            0.5280577
## 2     rpart/pca            0.3596415
## 3       gbm/pca            0.8123461
```

```
## 4        rf/pca            0.9772588
## 5     multinorm            0.6679392
## 6         rpart            0.5034705
## 7           gbm            0.9614646
## 8            rf            0.9938208
```

Testing the models in the data set we have:

```
Predict_multinorm    <-predict(FIT_multinorm , Final_testing)
Predict_rpart        <-predict(FIT_rpart , Final_testing)
Predict_boosting     <-predict(FIT_boosting , Final_testing)
Predict_rf           <-predict(FIT_rf , Final_testing)

Predict_multinorm2   <-predict(FIT_multinorm2, Final_testing)
Predict_rpart2       <-predict(FIT_rpart2 , Final_testing)
Predict_boosting2    <-predict(FIT_boosting2 , Final_testing)
Predict_rf2          <-predict(FIT_rf2 , Final_testing)
```

```
results_test<-data.frame(model=character(), test_accuracy=double())

results_test<-rbind(results_test,data.frame(model="multinorm/pca"
                  , test_accuracy=accur(Final_testing,Predict_multinorm  )$overall[1]))
results_test<-rbind(results_test,data.frame(model="rpart/pca"
                  , test_accuracy=accur(Final_testing,Predict_rpart      )$overall[1]))
results_test<-rbind(results_test,data.frame(model="gbm/pca"
                  , test_accuracy=accur(Final_testing,Predict_boosting   )$overall[1]))
results_test<-rbind(results_test,data.frame(model="rf/pca"
                  , test_accuracy=accur(Final_testing,Predict_rf         )$overall[1]))
results_test<-rbind(results_test,data.frame(model="multinorm"
                  , test_accuracy=accur(Final_testing,Predict_multinorm2 )$overall[1]))
results_test<-rbind(results_test,data.frame(model="rpart"
                  , test_accuracy=accur(Final_testing,Predict_rpart2     )$overall[1]))
results_test<-rbind(results_test,data.frame(model="gbm"
                  , test_accuracy=accur(Final_testing,Predict_boosting2  )$overall[1]))
results_test<-rbind(results_test,data.frame(model="rf"
                  , test_accuracy=accur(Final_testing,Predict_rf2        )$overall[1]))
results_test
```

```
##                    model test_accuracy
## Accuracy  multinorm/pca     0.5388733
## Accuracy1     rpart/pca     0.3800663
## Accuracy2       gbm/pca     0.8159572
## Accuracy3        rf/pca     0.9811369
## Accuracy4     multinorm     0.6507775
## Accuracy5         rpart     0.4825389
## Accuracy6           gbm     0.9655876
## Accuracy7            rf     0.9961764
```

What gives similar accuracy than the one found by cross validation, i.e. 0.9961763956156.

Apendix 2 pre-analaysing the data

this data contains 19622, 1 (-1) possible predictors and 19622, 1 observations.

```
names(data)
```

```
##   [1] "X"                        "user_name"
##   [3] "raw_timestamp_part_1"     "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"           "new_window"
##   [7] "num_window"               "roll_belt"
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
```

```
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

```
head(data$classe)
```

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

The variable we are interested in predicting is a factor, so I will conver it to factor for proper interpretability. After looking at the data there area several variables that do not contain more than 1% of the values, so I want to clean the data set from these variables:

```
Y<- c()
Z<- c()
for (i in c(1:length(names(training)))) {
    X<-(is.na(as.numeric(training[,i]))) #these are the NA values
    Z[i]<-(1-(dim(training[X,])[1]/dim(training[,])[1]))*100 # this is the percentage of "good observab
    Y[i]<-(dim(training[X,])[1]==0)   #This is the vector with the indexes of the complete data
}
    Clean_training<-training[,Y]
    dim(Clean_training)
```

```
## [1] 15699    57
```

```
    names(Clean_training)
```

```
##  [1] "X"                    "raw_timestamp_part_1" "raw_timestamp_part_2"
##  [4] "num_window"           "roll_belt"            "pitch_belt"
##  [7] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
## [10] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [13] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [16] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [19] "pitch_arm"            "yaw_arm"              "total_accel_arm"
```

```
## [22] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [25] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [28] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [31] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [34] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [37] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [40] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [43] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [46] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [49] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [52] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [55] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

The variables X, raw_timestamp_part_1, raw_timestamp_part_2, num_window are not correlated or has a trivial correlation with "classe" (as the case of X, that is an index), and, therefore, should not be taken into account).

