Exercise Manner Prediction Model

Kurt Eichinger

6/29/2018

Summary

The goal of this analysis will be to develop a predictive model for exercise manner. We will generate a tool to quantify how well people exercise, based on data obtained from accelerometers on the belt, arm, forearm, and dumbell of six participants who performed exercises in correct and incorrect ways. The data contain labels identifying the quality of the exercise (under the "classe" variable), and we will use this to predict the outcomes of activities in our test data. We will critique this model from its results when using it to evaluate 20 different test cases.

The data for this project come from this source: link.

Getting and Cleaning the Data

Let us load all of the packages we might need for this analysis as we download, clean, and partition our data (so that we can estimate the out of sample error). We will also set the seed to guarantee reproducibility.

```
set.seed(1029384756)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
library(knitr)
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.4.4
```

```
library(RColorBrewer)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(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 2.1.3
```

```
trainURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(trainURL), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testURL), na.strings=c("NA","#DIV/0!",""))
inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
trainingfortraining <- training[inTrain, ]
trainingfortesting <- training[-inTrain, ]
dim(trainingfortraining); dim(trainingfortesting); dim(testing)
```

```
## [1] 13737   160
```

```
## [1] 5885  160
```

```
## [1]  20 160
```

There is a great amount of extraneous information and missing data in this set. We must separate the wheat from the chaff. We will eliminate the variables that are unnecessary for the predictive model, mostly NA, or have nearly zero variance. These modifications are performed on all data subsets.

```
# We eliminate the variables that have nearly zero variance.
nearOvar <- nearZeroVar(trainingfortraining)
trainingfortraining <- trainingfortraining[, -nearOvar]
trainingfortesting <- trainingfortesting[, -nearOvar]

# We eliminate the first seven variables because they are not involved.
trainingfortraining <- trainingfortraining[, -(1:7)]
trainingfortesting <- trainingfortesting[, -(1:7)]

# We eliminate the variables that are mostly NA (> 95%).
eliminateNAs <- sapply(trainingfortraining, function(x) mean(is.na(x))) > 0.95
trainingfortraining <- trainingfortraining[, eliminateNAs == FALSE]
trainingfortesting <- trainingfortesting[, eliminateNAs == FALSE]
dim(trainingfortraining); dim(trainingfortesting)
```

```
## [1] 13737    52
```

```
## [1] 5885   52
```

```
# Let's remove the "problem_id" column from the testing data and format the set.
colremoval <- colnames(trainingfortraining[, -52])
testing <- testing[colremoval]
dim(testing)
```

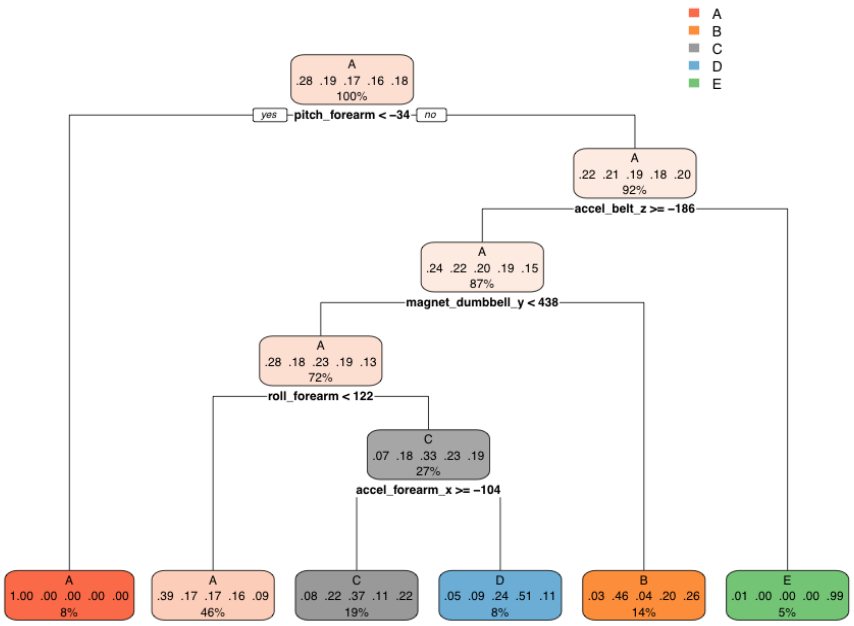
```
## [1] 20 51
```

Training the Model

We will now test the classification tree, random forest, and gradient boosting modeling methods. By utilizing cross validation, we seek to limit overfitting and improve efficiency.

Classification Tree

```
set.seed(48003)
trControl <- trainControl(method="cv", number=5)
classificationtreemodel <- train(classe~., data=trainingfortraining, method="rpart", trControl=trControl
rpart.plot(classificationtreemodel$finalModel)
```




```
prediction_model_1 <- predict(classificationtreemodel, newdata = trainingfortesting)
confusion_matrix_1 <- confusionMatrix(trainingfortesting$classe, prediction_model_1)
confusion_matrix_1$table;confusion_matrix_1$overall[1]
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1504   26  108   29    7
##          B  464  394  241   40    0
##          C  489   33  403  101    0
##          D  411  174  122  257    0
##          E  256  203  264   47  312
```

```
##  Accuracy
## 0.4876805
```

At under 49%, the accuracy achieved by classification trees is very low (expected out of sample error of 51%). Let's try another method.

