Prediction Assignment

Hosub Park

May 31, 2016

Overview

About dataset

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. Each set has 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.

The dataset used in this report is from below research:
* Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. (http://groupware.les.inf.puc-rio.br/work.jsf?p1=11201) Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013. ###About prediction model Random forest method were applied to randomly splitted subset of training set, which has 75% of cases of training set. Validation of the model was taken at another non-overlapping subset of training set(which has 25% of cases). Fitted model was applied to the test set.

Data aquisition

The dataset used in this report was downloaded from the links in assignment instruction page in the Couresa.

```
trainurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url=trainurl, destfile = "pml-training.csv")
download.file(url=testurl, destfile = "pml-testing.csv")
```

Loading and clean the data

```
train <- read.csv(file="pml-training.csv", header=T, stringsAsFactors = FALSE, na.strings = "")

train$classe <- factor(train$classe)
train$user_name <- factor(train$user_name)
```

This dataset have time-series data. Also it have summaries of the data in each windows(which rows have `new_window` is "yes"). However, It could be dangerous to use these summaries. Because there are many "#DIV/0!" and `NA` s. And condidering the size of test set file, which is 0.1% of that of training set, the test set could have only about 20 rows. It means that the model to aquire high accuracy in the test set need to be based on the values of each rows, not the summerized values of each window. So, I disgard the variables for summerized data for each windows. Remaining variables are from belt, arm, forearm, and dumbell. And each site has data from 3 motion sensors(gyros, accel, magnet). So I choose just the variables from these motion sensors to fit a model.

```
extsen <- c(grep("^gyros",  names(train)), grep("^accel",  names(train)), grep("^magnet",  names(train)))
train_sen <- train[,c(2,extsen,160)]
```

Examine the data

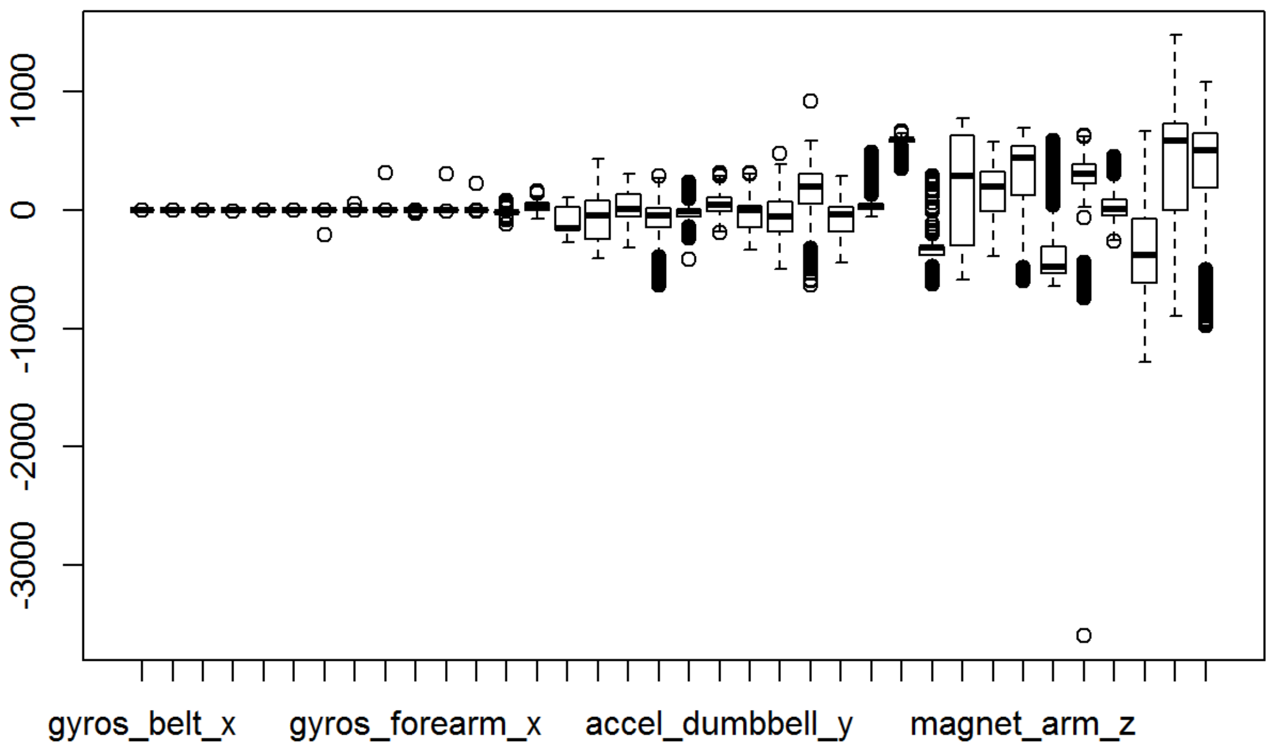
Examine the cleaned data

```
names(train_sen)
```

```
##  [1] "user_name"          "gyros_belt_x"       "gyros_belt_y"
##  [4] "gyros_belt_z"       "gyros_arm_x"        "gyros_arm_y"
##  [7] "gyros_arm_z"        "gyros_dumbbell_x"   "gyros_dumbbell_y"
## [10] "gyros_dumbbell_z"   "gyros_forearm_x"    "gyros_forearm_y"
## [13] "gyros_forearm_z"    "accel_belt_x"       "accel_belt_y"
## [16] "accel_belt_z"       "accel_arm_x"        "accel_arm_y"
## [19] "accel_arm_z"        "accel_dumbbell_x"   "accel_dumbbell_y"
## [22] "accel_dumbbell_z"   "accel_forearm_x"    "accel_forearm_y"
## [25] "accel_forearm_z"    "magnet_belt_x"      "magnet_belt_y"
## [28] "magnet_belt_z"      "magnet_arm_x"       "magnet_arm_y"
## [31] "magnet_arm_z"       "magnet_dumbbell_x"  "magnet_dumbbell_y"
## [34] "magnet_dumbbell_z"  "magnet_forearm_x"   "magnet_forearm_y"
## [37] "magnet_forearm_z"   "classe"
```

