Human Activity Recognition

Montassar H'Daya : Montassar.hdaya@gmail.com

July 26, 2015

Contents

Executive Summary

This detailed analysis has been performed to fulfill the requirements of the course project for the course Pratical Machine learning offered by the Johns Hopkins University on Data science specialization.

Human Activity Recognition - HAR - has emerged as a key research area in the last years and is gaining increasing attention by the pervasive computing research community (see picture below, that illustrates the increasing number of publications in HAR with wearable accelerometers), especially for the development of context-aware systems. There are many potential applications for HAR, like: elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises.

Human activity recognition research has traditionally focused on discriminating between different activities. However, 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 (http://groupware.les.inf.puc-rio.br/har).

For the prediction of how welll individuals performed the assigned exercise 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).

This report aims to use machine learning algoritmhs to predict the class of exercise the individuals was performing by using meaurements available from devices such as Jawbone Up, Nike FuelBand, and Fitbit.

Data

The data for this project come was obteined from http://groupware.les.inf.puc-rio.br/har. Two data set were available a training set and a test set for which 20 individuals without any classification for the class of exercise was available.

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Extract , transform and load the Data

```
setwd("~/R/R1/8.Machine Learning/Machine Learning Project")
pmlTrain<-read.csv("pml-training.csv", header=T, na.strings=c("NA", "#DIV/0!"))
pmlTest<-read.csv("pml-testing.csv", header=T, na.string=c("NA", "#DIV/0!"))
```

Training data was partitioned and preprocessed using the code described below. In brief, all variables with at least one "NA" were excluded from the analysis. Variables related to time and user information were excluded for a total of 51 variables and 19622 class measurements. Same variables were mainteined in the test data set (Validation dataset) to be used for predicting the 20 test cases provided.

```
## NA exclusion for all available variables
noNApmlTrain<-pmlTrain[, apply(pmlTrain, 2, function(x) !any(is.na(x)))]
dim(noNApmlTrain)
```

```
## [1] 19622    60
```

```
## variables with user information, time and undefined
cleanpmlTrain<-noNApmlTrain[,-c(1:8)]
dim(cleanpmlTrain)
```

```
## [1] 19622    52
```

```
## 20 test cases provided clean info - Validation data set
cleanpmltest<-pmlTest[,names(cleanpmlTrain[,-52])]
dim(cleanpmltest)
```

```
## [1] 20 51
```

Data Partitioning and Prediction Process

The cleaned downloaded data set was subset in order to generate a test set independent from the 20 cases provided set. Partitioning was performed to obtain a 75% training set and a 25% test set.

data cleaning

```
library(caret)
```

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

```
inTrain<-createDataPartition(y=cleanpmlTrain$classe, p=0.75,list=F)
training<-cleanpmlTrain[inTrain,]
test<-cleanpmlTrain[-inTrain,]
```

```
## Training and test set dimensions
```

```
dim(training)
```

```
## [1] 14718    52
```

```
dim(test)
```

```
## [1] 4904   52
```

Data analysis and estimate error with cross-validation

Random forest trees were generated for the training dataset using cross-validation. Then the generated algorithm was examnined under the partitioned training set to examine the accuracy and estimated error of prediction. By using 51 predictors for five classes using cross-validation at a 5-fold an accuracy of 99.2% with a 95% CI [0.989-0.994] was achieved accompanied by a Kappa value of 0.99.

```
library(caret)
library(e1071)
library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
set.seed(13333)
fitControl2<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
rffit<-train(classe~.,data=training, method="rf", trControl=fitControl2, verbose=F)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=26
## - Fold1: mtry=26
```

```
## + Fold1: mtry=51
## - Fold1: mtry=51
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=26
## - Fold2: mtry=26
## + Fold2: mtry=51
## - Fold2: mtry=51
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=26
## - Fold3: mtry=26
## + Fold3: mtry=51
## - Fold3: mtry=51
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=26
## - Fold4: mtry=26
## + Fold4: mtry=51
## - Fold4: mtry=51
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry=26
## - Fold5: mtry=26
## + Fold5: mtry=51
## - Fold5: mtry=51
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 26 on full training set
```

```
## Fitting mtry = 26 on full training set
predrf<-predict(rffit, newdata=test)
## Confusion Matrix and Statistics
confusionMatrix(predrf, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    8    0    0    0
##          B    0  934    5    0    0
##          C    0    5  850    7    1
##          D    0    1    0  796    4
##          E    0    1    0    1  896
##
## Overall Statistics
##
##                Accuracy : 0.9933
##                  95% CI : (0.9906, 0.9954)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9842   0.9942   0.9900   0.9945
## Specificity            0.9977   0.9987   0.9968   0.9988   0.9995
## Pos Pred Value         0.9943   0.9947   0.9849   0.9938   0.9978
## Neg Pred Value         1.0000   0.9962   0.9988   0.9981   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1905   0.1733   0.1623   0.1827
## Detection Prevalence   0.2861   0.1915   0.1760   0.1633   0.1831
## Balanced Accuracy      0.9989   0.9915   0.9955   0.9944   0.9970
```

```
pred20<-predict(rffit, newdata=cleanpmltest)
```

