Coursera Practical Machine Learning - The Course Project

Krishna Vaidyanathan

2015-01-25

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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Input Data

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. The data was downloaded to hard drive for this model.

Import libraries…

The most important library for this model is the caret package, that allows uniform evaluation and comparision of multiple models. This report addresses a random forest apprach and a KNN model, however more models can be chosen for the purpose of learning.

The model was done under Mac OS X, with 3.06 GHz Intel Core i3, 4 GB RAM. This allowed using the doMC library to utilize multiple cores that expedites the computation. Initially a 10-fold cross-validation was attempted, that consumed substantial time, therefore only a 5-fold cross-validation is adopted.

```
library(caret)
```

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

```
library(ggplot2)
library(gridExtra)
```

```
## Loading required package: grid
```

```
library(randomForest)
```

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

```
library(knitr)

library(doMC)
```

```
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel
```

```
registerDoMC(cores = 4)

set.seed(1408)
```

Data processing

The data is imported from hard drive and into to R memory. The data can also be imported directly from the URL as well.

```
training <- read.csv("~/Dropbox/Coursera/Practical Machine Learning/Project/pml-training.csv")
testing <- read.csv("~/Dropbox/Coursera/Practical Machine Learning/Project/pml-testing.csv")
```

Inspecting the data, several column vectors that are empty and with NA values are found. This can be dealt with by specifying na.strings during import itself.

```
training <- read.csv("~/Dropbox/Coursera/Practical Machine Learning/Project/pml-training.csv", na.strings = c(NA, '', '
testing <- read.csv("~/Dropbox/Coursera/Practical Machine Learning/Project/pml-testing.csv", , na.strings = c(NA, '', '
dim(training); dim(testing)
```

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

```
## [1]  20 160
```

Examining the data suggests there are still columns with high percentage of NA values. Columns with more than 90% of valid data will be selected.

```
training <- training[ , (nrow(training) - colSums(is.na(training)))/nrow(training) > 0.9]
testing  <-  testing[ , (nrow(testing) - colSums(is.na(testing)))/nrow(testing) > 0.9]
dim(training); dim(testing)
```

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

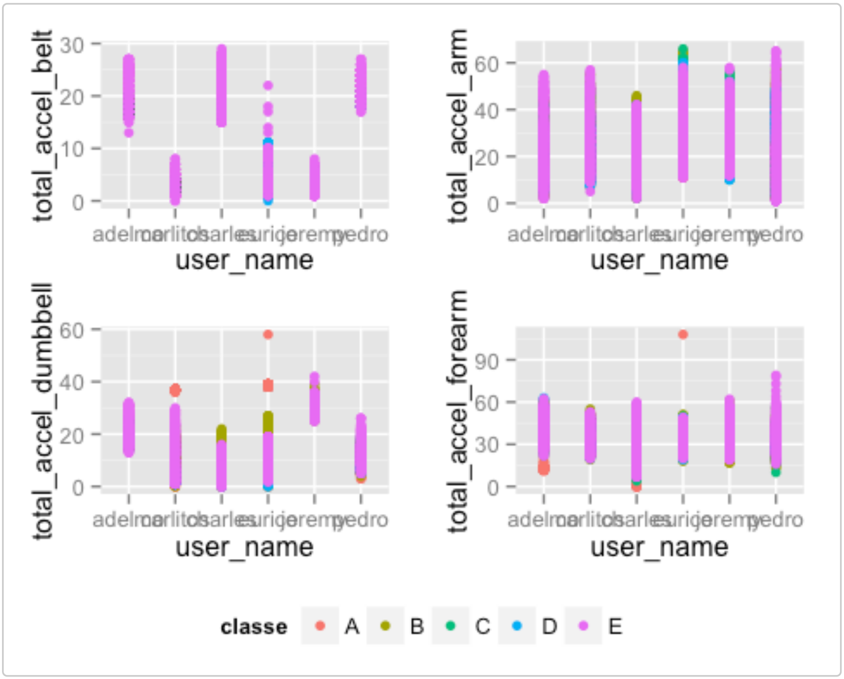
```
## [1] 20 60
```

Plot the interactions by user, by classe.

