Prediction Assignment Writeup

Shu Yan

February 22, 2015

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. In this project, we will be performing maching learning algorithms to predict the correctness of barbell lifting based on data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Data preprocessing

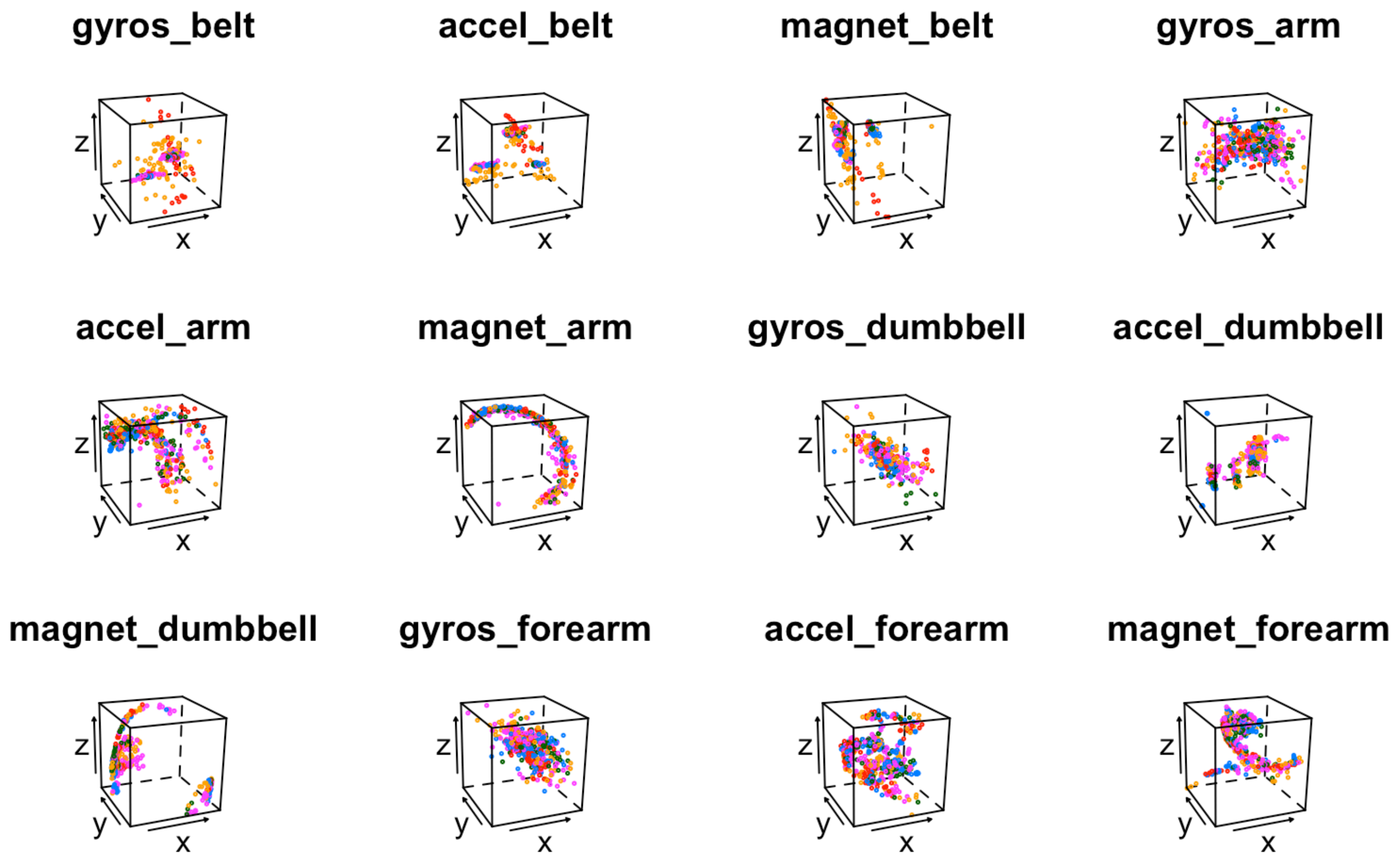
We first load the training data set from the given *URL*. It is resonable to believe that the prediction only depends on the body motions. Therefore we only select predictors coming from the data of the sensors. Meanwhile we should remove predictors that contain any *NA*s. (fortunately, we don't have any after the first cleaning step)

```r
urlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
pmlTrain <- read.csv(urlTrain,na.strings=c("NA", "#DIV/0!"))
feature <- c(grep("belt_x|arm_x|bell_x|belt_y|arm_y|bell_y|belt_z|arm_z|bell_z",
                  names(pmlTrain)),length(pmlTrain))
pmlTrain <- pmlTrain[,feature]
#pmlTrain <- pmlTrain[-which(sapply(pmlTrain,anyNA))]

urlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
pmlTest  <- read.csv(urlTest,na.strings=c("NA", "#DIV/0!"))
pmlTest  <- pmlTest[-which(sapply(pmlTest,anyNA))]

###uncomment the following two lines to test the source code in a small portion
#vis <- sample(1:dim(pmlTrain)[1],1000)
#pmlTrain <- pmlTrain[sort(vis),]
```

