Machine Learning on Human Activity Recognition

yc

June 20, 2018

Summary

This project is about Human Activity Recognition

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, your 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.

Goal of this project

The goal for this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. Any of the other variables may be used to predict with. The accelerometers variables on the belt, forearm, arm, and dumbell are used to predict. This document contains: - how the model is built - the cross validation - expected out of sample error - prediction of 20 different test cases

Weight Lifting Exercises Dataset

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 (like with the Daily Living Activities dataset above). The approach we propose for the Weight Lifting Exercises dataset 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.

In this work (see the paper) we first define quality of execution and investigate three aspects that pertain to qualitative activity recognition: the problem of specifying correct execution, the automatic and robust detection of execution mistakes, and how to provide feedback on the quality of execution to the user. We tried out an on-body sensing approach (dataset here), but also an "ambient sensing approach" (by using Microsoft Kinect - dataset still unavailable)

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).

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. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

2 type of dataset

The project assignment includes two data files (in csv format) 1. Training data: pml-training.csv 2. Test data: pml-testing.csv

Import training dataset and apply data cleansing on "NA" and "#DIV/0!"

```
pml_training_data = read.table("C:/Users/yckoong/Downloads/pml-training.csv",
                               header = TRUE, sep = ",",
                               na.strings = c("NA", "#DIV/0!"))
dim(pml_training_data)
```

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

The dataframe has 19622 rows (observations) and 160 columns (variables). Most of the variables (152 out of 160) correspond to sensor readings for one of the four sensors. Those sensor-reading variable names (columns 8 to 159) include one of the following strings to identify the corresponding sensor:

_belt _arm _dumbbell _forearm

The last column in the data frame (column 160) contains the values A to E of the classe variable that indicates the execution type of the exercise.

Restricting the Variables to Sensor-related Ones.

Thus, the data in the first seven columns are not sensor readings. For the prediction purposes of this analysis, we will remove the data in those columns from the data frame (using grep to select the sensor-related columns).

```
sensorColumns = grep(pattern = "_belt|_arm|_dumbbell|_forearm", names(pml_training_data))
length(sensorColumns)
```

```
## [1] 152
```

```
data = pml_training_data[, c(sensorColumns,160)]
dim(data)
```

```
## [1] 19622   153
```

Handling NA Values

The selected sensor data columns still include many variables whose values are NA for almost all obervations. To remove those variables we do the following:

```
missingData = is.na(data)
omitColumns = which(colSums(missingData) > 19000)
data = data[, -omitColumns]
dim(data)
```

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

As you can see, only 53 predictor variables (plus classe) remain in the data set. Next we check that the resulting data frame has no missing values with:

```
table(complete.cases(data))
```

```
##
##  TRUE
## 19622
```

All of the remaining predictor variables are of numeric type:

```
table(sapply(data[1,], class))
```

```
##
##  factor integer numeric
##       1      25      27
```

Data Splitting and Discussion of Preprocessing.

Following the usual practice in Machine Learning, we will split our data into a training data set (75% of the total cases) and a testing data set (with the remaining cases; the latter should not be confused with the data in the pml-testing.csv file). This will allow us to estimate the out of sample error of our predictor. We will use the caret package for this purpose, and we begin by setting the seed to ensure reproducibility.

```
set.seed(2014)
library(caret)
```

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

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

```
inTrain <- createDataPartition(y=data$classe, p=0.75, list=FALSE)

training <- data[inTrain,]
dim(training)
```

```
## [1] 14718    53
```

```
testing <- data[-inTrain,]
dim(testing)
```

```
## [1] 4904   53
```

Fitting a model

We fit a predictive model for activity recognition using Random Forest algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general. We will use 5-fold cross validation when applying the algorithm.

```
library(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
```

```
fitRf <- train(classe ~ ., data=training, method="rf", trControl=trainControl(method="cv", 5), ntree=250
fitRf
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11774, 11775, 11775, 11773
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9904196  0.9878794
##   27    0.9916425  0.9894272
##   52    0.9857993  0.9820365
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Applying the Model to the Testing dataset.

After training the predictor we use it on the testing dataset we constructed before, to get an estimate of its out of sample error.