Examining the following plots, where the sum total value from the sensor data are plotted for each user, factored by the classe variable, suggests some similarity in how the subjects reacted to each of the exercise class.

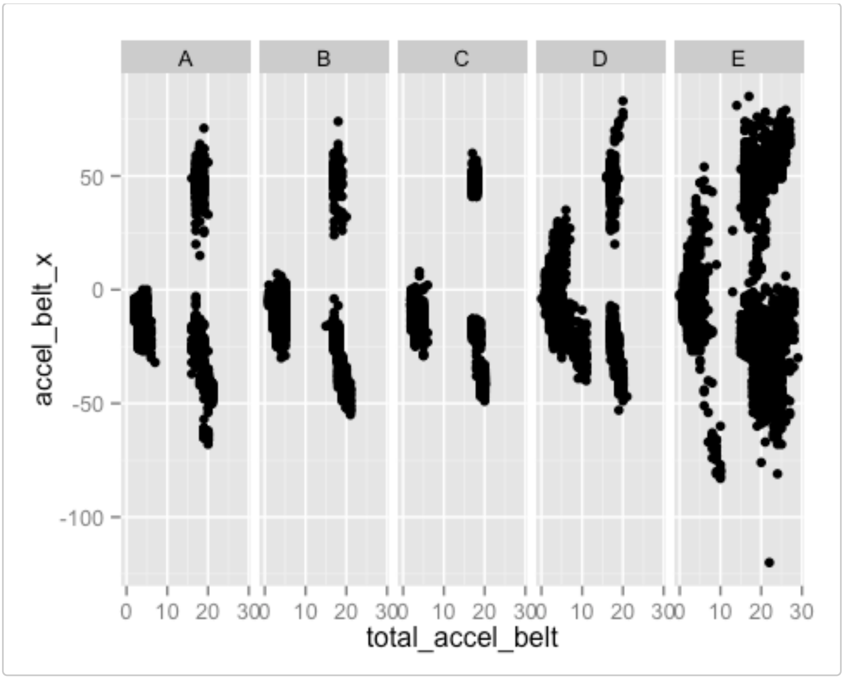
```
plot1 <- ggplot(data = training, aes(x = user_name, y = total_accel_belt)) + geom_point(aes(colour = classe))
plot2 <- ggplot(data = training, aes(x = user_name, y = total_accel_arm)) + geom_point(aes(colour = classe))
plot3 <- ggplot(data = training, aes(x = user_name, y = total_accel_dumbbell)) + geom_point(aes(colour = classe))
plot4 <- ggplot(data = training, aes(x = user_name, y = total_accel_forearm)) + geom_point(aes(colour = classe))
```

```
grid_arrange_shared_legend(plot1, plot2, plot3, plot4)
```

