Movement Classification with Wearable Devices

Jason Hooker

2/20/2020

Wearable motion tracking devices generate huge amounts of data to enable health-conscious people (or tech enthusiasts) to monitor their activity, thus allowing them to find patterns in their behavior that may help them improve their health. People often quantify how much of an activity they do, but rarer is quantifying the quality of that activity - the latter is the goal of this project. The data was collected from six participants who wore accelerometers on their belts, forearms, upper arms, and dumbbells and performed lifts correctly and in five different incorrect ways. The random forest model created is used to predict the manner in which participants did the exercise.

Reading in Data

Load the necessary libraries.

```
library(caret)
library(parallel)
library(doParallel)
library(dplyr)
library(randomForest)
```

Read in the data for the training and testing sets, and split the raw training data into training and validation sets.

```
set.seed(12345)

allTraining <- read.csv("../data/pml-training.csv", na.strings = c("NA", "", "#DIV/0!"))

inTrain <- createDataPartition(y=allTraining$classe, p = 0.7, list = FALSE)

training <- allTraining[inTrain,]
validation <- allTraining[-inTrain,]

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

Feature Selection

Begin feature selection by examining columns containing NAs. Each of those columns have NAs in all or nearly all rows, with only a fraction of the rows in the entire dataset containing values for these variables.

```
# find number of NAs in each column
sumNAs <- colSums(is.na(training))

# pull out the names of each column with at least one NA
NAnames <- names(which(sumNAs > 0))

# subset the training dataset to just include columns with at least one NA
NAcols <- training[NAnames]

# subset the training dataset to just include rows that have data in the columns containing NAs
```

```
NAdata <- filter(training, rowSums(!is.na(NAcols)) != 0)
dim(NAdata)
```

```
## [1] 283 160
```

Of the observations containing data for the aforementioned variables, the split between the five classes is relatively even, with there being somewhat more for classe 'A'.

```
##
##  A  B  C  D  E
## 74 51 53 50 55
```

Because of this, we exclude all of these variables from the datasets to be used in our model. The first seven variables, which contain descriptive information for each observation and are of no predictive value, are excluded as well. These exclusions are performed on the training, testing, and validation sets.

```
# subset the all datasets to exclude columns with at least one NA
subTraining <- training %>% select(which(!(names(training) %in% NAnames))) %>% select(-(1:7))
subValidation <- validation %>% select(which(!(names(validation) %in% NAnames))) %>% select(-(1:7))
subTesting <- testing %>% select(which(!(names(testing) %in% NAnames))) %>% select(-(1:7))
```

Now we check our smaller training dataset for any variables that may have near zero variance. We find that all have relatively significant variance, so we proceed with the datasets we have.

```
nsv <- nearZeroVar(subTraining, saveMetrics = TRUE)
sum(nsv$zeroVar)
```

```
## [1] 0
```

Model Fitting

We use random forests to train our prediction model, one without preprocessing and one preprocessing the data with Principal Component Analysis. We use cross validation with five resamples in order to get a better estimate of the accuracy of the models.

```
x <- subTraining[,-53]
y <- subTraining[,53]

# use cross validation
fitControl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)

#enable parallel processing
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)


modelFit <- train(x, y, method = "rf", data = subTraining, trControl = fitControl)
modelFitPCA <- train(x, y, method = "rf", data = subTraining, trControl = fitControl,
                     preProcess = "pca")

# disable parallel processing
stopCluster(cluster)
registerDoSEQ()
```

The model without preprocessing proves to be more accurate than the PCA model, so we will use this model for prediction.

```
modelFit$results
```

```
##   mtry  Accuracy     Kappa  AccuracySD     KappaSD
## 1    2 0.9911914 0.9888567 0.001697190 0.002147317
## 2   27 0.9902453 0.9876601 0.001104697 0.001397441
## 3   52 0.9846405 0.9805695 0.002443872 0.003093117
```

```
modelFitPCA$results
```

```
##   mtry  Accuracy     Kappa  AccuracySD     KappaSD
## 1    2 0.9694252 0.9613189 0.003924106 0.004973212
## 2   27 0.9549388 0.9429953 0.005812292 0.007354284
## 3   52 0.9566129 0.9451079 0.007983004 0.010106286
```

Model Accuracy on Validation Set and Out of Sample Error Estimation

We now use the model to predict classe values on the validation test set and find that the accuracy remains above 99%.

