Machine Learning Project

Gary Baggett (Isleguard)

Saturday, October 17, 2015

Executive Summary

The answer to the question: How to determine the "classe" output using various machine learning techniques and then use that information to determine the "classe" of proper weighlifting curls. Several methods were used to determine the correct output. It was finally determined to use the Random Forest method as it is a non-linear method.

A non-linear method was chosen because, non-linear can do linear but linear cannot do non-linear well.

The process to generate the answer to this question if below. Use the instructor provided script [see Appendix] to create the required answer files.

Background Information

Data used with permission comes from:

Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6. Cited by 2 (Google Scholar)

Movement Classifications involved from the above paper:

"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)."

From this we learn the Classification type;
A. Correctly performed exercise
B. Throwing the elbows to the front - Yaw
C. Lifting the dumbbell only halfway - partial movement
D. Lowering the dumbbell only halfway - partial movement
E. Throwing the hips to the front - belt moves too much

Notes: Variables in dataset
1. Yaw: one elbow moves to the front - should be weaker side
2. Pitch: moving the barbell up and down
3. Roll: Raising one arm up faster than the other

```
## Loading required package: lattice
## Loading required package: ggplot2
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'dplyr'
##
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
##
## The following object is masked from 'package:randomForest':
##
##     combine
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Exploratory Data Analysis

Multiple methods of analysis were performed the data from the different trials are not shown as part of this document.

Methods tested in the EDA 1.Rpart
2.Random Forrest
3.BagRboostR
4.Party
5.bagFDA

By checking the required variables many of the 159 data variables were removed. The 160th variable is the outcome as produced by the authors of the original work - cited above.

The linear prediction and generalized linear prediction models were not used due to the thinking that due to the nonlinear effects of the movement and types of movement involved that either the bagging or the Random Forest methods would be the best.

I chose the Random Forest because it is a non-linear method. It seemed to provide the best overall data selections due to the group (nearest neighbor) member selection.

The thought occurred to me also that non-linear models will show linear output when needed, but linear models will not do a good job predicting with non-linear data. Therefore choose the non-linear prediction method over any generalized linear prediction method unless the EDA shows otherwise

Within the EDA, the ability to show the importance of the various variables results in an interesting listing.

```
set.seed(1954)          #set the random number seed for reproducability
print(nvz<-nearZeroVar(trdata2,saveMetrics=TRUE))
```

```
##                     freqRatio percentUnique zeroVar   nzv
## roll_dumbbell        1.022388    84.2065029   FALSE FALSE
## roll_forearm        11.589286    11.0895933   FALSE FALSE
## pitch_belt           1.036082     9.3772296   FALSE FALSE
## pitch_arm           87.256410    15.7323412   FALSE FALSE
## pitch_dumbbell       2.277372    81.7449801   FALSE FALSE
## pitch_forearm       65.983051    14.8557741   FALSE FALSE
## yaw_belt             1.058480     9.9734991   FALSE FALSE
```

```
## yaw_arm              33.029126    14.6570176   FALSE FALSE
## yaw_dumbbell          1.132231    83.4828254   FALSE FALSE
## yaw_forearm          15.322835    10.1467740   FALSE FALSE
## total_accel_belt      1.063160     0.1477933   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
## total_accel_arm       1.024526     0.3363572   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
## total_accel_dumbbell  1.072634     0.2191418   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
## total_accel_forearm   1.128928     0.3567424   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
## classe                1.469581     0.0254816   FALSE FALSE
```

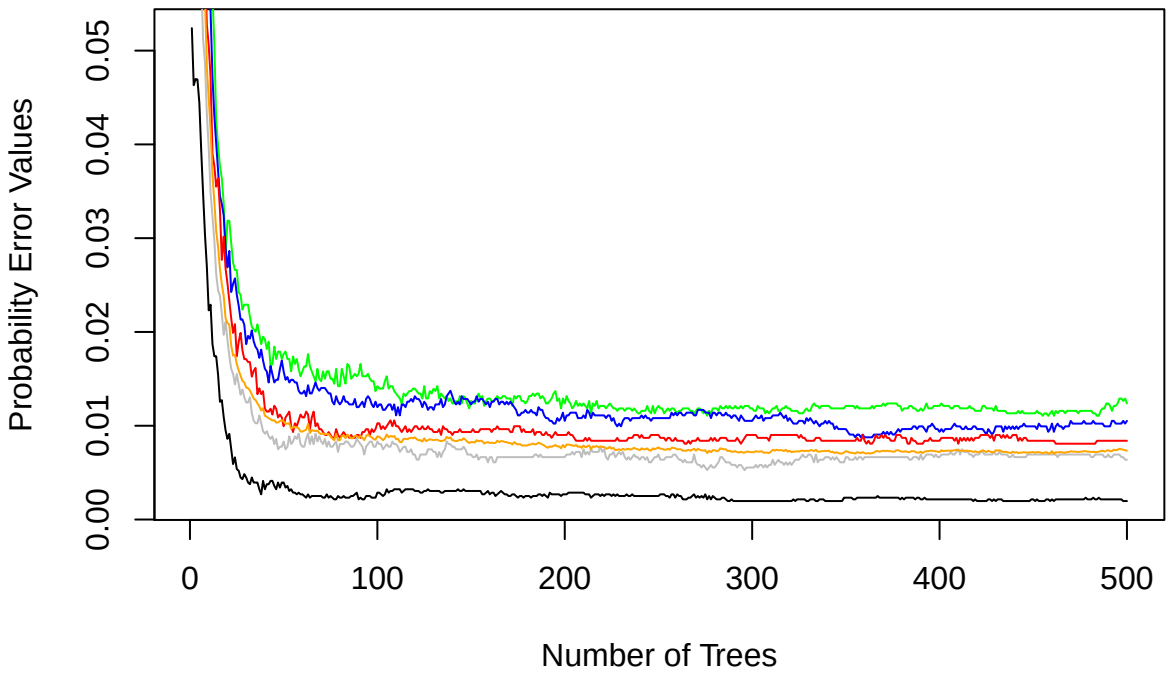
```
####  Now run the training setup and show the output
rf<-randomForest(formula=classe~. ,data=trdata2, ntree=500,replace=TRUE,importance=TRUE)

