Automatic analysis of exercise quality

Machine learning, course project

Synopsis

The problem

The data used for this project has been downloaded here:

http://groupware.les.inf.puc-rio.br/har

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions - the correct one (Class A) and with a four typical mistakes (Classes B, C, D, E). The goal of our analysis is to construct a prediction algorithm to automatically determine the class using other 159 variables as predictors.

Summary of analysis

We first looked at the given data and identified about 50 variables that were actually measured (the rest are either missing values or irrelevant things like the time of measurement). Then we constructed a simple decision tree several times based on a random subsample to identify the only 14 variables that affect the outcome.

Further, we split our data into a training set proper and a validation set, removed all the variables but the important 14 from the data and constructed a model based on boosting with trees. Its accuracy is 0.93. We used it to correctly guess 18 out of 20 test examplels.

Finally, we create a random forest model with accuracy about 0.99 and used it to correctly guess the remaining 2 test examples.

Exploratory analysis and cleaning the data.

Downloading etc.:

```
library(ggplot2)
library(knitr)
library(rpart)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked _by_ '.GlobalEnv':
##
##     best
```

```
library(lattice)
library(rpart.plot)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.3.0 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(plyr)
```

```
##
## Attaching package: 'plyr'
##
## The following object is masked _by_ '.GlobalEnv':
##
##     ozone
```

```
library(survival)
```

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

```
if (!file.exists("pml-training.csv")) {
        url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
        download.file(url,"pml-training.csv", method="curl")
}

if (!file.exists("pml-testing.csv")) {
        url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
        download.file(url,"pml-testing.csv", method="curl")
}

D <- read.csv("pml-training.csv")
T <- read.csv("pml-testing.csv")
set.seed(2603)
```

The training set contains 19622 rows and the test set contains 20 rows.

Now we'll remove the first 7 variables because they identify the subject, the time etc. - things not relevant to prediction).

```
training <- D[,8:160]
testing <- T[,8:160]
types <- data.frame(var=names(testing),type=as.character(lapply(testing,class)))
head(types)
```

```
##                   var    type
## 1           roll_belt numeric
## 2          pitch_belt numeric
## 3            yaw_belt numeric
## 4    total_accel_belt integer
## 5  kurtosis_roll_belt logical
## 6 kurtosis_picth_belt logical
```

```
head(testing[,1:8])
```

```
##   roll_belt pitch_belt yaw_belt total_accel_belt kurtosis_roll_belt
## 1    123.00      27.00    -4.75               20                 NA
## 2      1.02       4.87   -88.90                4                 NA
## 3      0.87       1.82   -88.50                5                 NA
## 4    125.00     -41.60   162.00               17                 NA
## 5      1.35       3.33   -88.60                3                 NA
## 6     -5.92       1.59   -87.70                4                 NA
##   kurtosis_picth_belt kurtosis_yaw_belt skewness_roll_belt
## 1                  NA                NA                 NA
## 2                  NA                NA                 NA
## 3                  NA                NA                 NA
## 4                  NA                NA                 NA
## 5                  NA                NA                 NA
## 6                  NA                NA                 NA
```

As we see, a lot of variables do not actually have any values. As it happens, only relevant variables are those numberic ones. We'll re-define the training and the testing sets:

```
relevant_var <- (types$type=='numeric')|(types$type=='integer')
training <- training[,relevant_var]
training$classe <- D$classe
testing <- testing[,relevant_var]
dim(training)
```

