HAR Sensors As A Predictor of Exercise Form

BACKGROUND Human Activity Recognition - HAR - has emerged as a key research area in the last years and is gaining increasing attention by the pervasive computing research community. 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. Accelerometers were placed on the belt, forearm, arm, and dumbell of 6 study participants. They were then asked to perform barbell lifts correctly and incorrectly in 5 different ways while the accelerometer components were recorded for each.

EXECUTIVE SUMMARY The goal of this project is to predict the manner in which the participants did the exercise. The Random Forest model was selected and used to correctly predict the values for 20 different test cases.

Initial Analysis An initial look at the data in both the training and prediction data sets shows that they each contain many variables that are unrelated to the measurement of movement. The final data set will contain only data collected from the accelerometers and an outcome variable (classe).

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

```
## Warning: package 'ggplot2' was built under R version 3.1.2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
pml.training <- read.csv("pml-training.csv", header=TRUE)
pml.training$classe<-factor(pml.training$classe)
is.na(pml.training) <- pml.training == ''
is.na(pml.training) <- pml.training == '#DIV/0!'
train <- pml.training[,colSums(is.na(pml.training)) == 0]
train<-train[,8:60]

pml.testing <- read.csv("pml-testing.csv", header=TRUE)
is.na(pml.testing) <- pml.testing == ''
is.na(pml.testing) <- pml.testing == '#DIV/0!'
test <- pml.testing[,colSums(is.na(pml.testing)) == 0]
test<-test[,8:59]
```

Next, the training data was divided into two sets, one for training and one for internal testing.

```
set.seed(123)
inTrain <- createDataPartition(y=train$classe,p=0.7, list=FALSE)
training <- train[inTrain,]
testing <- train[-inTrain,]
```

The expected outcome for these predictors is a categorical variable, so if we are to use a standard regression model, we will first need to encode "dummy variables" to accomodate the model. An better way is to construct a Random Forest, and make predictions with it.

```
#trainCtrl<-trainControl(method = "cv",
#                         number=4)
#modl <- train(classe ~ ., data = training,
#              method = "rf",
#              trControl = trainCtrl)
#modl

modFit <- randomForest(classe ~., training, method = "class")
prediction<-predict(modFit,testing,type="class")
confusionMatrix(prediction, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    5    0    0    0
##          B    1 1134   11    0    0
##          C    0    0 1015   15    0
##          D    0    0    0  948    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9956   0.9893   0.9834   1.0000
## Specificity            0.9988   0.9975   0.9969   1.0000   0.9998
## Pos Pred Value         0.9970   0.9895   0.9854   1.0000   0.9991
## Neg Pred Value         0.9998   0.9989   0.9977   0.9968   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1927   0.1725   0.1611   0.1839
## Detection Prevalence   0.2851   0.1947   0.1750   0.1611   0.1840
## Balanced Accuracy      0.9991   0.9965   0.9931   0.9917   0.9999
```

Random forests were constructed with both the randomForest package and with caret's method="rf". Both gave very excellent and very similar results. Because the randomForest package executed faster, it was used for the final model. The confusion matrix shows that very few incorrect predictions were made, yielding a Kappa statistic of 0.9929 and an out of sample error rate = .0012.

```
zz<-sort(modFit$importance,index=TRUE,decreasing=TRUE)
importance<-cbind(zz$x,names(training[,zz$ix]))
importance
```

```
##       [,1]               [,2]
```

```
##  [1,] "840.753858590441" "roll_belt"
##  [2,] "623.009117528011" "yaw_belt"
##  [3,] "549.794896469743" "pitch_forearm"
##  [4,] "510.510350357133" "magnet_dumbbell_z"
##  [5,] "473.994650223263" "magnet_dumbbell_y"
##  [6,] "472.634411288872" "pitch_belt"
##  [7,] "406.274813101744" "roll_forearm"
##  [8,] "339.587094906636" "magnet_dumbbell_x"
##  [9,] "306.402419614596" "accel_belt_z"
## [10,] "302.522730279951" "roll_dumbbell"
## [11,] "287.383038950061" "accel_dumbbell_y"
## [12,] "274.515023267593" "magnet_belt_y"
## [13,] "265.81805791284"  "magnet_belt_z"
## [14,] "236.638638749281" "accel_dumbbell_z"
## [15,] "231.338331626032" "accel_forearm_x"
## [16,] "226.279966094194" "roll_arm"
## [17,] "212.253263344801" "gyros_belt_z"
## [18,] "191.556054374901" "magnet_forearm_z"
## [19,] "189.660607351023" "magnet_arm_x"
## [20,] "185.355791028613" "total_accel_dumbbell"
## [21,] "184.66892590048"  "gyros_dumbbell_y"
## [22,] "178.00013106502"  "yaw_dumbbell"
## [23,] "173.329637577134" "accel_dumbbell_x"
## [24,] "172.986381167225" "magnet_belt_x"
## [25,] "172.783694682914" "accel_arm_x"
## [26,] "171.857433103299" "accel_forearm_z"
## [27,] "167.987770887595" "yaw_arm"
## [28,] "166.195046736228" "magnet_arm_y"
## [29,] "161.315046664245" "magnet_forearm_x"
## [30,] "156.054873536079" "magnet_forearm_y"
## [31,] "155.248767318736" "total_accel_belt"
## [32,] "135.157677501027" "magnet_arm_z"
## [33,] "126.186511193889" "pitch_arm"
## [34,] "123.490168884605" "pitch_dumbbell"
## [35,] "113.493992431878" "yaw_forearm"
## [36,] "112.876839038143" "accel_arm_y"
## [37,] "103.93681152236"  "accel_forearm_y"
## [38,] "97.4120859044993" "accel_arm_z"
## [39,] "94.404992098293"  "gyros_dumbbell_x"
## [40,] "94.3631364692738" "gyros_arm_y"
## [41,] "90.1593255139591" "gyros_forearm_y"
## [42,] "90.1554811809041" "gyros_arm_x"
## [43,] "89.8657437624896" "accel_belt_y"
## [44,] "84.2576883281141" "accel_belt_x"
## [45,] "79.9239814813498" "total_accel_forearm"
## [46,] "76.6597200631482" "gyros_belt_y"
## [47,] "70.7329630471652" "gyros_belt_x"
## [48,] "70.5896641918575" "total_accel_arm"
## [49,] "62.2587945717073" "gyros_dumbbell_z"
## [50,] "61.0983123520612" "gyros_forearm_z"
## [51,] "53.9442521082073" "gyros_forearm_x"
## [52,] "42.250941040597"  "gyros_arm_z"
```

