Machine Learning Course Project

Norma Ruiz

20-oct-2015

Introduction

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 the goal is 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 differente ways. The goal is to predict the manner in which they did the exercise. This is the “classe” variable in the training set, which can take the values: A, B, C, D, E, with the meanings:
- Class A = exactly according to the specification
- Class B = throwing the elbows to the front
- Class C = lifting the dumbbell only halfway
- Class D = lowering the dumbbell only halfway
- Class E = throwing the hips to the front

Source: Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedins of 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13). Stuttgart, Germany: ACM SIGCHI, 2013.
The results of these measurements were collected in a dataset with 160 variables and 19,622 observations. I will use this data to build a model capable of determine how well the exercise was done (variable “classe”). There is another dataset with 20 observations (without the classe variable) that I will use to apply my model to predict the class.

PreProcessing

Read the data

```
library(caret)
```

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

```
setwd("~/Documents/NRS_iMAC/norma_2015/infomedia/data science/8 Practical Machine Lea
rning/course project")
pmltrain <- read.csv("pml-training.csv",na.strings=c("","NA","#DIV/0!"))
pmltest  <- read.csv("pml-testing.csv", na.strings=c("","NA","#DIV/0!"))
```

During the exploratory data analysis I discovered that there are many missing values, so I will exclude those columns having more than 10% of missing values. I will also exclude columns with an absolute correlation factor > 0.9, Finally I exclude the first 7 columns of the data since they are irrelevant for the prediction purposes. After this reduction, there are 45 predictor variables available for the analysis, plus the response variable **classe**.

Split the data: training set and test set for my model.

```
dim(pmltrain)
```

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

```
na_count <- apply(pmltrain, 2, function(x) sum(is.na(x)))
cols <- names(which(na_count/dim(pmltrain[1])< 0.1, arr.ind=T))
train_new <- pmltrain[,cols]
train_new <- train_new[, -(1:7)]
train_cor <- cor(subset(train_new, select = -classe))
high_corr  <- findCorrelation(train_cor, cutoff=0.9)
train_new <- train_new[, -high_corr]
dim(train_new)
```

```
## [1] 19622    46
```

```
inTrain = createDataPartition(train_new$classe,p=3/4)[[1]]
trn  <- train_new[ inTrain,] # 75% para entrenar
tst  <- train_new[-inTrain,] # 25% para probar
dim(trn)
```

```
## [1] 14718    46
```

```
dim(tst)
```

```
## [1] 4904   46
```

Variable Selection

I built a model using **random forest** to select the 20 most important predictor variables, in order to have a simple model.

