Machine Learning Project

ch8stanford@gmail.com (mailto:ch8stanford@gmail.com)

December 26, 2017

Executive summary

The goal of this project is to predict the manner in which participants exercised based on the data from the following website: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har), which is the “classe” variable in the training set. Since it is a classification problem, we decided early on that a random forest model should be tried, with a Generalized Boosted Model (the “GBM” model) as a reserve. To our delight, the random forest model achieved an accuracy rate of over 99% on both the testing and the valuation data sets, which exceeded the performance of the backup GBM model. We therefore used our random forest prediction model to predict the 20 different test cases.

Loading relevant library and data

The following R code load up the necessary data and analytical packages.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

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

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

```
library (randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.3
```

```
## 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
```

```
test<-read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!",""))
train<-read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))
```

Data Cleaning

Next we remove columns that are obviously not predictors (i.e., the first 7 columns) and columns that have variables with mostly NAs (use threshold of >60%). Such data cleanups help both the speed and the accuracy of the model testings.

```
train_SUB <- train
train_SUB <- train_SUB[,8:length(train_SUB)]
for (i in 1:length(train)) {
  if (sum(is.na(train[ , i])) / nrow(train) >= .6) {
    for (j in 1:length(train_SUB)) {
      if (length(grep(names(train[i]), names(train_SUB)[j]))==1) {
        train_SUB <- train_SUB[ , -j]
      }
    }
  }
}

dim(train_SUB)
```

```
## [1] 19622    53
```

Create a bulding data set and validation set

Since this is a classification problem, an obvious candidate is the random forest approach. Random forest has certain possible advantages over other forms of classification models, including:

- Reduction in overfitting: by averaging several trees, there is a significantly lower risk of overfitting.
- Less variance: by using multiple trees, one reduces the chance of stumbling across a classifier that doesn't perform well because of the relationship between the train and test data.

As a consequence, random forests are often quite accurate. Yet, to be safe, we have also choosen the GBM model as the reserve, which can also be used in combination with the random forest approach as part of an ensembling method of learning. The GBM model uses boosting and is also known as one of the most widely-used and accurate prediction models.

Since we may wish to combine the random forest model with the GBM model as an ensembling method, we proceeded to divide the training data into 3 parts under the following R code: training, testing, and validation.

```
inBuild<-createDataPartition(y=train_SUB$classe,p=.7,list=FALSE)
validation<-train_SUB[-inBuild,]
buildData<-train_SUB[inBuild,]
inTrain<-createDataPartition(y=buildData$classe,p=.7,list=FALSE)
training<-buildData[inTrain,]
testing<-buildData[-inTrain,]
dim(training)
```

```
## [1] 9619   53
```

```
dim(testing)
```

```
## [1] 4118   53
```

```
dim(validation)
```

```
## [1] 5885   53
```

Model buildings & cross validations

The following R code builds a Random Forest model and tests its accuracy with the testing data set.

```
set.seed(12)
##Using randomForest instead of caret (with the "rf" method) because it is much faster.
mod1<-randomForest(classe~.,data=training)
prediction1 <- predict(mod1, testing)
cmrf <- confusionMatrix(prediction1, testing$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1169    4    0    0    0
##          B    2  791    4    0    0
##          C    0    2  709   12    2
##          D    0    0    5  662    3
##          E    0    0    0    1  752
##
## Overall Statistics
##
##                Accuracy : 0.9915
##                  95% CI : (0.9882, 0.9941)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9892
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9983   0.9925   0.9875   0.9807   0.9934
## Specificity            0.9986   0.9982   0.9953   0.9977   0.9997
## Pos Pred Value         0.9966   0.9925   0.9779   0.9881   0.9987
## Neg Pred Value         0.9993   0.9982   0.9973   0.9962   0.9985
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2839   0.1921   0.1722   0.1608   0.1826
## Detection Prevalence   0.2848   0.1935   0.1761   0.1627   0.1829
## Balanced Accuracy      0.9985   0.9953   0.9914   0.9892   0.9965
```

As one can see, the accuracy rate of the random forest model, when tested against the testing data set, is over 99%. To see whether we can get a better accuracy rate, we next try the GBM model.

```
set.seed(13)
fitControl <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
mod2 <- train(classe ~., method="gbm", data=training, trControl=fitControl, verbose=FALSE)

prediction2 <- predict(mod2, testing)
cmrf2 <- confusionMatrix(prediction2, testing$classe)
cmrf2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1156   21    0    1    1
##          B   13  741   13    2    9
##          C    1   29  693   16   14
##          D    1    4    9  649    8
##          E    0    2    3    7  725
##
## Overall Statistics
##
##                Accuracy : 0.9626
##                  95% CI : (0.9563, 0.9682)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9527
##  Mcnemar's Test P-Value : 0.006981
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9872   0.9297   0.9652   0.9615   0.9577
## Specificity            0.9922   0.9889   0.9824   0.9936   0.9964
## Pos Pred Value         0.9805   0.9524   0.9203   0.9672   0.9837
## Neg Pred Value         0.9949   0.9832   0.9926   0.9925   0.9905
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2807   0.1799   0.1683   0.1576   0.1761
## Detection Prevalence   0.2863   0.1889   0.1829   0.1629   0.1790
## Balanced Accuracy      0.9897   0.9593   0.9738   0.9775   0.9771
```

Final cross validation

As one can see, the accuracy rate of the GBM model is at about 96% and is significantly less than that of the random forest model. We decide, therefore, to forego the complexity of combining these two into a single ensembled method and will use the validation data set to double-check the accuracy of the random forest model instead.

```
prediction3 <- predict(mod1, validation)
cmrf3 <- confusionMatrix(prediction3, validation$classe)
cmrf3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1667   12    0    0    0
##          B    7 1123    6    0    0
##          C    0    4 1020   19    1
##          D    0    0    0  944    4
##          E    0    0    0    1 1077
##
## Overall Statistics
##
##                Accuracy : 0.9908
##                  95% CI : (0.988, 0.9931)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9884
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9860   0.9942   0.9793   0.9954
## Specificity            0.9972   0.9973   0.9951   0.9992   0.9998
## Pos Pred Value         0.9929   0.9886   0.9770   0.9958   0.9991
## Neg Pred Value         0.9983   0.9966   0.9988   0.9959   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2833   0.1908   0.1733   0.1604   0.1830
## Detection Prevalence   0.2853   0.1930   0.1774   0.1611   0.1832
## Balanced Accuracy      0.9965   0.9916   0.9946   0.9892   0.9976
```

Given that the random forest model has once again performed well with an accuracy rate around 99%, we decide to use it as our final model, since its error rate is already likely to be only around 1%.

Perform the final testing

The following R code applies the random forest model on the “test” dataset presented in the study.

```
prediction4 <- predict(mod1, test)
prediction4
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```