```
pred <- predict(modelFit, subValidation)

confusionMatrix(pred, subValidation$classe)
```

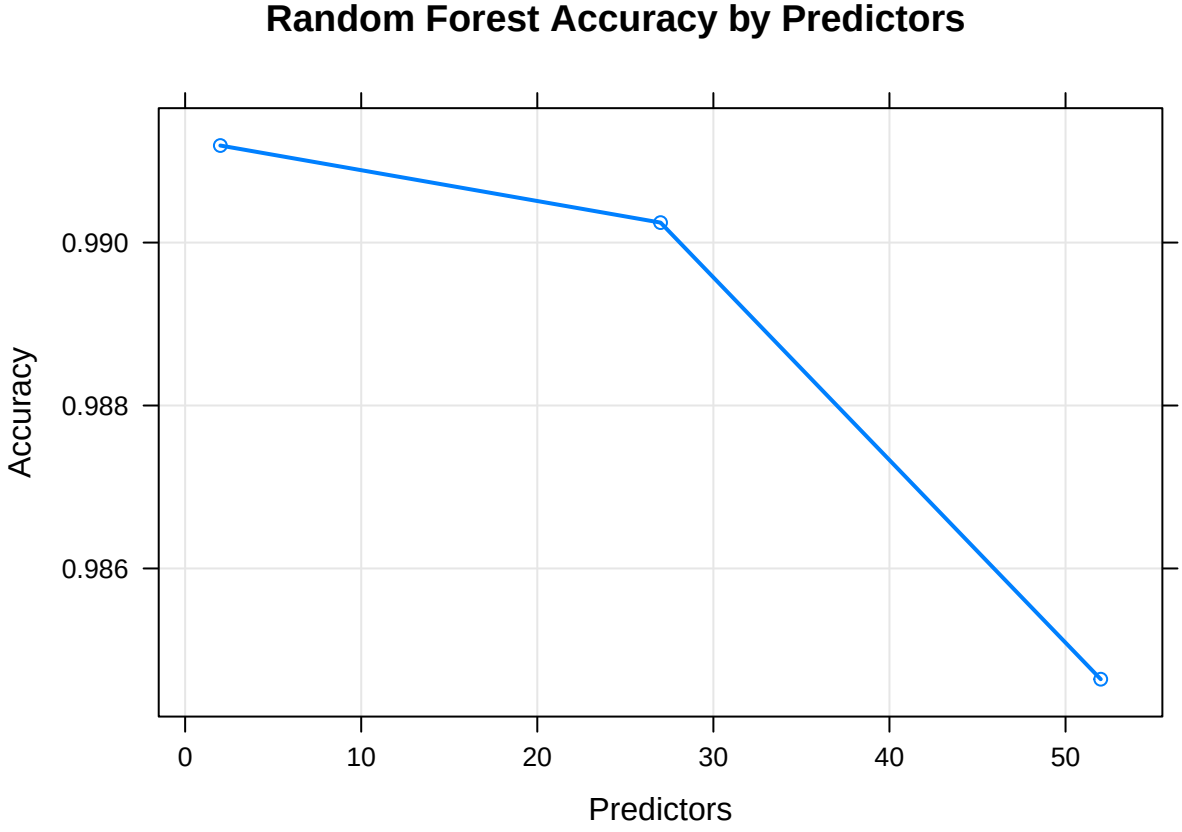
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    1 1135    8    0    0
##          C    0    0 1017   22    0
##          D    0    0    1  940    1
##          E    0    0    0    2 1081
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.991, 0.9953)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9965   0.9912   0.9751   0.9991
## Specificity            0.9991   0.9981   0.9955   0.9996   0.9996
## Pos Pred Value         0.9976   0.9921   0.9788   0.9979   0.9982
## Neg Pred Value         0.9998   0.9992   0.9981   0.9951   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1929   0.1728   0.1597   0.1837
## Detection Prevalence   0.2850   0.1944   0.1766   0.1601   0.1840
## Balanced Accuracy      0.9992   0.9973   0.9934   0.9873   0.9993
```

```
v <- postResample(pred, subValidation$classe)

OoSErrorPercent <- (1 - v[[1]]) * 100
```

This accuracy is used to estimate the out of sample error of the model, which comes out to be 0.66%.

The plot below shows that accuracy of the model peaks when 2 predictors are used, though the difference in accuracy is only marginal.



Prediction on the Testing Dataset

Finally, we use the model to predict values for classe in the testing dataset. These predictions passed the course final quiz with 100% (20/20) accuracy.

```
predTest <- predict(modelFit, subTesting)
predTest
```

```
##  [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
```