Prediction Assignment

Practical Machine Learning

Dennis van den Berg

2015-10-25

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. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

In this project, we used 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. We constructed and applied a Machine Learning algorithm that aims to predict activity quality based on the accelerometer data.

Exploratory Data Analysis

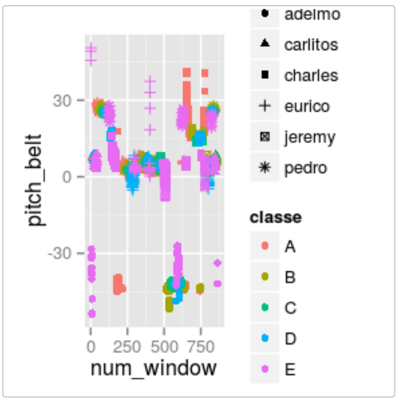
Raw data

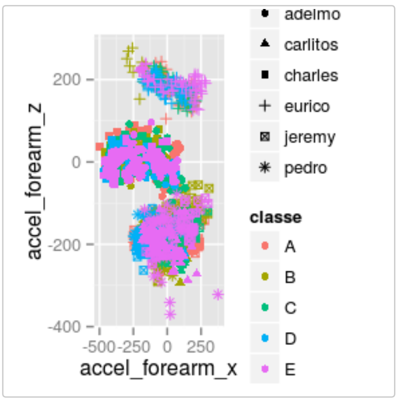
The input data set consists of 19622 measurements of 160 variables and the data set to use for prediction has 20 measurements. The outcome variable that we want to predict is `classe`, a factor variable with 5 different levels: A, B, C, D, E, denoting the quality of the exercise performed. Both data sets contain time series measurements with time variables `raw_timestamp_part_1`, `raw_timestamp_part_2`, `cvtd_timestamp` and measured in time windows denoted by `num_window`. At the end of each window aggregate data is calculated (denoted by `new_window=yes`) and was incorporated into the same dataset. Measurements have been done for 6 persons (variable: user_name).

This data was split into a training set, testing set and predicting set. Due to computation time constraints we decided to choose a smaller size for the training set (p=0.1), which eventually led to very good results.

The following example plots give an idea of some of the data:

```
qplot(num_window, pitch_belt, data=df.training, colour=classe, pch=user_name)
qplot(accel_forearm_x, accel_forearm_z, data=df.training, colour=classe, pch=user_name)
```





Data Cleaning

The data sets were cleaned up by removing variables that are correlated to classe due to experiment setup (such as `user_name`, timestamps and window number). We also decided to use only non-aggregate data (`new_window=="no"`). Furthermore we removed columns with only NA values.

This resulted in a more compact training set with 1934 measurements of 53 variables and a large testing set with 17282 such measurements of 53 variables, to be used for validation.

Model Building

Random Forest

We first used a simple tree model (method="rpart"), which gave very poor results (accuracy=0.5). A boosting model (method="gbm") gave a memory allocation error, which led us to a random forest model as an alternative. To speed up results we used the `prox=FALSE` option during the train process, as well as a relatively small training set as mentioned earlier.

```
model.rf.2 <- train(factor(classe) ~ ., data=df.training, method="rf", prox = FALSE)
```

Results

Accuracy

We find an accuracy of 99.3% on the training set (in-sample error 0.7%). Cross validating our model on the test set we find a 94.4% accuracy (out-of-sample error 5.6%).

```
##    list.training.results
##       A   B   C   D   E
##   A 544   0   0   2   0
##   B   2 361   5   0   0
##   C   0   0 342   0   0
##   D   1   0   2 313   0
##   E   0   1   1   0 356
```

```
## [1] 0.9927461
```

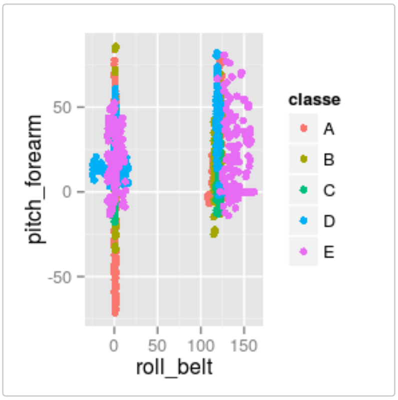
```
##    list.testing.results
##        A    B    C    D    E
##   A 4868   31   15    4    7
##   B  199 2967  134   41    9
##   C    1  128 2813   68    0
##   D   16   26  107 2666   16
##   E    9   93   34   36 2998
```

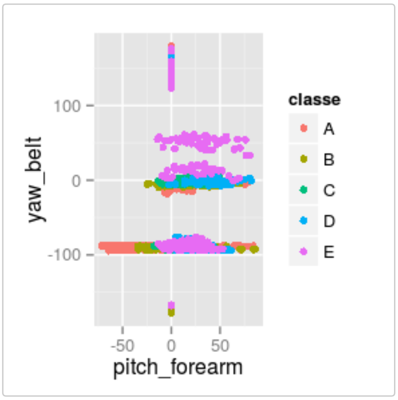
```
## [1] 0.9436538
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          70.18
## yaw_belt               50.69
## magnet_dumbbell_z      50.15
## pitch_belt             42.66
## magnet_dumbbell_y      40.87
## roll_forearm           37.17
## accel_dumbbell_y       25.29
## magnet_dumbbell_x      24.35
## roll_dumbbell          21.44
## accel_forearm_x        20.17
## magnet_belt_z          16.64
## magnet_belt_y          15.47
## accel_dumbbell_z       15.30
## gyros_dumbbell_y       12.17
## total_accel_dumbbell   12.12
## accel_belt_z           12.05
## magnet_belt_x          11.03
## gyros_belt_z           10.87
## magnet_forearm_z       10.55
```

