DS8 - Machine Learning Project

Irene Perez

20 March 2015

Synopsis

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, your 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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The goal of this 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.

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Prepare libraries

```
#Load the required Libraries


library(caret)
```

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

```
library(abind)
library(arm)
```

```
## Loading required package: MASS
## Loading required package: Matrix
## Loading required package: lme4
## Loading required package: Rcpp
##
## arm (Version 1.7-07, built: 2014-8-27)
##
## Working directory is /Users/ireneperez/datasciencecoursera/8 - Machine Learning
```

```r
library(kernlab)
library(klaR)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```r
library(rpart)
```

Retrieve the data set

```r
# Seed for pseudo-random generator and reproduceablity
set.seed(1970)

# Read the Training and Testing Data Sets...  only want to store the data files in memory

trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

trainDT <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))

testDT <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Cleaning the data

I'm interested in variables that predict the movement. The data set contains some variables that can be removed

```
# Step 1 - Identify Variables that are mostly NA's and remove them

# A number of variable contain (a lot of) NA's. Leaving them in the set not only makes the model
 # creation slower, but also results in lower accuracy. These variables will be
# removed from the set if contains 60%+ NA values

naprops <- colSums(is.na(trainDT))/nrow(trainDT)
mostlyNAs <- names(naprops[naprops > 0.60])

mostlyNACols <- which(naprops > 0.60)

length(mostlyNACols)  # there are 100 variables/columns with 60%+ NA values
```

```
## [1] 100
```

```
trainDT_V1 <- trainDT[,-mostlyNACols]

testDT_V1 <- testDT[,-mostlyNACols]  # variable/columns are the same in both data sets


# Step 2 - Identify and clean Near Zero Variance Variables

# some varialbes have near Xero variance which indicates that they do not contribute enough to
# the model, those are removed.

trainDT_V2 <- trainDT_V1[,-nearZeroVar(trainDT_V1)]

testDT_V2 <- testDT_V1[,-nearZeroVar(testDT_V1)]


# Step 3 - Remove variables related to data acquisition

# Variables such as id, timestamps, individual's names are not candidate predictors
# the first 5 variable/columns from the data set are removed

trainDT_V3 <- trainDT_V2[, -(1:5)]

testDT_V3 <- testDT_V2[, -(1:5)]
```

Spliting Train data into Train and Validation Sets

We now split the updated training dataset into a training dataset (70% of the observations) and a validation dataset (30% of the observations). This validation dataset will allow us to perform cross validation when developing our model.

```
Training = createDataPartition(y = trainDT_V3$classe, p = 0.7, list = FALSE)
train_data = trainDT_V3[Training, ]
val_data = trainDT_V3[-Training, ]
```

Correlation Analysis

Identify how many correlated variables there are.

```
Hcorr <- caret::findCorrelation(cor(train_data[,-54]),cutoff=0.8)
length(Hcorr)
```

```
## [1] 12
```

As you can see from above, many variables are highly correlated, hence, PCA will be used in the pre-processing.

```
#corMat <- cor(train_data[, -54])
#corrplot(corMat, order = "FPC", method = "color", type = "lower", tl.cex = 0.8, tl.col #= rgb(0
, 0, 0))
```

This grid shows the correlation between pairs of the predictors in our dataset. From a high-level perspective darker blue and darker red squares indicate high positive and high negative correlations, respectively. We choose to use a principal components analysis to produce a set of linearly uncorrelated variables to use as our predictors.

At this point, our dataset contains 54 variables, with the last column containing the 'classe' variable we are trying to predict.

```
names(train_data)
```

```
##  [1] "num_window"           "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

Model especification

To avoid overfitting and reduce out of sample errors, TrainControl is used to perform 10-fold cross validation

```
tc <- trainControl(method = "cv", number = 10, verboseIter=FALSE , preProcOptions="pca", allowPa
rallel=TRUE)
```

Five models are estimated:

1 -> Random forest

```
rf <- train(train_data$classe ~ ., data = train_data, method = "rf", trControl= tc)
```

2 & 3 -> Support Vector Machine (both radial and linear)

```
svmr <- train(classe ~ ., data = train_data, method = "svmRadial", trControl= tc)

svml <- train(classe ~ ., data = train_data, method = "svmLinear", trControl= tc)
```

4 -> Bayes Generalized linear model

```
bayesglm <- train(classe ~ ., data = train_data, method = "bayesglm", trControl= tc)
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in .bayesglm.fit.loop.printWarnings(Warning, state, family):
## fitted probabilities numerically 0 or 1 occurred
```

5 -> Logit Boosted model

```
logitboost <- train(classe ~ ., data = train_data, method = "LogitBoost", trControl= tc)
```

```
## Loading required package: caTools
```

Accuracy comparision

```
model <- c("Random Forest", "SVM (radial)","LogitBoost","SVM (linear)", "Bayes GLM")
Accuracy <- c(max(rf$results$Accuracy), max(svmr$results$Accuracy),max(logitboost$results$Accura
cy), max(svml$results$Accuracy),  max(bayesglm$results$Accuracy))

Kappa <- c(max(rf$results$Kappa),max(svmr$results$Kappa), max(logitboost$results$Kappa),
 max(svml$results$Kappa),  max(bayesglm$results$Kappa))

performance <- cbind(model,Accuracy,Kappa)

knitr::kable(performance)
```

model	Accuracy	Kappa
Random Forest	0.997306070751339	0.996592388002984
SVM (radial)	0.930479799161029	0.911943679607933
LogitBoost	0.927541235191649	0.907788654447583
SVM (linear)	0.789180732465452	0.732124621769375
Bayes GLM	0.401981244210405	0.235454588068098

Random forest provides the best results and will provide the predictions for the submission. Even if the Out of sample error cannot be estimated exactly, the in-sample error obtained through cross-validation is calculated over different test sets and should provide a better estimate of out-of sample error with respect to the case of no cross-validation.

The estimated accuracy of the model is 99.7% and the estimated out-of-sample error based on our fitted model applied to the cross validation dataset is 0.3%.

Cross Validation and Testing and Out-of-Sample Error Estimate

Apply the trained model to our cross validation test dataset. We'll see the resulting table in the 'Confusion Matrix' function's output to see how well the movel predicted/classified the values in the validation test set.

```
rf_val_data <- predict(rf, val_data)

Confus_val_data <- confusionMatrix(val_data$classe, rf_val_data)

Confus_val_data$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    2 1137    0    0    0
##          C    0    1 1025    0    0
##          D    0    0    4  960    0
##          E    0    0    0    2 1080
```

```
accur <- postResample(val_data$classe, rf_val_data)

model_accuracy <- accur[[1]]

model_accuracy
```

```
## [1] 0.9984707
```

```
out_of_sample_error <- 1 - model_accuracy

out_of_sample_error
```

```
## [1] 0.001529312
```

The estimated out-of-sample error is 1.000 minus the model's accuracy. The estimated accuracy of the model is 99.7% and the estimated out-of-sample error based on our fitted model applied to the cross validation dataset is 0.3%.

Prediction of "classe" variable for the test set

Finally, we apply our model to the original testing dataset, and display the predicted results.

```
rfPred <- predict(rf, testDT_V3)

rfPred
```

```
##  [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
```

Submission to Coursera

Function to generate files with predictions to submit for assignment

```
pred_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)
} }

 pred_write_files(rfPred)
```