Practical Machine Learning Class Project
Peter Pih
(predmachlearn-030)
July 2015

Summary

We are given a data set with measurements of subjects doing excercises both correctly and incorrectly. Using supervised machine learning techniques, we are asked to model a predictor for the correctness of the excecise.

We use two classifiers random forests and gener

Loading Data and PreProcessing

```
setwd("C:/R/PracticalMachineLearning")
```

```
library(lattice)
library(ggplot2)
library(caret)
library(rpart)
library(randomForest)
```

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

Load the data:

```
trainFile <- "pml-training.csv"
testFile <- "pml-testing.csv"
train <- read.csv(trainFile, stringsAsFactors=FALSE, na.strings="NA")        # explicitly set NAs
realtest <- read.csv(testFile, stringsAsFactors=FALSE, na.strings="NA")      # explicitly set NAs
```

Check for differences between the training and testing data sets:

```
train_names <- names(train)
realtest_names <- names(realtest)
length(names(train)) == length(names(realtest))
```

```
## [1] TRUE
```

```
sum(! train_names %in% realtest_names)
```

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

The difference between the two data set is one field: the training data has ***classe*** and the testing data set has ***problem_id***. Modify the data sets, excluding columns with NAs, the new datasets are called **newtrain** and **realtest**.

```r
exclude_names <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp",
                   "num_window", "problem_id")

# Get variable names which are relevent to the realtest set
use_variables <- as.character()
for (n in realtest_names){
    if (sum(!is.na(realtest[,n])) > 0){
        if (! n %in% exclude_names){
            use_variables <- c(use_variables, n)
        }
    }
}

# add back the prediction for training
use_name <- c("classe", use_variables)

newtrain <- train[, use_name]
newtrain$classe <- factor(newtrain$classe)  # have to factor() this since it's descriptive
                                            # and taken out in the read() statement
newrealtest <- realtest[, use_variables]
```

Now, split the traning set **newtrain** into a trainng and test data sets:

```r
set.seed(123)
inTrain <- createDataPartition(y=newtrain$classe, p=0.7, list=FALSE)

newtrain_train <- newtrain[inTrain, ]
newtrain_test <- newtrain[-inTrain, ]
```

Random Forest

Generate the random forest across the entire variable set

```r
newtrain_train.rf <-train(classe~.,data=newtrain_train)
```

Random forest summary table over all variables

```r
dim(newtrain_train)         # show dimensions and number of variables
```

```
## [1] 13737    53
```

```r
newtrain_train.predict <- predict(newtrain_train.rf)
confusionMatrix(newtrain_train.predict, newtrain_train$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
```

```
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Random forest has proved an excellent classifier, and we wanted to see if it could be imporved upon by decreasing the number of necessary predictors. We cut the variable number in half, based each the variable's **gini score** creating a new training set **newtrain_half** and retrained.

```
n <- newtrain_train.rf$finalModel$importance
n.df <- as.data.frame(n)
n2<-n[order(n.df,n.df$MeanDecreaseGini,decreasing=T),,drop=F]

halfnames.rf <- c("classe", rownames(n2)[1:(length(n2)/2)])
newtrain_half <- newtrain_train[,halfnames.rf]
```

```
newtrain_half.rf <-train(classe~.,data=newtrain_half)   # train on half the variables
```

Random forest summary table on half of the variables

```
dim(newtrain_half)      # show dimensions and number of variables
```

```
## [1] 13737    27
```

```
newtrain_half.predict <- predict(newtrain_half.rf)
confusionMatrix(newtrain_half.predict, newtrain_half$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Still showing a perfect score, we cut the table in half again to a quarter of the original variables, creating a new dataset **newtrain_qtr**

```
n <- newtrain_half.rf$finalModel$importance
n.df <- as.data.frame(n)
n2<-n[order(n.df,n.df$MeanDecreaseGini,decreasing=T),,drop=F]

qtrnames.rf <- c("classe", rownames(n2)[1:(length(n2)/2)])
newtrain_qtr <- newtrain_train[,qtrnames.rf]
```

```
newtrain_qtr.rf <-train(classe~.,data=newtrain_qtr)   # train on a quarter of the variables
```

Random forest summary table on a quarter(qtr) of the variables

```
dim(newtrain_qtr)      # show dimensions and number of variables
```

```
## [1] 13737    14
```

```
newtrain_qtr.predict <- predict(newtrain_qtr.rf)
confusionMatrix(newtrain_qtr.predict, newtrain_qtr$classe)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

We **crossvalidate** the trained random forest on our testing data set