#   R command:  varImp(rf, value = "rf", scale=TRUE)**
print(v<-varImp(rf, value = "rf", scale=TRUE))
```

```
##                            A        B        C        D        E
## roll_dumbbell        35.84084 48.02696 50.43836 45.17298 45.84497
## roll_forearm         44.26061 38.98363 60.18801 35.14512 37.72617
## pitch_belt           49.92270 82.18361 62.34172 57.59725 55.92528
## pitch_arm            27.48632 32.20269 29.18799 33.68853 26.74411
## pitch_dumbbell       17.33040 27.63964 25.09645 21.22078 20.67940
## pitch_forearm        41.67661 47.03785 49.07619 53.33034 48.99319
## yaw_belt             79.88020 71.32210 69.49020 84.94340 52.50318
## yaw_arm              44.58847 44.39594 41.28635 47.55774 35.97873
## yaw_dumbbell         32.42888 37.91066 40.56507 36.56770 37.68645
## yaw_forearm          35.35344 39.39333 36.04861 33.45656 36.84674
## total_accel_belt     26.31660 31.54307 29.33933 27.23731 31.57833
## accel_belt_x         25.01371 33.16626 34.37437 28.51031 28.82412
## accel_belt_y         20.35146 29.79594 26.27376 26.94310 24.67232
## accel_belt_z         38.74965 47.85406 52.13866 45.13395 39.88908
## total_accel_arm      17.40488 38.63905 35.34254 33.84585 33.54081
## accel_arm_x          29.47751 29.76698 29.46352 39.49924 25.09069
## accel_arm_y          31.56932 39.21136 33.89742 34.77545 30.95177
## accel_arm_z          23.43657 39.80219 36.28503 38.98568 35.37178
## total_accel_dumbbell 27.65207 30.62658 28.01097 28.41544 32.98266
## accel_dumbbell_x     24.62721 34.84553 31.76301 30.71821 31.30502
## accel_dumbbell_y     34.90174 42.33330 41.13997 39.40206 38.83615
## accel_dumbbell_z     33.31499 38.94170 37.41767 39.94173 43.64168
## total_accel_forearm  33.98628 32.82179 37.19041 31.29023 33.02075
## accel_forearm_x      27.10122 40.33970 37.70619 46.58363 37.06826
## accel_forearm_y      35.13244 40.05967 35.47756 35.46351 35.37432
## accel_forearm_z      36.22802 43.12597 50.32071 36.82522 39.79911
```

Show the plot of the errors.

```
pdataa<-as.matrix(rf$err.rate)
pdata<-as.data.frame(pdataa)
plot(x=1:500,y=pdata$A, xlab="Number of Trees",ylab="Probability Error Values", type="l")
lines(x=1:500,y=pdata$B,col="green")
lines(x=1:500,y=pdata$C,col="blue")
lines(x=1:500,y=pdata$D,col="red")
lines(x=1:500,y=pdata$E,col="grey")
lines(x=1:500,y=pdata$OOB,col="orange")
```



Conclusion

The Random Forest model correctly predicted and confirmed the statement by the authors about their ability to predict the performance of the "curl" exercise.

The "Confusion Matrix" for the data;

```
print(rf$confusion)
```

```
##      A    B    C    D    E class.error
## A 5569    6    1    4    0 0.001971326
## B   28 3750   16    1    2 0.012378193
## C    1   20 3386   15    0 0.010520164
## D    0    3   22 3189    2 0.008395522
## E    0    7    7    9 3584 0.006376490
```

Appendix

Notes

```
####################################
#   Instructor Provided Script below
#   to generate the answer files
####################################
#pml_write_files = function(x){
#  n = length(x)
#  for(i in 1:n){
#    filename = paste0("problem_id_",i,".txt")
#    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
#  }
#}
# pml_write_files(rf.prediction)
```