```
predictRf <- predict(fitRf, testing)
confusionMatrix(testing$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    2    0    0    1
##          B    8  940    1    0    0
##          C    0    2  848    5    0
##          D    0    1   13  789    1
##          E    0    0    3    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.9896, 0.9947)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9943   0.9947   0.9803   0.9937   0.9978
```

```
## Specificity            0.9991   0.9977   0.9983   0.9964   0.9993
## Pos Pred Value         0.9978   0.9905   0.9918   0.9813   0.9967
## Neg Pred Value         0.9977   0.9987   0.9958   0.9988   0.9995
## Prevalence             0.2855   0.1927   0.1764   0.1619   0.1835
## Detection Rate         0.2838   0.1917   0.1729   0.1609   0.1831
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9967   0.9962   0.9893   0.9950   0.9985
```

```
accuracy <- postResample(predictRf, testing$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9924551 0.9904554
```

```
oose <- 1 - as.numeric(confusionMatrix(testing$classe, predictRf)$overall[1])
oose
```

```
## [1] 0.007544861
```

So, the estimated accuracy of the model is 99.45, 99.3% and the estimated out-of-sample error is 0.55%.

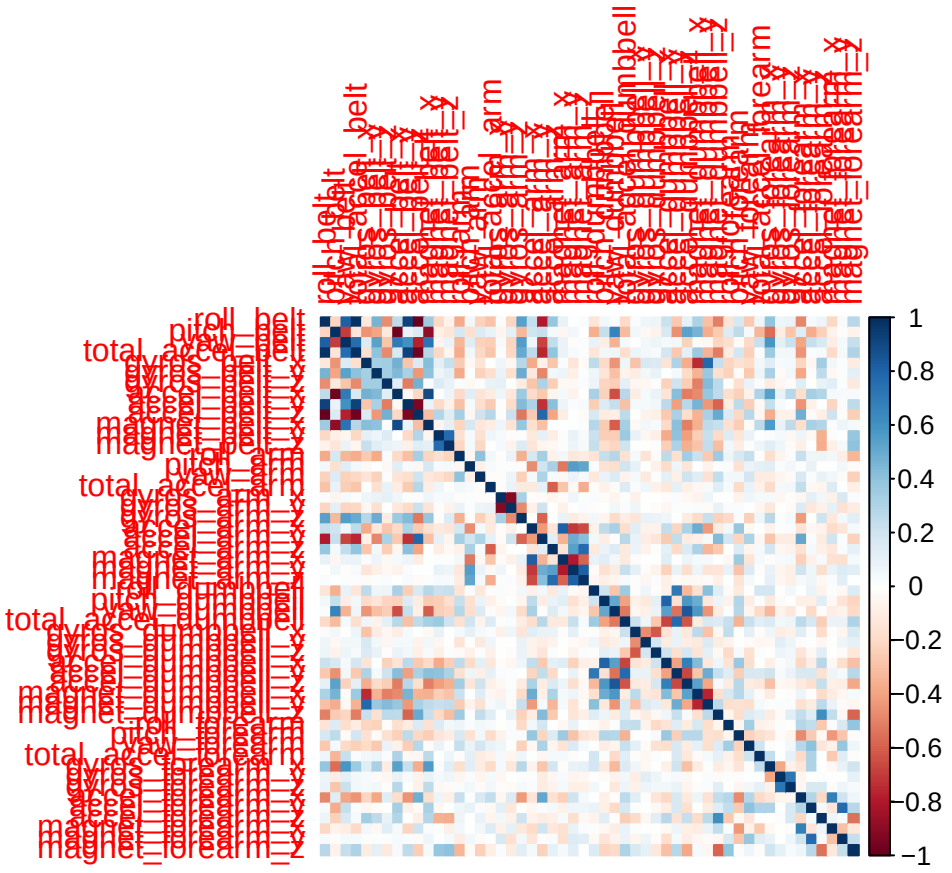
Plots

Correlation Matrix Visualization

