Practical Machine Learning: Course Project

Johns Hopkins University / Coursera

April 2015

Summary

We use the UCI Human Activity Recognition dabase of Weight Lifting Exercises to build predictor for classifying correctness of performed exercises. **The predictor is trained via a random forest methodology** using data about the exercise performance itself, and no data about the subject or time window, in hopes of creating a model which can be generalized beyond the subjects in the database.

The model achieved **99.3% cross validation accuracy (collected in building the forest), or an expected error rate of 0.7%. Out of sample testing achieved accuracy of 99.3% (0.7% error rate)**.

Model Construction

We first read the data, and exclude any variables that 1) have measurements for less than half the samples, or 2) are solely descriptive of the data collection (including subject name, timestamps, or timing windows). What remains are measurements taken from the sensors.

```
options (warn = -1)

library (caret)
library (randomForest)

dataTrainingRaw <- read.csv ("pml-training.csv", na.strings = "NA")
fKeep <- sapply (1:ncol (dataTrainingRaw), function (i) { if ( 0.5 < length (which (is.na (dataTrainingR
dataTraining <- dataTrainingRaw [ , fKeep]
```

We then partition the data into a training set (80%) and testing set (20%) and use the training set to create a random forest (up to 50 trees). The forest is resampled 5 times. The most important variables are examined.

```
set.seed (314159)
indexTraining <- createDataPartition (dataTraining$classe, p = 0.80, list = FALSE)
trainingSet <- dataTraining [indexTraining, ]
testingSet <- dataTraining [-indexTraining, ]

trainOptions <- trainControl(method = "oob", 5)
model <- train (classe ~ ., method = "rf", data = trainingSet, ntree=50, trControl = trainOptions)

variableImportance <- varImp (model)$importance
varOrder <- order (-variableImportance [ , 1])
head (cbind (variable = names (trainingSet) [varOrder], importance = variableImportance [varOrder, 1]),
```

```
##       variable            importance
##  [1,] "roll_belt"         "100"
##  [2,] "pitch_forearm"     "65.0059204832701"
##  [3,] "yaw_belt"          "55.1186973108158"
```

```
##  [4,] "magnet_dumbbell_y" "46.3946469160535"
##  [5,] "pitch_belt"        "44.4614752151486"
##  [6,] "magnet_dumbbell_z" "42.4943587375163"
##  [7,] "roll_forearm"      "41.7200727870812"
##  [8,] "accel_dumbbell_y"  "22.4244488703446"
##  [9,] "magnet_belt_z"     "18.0526292659214"
## [10,] "accel_belt_z"      "17.9523667895972"
```

Here we see that key data involve belt movement (roll_belt, yaw_belt, pitch_belt), forearm position (pitch_forearm, roll_forearm), and weight movement (magnet_dumbbell_y, magnet_dumbbell_z). Aside from these variables, importance drops quickly.

Predictive Power

We estimate out of sample performance in two ways. First, we look at the cross validation accuracy and error as collected during creation of the random forest (in this case, the data was resampled 5 times). Second, we use the model to predict the testing set held back from training, and examine its performance.

```
model$results
```

```
##    Accuracy     Kappa mtry
## 1 0.9892987 0.9864607    2
## 2 0.9933754 0.9916199   27
## 3 0.9856679 0.9818678   52
```

```
predictRF <- predict (model, testingSet)
testResults <- confusionMatrix (predictRF, testingSet$classe)
testResults$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9933724      0.9916145      0.9903040      0.9956662      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Here we see the model achieved 99.3% cross validation accuracy, or an expected error rate of 0.7%.

Out of sample testing achieved accuracy of 99.3%, or 0.7% error rate with 95% confidence interval of 1-0.4%.

Conclusions

A random forest proved to be an effective method for training a predictor of exercise quality, with near-perfect accuracy. However, it should be noted that all data (training and testing) has come from only 6 subjects. It would be interesting to look at performance of the model against other subjects not part of the training database at all.

Appendix

The full code for the assignment is provided below. Output is held back (already used in the report above).

```
##
## Johns Hopkins University / Coursera
## Practical Machine Learning
## Course Project
## April 2015
##
## This script implements the Prediction Assignment for the Johns Hopkins / Coursera Practical
## Machine Learning course.  It imports the Weight Lifting Exercise dataset from the UCI Human
## Activity Recognition database, trains a Random Forest for prediction of exercise correctness,
## and tests the model on the course testing dataset.
##

##
## 1. Read and clean the data by removing any variables which have more than half missing
##    values or are descriptive of the data collection.  This limits the data available
##    prediction to just features collected,
##
##    The script assumes the files are in the current working directory.
##

library (caret)
library (randomForest)

dataTrainingRaw <- read.csv ("pml-training.csv", na.strings = "NA")
fKeep <- sapply (1:ncol (dataTrainingRaw), function (i) { if ( 0.5 < length (which (is.na (dataTrainingR
dataTraining <- dataTrainingRaw [ , fKeep]

##
## 2. Segment the data for training (80%) and testing (20%) then train a random forest
##    with 50 trees per run.  This keeps the training time manageable and (after some
##    experimentation) preserves accuracy.  Training uses out-of-bag resampling and
##    performs 5 iterations
##
##    A seed is set for reproducibility, and variable importance is examined.
##

set.seed (314159)
indexTraining <- createDataPartition (dataTraining$classe, p = 0.80, list = FALSE)
trainingSet <- dataTraining [indexTraining, ]
testingSet <- dataTraining [-indexTraining, ]

trainOptions <- trainControl(method = "oob", 5)
model <- train (classe ~ ., method = "rf", data = trainingSet, ntree=50, trControl = trainOptions)

variableImportance <- varImp (model)$importance
varOrder <- order (-variableImportance [ , 1])
#head (cbind (variable = names (trainingSet) [varOrder], importance = variableImportance [varOrder, 1])

##
## 3. Apply the model to predict the held-back testing data, to estimate out-of-sample
##    accuracy.
##
```

```
predictRF <- predict (model, testingSet)
testResults <- confusionMatrix (predictRF, testingSet$classe)
```