Weight Lifting Exercise Analysis

Eric Scuccimarra

13 January 2018

Executive Summary

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. Data was gathered from accelerometers on the belt, forearm, arm and dumbell of 6 participants while they performed barbell lifts both correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The goal of this analysis is to create a prediction algorithm to determine whether the participants peformed the exercise correctly or not, based on the data from the accelerometers.

Loading and Preprocessing Data

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", dest="./data/pml-t
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", dest="./data/pml-te
training <- read.csv("./data/pml-training.csv")
testing <- read.csv("./data/pml-testing.csv")
```

The data has a large number of features, so I will begin by removing features which may not be useful. In addition to columns which do not contain acceleromter data, I will remove features with near zero variance and those which contain a large proportion of missing values, setting the threshold at 95%.

```
training$user_name <- NULL
training$X <- NULL
training$raw_timestamp_part_1 <- NULL
training$raw_timestamp_part_2 <- NULL
training$cvtd_timestamp <- NULL
training$num_window <- NULL
nearzerovar <- nearZeroVar(training, saveMetrics = T)
training <- training[, !nearzerovar$nzv]
colswithNAs <- colSums(is.na(training)) > (0.95 * nrow(training))
training <- training[, !colswithNAs]
```

This leaves a more manageable 52 features to use for our model. Next I will preprocess the data by scaling and centering.

```
preprocessObj <- preProcess(training, method=c("center", "scale"))
training <- predict(preprocessObj, training)
testing <- predict(preprocessObj, testing)
```

And finally I will split the training data into a training set and a cross-validation set:

```
inTrain <- createDataPartition(y=training$classe,p=0.7, list=FALSE)
crossv <- training[-inTrain,]
training <- training[inTrain,]
```

Exploratory Data Analysis

Now I will look at a correlation matrix between classe and the remaining features. The matrix is included in the appendix in Figure 1.

```
apply(training, 2, function(col) cor(as.numeric(col), as.numeric(training$classe), method="spearman"))
```

```
## Warning in is.data.frame(x): NAs introduced by coercion
```

The facts that this is a classification problem and that none of the features has a high correlation to classe make linear models likely to be a poor choice for modeling the problem. I will instead try to use random forests and boosting models.

Prediction Models

Random Forest Model

To begin I will train a random forest model using 5-fold cross validation:

```
rfModel <- train(classe ~ ., method="rf", data=training, trControl = trainControl(method = "cv", number
```

Now we will look at a summary of the model:

```
rfModel
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10990, 10991, 10988
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9896622  0.9869229
##   27    0.9909729  0.9885804
##   52    0.9864598  0.9828707
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

The random forest model has a very high accuracy rating of 99%. A confusion matrix for the actual values of classe vs the predictions using our random forest model are in Figure 2 of the Appendix. The accuracy is reported at 100%, with a very tight confidence interval of .03%.

Boosting Model

Next we will try a boosting model using 5-fold cross validation:

```
boostModel <- train(classe ~ ., method="gbm", verbose=FALSE, data=training, trControl = trainControl(me
```

Now we will look at the results of the boosting model:

```r
boostModel
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10990, 10990, 10988, 10991
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7535141  0.6875101
##   1                  100      0.8203399  0.7726049
##   1                  150      0.8545545  0.8159358
##   2                   50      0.8576862  0.8196702
##   2                  100      0.9055844  0.8805440
##   2                  150      0.9313544  0.9131506
##   3                   50      0.8929906  0.8645383
##   3                  100      0.9396530  0.9236568
##   3                  150      0.9599635  0.9493547
##
## 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.
```

The confusion matrix for the predictions of the boost model on the training data versus the actual label of the training data is included in Figure 3 of the Appendix. The accuracy is reported at 97%, with a confidence interval of less than 1%.

Model Selection

Based on the performance on the training data, the random forest model looks like a better predictor. However, to confirm this we will compare the performance of each model on the cross-validation data set.

Let's start by using our two competing models to predict the classe of the cross validation set and look at the confusion matrices. I will start with the random forest confusion matrix.

```r
rfPredictions <- predict(rfModel, crossv)
boostPredictions <- predict(boostModel, crossv)
confusionMatrix(rfPredictions, crossv$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    7    0    0    0
##          B    1 1131    1    0    1
##          C    0    1 1021    8    0
##          D    0    0    4  956    2
```

```
##          E    1    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9930   0.9951   0.9917   0.9972
## Specificity            0.9983   0.9994   0.9981   0.9988   0.9998
## Pos Pred Value         0.9958   0.9974   0.9913   0.9938   0.9991
## Neg Pred Value         0.9995   0.9983   0.9990   0.9984   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1922   0.1735   0.1624   0.1833
## Detection Prevalence   0.2853   0.1927   0.1750   0.1635   0.1835
## Balanced Accuracy      0.9986   0.9962   0.9966   0.9952   0.9985
```

The random forest model has an accuracy of 99.24% on the cross validation data set.

Now we look at the confusion matrix for the boosting model:

```r
confusionMatrix(boostPredictions, crossv$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1648   43    0    0    4
##          B   18 1068   36    2   12
##          C    5   28  975   30    6
##          D    2    0   11  931   14
##          E    1    0    4    1 1046
##
## Overall Statistics
##
##                Accuracy : 0.9631
##                  95% CI : (0.958, 0.9678)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9533
##  Mcnemar's Test P-Value : 3.886e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9845   0.9377   0.9503   0.9658   0.9667
## Specificity            0.9888   0.9857   0.9858   0.9945   0.9988
## Pos Pred Value         0.9723   0.9401   0.9339   0.9718   0.9943
```

```
## Neg Pred Value         0.9938   0.9850   0.9895   0.9933   0.9926
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2800   0.1815   0.1657   0.1582   0.1777
## Detection Prevalence   0.2880   0.1930   0.1774   0.1628   0.1788
## Balanced Accuracy      0.9867   0.9617   0.9680   0.9801   0.9827
```

The random forest model performs better on the cross validation data set than the boosting model, so we will select that as the final model.