```
qtrnames <- names(newtrain_qtr)
newtrain_qtr_test <- newtrain_test[,qtrnames]
p <- predict(newtrain_qtr.rf, newdata=newtrain_qtr_test)
confusionMatrix(p, newtrain_qtr_test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    4    0    0    1
##          B    2 1131    6    0    0
##          C    0    4 1019    6    0
##          D    1    0    1  957    1
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9930   0.9932   0.9927   0.9982
## Specificity            0.9988   0.9983   0.9979   0.9994   0.9998
## Pos Pred Value         0.9970   0.9930   0.9903   0.9969   0.9991
## Neg Pred Value         0.9993   0.9983   0.9986   0.9986   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1922   0.1732   0.1626   0.1835
## Detection Prevalence   0.2848   0.1935   0.1749   0.1631   0.1837
## Balanced Accuracy      0.9985   0.9956   0.9956   0.9961   0.9990
```

Since we notice a slight degradation in **Accuracy** of 1%, we stop paring variables, and run against the real test data set. These predictions were submitted as Part 2 of this Class Project.

```r
t <- qtrnames[2:length(qtrnames)]                 # test data set does not have classe variable
realtest_qtr <- realtest[, t]                     # get the surviving variable columns
p.rf <- predict(newtrain_qtr.rf, realtest_qtr)    # predict against the model
p.rf                                              # show the 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
```

GBM Boosting

We used the same technique of evaluation using GBM Boosting. We only show the Summary Tables for brevity.

GBM Summary over all the variables:

```r
confusionMatrix(newtrain_train.predict_gbm, newtrain_train$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3866   62    0    3    2
##          B   33 2537   56    4   15
##          C    5   59 2312   56   17
##          D    0    0   22 2181   30
##          E    2    0    6    8 2461
##
## Overall Statistics
##
##                Accuracy : 0.9723
##                  95% CI : (0.9695, 0.975)
##     No Information Rate : 0.2843
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.965
##  Mcnemar's Test P-Value : 7.727e-11
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9898   0.9545   0.9649   0.9685   0.9747
## Specificity            0.9932   0.9903   0.9879   0.9955   0.9986
## Pos Pred Value         0.9830   0.9592   0.9441   0.9767   0.9935
## Neg Pred Value         0.9959   0.9891   0.9926   0.9938   0.9943
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2814   0.1847   0.1683   0.1588   0.1792
## Detection Prevalence   0.2863   0.1925   0.1783   0.1626   0.1803
## Balanced Accuracy      0.9915   0.9724   0.9764   0.9820   0.9866
```

Although training of GBM only took half the time to train than the random forest, we immediately see that the Accuracy of **GBM** is not as good as random forests. So, we stop here and see how well it will predict against the test data set.

Crossvalidate BGM on the test set

```r
p <- predict(newtrain_train.gbm, newtrain_test)     # run on the test set
```

```
## Loading required package: gbm
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
## Loading required package: plyr
```

```r
confusionMatrix(p, newtrain_test$classe)           # how good were the predictions?
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1652   42    0    1    2
##          B   18 1065   36    3   10
##          C    3   32  979   34    7
##          D    1    0    8  926   13
##          E    0    0    3    0 1050
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9638
##                  95% CI : (0.9587, 0.9684)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9542
##  Mcnemar's Test P-Value : 6.867e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9869   0.9350   0.9542   0.9606   0.9704
## Specificity            0.9893   0.9859   0.9844   0.9955   0.9994
## Pos Pred Value         0.9735   0.9408   0.9280   0.9768   0.9972
## Neg Pred Value         0.9947   0.9844   0.9903   0.9923   0.9934
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2807   0.1810   0.1664   0.1573   0.1784
## Detection Prevalence   0.2884   0.1924   0.1793   0.1611   0.1789
## Balanced Accuracy      0.9881   0.9605   0.9693   0.9781   0.9849
```

Now run on the real data

```
p <- predict(newtrain_train.gbm, realtest)         # run it on the real test set
p                                                  # show the predictions of gbm
```

```
##  [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
```

```
p.rf                                               # show predictions of random forest
```

```
##  [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
```

Conclusion

In this particular case, both models forecast the same outcomes (which we know are correct). However, the lower **Accuracy** score of **GBM** gives less confidence than random forests and because we were able to reduce the number of predictors for the random forest.

Only a quarter of the variables were necessary for random forest, with still a higher level of expected Accuracy than BGM. This allowed computing times for random forest to be less than BGM.

The reason for hihgher Accuracy in random forest would still be an area of interest for further investigation. Possibly it is due to the many more forests (500) in random forest versus the iterations (150) in BGM.