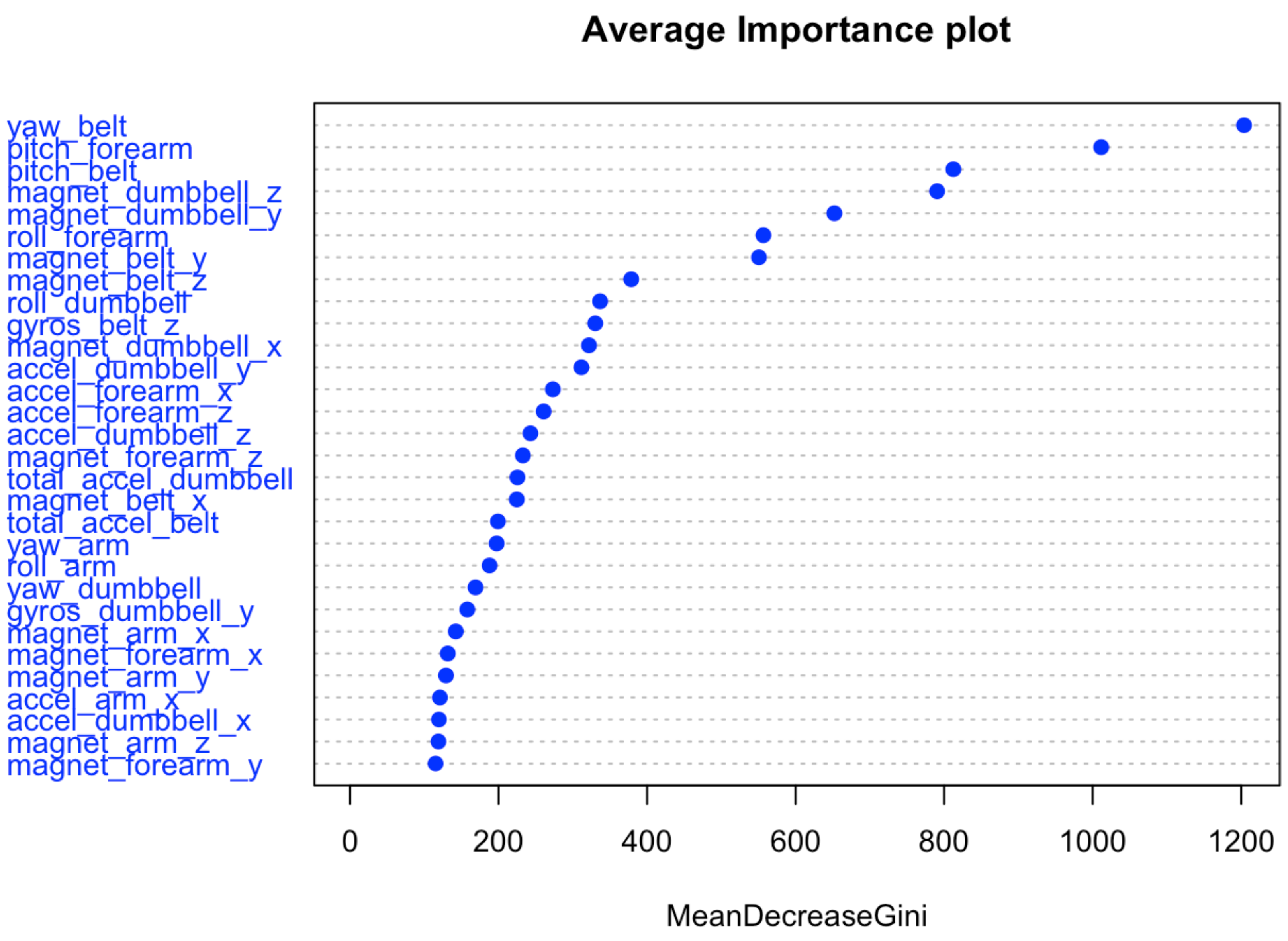
```
library(doMC)
```

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

```
registerDoMC(cores=3)
model_rf <- train(classe ~ ., data=trn, method="rf")
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
varImpPlot(model_rf$finalModel, main="Average Importance plot",
           col="blue", pch =19, cex=0.9)
```




```
top20 <- varImp(model_rf)[[1]]
x <- order(-top20$Overall)
top20vars <- row.names(top20)[x][1:20]
trn20 <- cbind(trn[,top20vars], trn$classe)
names(trn20)[21] <- "classe"
```

Training

Lets train 2 different models using the training data with 20 predictor variables: boosting(gbm) and random forest(rf). The accuracy for each model is: boosting accuracy=0.9492; random forest accuracy=0.99 (this numbers can change a little bit each next execution).

```
# first model - boosting
train_gbm <- train(classe ~ ., data=trn20, method="gbm", verbose=F)
```

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

```
pred_gbm  <- predict(train_gbm, newdata=tst)
confusionMatrix(pred_gbm, tst$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1374   31    0    1    0
##          B   13  873   26    3    5
##          C    4   31  808   30    9
##          D    3    7   18  766   14
##          E    1    7    3    4  873
##
## Overall Statistics
##
##                Accuracy : 0.9572
##                  95% CI : (0.9511, 0.9627)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9458
##  Mcnemar's Test P-Value : 0.002342
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9849   0.9199   0.9450   0.9527   0.9689
## Specificity            0.9909   0.9881   0.9817   0.9898   0.9963
## Pos Pred Value         0.9772   0.9489   0.9161   0.9480   0.9831
## Neg Pred Value         0.9940   0.9809   0.9883   0.9907   0.9930
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2802   0.1780   0.1648   0.1562   0.1780
## Detection Prevalence   0.2867   0.1876   0.1799   0.1648   0.1811
## Balanced Accuracy      0.9879   0.9540   0.9634   0.9712   0.9826
```

```
# second model - random forest
train_rf  <- train(classe ~ ., data=trn20, method="rf")
pred_rf   <- predict(train_rf, newdata=tst)
confusionMatrix(pred_rf, tst$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    9    0    0    0
##          B    1  937    6    0    0
##          C    1    2  844   11    1
##          D    0    1    5  793    7
##          E    0    0    0    0  893
##
## Overall Statistics
##
##                Accuracy : 0.991
##                  95% CI : (0.988, 0.9935)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9886
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9874   0.9871   0.9863   0.9911
## Specificity            0.9974   0.9982   0.9963   0.9968   1.0000
## Pos Pred Value         0.9936   0.9926   0.9825   0.9839   1.0000
## Neg Pred Value         0.9994   0.9970   0.9973   0.9973   0.9980
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1911   0.1721   0.1617   0.1821
## Detection Prevalence   0.2859   0.1925   0.1752   0.1644   0.1821
## Balanced Accuracy      0.9980   0.9928   0.9917   0.9916   0.9956
```