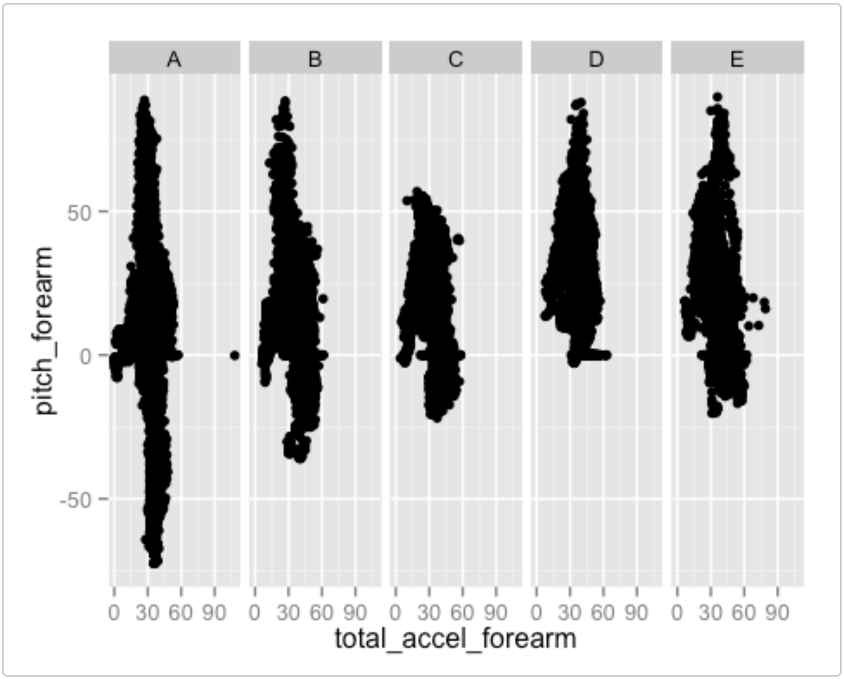


On a much closer look, faceting by the classe variable, the derivate feature of the accelerometer, suggests that each classe had an unique signature. This lends promise for further modeling and the type of activity can be predicted from the derivate accelerometer data, which happens to be the objective of this exercise.

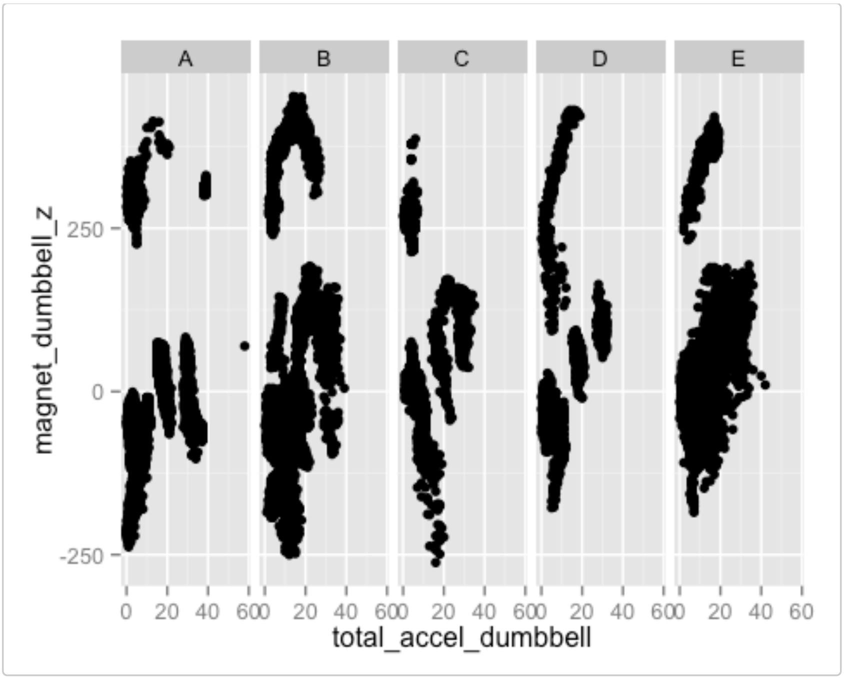
```
ggplot(data = training, aes(x = total_accel_belt, y = accel_belt_x )) + geom_point() + facet_wrap(~classe, nrow =1)
```



```
ggplot(data = training, aes(x = total_accel_forearm, y = pitch_forearm )) + geom_point() + facet_wrap(~classe, nrow =1)
```



```
ggplot(data = training, aes(x = total_accel_dumbbell, y = magnet_dumbbell_z )) + geom_point() + facet_wrap(~classe, nrc
```



Now lets remove the columns that will not be used in prediction. The first 7 columns contain user information and cannot be used in model predictions.

```
training <- training[ , 8:60]
testing  <-  testing[ , 8:60]
dim(training); dim(testing)
```