Examine the data by boxplot.

```
boxplot(train_sen[,-c(1,38)])
```

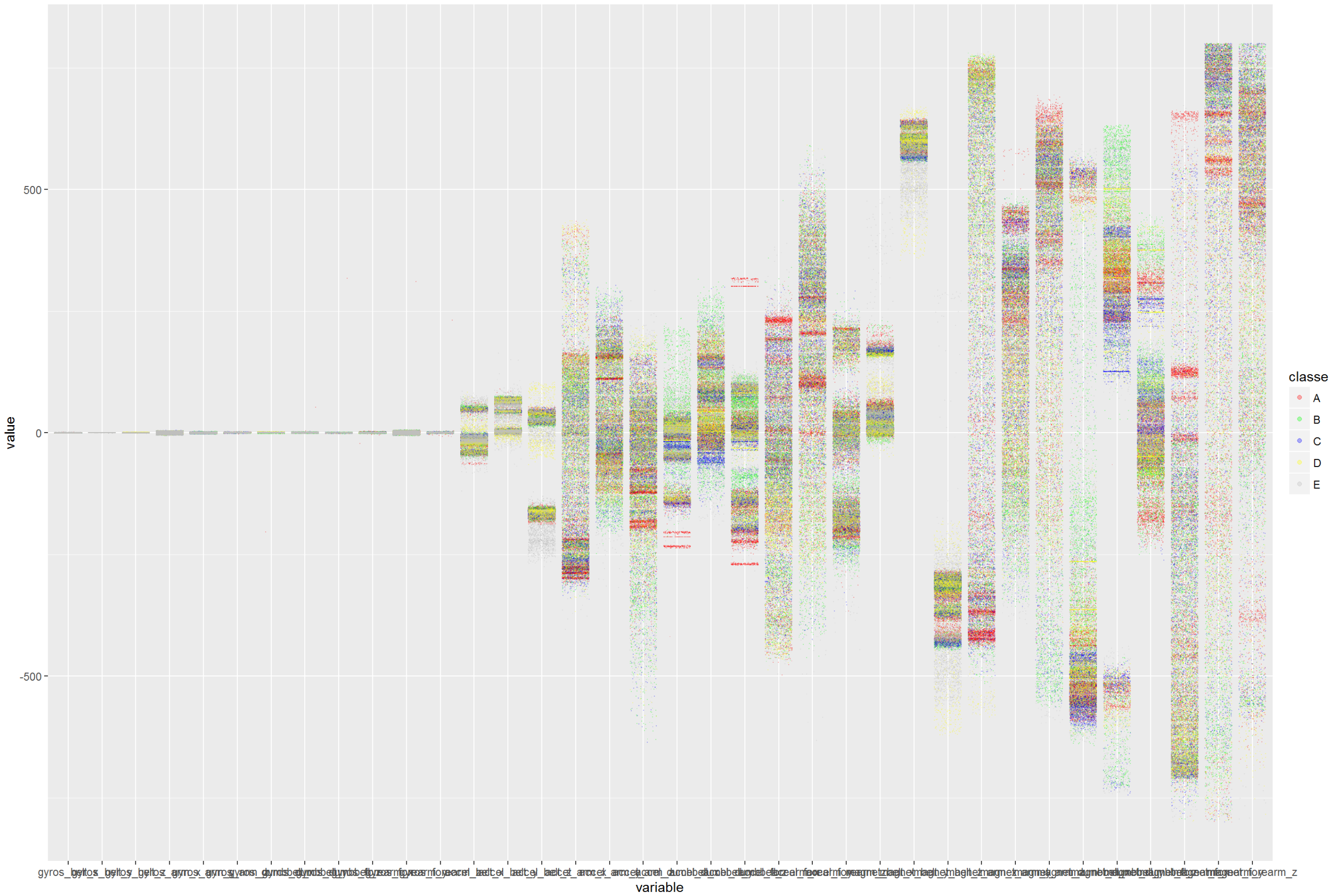


The variables in the data set has many outliers. In general, the data from gyrosensor has relatively more condensed values than those from accelometer and magnetic sensor. The values from magnetic sensor have most various values.

Examine the distributions of the values by the `classe`