```
library(corrplot)
```

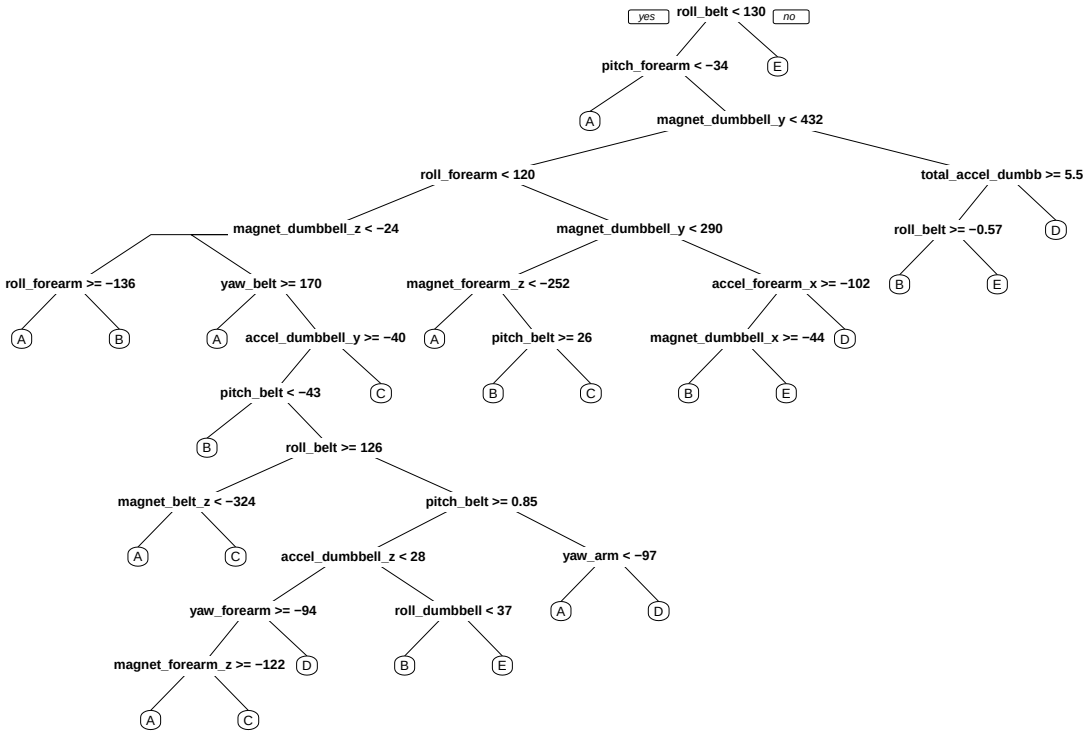
```
## corrplot 0.84 loaded
```

```
corrPlot <- cor(training[, -length(names(training))])
corrplot(corrPlot, method="color")
```



Tree Visualization

```
library(rpart)
library(rpart.plot)
treeModel <- rpart(classe ~ ., data=training, method="class")
prp(treeModel)
```


yes
roll_belt < 130
no
pitch_forearm < −34
E
A
magnet_dumbbell_y < 432
roll_forearm < 120
total_accel_dumbb >= 5.5
magnet_dumbbell_z < −24
magnet_dumbbell_y < 290
roll_belt >= −0.57
D
roll_forearm >= −136
yaw_belt >= 170
magnet_forearm_z < −252
accel_forearm_x >= −102
B
E
A
B
A
accel_dumbbell_y >= −40
A
pitch_belt >= 26
magnet_dumbbell_x >= −44
D
pitch_belt < −43
C
B
C
B
E
B
roll_belt >= 126
magnet_belt_z < −324
pitch_belt >= 0.85
A
C
accel_dumbbell_z < 28
yaw_arm < −97
yaw_forearm >= −94
roll_dumbbell < 37
A
D
magnet_forearm_z >= −122
D
B
E
A
C