Combining models

Finally, I combined both models fitting a model that combine the predictors using the training data. Lets measure the accuracy of this combined model using the test data: combined accuracy=0.9904 (this number can change a little bit each execution).

```
pred_gbm     <- predict(train_gbm, newdata=trn)
pred_rf      <- predict(train_rf, newdata=trn)
trn_dat_comb <- data.frame(GBM=pred_gbm,RF=pred_rf,classe=trn$classe)
train_comb   <- train(classe ~ ., method="rf",data=trn_dat_comb)
pred_gbm     <- predict(train_gbm, newdata=tst)
pred_rf      <- predict(train_rf, newdata=tst)
tst_dat_comb <- data.frame(GBM=pred_gbm,RF=pred_rf,classe=tst$classe)
pred_comb    <- predict(train_comb, newdata=tst_dat_comb)
confusionMatrix(pred_comb, tst$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    9    0    0    0
##          B    1  937    6    0    0
##          C    1    2  844   11    1
##          D    0    1    5  793    7
##          E    0    0    0    0  893
##
## Overall Statistics
##
##                Accuracy : 0.991
##                  95% CI : (0.988, 0.9935)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9886
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9874   0.9871   0.9863   0.9911
## Specificity            0.9974   0.9982   0.9963   0.9968   1.0000
## Pos Pred Value         0.9936   0.9926   0.9825   0.9839   1.0000
## Neg Pred Value         0.9994   0.9970   0.9973   0.9973   0.9980
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1911   0.1721   0.1617   0.1821
## Detection Prevalence   0.2859   0.1925   0.1752   0.1644   0.1821
## Balanced Accuracy      0.9980   0.9928   0.9917   0.9916   0.9956
```

Conclusion

Since the combined model has the same accuracy as the random forest I will choose the random forest model since it is simpler than the combined model.

```
head(getTree(train_rf$finalModel, k=1, labelVar=T))
```

```
##   left daughter right daughter         split var split point status
## 1             2              3 magnet_dumbbell_y       422.5      1
## 2             4              5  accel_dumbbell_y       -40.5      1
## 3             6              7     magnet_belt_z      -345.0      1
## 4             8              9  accel_dumbbell_z        26.5      1
## 5            10             11 magnet_dumbbell_z        60.5      1
## 6            12             13   accel_forearm_x       126.0      1
##   prediction
## 1       <NA>
## 2       <NA>
## 3       <NA>
## 4       <NA>
## 5       <NA>
## 6       <NA>
```

```
head(getTree(train_rf$finalModel, k=2, labelVar=T))
```

```
##   left daughter right daughter            split var split point status
## 1             2              3     accel_dumbbell_y   -40.50000      1
## 2             4              5             yaw_belt     6.05000      1
## 3             6              7        roll_dumbbell    63.52522      1
## 4             8              9      accel_forearm_z   -56.50000      1
## 5             0              0                 <NA>     0.00000     -1
## 6            10             11 total_accel_dumbbell    29.50000      1
##   prediction
## 1       <NA>
## 2       <NA>
## 3       <NA>
## 4       <NA>
## 5          E
## 6       <NA>
```

```
library(inTrees)
treeList <- RF2List(train_rf$finalModel)
exec <- extractRules(treeList, trn20)
```

```
## 4327 rules (length<=6) were extracted from the first 100 trees.
```

```
ruleMetric <- getRuleMetric(exec, trn20, trn20$classe)
ruleMetric <- pruneRule(ruleMetric, trn20, trn20$classe)
ruleMetric <- selectRuleRRF(ruleMetric, trn20, trn20$classe)
rules <- presentRules(ruleMetric,colnames(trn20))
head(rules)
```

```
##      len freq    err
## [1,] "5" "0.148" "0.119"
## [2,] "2" "0.082" "0.151"
## [3,] "4" "0.397" "0.651"
## [4,] "2" "0.041" "0.07"
## [5,] "4" "0.056" "0.192"
## [6,] "4" "0.07"  "0.282"
##      condition
## [1,] "pitch_forearm<=24.15 & pitch_belt>3.785 & magnet_dumbbell_y<=422.5 & roll_fo
rearm<=116.5 & roll_forearm>-44.85"
## [2,] "magnet_dumbbell_z<=120 & magnet_belt_z<=-382.5"
## [3,] "magnet_dumbbell_z>-42.5 & magnet_dumbbell_z<=284.5 & magnet_belt_y>555.5 & r
oll_dumbbell<=63.63849661"
## [4,] "yaw_belt>168.5 & pitch_belt>-45.1"
## [5,] "pitch_forearm<=4.865 & magnet_belt_y>559.5 & magnet_belt_z<=-323.5 & magnet_
forearm_z<=340.5"
## [6,] "pitch_belt<=15.05 & magnet_belt_z>-350.5 & roll_dumbbell<=-50.249429785 & ma
gnet_dumbbell_x<=-420.5"
##      pred impRRF
## [1,] "A"  "1"
## [2,] "E"  "0.701136054460623"
## [3,] "C"  "0.358015819740995"
## [4,] "A"  "0.298802626931149"
## [5,] "A"  "0.24639112427363"
## [6,] "C"  "0.153033371715878"
```