A Prediction of Activity Quality

Summary

With devices such as Jawbone, Fitbit and so one, people regularly measures how much of a particular activity they do, but they rarely quantify how well they do it. In this small project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to build a supervised classifier algorithm to predict activity quality. The data for this project is provided by Groupware@LES (mailto:Groupware@LES) on the website http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Three machine learning algorithm (Neural Networks, Boosted Trees and Random Forest) were applied. Results shows that a random forest model fits best the data with an accuracy of 0,974% on the validation dataset and 100% on the test dataset.

Data Processing

Setup directory, default parameters and required packages

```
library(knitr)
opts_chunk$set(echo = TRUE, cache = FALSE, eval = FALSE)
```

```
setwd("~/Desktop/Machine Learning")
library(caret); library(caTools); library(randomForest)
```

Load data

```
trainRawData <- read.csv("pml-training.csv", na.strings = c("NA", ""))
dim(trainRawData)

testRawData <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
dim(testRawData)
```

Delete covariate with NA

NAs are unbalanced in the training dataset. Covariates have either NA = 0 or NA = 19220. These can't be imputed easily. I've therefore decided to delete each covariate with NA > 19000 (which in this case also means NA >0). It results in a training dataset of 60 covariates. The same variables were then also removed from the testing dataset.

```
#Training
NAtrain <- apply(trainRawData, 2, function(x) { sum(is.na(x)) })
trainRawData2 <-  trainRawData[ , NAtrain < 19000]
dim(trainRawData2)

#Testing
testRawData2 <- testRawData[ , NAtrain < 19000]
dim(testRawData2)
```

Change the data types

For later purpose (i.e. to ease the ACP analysis), I transform the factor variable "user_name" into numeric for both the training and testing dataset.

```
#Training
trainRawData2$user_name <- as.numeric(trainRawData2$user_name)
#Testing
testRawData2$user_name <- as.numeric(testRawData2$user_name)
```

Remove unrequired variables

To reduce overfitting issues, 2 variables (i.e. "X" and "new-window") which ahve no potential explaination power were removed from both datset. Finally, to avoid carefull care required by panel data, timestamps were also removed from training and testing dataset.

```
#training
unrequired <- grep("X|new_window|timestamp", names(trainRawData2))
trainRawData2 <- trainRawData2[, -unrequired]
dim(trainRawData2)

#testing
unrequired <- grep("X|new_window|timestamp", names(testRawData2))
testRawData2 <- testRawData2[, -unrequired]
dim(testRawData2)
```

Store the processed Data

First, the training set was divided into a Training set and a Validation set.

```
set.seed <- 100
inTrain  <-  createDataPartition(trainRawData2$classe, p = 4/5)[[1]]
training <- trainRawData2[inTrain, ]
validation <- trainRawData2[-inTrain, ]
```

Then, the processed data was stored for testing. The size of each dataset is shown below.

```
testing <-   testRawData2
dim(training); dim(validation); dim(testing)
```

Store the processed Data

First, the training set was divided into a Training set and a Validation set.

```
set.seed <- 100
inTrain  <-  createDataPartition(trainRawData2$classe, p = 4/5)[[1]]
training <- trainRawData2[inTrain, ]
validation <- trainRawData2[-inTrain, ]
```

Then, the processed data was stored for testing. The size of each dataset is shown below.

```
testing <-   testRawData2
dim(training); dim(validation); dim(testing)
```

Running the Principal Component Analysis

Since only the “user_name” variable is a categorical variable, I apply a PCA on the traiing dataset to reduce the size to avoid multicollinearity and reduce overfiiting issues. More appropriate technique to deal with mixed data types, namely Multiple Factor Analysis for mixed data available in the FactoMineR R package (cfr. FAMD) were not really justify in the case.

The first 13 components were selected according to their variances >= 1. The same size reduction was applied on the validation and testing datasets.

```
# training
#http://stats.stackexchange.com/questions/72839/how-to-use-r-prcomp-results-for-prediction
PCA <- prcomp(training[, 1:54], retx=TRUE, center=TRUE, scale = TRUE)
summary(PCA)
predictors <- PCA$x[, 1:13] #rotated data for the 14 component
outcome  <- training[,55]
training2  <- data.frame(outcome, predictors)

#validation
predictors <- predict(PCA, validation)  #rotated data using the same PCA
predictors <- predictors[, 1:13]
outcome  <- validation[,55]
validation2  <- data.frame(outcome, predictors)

#testing
predictors <- predict(PCA, testing)  #rotated data using the same PCA
predictors <- predictors[, 1:13]
outcome  <- testing[,55]
testing2  <- data.frame(outcome, predictors)
```

Train prediction model

The outcome “classe” were predicted by training 3 classifier algorithms (Neural networks, Boosted Trees and Random Forest) on the 13 factors from the PCA.

```
model_nnet <- train(outcome ~ ., method = "nnet", data = training2)
model_gbm <- train(outcome ~ ., method = "gbm", data = training2)
model_rf <- train(outcome ~ ., method = "rf", data = training2, prox = TRUE)

#pred_nnet <- predict(model_nnet, training2)
#pred_gbm  <- predict(model_gbm,  training2)
#predDF    <- data.frame(pred_nnet, pred_gbm, outcome = training2$outcome) #TROP long
#model_combiner <- train(outcome ~ ., method = "rf", data = predDF)
```

The best predictor is by far the Random Forest with an accuracy of 0.974% (i.e. “nnet” is 0.55% and “gbm” is 0.776%). Because of it’s dominance, it is not really worth combining them together using a model stacking.

```
validation_nnet <- predict(model_nnet,validation2)
accuracy_nnet <- confusionMatrix(validation2$outcome, validation_nnet)
accuracy_nnet$overall[1]

validation_gbm<- predict(model_gbm,validation2)
accuracy_gbm <- confusionMatrix(validation2$outcome, validation_gbm)
accuracy_gbm$overall[1]

validation_rf<- predict(model_rf,validation2)
accuracy_rf <- confusionMatrix(validation2$outcome, validation_rf)
accuracy_rf$overall[1]

#validation_predDF <- data.frame(pred1= validation_nnet, pred2= validation_gbm)
#validation_gam <- predict(model_combiner, validation_predDF)
#accuracy_gam <- confusionMatrix(validation2$outcome, validation_gam)
#accuracy_gam$overall[1]
```

Results

The best model according to the cross valisation test is the Random Forest with an accurancy of 0,974% . Let’s now predict the outcome on the testing dataset.

```
testing_rf<- predict(model_rf,testing2)
```

Write up

Finally, results are exported in the working directory.

```r
answers <- testing_rf

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(answers)
```