Course Project: Practical Machine Learning

Ryan R. Squires

25 April 2016

Synopsis

We use a dataset from prior academic work on human activity monitoring in order to create an algorithm that will determine the method in which a subject is performing an exercise activity.

Data Cleaning and Pre-Processing

Code to download, clean and create our datasets is shown below. We create both a training and a test set from the provided file pml-training.csv. We download the provided test-submission set for later use. We use the dplyr package to select the appropriate variables from the original data. We elected not to use timestamp variables or those that seem to identify the particular experimental subject as those would seem to limit the generality of our model. We also eliminate those variables that were mostly blank or NA.

```r
library(curl)

download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
        destfile="train.csv", method="curl")

# The test-submission file is below.

download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
        destfile="test.csv", method="curl")

ham <- read.csv("train.csv")

library(plyr)

library(dplyr)

library(caret)

ham <- select(ham,-c(1:7,12:36,50:59,69:83,87:101,103:112,125:139,141:150))

ham <- ham[complete.cases(ham),]

inTrain <- createDataPartition(y=ham$classe, p= 0.75, list=FALSE)

train_set <- ham[inTrain,]

test_set <- ham[-inTrain,]
```

Experimental Design

The five-level 'classe' (A-E) variable the training data identifies the manner in which the experimental subject performed the expercize. We would like to predict this outcome based on sensor measurements. An obvious means to do this is a machine learning algorithm, however, there are many different machine learning algorithms with varying degrees of flexibility and interpretability. In his tutorial, "Predictive Modeling with R and the caret Package (2013), Kuhn suggests that one start with a more flexible model structure to identify the upper limits of prediction and then try a less flexible, more interpretable model. If the difference in performance is acceptable and interpretability is important, one can then elect to use the less flexible, more interpretable model. Applying this approach, we first attempt to predict classe by creating a random forest of classification trees to predict the outcome. After estimating the performance of this method, we then attempt to predict classe using a single classification tree.

Basic Data Exploration

Here we take a brief look at the contents of the training set. Our training set contains 14718 observations. Each observation has 53 variables. A table of number of observations for each level of the classe variable follows.

```
table(train_set$classe)
```

```
##
##    A    B    C    D    E
## 4185 2848 2567 2412 2706
```

Random Forest Modeling

We first train a random forest to predict the classe variable within our training set. We use a 10-fold cross validation scheme, selecting the model based on cross-validated accuracy. We then estimate our out-of sample accuracy using the hold-out test set (this is not the submission test set).

```
fitControl <- trainControl(method = "cv", number = 10)

set.seed(1802)

rf_fit <- train(classe~., data=train_set, trControl=fitControl,
                    method="rf",verbose=FALSE)

# In sample
# 1

confusionMatrix(predict(rf_fit),train_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## 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   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```r
# out of sample
# 0.9932708

pred_classe <- predict(rf_fit, newdata=test_set)
```

Classification Tree Modeling

Having obtained excellent results using the random forest algorithm, we would like to see if we can obtain acceptable results from a more interpretable model. The base classifier in the random forest is a classification tree. We now grow a single tree and evaluate its performance as above.

```r
set.seed(1802)

rpart_fit <- train(classe~., data=train_set, trControl=fitControl,
                   method="rpart")

# In sample

confusionMatrix(predict(rpart_fit),train_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3817 1239 1208 1069  381
##          B   68  723   44  399  161
##          C  287  886 1315  944  948
##          D    0    0    0    0    0
##          E   13    0    0    0 1216
##
## Overall Statistics
##
##                Accuracy : 0.4804
##                  95% CI : (0.4723, 0.4885)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3212
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9121  0.25386  0.51227   0.0000  0.44937
## Specificity            0.6300  0.94339  0.74776   1.0000  0.99892
## Pos Pred Value         0.4948  0.51828  0.30023      NaN  0.98942
## Neg Pred Value         0.9475  0.84050  0.87889   0.8361  0.88954
## Prevalence             0.2843  0.19350  0.17441   0.1639  0.18386
## Detection Rate         0.2593  0.04912  0.08935   0.0000  0.08262
## Detection Prevalence   0.5241  0.09478  0.29759   0.0000  0.08350
## Balanced Accuracy      0.7710  0.59862  0.63001   0.5000  0.72414
```

```r
# out of sample

pred_classe <- predict(rpart_fit, newdata=test_set)

confusionMatrix(pred_classe, test_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1277  399  407  391  145
##          B   12  236    9  143   49
##          C  105  314  439  270  292
##          D    0    0    0    0    0
##          E    1    0    0    0  415
##
## Overall Statistics
##
##                Accuracy : 0.4827
##                  95% CI : (0.4686, 0.4968)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3231
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9154  0.24868  0.51345   0.0000  0.46060
## Specificity            0.6176  0.94614  0.75772   1.0000  0.99975
## Pos Pred Value         0.4876  0.52561  0.30915      NaN  0.99760
## Neg Pred Value         0.9484  0.83996  0.88060   0.8361  0.89171
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2604  0.04812  0.08952   0.0000  0.08462
## Detection Prevalence   0.5341  0.09156  0.28956   0.0000  0.08483
## Balanced Accuracy      0.7665  0.59741  0.63558   0.5000  0.73017
```