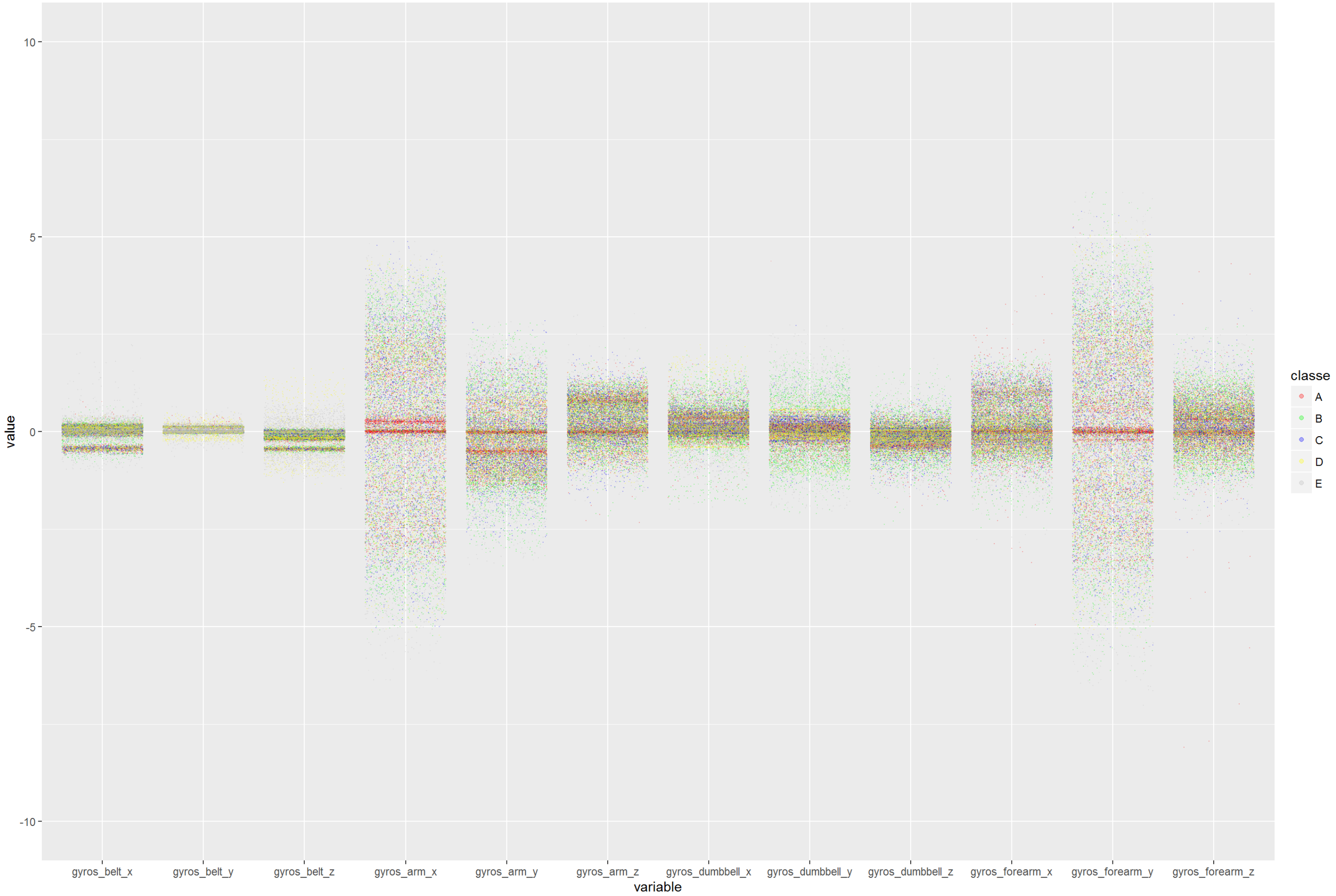
```
library(ggplot2)
library(reshape2)

mt <- melt(train_sen[,-1])
g <- ggplot(mt)
g <- g + geom_jitter(aes(x=variable, y=value, color=classe), alpha=0.3, shape=46) + ylim(-800,800)
g <- g + scale_color_manual(values=c("red", "green", "blue","yellow","gray"))
g + guides(color = guide_legend(override.aes = list(shape = 19)))
```



