Machine Learning - Class Project

Chris Serrano

January 25, 2015

Executive Summary

With the advances in technology miniaturization is sensor technology it is now possible to collect vast amounts of physical activity data in cost effective way. Currently the majority of in the data being collected is being used to track and analyze how much of a particular activity or exercise or movements is being done, however this collected data is hardly used to determine if a particular exercise is been done correctly. This class project study will use data from accelerometers to determine how well a weight lifting exercise is done. Based on the data provided, the random forest algorithm can predict with a high degree of accuracy whether an individual is doing correctly or not the weight lifting exercise. The main problem with random forest algorithm is performance scalability. This performance scalability could be improvded by reduing the number of sensors. For the specifica weight exercise locating a ssnor on the belt could provide an accurancy of approcimately 90%. If higher accuracies are required combining 2 sensor could potentially produce accuracies above 95%.

Study Objective

The objective of this study is to determine if a machine learning algorithm can be used to determine or classify if a weight lifting exercise is being done in a correct way or not, “how well it is being done. Data from accelerometers on the belt, arm, forearm, and dumbbell of 6 participants will be used for this study

Data Description

Data for this project comes originally from the web site groupware.les.inf.puc-rio.br/har. Data was collected from accelerometers on the belt, arm, forearm, and dumbbell of six (6) participants. “Six (6) young health participants (ages between 20-28 with little weight experience) were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.
The training and test data for this project was downloaded from the following web sites:

- Training: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
- Testing: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The training dataset consist of 19622 observations and 160 variables, while the test dataset consist of 20 observations and 160 variables. The training dataset will be used to train the model and validate the model. The test dataset will be used in the final step to apply the machine-learning algorithm and to submit predictions for course project grading.

Data Selection and transformation

By simple observation using Microsoft Excel (opening the .csv training and test file) the following problems were identified:

- Variables with blanks or NAs or #DIV/0! Data
- Variables that may have been derived from the original data such max, min, var, std, amplitude etc.
- Variables that are not relevant to this project, such as name of participants, observation numbers, and windows are excluded from the datasets.

The following R code cleans up and prepares the data to be used for the project. Although timestamps may be relevant for this project, since we do not have a description of they represent and how it was taken, they will be removed from the data set

```
# SETUP WORKING DIRECTORY AND LOAD CARET LIBRARY
setwd("~/Documents/Machine Learning - Coursera/Project");
library("caret");
```

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

```
# CLEAN UP TRAINING DATASET AND SELECT DATASET VARIABLES FOR THE STUDY
fileName <- "pml-training.csv";
train <- read.csv(fileName, header=TRUE, sep=",", stringsAsFactors=FALSE, na.strings= c("NA
","","#DIV/0!"));
exclude_cols <- grep("^var|^avg|^max|^min|^std|^amplitude|^skewness|^kurtosis", names(train
));
train <- train[ , -exclude_cols];
train <- train[ , 8:60];
train <- train[ , c(53, 1:52)]
train$classe <- as.factor(train$classe);

# CLEAN UP TEST DATASET AND SELECT DATASET VARIABLES FOR THE STUDY
fileName <- "pml-testing.csv";
test <- read.csv(fileName, header=TRUE, sep=",", stringsAsFactors=FALSE, na.strings= c("NA"
,"","#DIV/0!"));
test <- test[ ,-exclude_cols];
test <- test[ , 8:60];
test <- test[ , c(53, 1:52)]
```

```
# SUMMARY CLASSE
summary(train$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
# VERIFY IF SELECTED VARIABLES HAVE UNIQUE VALUES
nearZeroVar(train[ ,-1], saveMetrics= TRUE);
```

```
##                     freqRatio percentUnique zeroVar   nzv
## roll_belt            1.101904     6.7781062   FALSE FALSE
## pitch_belt           1.036082     9.3772296   FALSE FALSE
## yaw_belt             1.058480     9.9734991   FALSE FALSE
## total_accel_belt     1.063160     0.1477933   FALSE FALSE
## gyros_belt_x         1.058651     0.7134849   FALSE FALSE
## gyros_belt_y         1.144000     0.3516461   FALSE FALSE
## gyros_belt_z         1.066214     0.8612782   FALSE FALSE
## accel_belt_x         1.055412     0.8357966   FALSE FALSE
## accel_belt_y         1.113725     0.7287738   FALSE FALSE
## accel_belt_z         1.078767     1.5237998   FALSE FALSE
## magnet_belt_x        1.090141     1.6664968   FALSE FALSE
```

```
## magnet_belt_y         1.099688     1.5187035   FALSE FALSE
## magnet_belt_z         1.006369     2.3290184   FALSE FALSE
## roll_arm             52.338462    13.5256345   FALSE FALSE
## pitch_arm            87.256410    15.7323412   FALSE FALSE
## yaw_arm              33.029126    14.6570176   FALSE FALSE
## total_accel_arm       1.024526     0.3363572   FALSE FALSE
## gyros_arm_x           1.015504     3.2769341   FALSE FALSE
## gyros_arm_y           1.454369     1.9162165   FALSE FALSE
## gyros_arm_z           1.110687     1.2638875   FALSE FALSE
## accel_arm_x           1.017341     3.9598410   FALSE FALSE
## accel_arm_y           1.140187     2.7367241   FALSE FALSE
## accel_arm_z           1.128000     4.0362858   FALSE FALSE
## magnet_arm_x          1.000000     6.8239731   FALSE FALSE
## magnet_arm_y          1.056818     4.4439914   FALSE FALSE
## magnet_arm_z          1.036364     6.4468454   FALSE FALSE
## roll_dumbbell         1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell        2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell          1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell  1.072634     0.2191418   FALSE FALSE
## gyros_dumbbell_x      1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y      1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z      1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x      1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y      1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z      1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x     1.098266     5.7486495   FALSE FALSE
## magnet_dumbbell_y     1.197740     4.3012945   FALSE FALSE
## magnet_dumbbell_z     1.020833     3.4451126   FALSE FALSE
## roll_forearm         11.589286    11.0895933   FALSE FALSE
## pitch_forearm        65.983051    14.8557741   FALSE FALSE
## yaw_forearm          15.322835    10.1467740   FALSE FALSE
## total_accel_forearm   1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x       1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y       1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z       1.122917     1.5645704   FALSE FALSE
## accel_forearm_x       1.126437     4.0464784   FALSE FALSE
## accel_forearm_y       1.059406     5.1116094   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x      1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y      1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z      1.000000     8.5771073   FALSE FALSE
```

