PML Assingment

Beatriz JimÃ©nez Franco

14/11/2020

Predict activity quality from accelerometers sensors,

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 of 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. In this project, the goal will be 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 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har(see the section on the Weight Lifting Exercise Dataset).

Goal of the proyect

The goal of the project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. I will also use the prediction model to predict 20 different test cases.

Data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Data processing

Download data. The pml-training.csv data is used as training and the pml-test.csv data as testing (is used to predict 20 cases based on the best trained model).

```
getwd()
```

```
## [1] "C:/Users/pruebas/Documents/practicalmachinelearning"
```

```
setwd("C:/Users/pruebas/Documents/practicalmachinelearning")
urltr <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlts <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training   <-read.csv(url(urltr), na.strings=c("NA","#DIV/0!", ""), stringsAsFactors = F)
testing <-read.csv(url(urlts), na.strings=c("NA","#DIV/0!", ""))
dim(training)
```

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

```
dim(testing)
```

```
## [1]  20 160
```

NA values are omitted in both datasets. Irrelevant columns such as user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, and num_window (columns 1 to 7) will be removed in the subset.

```
training<-training[,colSums(is.na(training)) == 0]
testing <-testing[,colSums(is.na(testing)) == 0]
training   <-training[,-c(1:7)]
testing <-testing[,-c(1:7)]
```

Exploratory analysis

I plot a correlogram to see whether we could simplify the data set by identifying pairs of strongly correlated variables that are shown with the dark red and blue dark colours:

See the pdf in the repo files.

```
pdf("multi-corr.pdf",50,50)
library(corrgram)
corrgram(training, order = TRUE, lower.panel=panel.pie)
dev.off()
```

```
## pdf
##   2
```

We donÂ´t observe a hight number of correlation variables, so we will try two preprocessings before training the model: - Preprocessing using PCA analysis before training the model(we will used a 95% of thresh, it is the default % of variance that should have the principal components). - No preprocessing with PCA, so we used all the variables to train the model.

Cross validation

Cross validation uses 5 folds for resampling to compute the accuracy in order to select the optimal model using the largest value tuning parameter(mtry).

Prediction mnodels on the training set

The reason i use random forest algorithm to train the model is beacause it is one of the most accurate prediction ones. I will test a version with all variables as well as a version with feature space reduced by PCA. In both cases, we will train the model with 5-folds cross-validation to estimate an out-of-sample error.

First randomforest cross validated trained model without pca

Do parallel is to meke computations more quickly....

```
library(caret)
```

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

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:corrgram':
##
##     panel.fill
```

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

```
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
set.seed(1234)
fit.nopca <- train(classe ~ ., data = training, ntree = 100, method = 'rf',
    trControl = trainControl(method = "cv", number = 5,allowParallel = TRUE))
stopCluster(cluster)
registerDoSEQ()
```

Secomd randomforest cross validated trained model with pca

```
fit.pca <- train(classe ~ ., data = training, ntree = 100, method = 'rf',
    preProcess = "pca", trControl = trainControl(method = "cv", number = 5))
```

Expected out-of-sample error.

In order to select the best final training model that i will use to predict with, i take the smallest OOB estimate of error rate (0.52 %) . I fitted two random forest models with and without principal component preprocessing to see which one improves my trainin model fit. The OOB estimate error uses unknown cases in each split for the training model because they were not used to fit the model, instead they were used to estimate the best performance of the model that we can see in the confusion matrix that shows the errors of the prediction algorithm.

```
fit.nopca$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 100, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.49%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 5572    7    1    0    0 0.001433692
## B   20 3770    6    1    0 0.007110877
## C    1   10 3401   10    0 0.006136762
## D    0    0   27 3186    3 0.009328358
## E    0    1    3    6 3597 0.002772387
```

```
fit.pca$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 100, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.13%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 5531   15   23    8    3 0.008781362
## B   53 3691   39    3   11 0.027916776
## C    6   45 3333   29    9 0.026008182
## D    4    3  105 3097    7 0.037002488
## E    1   22   16   15 3553 0.014970890
```

The model with all features performs better than the one with PCA pre-processing.

Compute the accuracy of testing

We will use this code but in this case it is not necessary because i displayed above the same confussion matrix and out of error with the final model option:

confusionMatrix((testing$classe), predict(fit.nopca, testing))

Predictions

We use to predict 20 cases the testing data set, with the best final model that we select above (fit.nopca):

```
predictions = predict(fit.nopca, testing)
predictions
```

```
##  [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
```