Practical Machine Learning Wk 4

ekonomix

25 January 2017

Executive Summary

We have data from 4 sensors placed on participants bodies and object (belt, forearm, arm and the dumbell); they measure how the different body parts and the dumbell itself are moving as the participant is attempting to lift it.

Participants were asked to lift the dumbell in 5 different ways, 1 correct way and 4 'wrong' ways. Our aim is to distinguish "how well" the exercise is taking place, hence using the sensor data distinguish between these different types of lift.

load some libraries we are likely to be using

```
library(caret)

## Warning: package 'caret' was built under R version 3.3.2

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.2

library(ggplot2)
```

Get data - download as csv and load

```
Urla <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
Urlb <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(Urla), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(Urlb), na.strings=c("NA","#DIV/0!",""))
```

take a look at the data

```
dim(training)

## [1] 19622   160

#str(training) # not shown in output
dim(testing)

## [1]  20 160

#str(testing) # not shown in output
```

it looks like the first 7 variables have no predictive value for this exercise

get rid of variables with many NAs and variables expected to have no predictive value in this case

```
NA_Count = sapply(1:dim(training)[2],function(x)sum(is.na(training[,x])))
NA_Count

##   [1]     0     0     0     0     0     0     0     0     0     0     0
##  [12] 19226 19248 19622 19225 19248 19622 19216 19216 19226 19216 19216
##  [23] 19226 19216 19216 19226 19216 19216 19216 19216 19216 19216 19216
##  [34] 19216 19216 19216     0     0     0     0     0     0     0     0
##  [45]     0     0     0     0     0 19216 19216 19216 19216 19216 19216
##  [56] 19216 19216 19216 19216     0     0     0     0     0     0     0
##  [67]     0     0 19294 19296 19227 19293 19296 19227 19216 19216 19216
##  [78] 19216 19216 19216 19216 19216 19216     0     0     0 19221 19218
##  [89] 19622 19220 19217 19622 19216 19216 19221 19216 19216 19221 19216
## [100] 19216 19221     0 19216 19216 19216 19216 19216 19216 19216 19216
## [111] 19216 19216     0     0     0     0     0     0     0     0     0
## [122]     0     0     0 19300 19301 19622 19299 19301 19622 19216 19216
## [133] 19300 19216 19216 19300 19216 19216 19300     0 19216 19216 19216
## [144] 19216 19216 19216 19216 19216 19216 19216     0     0     0     0
## [155]     0     0     0     0     0     0

NA_list = which(NA_Count>0)
```

modify the training and test data sets to remove unnecessary columns and transforming the class into a factor

```
training_clean <- training[,-NA_list]
training_clean <- training_clean[,-c(1:7)]
training_clean$classe = factor(training_clean$classe)

testing_clean <- testing[,-NA_list]
testing_clean <- testing_clean[,-c(1:7)]


# head(testing_clean) # not shown in output
```

build models and deciding which works best

this is a classification problem, and we'll try random forest and classification tree

```
set.seed(2501)
```

Random Forest

```
rfFit <- train(classe ~ ., method = "rf", data = training_clean, importance =
T, trControl = trainControl(method = "cv", number = 3))
```

```
## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 3.3.2

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

## 
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
## 
##     margin
```

```
#training performance
training_rfpred <- predict(rfFit, newdata=training_clean)
rf_confusion <-confusionMatrix(training_rfpred,training_clean$classe)
rf_confusion
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    0    0    0    0
##          B    0 3797    0    0    0
##          C    0    0 3422    0    0
##          D    0    0    0 3216    0
##          E    0    0    0    0 3607
## 
## Overall Statistics
## 
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```r
#looks good
```

Random Forest Results look good!