data visualization

Our data now contains 12 features, each of which has 3 spacial components, we can visualize the features in 3D plots. Here we randomly select a small portion of all observations. (Note that the figures are not drawn on same scales.)

```
library(lattice)
vis <- sample(1:dim(pmlTrain)[1],500)
i <- 1
has_more = TRUE
par.set <- list(axis.line = list(col = "transparent"), clip = list(panel = "off"))
while(has_more) {
    if(i > 33) {
        has_more = FALSE
    }
    j = (i+2)/3
    str <- names(pmlTrain[i])
    print(cloud(pmlTrain[vis,i+2] ~ pmlTrain[vis,i] * pmlTrain[vis,i+1],
            data = pmlTrain[vis,], cex = .2, groups = classe,
            xlab = "x", ylab = "y", zlab = "z",
            main = substr(str, 1, nchar(str)-2),
            screen = list(z = 20, x = -70, y = 3),
            par.settings = par.set, scales = list(col = "black")),
          split = c((j-1)%%4+1,ceiling(j/4),4,3), more = has_more)
    i = i + 3
}
```



Data slicing

The data sets are splitted into training set and testing set as follow

```
library(caret)
```

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

```
inTrain <- createDataPartition(y=pmlTrain$classe, p=.7, list=FALSE)
training <- pmlTrain[inTrain,]
testing  <- pmlTrain[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 13737    37
```

```
## [1] 5885   37
```

Data training

The **rpart** is fast but gives poor prediction. Here we will be using **gbm** and **rf** methods, which are more accurate but also very time consuming.

```
library(caret)
gbmFit <- train(classe~., method="gbm",data=training,verbose=FALSE)
```

```
## Loading required package: gbm
## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
## Loaded gbm 2.1
## Loading required package: plyr
```

```
gbmPred <- predict(gbmFit,newdata=testing)
confusionMatrix(gbmPred,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1616   98   30   24   14
##          B   16  947   48    6   25
##          C   18   71  924   65   29
##          D   22   12   18  857   27
##          E    2   11    6   12  987
##
## Overall Statistics
##
##                Accuracy : 0.9059
##                  95% CI : (0.8981, 0.9132)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8807
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9654   0.8314   0.9006   0.8890   0.9122
## Specificity            0.9606   0.9800   0.9623   0.9839   0.9935
## Pos Pred Value         0.9068   0.9088   0.8347   0.9156   0.9695
## Neg Pred Value         0.9859   0.9604   0.9787   0.9784   0.9805
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2746   0.1609   0.1570   0.1456   0.1677
## Detection Prevalence   0.3028   0.1771   0.1881   0.1590   0.1730
## Balanced Accuracy      0.9630   0.9057   0.9315   0.9365   0.9529
```

```
rfFit<-train(classe~.,data=training,method="rf",
                trControl=trainControl(method="cv",number=5),
                prox=TRUE,allowParallel=TRUE)
```

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

```
rfPred <- predict(rfFit,newdata=testing)
confusionMatrix(rfPred,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669   14    1    2    0
##          B    2 1110   14    0    0
##          C    0   15 1011   30    3
##          D    2    0    0  932    4
##          E    1    0    0    0 1075
##
## Overall Statistics
##
##                Accuracy : 0.985
##                  95% CI : (0.9816, 0.988)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9811
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9745   0.9854   0.9668   0.9935
## Specificity            0.9960   0.9966   0.9901   0.9988   0.9998
## Pos Pred Value         0.9899   0.9858   0.9547   0.9936   0.9991
## Neg Pred Value         0.9988   0.9939   0.9969   0.9935   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1886   0.1718   0.1584   0.1827
## Detection Prevalence   0.2865   0.1913   0.1799   0.1594   0.1828
## Balanced Accuracy      0.9965   0.9856   0.9878   0.9828   0.9967
```

```
accuracy <- function(test, pred) {
    acc <- sum(test==pred)/length(test)
    return(acc)
}
```

The accuracy for **gbm** is 0.9058624 and for **rf** is 0.9850467. The latter is pretty accurate.

Predicting

Finally we use the two trained models to predict the given test data set

```
predict(gbmFit,newdata=pmlTest)
```

```
##  [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
```

```
predict(rfFit,newdata=pmlTest)
```

```
##  [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
```