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

```
## [1] 20 53
```

Convert the response variable to a factor variable, to be predicted.

```
training$classe <- factor(training$classe)
```

Create partition of the training data for model building and validation.

```
inTrain <- createDataPartition(training$classe, p = 3/4, list = FALSE)
myTraining <- training[ inTrain, ]
myTesting  <- training[-inTrain, ]
```

Create models for prediction:

The goal of the project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

caret package allows us to specify a trainControl function that can be applied across all model predictions. Lets define the trainControl parameter, using a 5-fold cross-validation

```
trnCtrl <- trainControl(method = 'cv',
                         number = 5,
                         allowParallel = TRUE,
                         verboseIter = TRUE)
```

Model 1 - Random Forest

```
model1 <- train(classe ~ .,
                data = myTraining,
                method = 'rf',
                trControl = trnCtrl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 2 on full training set
```

Model 2 - K-nearest neighbor

```
model2 <- train(classe ~ .,
                data = myTraining,
                method = 'knn',
                trControl = trnCtrl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting k = 5 on full training set
```

Final results

Comparison of the two models

```
model1Pred <- predict(model1, myTesting)
model2Pred <- predict(model2, myTesting)

confusionMatrix(model1Pred, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    5    0    0    0
##          B    2  944    4    0    0
##          C    1    0  850   16    0
##          D    0    0    1  787    2
##          E    0    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9908, 0.9955)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9917
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9947   0.9942   0.9789   0.9978
## Specificity            0.9986   0.9985   0.9958   0.9993   0.9998
## Pos Pred Value         0.9964   0.9937   0.9804   0.9962   0.9989
## Neg Pred Value         0.9991   0.9987   0.9988   0.9959   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1925   0.1733   0.1605   0.1833
## Detection Prevalence   0.2849   0.1937   0.1768   0.1611   0.1835
## Balanced Accuracy      0.9982   0.9966   0.9950   0.9891   0.9988
```

```
confusionMatrix(model2Pred, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1345   45    8    8   17
##          B    8  845   30    3   24
##          C   14   31  779   64   18
##          D   25   23   19  716   22
##          E    3    5   19   13  820
##
## Overall Statistics
##
##                Accuracy : 0.9186
##                  95% CI : (0.9106, 0.9261)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.897
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9642   0.8904   0.9111   0.8905   0.9101
## Specificity            0.9778   0.9836   0.9686   0.9783   0.9900
## Pos Pred Value         0.9452   0.9286   0.8598   0.8894   0.9535
## Neg Pred Value         0.9856   0.9740   0.9810   0.9785   0.9800
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2743   0.1723   0.1588   0.1460   0.1672
## Detection Prevalence   0.2902   0.1856   0.1847   0.1642   0.1754
## Balanced Accuracy      0.9710   0.9370   0.9399   0.9344   0.9501
```

The summary of accuracies of the two models are:

```
kable(data.frame(Model = c('RF', 'KNN'),
                 Accuracy = c(confusionMatrix(model1Pred, myTesting$classe)$overall[1],
                              confusionMatrix(model2Pred, myTesting$classe)$overall[1])))
```

Model	Accuracy
RF	0.9934747
KNN	0.9186378

The RF model is more accuracate than the KNN model.

Prediction on the provided testing dataset

```
model1Pred.test <- predict(model1, testing)
model2Pred.test <- predict(model2, testing)
```

The two models compares as follows on the test data:

```
knitr::kable(data.frame(rf.pred = model1Pred.test, knn.pred = model2Pred.test))
```

They conform with each other well, though the accuracy of RF model is slightly higher than KNN model and should be chosen between the two.

rf.pred	knn.pred
B	B
A	A
B	B
A	A
A	A
E	E
D	D
B	B
A	A
A	A
B	B
C	C
B	B
A	A
E	E
E	E
A	A
B	B
B	B

B	B

Function to generate files with prediction for submission

```
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(model1Pred.test)
```