Looking at the variable importance, it looks as if a significant number of features can be eliminated. Executing

randomForest's cross-validation cvmodel confirms that somewhere between 26 and 52 features produces very good, almost indistinguishable results. Elminating all variables with a score less than 100 from the model leaves 38 features in the model.

```
nv<-c("roll_belt","yaw_belt","pitch_forearm","magnet_dumbbell_z","magnet_dumbbell_y","pitch_belt","roll

newtraining<-subset(train,select=nv)
newtesting<-subset(testing,select=nv)
modFit <- randomForest(classe ~., newtraining, method = "class")
prediction<-predict(modFit,newtesting,type="class")
confusionMatrix(prediction, newtesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    0 1026    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9994, 1)
##     No Information Rate : 0.2845
##     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             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

This prodces a Kappa statistic of 1 and an out of sample error rate = 0!

PREDICTION Utilizing the model, the outcomes of the twenty unkown test cases are predicted.

```
unkPredict<-predict(modFit,test,type="class")
unkPredict
```

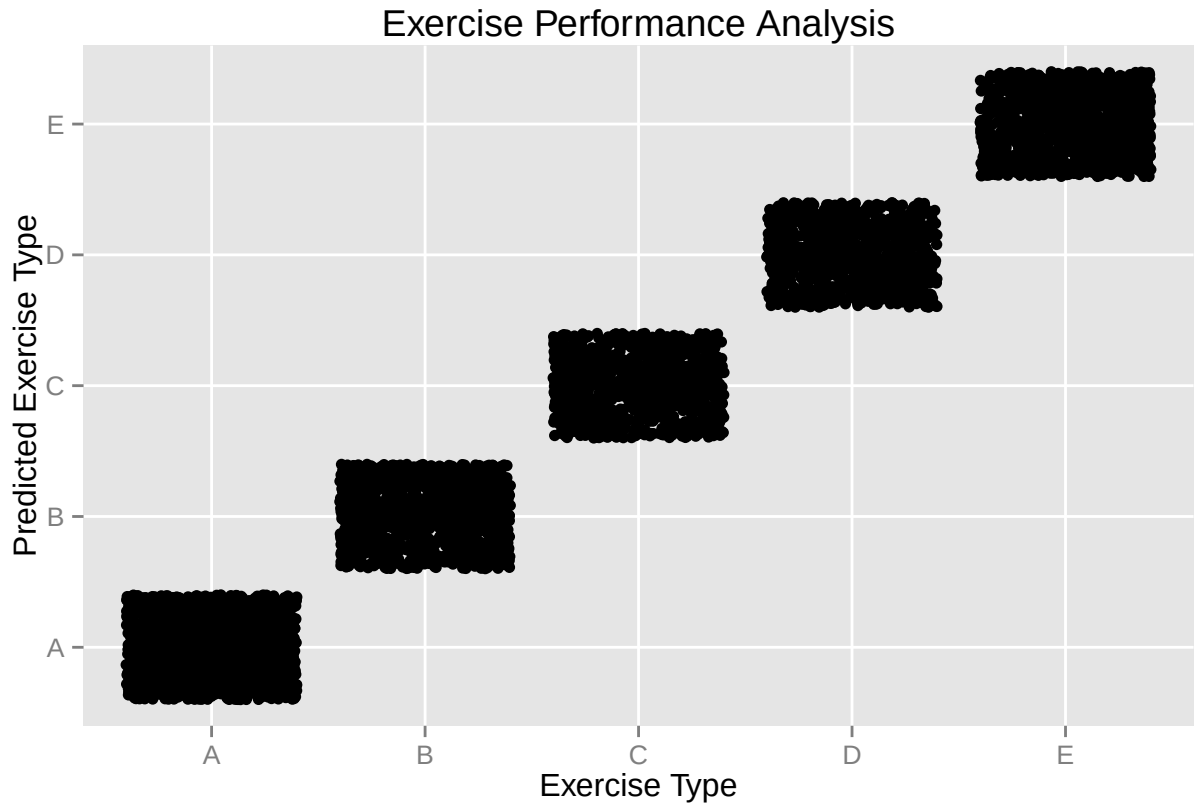
```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

CONCLUSION The original model produced very impressive results, but eliminating features that do not relatively contribute to the end result produces a much better model.

Appendix

Plot of Actual Versus Predicted Values

```
testQ<-cbind(testing,prediction)
qplot(testQ$classe,testQ$prediction,geom=c("boxplot", "jitter"),  main="Exercise Performance Analysis",
xlab="Exercise Type", ylab="Predicted Exercise Type")
```



Data set obtained from: http://groupware.les.inf.puc-rio.br/static/har/dataset-har-PUC-Rio-ugulino.zip