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

```
dim(testing)
```

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

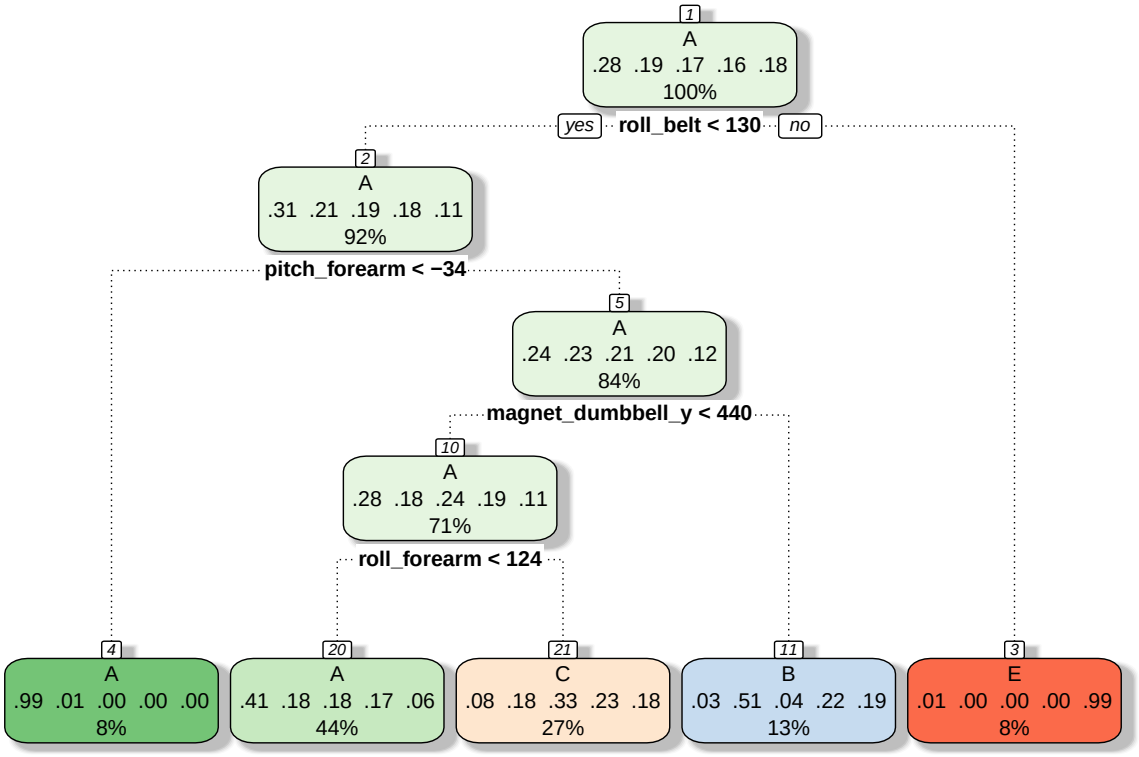
Decision tree that gives us important variables

Since the variable we are predicting is categorical, we won't even look at algorithms related to linear models and build our analysis entirely on trees. We'll construct a simple decision tree first to see which variables are imporant for prediction.

```
modTree <- train(classe ~ ., method="rpart", data=training)
```

```
## Loading required namespace: e1071
```

```
fancyRpartPlot(modTree$finalModel)
```