Classification Tree

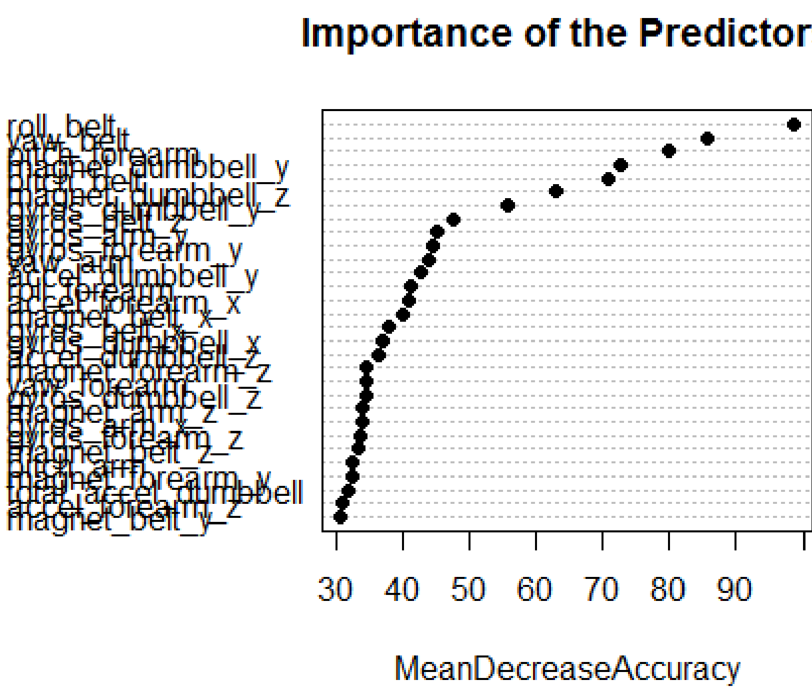
```r
rpartFit <- train(classe ~ ., method = "rpart", data = training_clean)

## Loading required package: rpart

#training performance
training_rpartpred <- predict(rpartFit, newdata=training_clean)
confusionMatrix(training_rpartpred,training_clean$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5080 1581 1587 1449  524
##          B   81 1286  108  568  486
##          C  405  930 1727 1199  966
##          D    0    0    0    0    0
##          E   14    0    0    0 1631
##
## Overall Statistics
##
##                Accuracy : 0.4956
##                  95% CI : (0.4885, 0.5026)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3407
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9104  0.33869  0.50468   0.0000  0.45218
## Specificity            0.6339  0.92145  0.78395   1.0000  0.99913
## Pos Pred Value         0.4970  0.50850  0.33040      NaN  0.99149
## Neg Pred Value         0.9468  0.85310  0.88225   0.8361  0.89008
## Prevalence             0.2844  0.19351  0.17440   0.1639  0.18382
## Detection Rate         0.2589  0.06554  0.08801   0.0000  0.08312
## Detection Prevalence   0.5209  0.12889  0.26638   0.0000  0.08383
## Balanced Accuracy      0.7721  0.63007  0.64431   0.5000  0.72565

#not as good
```

Regressions trees are nto as good as random forest here.

Hence Random Forest is Chosen!

important variables, expected error (1-accuracy) and predictions for test data:

```
#Important Variables
imp_rf <- varImp(rfFit)$importance
varImpPlot(rfFit$finalModel, sort = TRUE, type = 1, pch = 19, col = 1, cex =
1, main = "Importance of the Predictors")
```




```
#accuracy and expected error
attributes(rf_confusion)

## $names
## [1] "positive" "table"    "overall"  "byClass"  "mode"     "dots"
##
## $class
## [1] "confusionMatrix"

rf_confusion$overall

##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9998120      1.0000000      0.2843747
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
rf_confusion$overall['Accuracy']

## Accuracy
##        1

rf_confusion$overall['AccuracyUpper']

## AccuracyUpper
##             1

rf_confusion$overall['AccuracyLower']

## AccuracyLower
##      0.999812

testing_rfpred <- predict(rfFit, newdata=testing_clean)
testing_rfpred

##  [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
```

writing out the predictions

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("./practical_ml_wk4_",i,".txt")

write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(testing_rfpred)

testing_rfpred

##  [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
```