Peer_Asignment

Aurora González Vidal

Mon Jun 13 10:11:05 2016

```
library("caret")
```

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

```
library("xtable")
```

In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participant.

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Loading the data

```
train <- read.table("pml-training.csv", header = T, sep = ",", dec=".")
test <- read.table("pml-testing.csv", header = T, sep = ",", dec=".")
```

Missing values treatment

There are some attributes that are `NA` for all observations in both sets so we are going to discard those and create new train and test sets:

```
sapply(test, function(x) sum(is.na(x)))/nrow(test)*100
sapply(train, function(x) sum(is.na(x)))/nrow(train)*100

index <- as.vector(sapply(train, function(x) sum(is.na(x)))/nrow(test)*100!=0)

train2 <- train[!index]
test2 <- test[!index]
```

Standarization of variables

Variables are measured in different scales so in order to avoid that they weight different we have to standarize them:

```r
procValues <- preProcess(train2, method = c("center", "scale"))
trainNormalized <- predict(procValues, train2)
testNormalized <- predict(procValues, test2)
```

```r
inputs_train <- subset(trainNormalized, select = -c(classe))   # get rid of the classe
inputs_test <- subset(testNormalized, select = -c(problem_id))  # get rid of the id
output_train <- trainNormalized["classe"]
```

Reduction of dimensionality

There are still too many variables:

```r
dim(inputs_train)
```

```
## [1] 19622    92
```

so we are going to use PCA in order to reduce the dimensionality of the problem.

```r
precPCA <- preProcess(inputs_train[,1:ncol(inputs_train)-1], method = "pca",thresh = 0.9)
inputs_trainPCA <- predict(precPCA,  inputs_train )
inputs_testPCA <- predict(precPCA, inputs_test )
```

Cross validation, building the model

The train set is going to be splitted itself into a train and a test set in order to evaluate the results of the model. So, now we have `train3` and `test3` sets.

```r
set.seed(1234)
ctrl <- trainControl(method = "cv", repeats = 5
                     , returnResamp = "all")

nTraining <- as.integer(nrow(inputs_trainPCA) * 0.6)
indices <- sample(1:nrow(inputs_trainPCA), nTraining)

train3 <- cbind(inputs_trainPCA[indices,], classe = output_train[indices,])
test3 <- cbind(inputs_trainPCA[-indices,], classe = output_train[-indices,])
```

We use the `train3` set when looking for the best hyperparameters. We define a grid of feasible optimal parameters. We save the model.

```r
paramGrid <- expand.grid(.mtry = c(2,3,4,5,10,15))
set.seed(1234)
rf.final <- caret::train(
  classe~.,
  data = train3[38:59], # Inputs + Outputs
  method = "rf",
  metric = "Accuracy", # Metric to evaluate
  tuneGrid = paramGrid, # Parameters for tunning
  trControl = ctrl # Parameters of control
)


save(rf.final, file = "my_model2.rda")
```

This are the outputs of the model:

```r
load("my_model2.rda")
rf.final
```

```
## Random Forest
##
## 11773 samples
##    21 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10595, 10596, 10595, 10595, 10596, 10595, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9862409  0.9825984  0.005198837  0.006576077
##    3    0.9864956  0.9829206  0.005463891  0.006910835
##    4    0.9866655  0.9831358  0.004401242  0.005567292
##    5    0.9865799  0.9830283  0.004578089  0.005790475
##   10    0.9820779  0.9773335  0.004355820  0.005509380
##   15    0.9783402  0.9726070  0.005270101  0.006666212
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 4.
```

We see that the best `Accuracy` is obtained for mtry = `rf.final$bestTune`, so this one is the chosen one.

Evaluation

Now, we use the `test3` set in order to predict the `classe` variable and evaluate the performance of the model:

```r
rf.pred <- predict(rf.final, test3)
```

```
## Loading required package: randomForest
## 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
```

```
confusionMatrix(test3$classe, rf.pred)$overall['Accuracy']
```

```
##  Accuracy
## 0.9889158
```

Final prediction

As we don't have the real values for the subjects we cannot check the performance of the evaluation. Anycase, we expect a very good `Accuracy` because we checked it with a part of the train set.

```
rf.predF <- predict(rf.final, inputs_testPCA)
data.frame(id = test$user_name, result= rf.predF)
```

```
##          id result
## 1     pedro      A
## 2    jeremy      A
## 3    jeremy      A
## 4    adelmo      A
## 5    eurico      A
## 6    jeremy      A
## 7    jeremy      A
## 8    jeremy      B
## 9  carlitos      A
## 10  charles      A
## 11 carlitos      B
## 12   jeremy      A
## 13   eurico      B
## 14   jeremy      A
## 15   jeremy      A
## 16   eurico      A
## 17    pedro      A
## 18 carlitos      B
## 19    pedro      B
## 20   eurico      B
```