We can visualize the classification tree by plotting.

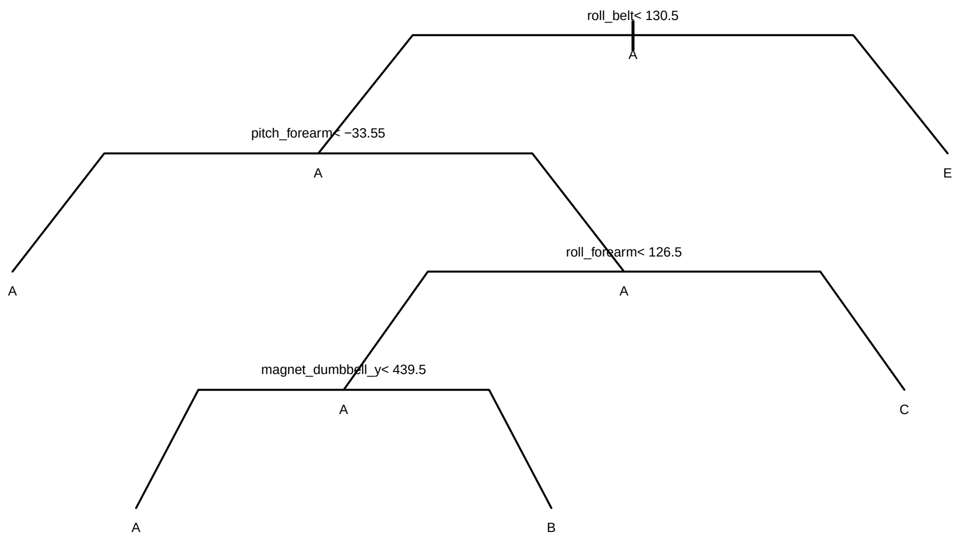

Figure: Classification Tree

Our classification tree is outperformed by our random forest by a large margin. Although the single tree is probably better than random guess. Interstingly, no leaves of our tree give the result ‘D’.

Submission Test Set

The submission test set demands predictive accuracy only, so it only makes sense to use our random forest model to predict the classe variable for each of these instances.

```
ham_test <- read.csv("test.csv")

ham_test <- select(ham_test,-c(1:7,12:36,50:59,69:83,87:101,103:112,125:139,141:150))

ham_test <- ham_test[complete.cases(ham_test),]

predict(rf_fit, newdata=ham_test)
```

```
##  [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
```

Conclusions

The power of aggregated classifiers is quite clear in this experiment. The accuracy achieved by the random forest is remarkable. The far lower predictive accuracy achieved by the classification tree demonstrates the trade-off for interpretability. We initially used boosting for our flexible model, but found random forest to be more accurate. While it would be possible to consider other flexible models such as boosting or support vector machines, the results achieved using the random forest would make this an entirely academic exercise.

The utility of the caret package is also evident. It was very easy to interact with different R packages through the uniform caret interface.