To interpret this model we will extract the importance of the features from the model, order it and look at the top ten results. This will be the 10 features most important to the prediction.

```
importance <- varImp(rfModel)$importance
importance$val <- importance$Overall
importance <- importance[order(importance$Overall, decreasing = T), ]
importance$val <- NULL
head(importance, 10)
```

```
##                     Overall
## roll_belt         100.00000
## pitch_forearm      60.98137
## yaw_belt           56.61885
## magnet_dumbbell_z  44.27013
## pitch_belt         43.90685
## magnet_dumbbell_y  41.55956
## roll_forearm       37.46332
## accel_dumbbell_y   20.47021
## roll_dumbbell      17.67370
## magnet_dumbbell_x  16.82547
```

Finally I will look at the final model from the random forest algorithm:

```
rfModel$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.68%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    5    0    0    0 0.001280082
## B   19 2630    8    1    0 0.010534236
## C    0   11 2379    6    0 0.007095159
## D    0    1   31 2218    2 0.015097691
## E    0    0    4    6 2515 0.003960396
```

The model results in 500 trees with 2 variables at each split. The estimated error rate is 0.71% which is quite close to the actual error rate on the cross validation data set.

Predictions

Lastly we will apply the final model to the testing data set. The data has already been preprocessed so we simply predict with our model:

```r
testPredictions <- predict(rfModel, testing)
testPredictions
```

```
##  [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
```

Appendix

Figure 1 - Correlation Matrix of features to classe

```r
apply(training, 2, function(col) cor(as.numeric(col), as.numeric(training$classe), method="spearman"))
```

```
## Warning in is.data.frame(x): NAs introduced by coercion
```

```
##            roll_belt           pitch_belt             yaw_belt 
##          0.132993554         -0.036467484          0.072605526 
##     total_accel_belt         gyros_belt_x         gyros_belt_y 
##          0.091902483          0.006970923         -0.001963170 
##         gyros_belt_z         accel_belt_x         accel_belt_y 
##         -0.005657527          0.031291886         -0.009453339 
##         accel_belt_z        magnet_belt_x        magnet_belt_y 
##         -0.143612844         -0.011958295         -0.198107845 
##        magnet_belt_z             roll_arm            pitch_arm 
##         -0.136124734          0.055708237         -0.179393040 
##              yaw_arm      total_accel_arm          gyros_arm_x 
##          0.029662067         -0.158430988          0.031895521 
##          gyros_arm_y          gyros_arm_z          accel_arm_x 
##         -0.041736497          0.017135728          0.259236911 
##          accel_arm_y          accel_arm_z         magnet_arm_x 
##         -0.085998729          0.108097167          0.283207827 
##         magnet_arm_y         magnet_arm_z        roll_dumbbell 
##         -0.267980962         -0.153072959          0.080433031 
##       pitch_dumbbell         yaw_dumbbell total_accel_dumbbell 
##          0.092679529          0.008111106         -0.013764362 
##     gyros_dumbbell_x     gyros_dumbbell_y     gyros_dumbbell_z 
##         -0.010813966          0.016767907          0.011553530 
##     accel_dumbbell_x     accel_dumbbell_y     accel_dumbbell_z 
##          0.123191401         -0.021520435          0.080549570 
##    magnet_dumbbell_x    magnet_dumbbell_y    magnet_dumbbell_z 
##          0.144608054          0.045053047          0.189655860 
##         roll_forearm        pitch_forearm          yaw_forearm 
##          0.044564211          0.321396387         -0.057222760 
##  total_accel_forearm      gyros_forearm_x      gyros_forearm_y 
##          0.114671640         -0.017014960          0.007948446 
##      gyros_forearm_z      accel_forearm_x      accel_forearm_y 
##         -0.003629098         -0.204196705          0.013249508 
##      accel_forearm_z     magnet_forearm_x     magnet_forearm_y 
##         -0.005565168         -0.192623429         -0.118291836 
##     magnet_forearm_z               classe 
##         -0.049319292                   NA
```

Figure 2 - Confusion Matrix for random forest model on training data

```r
confusionMatrix(predict(rfModel, training), training$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
```

Figure 3 - Confusion Matrix for boost model on training data

```r
confusionMatrix( predict(boostModel, training) , training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3864   59    0    0    2
##          B   31 2544   51    3   11
##          C    9   54 2322   62   13
##          D    1    1   21 2178   30
##          E    1    0    2    9 2469
##
## Overall Statistics
##
##                Accuracy : 0.9738
```

```
##                  95% CI : (0.971, 0.9764)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9668
##  Mcnemar's Test P-Value : 3.165e-11
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9892   0.9571   0.9691   0.9671   0.9778
## Specificity            0.9938   0.9913   0.9878   0.9954   0.9989
## Pos Pred Value         0.9845   0.9636   0.9439   0.9762   0.9952
## Neg Pred Value         0.9957   0.9897   0.9934   0.9936   0.9950
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2813   0.1852   0.1690   0.1585   0.1797
## Detection Prevalence   0.2857   0.1922   0.1791   0.1624   0.1806
## Balanced Accuracy      0.9915   0.9742   0.9785   0.9813   0.9884
```