Output for the prediction of the 20 cases provided

```
pred20
```

```
##  [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
```

A boosting algorithm was also run to confirm and be able to compare predictions. Data is not shown but the boosting approach presented less accuracy (96%) (Data not shown). However, when the predictions for the 20 test cases were compared match was same for both ran algorimths.

```
library (gbm)
```

```
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
```

```
fitControl2<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
gmbfit<-train(classe~.,data=training, method="gbm", trControl=fitControl2, verbose=F)
```

```
## Loading required package: plyr
```

```
## + Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
```

```
## + Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.minobsinnode = 10 on full training s
```

```
gmbfit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 51 predictors of which 41 had non-zero influence.
```

```
class(gmbfit)
```

```
## [1] "train"         "train.formula"
```

```
predgmb<-predict(gmbfit, newdata=test)
confusionMatrix(predgmb, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1372   36    0    0    0
##          B   14  874   21    1   12
##          C    6   34  818   28    1
##          D    2    1   16  765   13
##          E    1    4    0   10  875
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9592
##                  95% CI : (0.9533, 0.9646)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9484
##  Mcnemar's Test P-Value : 0.000732
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9835   0.9210   0.9567   0.9515   0.9711
## Specificity            0.9897   0.9879   0.9830   0.9922   0.9963
## Pos Pred Value         0.9744   0.9479   0.9222   0.9598   0.9831
## Neg Pred Value         0.9934   0.9812   0.9908   0.9905   0.9935
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2798   0.1782   0.1668   0.1560   0.1784
## Detection Prevalence   0.2871   0.1880   0.1809   0.1625   0.1815
## Balanced Accuracy      0.9866   0.9544   0.9698   0.9718   0.9837
```

```
predtrain<-predict(gmbfit, newdata=training)
confusionMatrix(predtrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4153   70    0    3    2
##          B   18 2731   62   10   16
##          C   11   45 2473   64   18
##          D    2    0   26 2320   25
##          E    1    2    6   15 2645
##
## Overall Statistics
##
##                Accuracy : 0.9731
##                  95% CI : (0.9704, 0.9756)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.966
##  Mcnemar's Test P-Value : 4.472e-15
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9924   0.9589   0.9634   0.9619   0.9775
## Specificity            0.9929   0.9911   0.9886   0.9957   0.9980
## Pos Pred Value         0.9823   0.9626   0.9471   0.9777   0.9910
## Neg Pred Value         0.9969   0.9902   0.9922   0.9925   0.9949
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
```

```
## Detection Rate         0.2822   0.1856   0.1680   0.1576   0.1797
## Detection Prevalence   0.2873   0.1928   0.1774   0.1612   0.1813
## Balanced Accuracy      0.9926   0.9750   0.9760   0.9788   0.9877
```

```
predtrain<-predict(gmbfit, newdata=training)
confusionMatrix(predtrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4153   70    0    3    2
##          B   18 2731   62   10   16
##          C   11   45 2473   64   18
##          D    2    0   26 2320   25
##          E    1    2    6   15 2645
##
## Overall Statistics
##
##                Accuracy : 0.9731
##                  95% CI : (0.9704, 0.9756)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.966
##  Mcnemar's Test P-Value : 4.472e-15
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9924   0.9589   0.9634   0.9619   0.9775
## Specificity            0.9929   0.9911   0.9886   0.9957   0.9980
## Pos Pred Value         0.9823   0.9626   0.9471   0.9777   0.9910
## Neg Pred Value         0.9969   0.9902   0.9922   0.9925   0.9949
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2822   0.1856   0.1680   0.1576   0.1797
## Detection Prevalence   0.2873   0.1928   0.1774   0.1612   0.1813
## Balanced Accuracy      0.9926   0.9750   0.9760   0.9788   0.9877
```

Generating Files to submit as answers for the Assignment:

Once, the predictions were obtained for the 20 test cases provided, the below shown script was used to obtain single text files to be uploaded to the courses web site to comply with the submission assigment. 20 out of 20 hits also confirmed the accuracy of the obtained models.

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
```

```
pml_write_files(pred20)
```