```
(Clean_training[,c(1:4,dim(Clean_training)[2])]) %>%
  gather(-classe, key = "var", value = "value") %>%
  ggplot(aes(x = value, y = classe, color = classe)) +
    geom_point() +
    facet_wrap(~ var, scales = "free") +
    theme_bw()
```

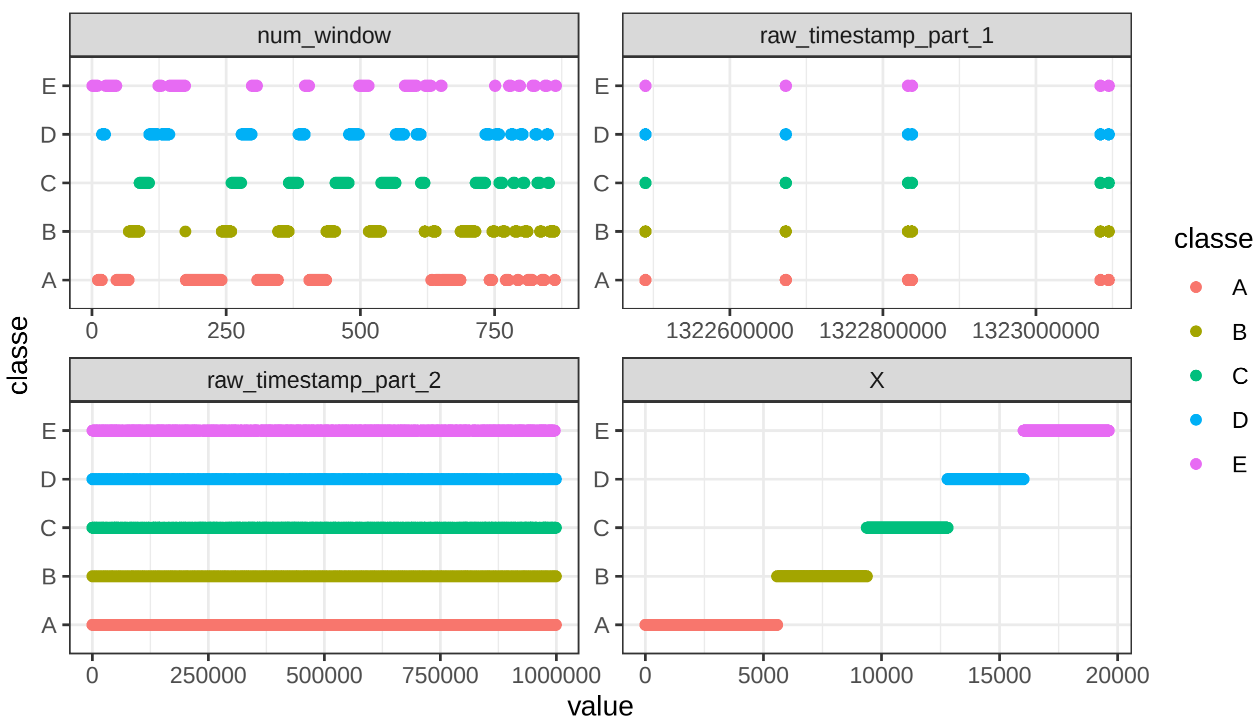

Figure 1: Fig