Can You Predict Weight Lifting Mistakes from Accelerometer Data?

by Scott Semel

January 28, 2015

Executive Summary

Six young health participants 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. We trained and tested several classifiers to find out that accurate prediction of the Class type was possible depending on the method with between 70 and 97% accuracy.

Read more: http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises#ixzz3yZxNTUIP

Data Processing

```
load("workspace2016-01-26")
#  The Weightlifting data set as 19622 records with 160
columns.
library(AppliedPredictiveModeling)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

dat = read.csv("pml-training.csv")
test = read.csv("pml-testing.csv")
# Several columns in the test set has only NA or mostly NA so
we removed those.
test2 = test[ , ! apply( test , 2 , function(x) any(is.na(x)) )
]
# And we realized it was only necessary to train on those
columns in the test set. There were more columns in the
training set that would never get used anyway.
keepvar = names(test2)
keepvar = keepvar[-60]
keepvar = c(keepvar,"classe")
# It was important to also remove the first 8 columns from the
```

```r
training and test set because there were counters and
timestamps that we did not want to affect the outcome. Leaving
those in confused the classifiers which thought they were doing
better than they were.
keepvar = keepvar[8:60]
dat2 = dat[, keepvar]
# We trained on 75% and tested on the other 25%. It did not
appear to matter where the seed started and still got very
close solutions.
inTrain = createDataPartition(y=dat2$classe, p = .75,
list=FALSE)
training = dat2[ inTrain,]
testing = dat2[-inTrain,]
# The linear discriminant analysis did not work as well only
producting 70% accurcay
# modFit1 = train(classe~ .,data=training, method="lda")
pred = predict(modFit1,testing)

## Loading required package: MASS

#confusionMatrix(pred, testing$classe)
predict(modFit1,test)

##  [1] B A B C C E D D A A D A B A E A A B B B
## Levels: A B C D E

# We tried it with and without cross-validation and it only
increased accuracy slightly. Also performing the centering and
standardizing between 0 and 1 did not make a difference in the
prediction so we didn't continue to do that.
tc = trainControl("repeatedcv",
                    number=10,
                    repeats=10,
                    classProbs=TRUE,
                    savePred=T)
# modFit2 = train(classe~ .,data=training,
method="lda",trControl=tc,
#                  preProc=c("center", "scale"))
pred = predict(modFit2,testing)
#table(pred, testing$classe)
#confusionMatrix(pred, testing$classe)
predict(modFit2,test)

##  [1] B A B C C C D D A A D A B A E A A B B B
## Levels: A B C D E

# The boosted trees method worked with high accuracy around 97%
# modFit3 = train(classe~ .,data=training, method="gbm")
# It was helpful to see the importance of each variable. At
first we tried very small subsets of important variables. Then
we realized we could test all of them. It is interesting that
```

```
it has different lists of importance each time so we couldnt
just use this function to tell us which variables to use.
gbmImp = varImp(modFit3, scale = FALSE)

## Loading required package: gbm

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1

## Loading required package: plyr

pred = predict(modFit3,testing)
confusionMatrix(pred, testing$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1378   25    0    0    3
##          B   14  908   26    0    6
##          C    1   15  812   32    6
##          D    1    1   13  769   18
##          E    1    0    4    3  868
##
## Overall Statistics
##
##                Accuracy : 0.9655
##                  95% CI : (0.96, 0.9705)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9564
##  Mcnemar's Test P-Value : 0.0001159
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D
Class: E
## Sensitivity            0.9878   0.9568   0.9497   0.9565
0.9634
```

```
## Specificity            0.9920   0.9884   0.9867   0.9920
0.9980
## Pos Pred Value         0.9801   0.9518   0.9376   0.9589
0.9909
## Neg Pred Value         0.9951   0.9896   0.9894   0.9915
0.9918
## Prevalence             0.2845   0.1935   0.1743   0.1639
0.1837
## Detection Rate         0.2810   0.1852   0.1656   0.1568
0.1770
## Detection Prevalence   0.2867   0.1945   0.1766   0.1635
0.1786
## Balanced Accuracy      0.9899   0.9726   0.9682   0.9742
0.9807

predict(modFit3,test)

##  [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

# The rpart method did only produced around 65% accuracy for
some reason.
# modFit4 = train(classe~ .,data=training, method="rpart",)
pred = predict(modFit4,testing)

## Loading required package: rpart

#confusionMatrix(pred, testing$classe)
predict(modFit4,test)

##  [1] C D D C C C D D A A C D C A D D C D C D
## Levels: A B C D E

# The random forest method worked the best with possibly more
than 97% accuracy.
# modFit5 = train(classe~ .,data=training, method="rf")
pred = predict(modFit5,testing)

## Loading required package: randomForest

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

confusionMatrix(pred, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    0  855    3    0
##          D    0    0    0  801    2
##          E    0    0    0    0  899
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9976, 0.9997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##  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   0.9963
0.9978
## Specificity            1.0000   1.0000   0.9993   0.9995
1.0000
## Pos Pred Value         1.0000   1.0000   0.9965   0.9975
1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   0.9993
0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639
0.1837
## Detection Rate         0.2845   0.1935   0.1743   0.1633
0.1833
## Detection Prevalence   0.2845   0.1935   0.1750   0.1637
0.1833
## Balanced Accuracy      1.0000   1.0000   0.9996   0.9979
0.9989
```

```
predict(modFit5,test)
```

```
##  [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
```

```
# The next step would be to combine all the models and either
average the answers or vote with them. But in this case random
forests scored 100% on the 20 question validation set so it was
not necessary.
```

Results

A relationship was found between the class and the predictor variables.

We successfully reduced the number of useful variables to these:

```
gbmImp

## gbm variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                     Overall
## roll_belt           2907.6
## pitch_forearm       1529.4
## yaw_belt            1256.5
## magnet_dumbbell_z    940.0
## magnet_dumbbell_y    802.7
## roll_forearm         733.5
## magnet_belt_z        567.2
## gyros_belt_z         453.4
## roll_dumbbell        387.4
## pitch_belt           370.7
## accel_forearm_x      367.4
## gyros_dumbbell_y     321.6
## accel_dumbbell_y     286.1
## magnet_belt_y        228.5
## magnet_forearm_z     211.2
## accel_dumbbell_x     206.7
## yaw_arm              188.4
## accel_belt_z         167.6
## magnet_dumbbell_x    148.7
## magnet_arm_z         144.3
```

Even from this small data set the work appears promising.

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises#ixzz3yZx0m9BL