The “nearZeroVar” R function diagnosis variables (predictors) that have one unique value or very unique values relatve to the number of samples. These are variables that have very small variance relative to the number of samples.

The following R code splits the “train dataset” into 60% used for training purposes (trainDS) and 40% used for validation purposes. (validateDS). In addition, to estimate the performance of the algorithm on the dataset, the “cross validation (method=cv)” will be used.

```
# SPLIT TRAIN DATASET. 60% TRAINING & 40% VALIDATION
set.seed(20150125);
trainIndex <- createDataPartition(train$class, p = 0.60, list=FALSE);
trainDS <- train[trainIndex, ];
validateDS <- train[-trainIndex, ];

# SETUP CROSS VALIDATION
foldControl <- trainControl(method="cv", number=5)
```

Machine Learning Models

Since we are looking to determine or classify how well a weight lifting activity is done based on sensor information collected from four (4) accelerometers (dumbbell, forearm, arm, & waist belt), two classification models will be explored:

- Classification tree (method = rpart)
- Random forest (method = rf)

The Classification tree performs well with large datasets, and it is also very easy to understand and interpret On the other hand, it can create complex trees that do not generalize well or over-fitting. Random forests correct for decision trees habit of over-fitting to their training set, however it requires more processing time.

The random forest algorithm constructs multiple decision trees at the training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

```
# MODEL No 1: CLASSIFICATION TREE
startTimeModel <- format(Sys.time(), "%a %b %d, %Y - %X")
library("rpart")
modelFit1 <- train(classe ~ ., method="rpart", trControl = foldControl, data=trainDS);
predictor1 <- predict(modelFit1, newdata = validateDS[,-1]);
confMatrixModel1 <- confusionMatrix(validateDS$classe, predictor1);
stopTimeModel <- format(Sys.time(), "%a %b %d, %Y - %X")
modelFit1
```

```
## CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 9420, 9421, 9421, 9420, 9422
##
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa       Accuracy SD  Kappa SD
##   0.02586616  0.5588469  0.44109081  0.01538933   0.02428920
##   0.04167062  0.4541481  0.27830810  0.08470629   0.14679051
##   0.11343142  0.3144555  0.04591075  0.04127363   0.06287455
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.02586616.
```

```
confMatrixModel1;
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1355  301  430  140    6
##          B  243  748  273  254    0
##          C   42  185 1107   34    0
##          D   78  208  354  646    0
##          E   20  365  291   99  667
##
## Overall Statistics
##
##                Accuracy : 0.5765
##                  95% CI : (0.5654, 0.5874)
##     No Information Rate : 0.3129
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.469
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7796  0.41395   0.4509  0.55072  0.99108
## Specificity            0.8564  0.87250   0.9516  0.90409  0.89196
## Pos Pred Value         0.6071  0.49275   0.8092  0.50233  0.46255
## Neg Pred Value         0.9318  0.83265   0.7919  0.91966  0.99906
## Prevalence             0.2215  0.23031   0.3129  0.14950  0.08578
## Detection Rate         0.1727  0.09534   0.1411  0.08233  0.08501
## Detection Prevalence   0.2845  0.19347   0.1744  0.16391  0.18379
## Balanced Accuracy      0.8180  0.64322   0.7013  0.72741  0.94152
```

```
## [1] "********** Model No 1 *********"
```

```
## [1] "Start date & time:"          "Sun Jan 25, 2015 - 18:25:46"
```

```
## [1] "Stop date & time:"           "Sun Jan 25, 2015 - 18:25:58"
```

```
# MODEL No 2: RANDOM FOREST
startTimeModel <- format(Sys.time(), "%a %b %d, %Y - %X")
library("randomForest")
```