Random Forest

```
randomforestmodel <- train(classe ~ ., data = trainingfortraining, method = "rf", trControl = trControl
print(randomforestmodel)
```

```
## Random Forest
##
## 13737 samples
##    51 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10989, 10989, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9913375  0.9890414
##   26    0.9917741  0.9895939
##   51    0.9883523  0.9852640
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 26.
```

The greatest accuracy is achieved with 26 predictors.

```
prediction_model_2 <- predict(randomforestmodel, newdata = trainingfortesting)
confusion_matrix_2 <- confusionMatrix(trainingfortesting$classe, prediction_model_2)
confusion_matrix_2$table;confusion_matrix_2$overall[1]
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    4    0    0    1
##          B    9 1128    2    0    0
##          C    0    7 1017    2    0
##          D    0    0    8  956    0
##          E    0    0    3    4 1075
```

```
##  Accuracy
## 0.9932031
```

Using five folds for our cross validation did not result in much accuracy under the classification tree method, but it helped achieve great accuracy for our random forest model. We did, however, have to sacrifice time, but getting 99% accuracy was worth the wait (expected out of sample error of 0.68%). Let's see if gradient boosting can keep up.

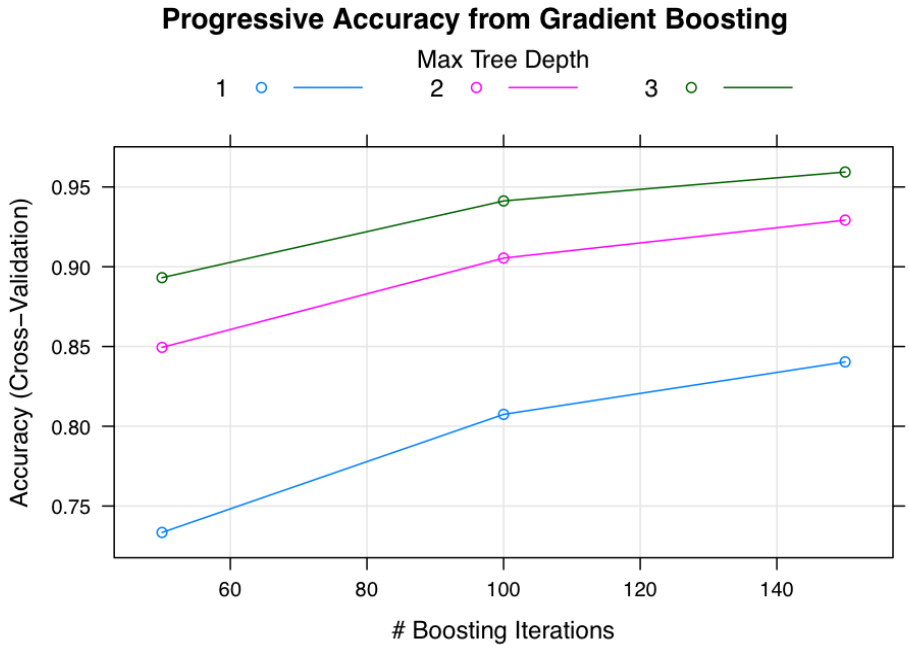
Gradient Boosting

```
gradientboostingmodel <- train(classe ~ ., data = trainingfortraining, method = "gbm", trControl = trCo
print(gradientboostingmodel)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    51 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10991, 10990, 10989, 10990, 10988
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7334194  0.6619464
##   1                  100      0.8074534  0.7562767
##   1                  150      0.8403575  0.7979034
##   2                   50      0.8494573  0.8093067
##   2                  100      0.9054378  0.8803153
##   2                  150      0.9292420  0.9104648
##   3                   50      0.8931357  0.8646879
##   3                  100      0.9412536  0.9256585
##   3                  150      0.9593807  0.9486071
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
plot(gradientboostingmodel, main = "Progressive Accuracy from Gradient Boosting")
```



An increase in interaction depth and the number of boosting iterations resulted in increased accuracy, but the maximum tree depth never got unwieldy to deliver satisfactory results.

```
prediction_model_3 <- predict(gradientboostingmodel, newdata = trainingfortesting)
confusion_matrix_3 <- confusionMatrix(trainingfortesting$classe, prediction_model_3)
confusion_matrix_3$table;confusion_matrix_3$overall[1]
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1650   17    6    1    0
##          B   28 1080   27    1    3
##          C    0   27  984   10    5
##          D    2    1   34  918    9
##          E    3    4   10   11 1054
```

```
##  Accuracy
## 0.9661852
```

We once again featured 5-fold cross validation. This model took longer than the random forest model to generate, and the results are slightly less impressive (expected out of sample error of 3.38%).

Conclusion

The greatest degree of accuracy was obtained with the random forest model. Now, we will employ it in our evaluation of the test data.

```
test_model_2 <- predict(randomforestmodel, newdata = testing)
print(test_model_2)
```

```
##  [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
```