Rattle 2014−Dec−19 12:47:44 orlenkoirina

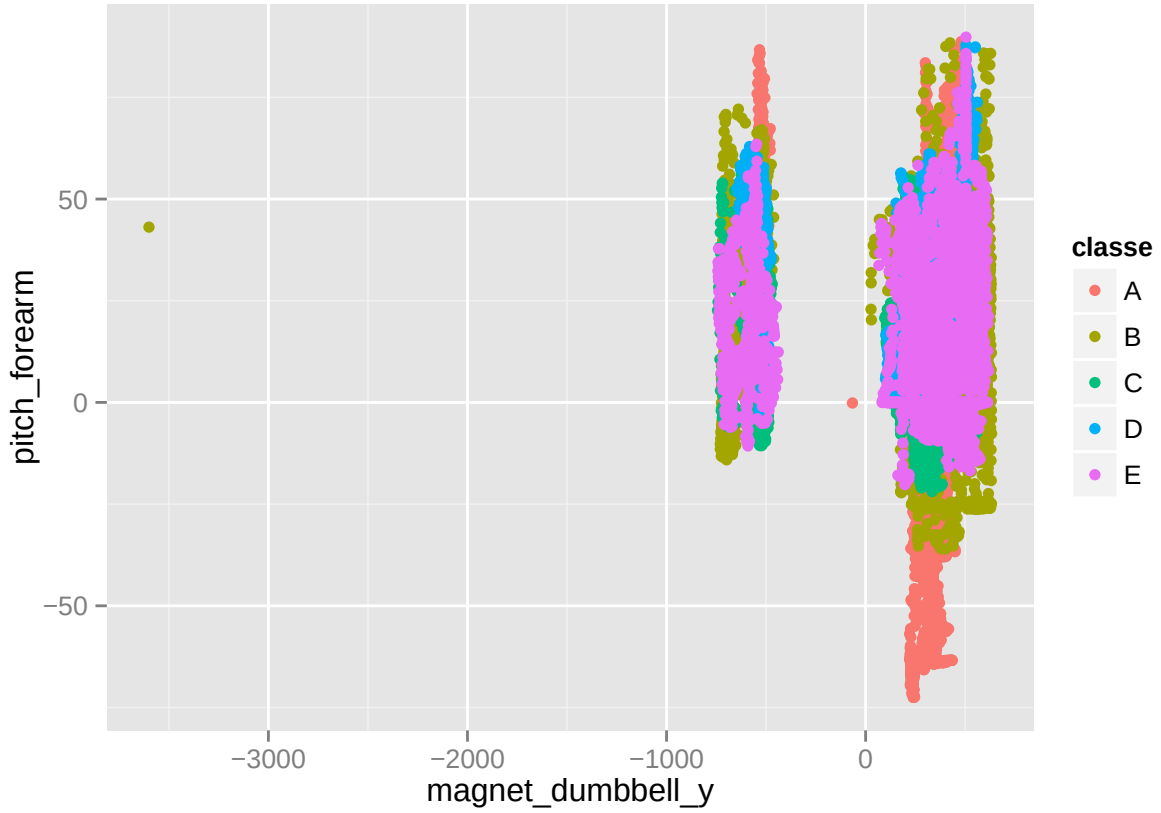
```
var.imp <- varImp(modTree)
print(var.imp)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                    Overall
## pitch_forearm       100.0
## roll_forearm         73.0
## roll_belt            70.4
## magnet_dumbbell_y    49.5
## accel_belt_z         43.0
## magnet_belt_y        40.8
## yaw_belt             40.4
## magnet_dumbbell_z    36.4
## total_accel_belt     35.3
## magnet_arm_x         27.1
## accel_arm_x          26.3
## roll_dumbbell        18.7
## accel_dumbbell_y     15.6
## roll_arm             15.3
## gyros_forearm_y       0.0
## yaw_arm               0.0
## accel_arm_z           0.0
```

```
## accel_belt_x          0.0
## yaw_dumbbell          0.0
## gyros_belt_z          0.0
```

I tried several times and the set of 14 important variables is always the same, so we'll use these 14 in to create our model. Here is a dot plot of the two most important variables colored by the class.

```
good.var <- row.names(var.imp$importance)[1:14]
qplot(data=training, x=magnet_dumbbell_y, y=pitch_forearm, colour=classe)
```



However, any particular decision tree itself has a very low accuracy. We'll try a boosting algorithm based

The boosting model.

Creating the model

We'll use a 'GBM' model because it is also based on trees so it seems reasonable to combine it with the variables that we got from obtaining a simple decision tree.

First, we'll remove all the variables but the relevant 14 and create a training subset and a validation set to be able to estimate the accuracy of our algorithm later.

```
inTrain <- createDataPartition(y=training$classe,
                               p=0.7, list=FALSE)
training.set <- training[inTrain,]
validation.set <- training[-inTrain,]
dim(training.set)
```

```
## [1] 13737    53
```

```
dim(validation.set)
```

```
## [1] 5885   53
```

```
training.set <- training.set[,c(good.var,"classe")]
validation.set <- validation.set[,c(good.var,"classe")]
testing <- testing[,c(good.var,"problem_id")]
```

We'll predict the class on the validation set and compare with the known answer:

```
modGBM <- train(classe ~ ., method="gbm",data=training.set)
```

```
## Loading required package: gbm
## Loading required package: parallel
## Loaded gbm 2.1
```

```
print(modGBM)

predictGBM <- predict(modGBM, newdata=validation.set)
C <- confusionMatrix(predictGBM,validation.set$classe)
```

Here is the whole confusion matrix

```
C$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1620   43    2    5    1
##          B   23 1009   57    8   11
##          C   20   61  937   50    8
##          D   10   23   30  893   19
##          E    1    3    0    8 1043
```

The overall accuracy is 0.9349. The actual accuracy on the given test set is 18 out of 20, i.e., 0.95.

Random forest

We also constructed a random forest model with 0.99 accuracy and used it to correct the two answers that were guessed wrongly by the GBM algorithm. However, it takes a long time to actually construct it (20 minutes or so on my computer), so I won't include it ito my analysis.

System info

Our analysis has been originally created and run in RStudio v. 0.98.1080 under OS X 10.9.5.

Time and date the report has been generated: 2014-12-19 12:58:19 (Central European time).