According to this model the most important variables affecting the outcome are: roll_belt, pitch_forearm, yaw_belt and magnet_dumbbell_z. Plotting some of these shows patterns that help explain their relevance for determining the outcome variable `classe`:

```
qplot(roll_belt, pitch_forearm, data=df.training, colour=classe)
qplot(pitch_forearm, yaw_belt, data=df.training, colour=classe)
```





Test Case Results

Applying the model we just trained to 20 different test cases with unknown `classe`, we find:

```
predict(model.best, df.predicting)
```

```
##  [1] B A B A A E D D A A B C B A E E A B A B
## Levels: A B C D E
```

We expect an accuracy of 94.4% here. It is therefor likely that ~1 of these 20 cases has been predicted incorrectly.

Discussion

We made the following choices in our analysis:

- Small training set (p=0.1), mainly to speed up the training process.
- Removed aggregated results from data set and only focus on single measurements. This also simplified the number of variables in the input data set.
- Removed variables that are correlated to the outcome variable due to experiment setup.
- Since this is a classification problem with a lot of non-linearity in the data we chose a random forest model. This gave clearly better results than a simple tree model. A boosting model would have been a potential alternative but required more memory space than we had available.
- To speed up results we used the `prox=FALSE` option during the train process.
- Cross validation was done on a fairly large testing set, also due to the fact that we chose our training set to be small. This gives us confidence in our estimate of the model accuracy.

Overall these choices gave us a very satisfactory accuracy.

Data Source

We used the Weight Lifting Exercises Dataset available at http://groupware.les.inf.puc-rio.br/har provided by:

- Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Appendix A: R Code

```
### Practical Machine Learning - Prediction assignment R code

## Preparations

# Variables
set.seed(1234)
dir.work <- "~/git/practical_machine_learning/prediction_assignment/data"
file.train <- "pml-training.csv"
file.predict <- "pml-testing.csv"
# Set p small to speed up training process (with risk of lower accuracy)
p.training <- 0.1

# Libraries
library(caret)
library(ggplot2)

# Load data
setwd(dir.work)
df.train.file <- read.csv(file.train, na.strings=c("","#DIV/0!","NA"))
df.predict.file <- read.csv(file.predict, na.strings=c("","#DIV/0!","NA"))

# Divide into training, testing and predicting set
m.train <- createDataPartition(df.train.file$classe, p=p.training, list = FALSE)
df.training <- df.train.file[m.train,]
df.testing <- df.train.file[-m.train,]
df.predicting <- df.predict.file


## Exploring raw data sets

# Check data sets
dim(df.training)
dim(df.testing)
names(df.training)
head(df.training)
lapply(df.training, class)
lapply(df.testing, class)

# Some plots based on training
qplot(num_window, pitch_belt, data=df.training, colour=classe, pch=user_name)
qplot(accel_forearm_x, accel_forearm_z, data=df.training, colour=classe, pch=user_name)
```

```
## Data Cleaning

# Remove variables from training dataset that are correlated to classe due to experiment setup
# Furthermore, only use non-aggregated data (new_window=="no")
df.training <- subset(df.training, new_window=="no", select=-c(X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cv
df.testing <- subset(df.testing, new_window=="no", select=-c(X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd

# Remove columns with only NA values
df.training <- df.training[,colSums(is.na(df.training))<nrow(df.training)]
df.testing <- df.testing[,colSums(is.na(df.testing))<nrow(df.testing)]


## Model 1: rpart

# rpart model gave poor results (accuracy=0.5), not used anymore
#model.rpart <- train(factor(classe) ~ ., method="rpart", data=df.training)


## Model 2: gbm

# gbm model gave memory allocation error, not used for now
#model.gbm <- train(factor(classe) ~ ., method="gbm", data=df.training)


## Model 3: rf

# Train model
#model.rf.1 <- train(factor(classe) ~ ., data=df.training, method="rf")
model.rf.2 <- train(factor(classe) ~ ., data=df.training, method="rf", prox = FALSE)
#model.rf.3 <- train(factor(classe) ~ ., data=df.training, method="rf", trControl=trainControl(method="cv", number=5), prox


## Investigate model

# Set selected model and display
model.best <- model.rf.2
model.best
model.best$finalModel

# Confusion table and accuracy for training set
list.training.results <- predict(model.best, newdata=df.training)
table(df.training$classe, list.training.results)
sum(df.training$classe == list.training.results) / length(df.training$classe)

# Confusion table and accuracy for testing set
list.testing.results <- predict(model.best, newdata=df.testing)
table(df.testing$classe, list.testing.results)
sum(df.testing$classe == list.testing.results) / length(df.testing$classe)

# Most important variables
varImp(model.best)

# Save/load model to/from file
saveRDS(model.best, "model_rf.rds")
# model <- readRDS("model_rf.rds")


## Making results more insightful

# Plot a few important variables
qplot(roll_belt, pitch_forearm, data=df.training, colour=classe)
qplot(roll_belt, yaw_belt, data=df.training, colour=classe)
qplot(pitch_forearm, yaw_belt, data=df.training, colour=classe)


## Predict

# Apply best model to predicting set
list.predicting.results <- predict(model.best, df.predicting)
list.predicting.results

# Write to files
pml_write_files = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0("problem_id_",i,".txt")
        write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
    }
}
pml_write_files(list.predicting.results)
```