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

```
modelFit2 <- train(classe ~ ., method="rf", trControl = foldControl, data=trainDS);
predictor2 <- predict(modelFit2, newdata = validateDS[,-1]);
confMatrixModel2 <- confusionMatrix(validateDS$classe, predictor2);
stopTimeModel <- format(Sys.time(), "%a %b %d, %Y - %X")
modelFit2
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 9421, 9421, 9421, 9420, 9421
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9894699  0.9866786  0.001629179  0.002062938
##   27    0.9896399  0.9868940  0.001817054  0.002299871
##   52    0.9838652  0.9795872  0.005371946  0.006801959
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
confMatrixModel2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    2    0    0    0
##          B   10 1501    7    0    0
##          C    0   16 1342   10    0
##          D    0    0   23 1263    0
##          E    0    1    0    7 1434
##
## Overall Statistics
##
##                Accuracy : 0.9903
##                  95% CI : (0.9879, 0.9924)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9877
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9955   0.9875   0.9781   0.9867   1.0000
## Specificity            0.9996   0.9973   0.9960   0.9965   0.9988
## Pos Pred Value         0.9991   0.9888   0.9810   0.9821   0.9945
## Neg Pred Value         0.9982   0.9970   0.9954   0.9974   1.0000
## Prevalence             0.2855   0.1937   0.1749   0.1631   0.1828
## Detection Rate         0.2842   0.1913   0.1710   0.1610   0.1828
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9976   0.9924   0.9871   0.9916   0.9994
```

```
## [1] "********** Model No 2 *********"
```

```
## [1] "Start date & time:"          "Sun Jan 25, 2015 - 18:25:58"
```

```
## [1] "Stop date & time:"           "Sun Jan 25, 2015 - 18:36:25"
```

The Classification Tree algorithm has an estimated accuracy of only 57.65% but it in total it required about 12 seconds to train, On the other hand, the random tree algorithm has an estimated accuracy of 99% but it required about 600 seconds to train. Therefore, for this specific situation we will use the random forest algorithm.

In order to reduce the amount of processing time required to we will explore only using data from one of the sensors at a time to determine what accuracy levels can be achieved. The following R code, using the random forest algorithm, determines for each individual sensor what accuracy levels can be achieved.

```
sensorLoc <- c("_dumbbell", "_forearm", "_arm", "_belt");
for (i in 1:4) {
  print(c("************ Sensor Location: ", sensorLoc[i]));
  startTimeModel <- format(Sys.time(), "%a %b %d, %Y - %X")
  include_cols <- grep(sensorLoc[i], names(train));
  sensorLocTrainDS <- trainDS[ ,c(1,include_cols)];
  sensorLocValidateDS <- validateDS[ ,c(1,include_cols)];
  sensorLocModelFit <- train(classe ~ ., method="rf", trControl = foldControl, data=sensorL
ocTrainDS);
  sensorLocPredictor <- predict(sensorLocModelFit, newdata = sensorLocValidateDS[,-1]);
  confMatrixSensorLoc <- confusionMatrix(sensorLocValidateDS$classe, sensorLocPredictor)$ov
erall[1:6];
  print(confMatrixSensorLoc)
  stopTimeModel <- format(Sys.time(), "%a %b %d, %Y - %X")
  print(c("Start date & time:", startTimeModel, " & Stop date & time:", stopTimeModel ))
}
```

```
## [1] "************ Sensor Location: " "_dumbbell"
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.8877135      0.8577690      0.8805193      0.8946182      0.2950548
## AccuracyPValue
##      0.0000000
## [1] "Start date & time:"          "Sun Jan 25, 2015 - 18:36:25"
## [3] " & Stop date & time:"        "Sun Jan 25, 2015 - 18:39:21"
## [1] "************ Sensor Location: " "_forearm"
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.8926842      0.8641683      0.8856254      0.8994498      0.2876625
## AccuracyPValue
##      0.0000000
## [1] "Start date & time:"          "Sun Jan 25, 2015 - 18:39:21"
## [3] " & Stop date & time:"        "Sun Jan 25, 2015 - 18:42:24"
## [1] "************ Sensor Location: " "_arm"
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.8706347      0.8363973      0.8630072      0.8779857      0.2829467
## AccuracyPValue
##      0.0000000
## [1] "Start date & time:"          "Sun Jan 25, 2015 - 18:42:24"
## [3] " & Stop date & time:"        "Sun Jan 25, 2015 - 18:45:43"
## [1] "************ Sensor Location: " "_belt"
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9156258      0.8932759      0.9092563      0.9216845      0.2847311
## AccuracyPValue
##      0.0000000
## [1] "Start date & time:"          "Sun Jan 25, 2015 - 18:45:43"
## [3] " & Stop date & time:"        "Sun Jan 25, 2015 - 18:48:43"
```

Based on the data provided using the random forest algorithm the best sensor to determine whether an individual is doing correctly a weight lifting exercise is to located in the belt, with a 91.6 accuracy. Since this technology can be used to advice weight lifters and it does not have pose any risk to the user it may be acceptable. Accuracy could potentially increase above 95% by combining measurements taken by 2 sensors, such as the belt and forearm sensor.