Zoom first 12 variables from gyrosensor.

```
mt <- melt(train_sen[,c(2:13,38)])
g <- ggplot(mt)
g <- g + geom_jitter(aes(x=variable, y=value, color=classe), alpha=0.3, shape=46) + ylim(-10,10)
g <- g + scale_color_manual(values=c("red", "green", "blue","yellow","gray"))
g + guides(color = guide_legend(override.aes = list(shape = 19)))
```



Above figures are jitter plots for the values of each variable. Each classes have different color. In these figures, there are several streaks that have specifically high proportion of each classes. But these streaks are scattered, and do not show strong pattern.

Split train data

Before fit a model, split subset of training set to validate the model.
Randomly split train data into train(75%) and test(25%) subset again without overlapping.

```
library(caret)
```

```
set.seed=(1002)
subtrad <- createDataPartition(y=train_sen$classe, p=0.75, list=FALSE)
subtrn <- train_sen[subtrad,]
subtes <- train_sen[-subtrad,]
```

Training model

The dataset has outcome that has 5 classes. And the predictors are all continuous numeric variable with many outliers. Since the variables have many out liers, the boosting algorithm would be effected by these outliers. And random forest tend to have high accuracy, which would be some advantages in later quiz for this assignment. Therefore, I choose the random forest algorythm.

Training Random forest model

Because the random forest uses randomly re-sampled subset of the data, it does not need cross validation.

Fitting model with `subtrn`

```
set.seed=(1018)
mdl_rf <- train(classe~., method="rf", data=subtrn[,-1])
mdl_rf
```

```
## Random Forest
##
## 14718 samples
##    36 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9815798  0.9766968
##   19    0.9778857  0.9720225
##   36    0.9705507  0.9627401
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
mdl_rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 1.17%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4176    2    0    6    1 0.002150538
## B   24 2806   16    2    0 0.014747191
## C    1   24 2539    3    0 0.010907674
## D    5    0   72 2332    3 0.033167496
## E    0    0    5    8 2693 0.004804139
```

This model has accuracy 0.9814, Kappa 0.9765, In-sample error is 0.019, and OOB estimate is 1.21%.

Validating model with `subtes`

```
pre_rf <- predict(mdl_rf, newdata=subtes)
confusionMatrix(pre_rf, subtes[,38])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1390   18    0    1    0
##          B    3  927   10    0    1
##          C    1    4  845   19    3
##          D    1    0    0  781    0
##          E    0    0    0    3  897
##
## Overall Statistics
##
##                Accuracy : 0.9869
##                  95% CI : (0.9834, 0.9899)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9835
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9768   0.9883   0.9714   0.9956
## Specificity            0.9946   0.9965   0.9933   0.9998   0.9993
## Pos Pred Value         0.9865   0.9851   0.9690   0.9987   0.9967
## Neg Pred Value         0.9986   0.9944   0.9975   0.9944   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2834   0.1890   0.1723   0.1593   0.1829
## Detection Prevalence   0.2873   0.1919   0.1778   0.1595   0.1835
## Balanced Accuracy      0.9955   0.9866   0.9908   0.9856   0.9974
```

The accuracy is 0.988, and kappa value is 0.9848. Out-of-sample error is 0.002, smaller than In-sample error and estimated OOB.

Check subset

In most cases, Out-of-sample error is larger than In-sample error. In this report, however, In-sample error is larger than Out-of-sample error. It could be happend when the rare outcome is un-equally distributed between testing and training subsets.

```
summary(subtrn$classe)/length(subtrn$classe)
```

```
##         A         B         C         D         E
## 0.2843457 0.1935046 0.1744123 0.1638810 0.1838565
```

```
summary(subtes$classe)/length(subtes$classe)
```

```
##         A         B         C         D         E
## 0.2844617 0.1935155 0.1743475 0.1639478 0.1837276
```

As you can see, both subset have almost same portion of each classes.

Applying test set

```
test <- read.csv(file="pml-testing.csv", header=T, stringsAsFactors = FALSE, na.strings = "")

test_result <- predict(mdl_rf, newdata=test)
test_result
```

```
##  [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
```