Barbell Analysis

Santosh

March 21, 2015

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 collected from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Our goal is to predict the manner in which the exercise was actually performed.

Pre-processing

Training and testing data are read from disk. After taking a summary of the training data, many columns were removed. For sake of brevity, the summary output isn't shown here. Most of the columns were removed because the vast majority of values were either NA or blank. Also, several columns were removed because they don't appear to be relevant, because they're either a record ID, user name, or timestamp.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2
```

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.2
```

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

```
training = read.csv("pml-training.csv")
testing = read.csv("pml-testing.csv")

summary(training)

useful_columns = c("num_window", "roll_belt", "pitch_belt", "yaw_belt",
  "total_accel_belt", "gyros_belt_x", "gyros_belt_y", "gyros_belt_z",
  "accel_belt_x", "accel_belt_y", "accel_belt_z", "magnet_belt_x", "magnet_belt_y",
  "magnet_belt_z",
  "roll_arm", "pitch_arm", "yaw_arm", "total_accel_arm", "gyros_arm_x",
  "gyros_arm_y", "gyros_arm_z",
  "accel_arm_x", "accel_arm_y", "accel_arm_z", "magnet_arm_x", "magnet_arm_y",
  "magnet_arm_z", "roll_dumbbell", "pitch_dumbbell", "yaw_dumbbell",
  "total_accel_dumbbell", "gyros_dumbbell_x", "gyros_dumbbell_y", "gyros_dumbbell_z",
  "accel_dumbbell_x", "accel_dumbbell_y", "accel_dumbbell_z",
  "magnet_dumbbell_x", "magnet_dumbbell_y", "magnet_dumbbell_z", "roll_forearm",
```

```
  "pitch_forearm", "yaw_forearm", "total_accel_forearm", "gyros_forearm_x",
  "gyros_forearm_y", "gyros_forearm_z",
  "accel_forearm_x", "accel_forearm_y", "accel_forearm_z", "magnet_forearm_x",
  "magnet_forearm_y", "magnet_forearm_z")
# in addition to the columns above, the training data has a class label
training = training[, c(useful_columns, "classe")]
# in addition to the columns above, the testing data has a problem id
testing = testing[, c(useful_columns, "problem_id")]
```

Cross-validation

80% of training data will be used for cross-validation. The remaining 20% of data will be used as a holdout/validation set.

```
# set seed for reproducibility
set.seed(11611)

# use subset of training set for training, and the rest as the holdout/validation set
train_cv_indices = createDataPartition(y = training$classe, p = 0.8, list = FALSE)
training_cv = training[train_cv_indices, ]
validation_set = training[-train_cv_indices, ]
```

Random forests will be used for training. This method creates multiple trees and uses voting. The below code performs cross validation and outputs the cross-validation error.

```
# Use the randomForest package because caret is very slow when creating random forests
# Because caret is not being used, cross-validation functionality will be recreated below.

# Function to compute number of incorrectly classified examples on a single fold
# during cross-validation
getNumWrong = function(modelFit) {
  correct = sum(diag(modelFit$test$confusion))
  total = sum(modelFit$test$confusion[, 1:5])
  total-correct
}

# create folds for cross-validation
folds = createFolds(y = training_cv$classe, k = 10, list = FALSE)
num_wrong = 0
for (fold_number in 1:10) {
  training_folds = training_cv[folds != fold_number, ]
  testing_fold = training_cv[folds == fold_number, ]
  # 50 trees are used for the random forest
  modelFit = randomForest(training_folds[, -54], training_folds$classe, ntree=50,
                          xtest = testing_fold[, -54], ytest = testing_fold$classe)
  fold_num_wrong = getNumWrong(modelFit)
  # weight the error by the fold size (the fold sizes are almost equal to each other)
  num_wrong = num_wrong + fold_num_wrong
}

# Divide the number wrong by the total examples used for cross validation, since every example
# would have been in one of the folds used for testing the model built on the remaining folds
num_wrong/length(training_cv$classe)
```

```
## [1] 0.003184916
```

The cross-validation error is 0.32%. Because this is a very low number, this model and parameters are sufficient. To estimate the out of sample error, create the final model using the entire cross-validation data set, and run it on the validation/holdout set.

Out-of-sample error estimate

```
finalModelFit = randomForest(training_cv[, -54], training_cv$classe, ntree=50,
                             xtest = validation_set[, -54], ytest = validation_set$classe,
                             keep.forest=TRUE)
finalModelFit
```

```
##
## Call:
##  randomForest(x = training_cv[, -54], y = training_cv$classe,      xtest = validation_set[, -54], yt
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.42%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4460    3    1    0    0 0.0008960573
## B    9 3020    9    0    0 0.0059249506
## C    0    9 2725    4    0 0.0047479912
## D    0    0   22 2550    1 0.0089389817
## E    0    0    1    7 2878 0.0027720028
##                 Test set error rate: 0.18%
## Confusion matrix:
##      A   B   C   D   E class.error
## A 1116   0   0   0   0 0.000000000
## B    0 759   0   0   0 0.000000000
## C    0   0 684   0   0 0.000000000
## D    0   0   6 637   0 0.009331260
## E    0   0   0   1 720 0.001386963
```

The estimated out of sample error is 0.18%.

Prediction on test set

The final model will be used to make predictions on the test set. The output of the prediction is not shown.

```
predict(finalModelFit, testing[, -54])
```

Conclusion

Using accelerometer data, predictions were made to determine how barbell exercises were performed. The machine learning technique used is random forests. 80% of training data was used for cross-validation. The

remaining 20% was used as a validation/holdout set. The cross-validation error was 0.32%. Then, a model was created using all data used for cross-validation. This model was run on the validation/holdout set, resulting in an out-of-sample error estimate of 0.18%. Finally, this model was run on a test set.