WLE.rmd

David

Monday, August 18, 2014

Examination of Weight Lifting Exercises Dataset

Introduction

Six young healthy 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).

Their movements were monitored using four detectors strapped to their forearm, upperarm, waist and to the dumbbell itself. For feature extraction a sliding window approach was used with different lengths from 0.5 second to 2.5 seconds, with 0.5 second overlap. In each step of the sliding window approach the features on the Euler angles (roll, pitch and yaw), as well as the raw accelerometer, gyroscope and magnetometer readings were calculated.

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this report comes from this source: http://groupware.les.inf.puc-rio.br/har.

Method

The algorithm selected to predict the quality of execution variable (classe) is random forest. This uses a decision tree approach with bootstrapping to generate a "forest" of potentail trees. New trees are developed by restricting the number of variables that can be randomly selected at each node. This increases the variance between the trees improving the prediction accuracy.

The random forest methodology was chosen because: -It is unexcelled in accuracy among current algorithms for class prediction. -It runs efficiently on large data bases. -It can handle thousands of input variables without variable deletion. -It gives estimates of what variables are important in the classification.

Download and read files

The files are downloaded and read. The training file is named "training"" and the testing file is named "validation"

```
if (!file.exists("data")) {
        dir.create("data")
}
fileUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
fileUrltest <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

download.file(fileUrl, destfile = "./data/training.csv")
download.file(fileUrltest, destfile = "./data/testing.csv")

#Read the relevant tables.
```

```r
training<- read.csv( "./data/training.csv")
validation <- read.csv( "./data/testing.csv")
```

Preprocess files

Some rows represent a summary of previous observations (the 2.5sec window as opposed to the 0.5). As the test data file does not provide this data our training and validation data files need to exclude them as well.

The measurements described in the introduction represent four measurement devices providing four measures each in three dimensions. This provides 48 predictors plus the predicted variable (classe). The next step of the preprocessing is to reduce the files to these 49 rows. This requires the removal of all extraneous variables (names, time, etc) which might give invalid correlations.

No further preprocessing was undertaken as the random forest method is not sensitive to issues of skewness etc..

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.1

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

```r
training <- training[training$new_window=="no",]
training <- training[,c(8:10,37:48,60:68,84:86,113:124,151:160)]
nzv <- nearZeroVar(training,saveMetrics=T)
dim(training)
```

```
## [1] 19216    49
```

```r
validation <- validation[,c(8:10,37:48,60:68,84:86,113:124,151:160)]
```

Finally we checked that the remaining predictors are valid by ensuring that they do not have a near zero variance (none do, see appendix). This gave a data frame of 19216 rows and 49 columns.

Create a training and testing set

The random forest uses a bootstrap method to estimates out of bag error (oob). This is used to estimate the optimal mtry (number of variables available for selection at each node)and hence does not provide an independent test of the accuracy of the final model.

Therefore the training file needs to be split into a training file (on which the model is built) and a teating file which will allow for cross validation and the calculation of the out of sample error

```r
set.seed(1334)

inTraining <- createDataPartition(y=training$classe,p=0.6,list=FALSE)
training <- training[inTraining,]
testing <- training[-inTraining,]
```

Build model

As already explained the random forest method was adopted. However because the default approach was too computationally demanding for my PC the "traincontrol" argument was used. This provided for a 4 fold cross validation.

```
modFit <- train(classe~.,data=training,method="rf",trControl = trainControl(method = "cv", number = 4, a
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.1.1
```

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

```
## Warning: package 'e1071' was built under R version 3.1.1
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=25
## - Fold1: mtry=25
## + Fold1: mtry=48
## - Fold1: mtry=48
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=25
## - Fold2: mtry=25
## + Fold2: mtry=48
## - Fold2: mtry=48
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=25
## - Fold3: mtry=25
## + Fold3: mtry=48
## - Fold3: mtry=48
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=25
## - Fold4: mtry=25
## + Fold4: mtry=48
## - Fold4: mtry=48
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 25 on full training set
```

```
print(modFit)
```

```
## Random Forest
##
## 11532 samples
##    48 predictors
```

```
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 8649, 8648, 8649, 8650
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     1         1      9e-04        0.001
##   20    1         1      7e-04        9e-04
##   50    1         1      0.002        0.003
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 25.
```

The model indicated the optimal mtry was 20 and this adopted model gives an accuracy of 0.988.

Predict result on test data

To calculate out of sample error this model was applied to the testing data and its results compared to the actual classe values using the confusion matrix function.

```r
pred <- predict(modFit,testing)
print(confusionMatrix(pred,testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1345    0    0    0    0
##          B    0  863    0    0    0
##          C    0    0  818    0    0
##          D    0    0    0  731    0
##          E    0    0    0    0  867
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.291
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
```

```
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.291    0.187    0.177    0.158    0.188
## Detection Rate          0.291    0.187    0.177    0.158    0.188
## Detection Prevalence    0.291    0.187    0.177    0.158    0.188
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

The confusion matrix predicts 0 out of sample error and therefore there is no opportunity to improve the model.

Predict results for 20 test cases

Finally the model is used to predict the classe variable for the 20 test cases provided. It correctly predicts all 20.

```r
answers <- predict(modFit,validation)
print(answers)
```

```
##  [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
```

Appendix

```r
print(nzv)
```

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt                1.086       6.83805   FALSE FALSE
## pitch_belt               1.037       9.53893   FALSE FALSE
## yaw_belt                 1.047      10.10616   FALSE FALSE
## gyros_belt_x             1.051       0.72336   FALSE FALSE
## gyros_belt_y             1.149       0.35908   FALSE FALSE
## gyros_belt_z             1.071       0.87948   FALSE FALSE
## accel_belt_x             1.059       0.85346   FALSE FALSE
## accel_belt_y             1.115       0.74417   FALSE FALSE
## accel_belt_z             1.081       1.55600   FALSE FALSE
## magnet_belt_x            1.089       1.70171   FALSE FALSE
## magnet_belt_y            1.097       1.55079   FALSE FALSE
## magnet_belt_z            1.019       2.37302   FALSE FALSE
## roll_arm                51.154      13.75937   FALSE FALSE
## pitch_arm               85.282      15.96066   FALSE FALSE
## yaw_arm                 32.282      14.89904   FALSE FALSE
## gyros_arm_x              1.024       3.34617   FALSE FALSE
## gyros_arm_y              1.451       1.95150   FALSE FALSE
## gyros_arm_z              1.119       1.29059   FALSE FALSE
## accel_arm_x              1.018       4.04351   FALSE FALSE
## accel_arm_y              1.169       2.78414   FALSE FALSE
## accel_arm_z              1.139       4.12157   FALSE FALSE
## magnet_arm_x             1.012       6.96295   FALSE FALSE
## magnet_arm_y             1.045       4.53268   FALSE FALSE
## magnet_arm_z             1.028       6.57785   FALSE FALSE
```

```
## roll_dumbbell         1.038       83.75312   FALSE FALSE
## pitch_dumbbell        2.248       81.22398   FALSE FALSE
## yaw_dumbbell          1.132       83.02456   FALSE FALSE
## gyros_dumbbell_x      1.010        1.25416   FALSE FALSE
## gyros_dumbbell_y      1.271        1.44151   FALSE FALSE
## gyros_dumbbell_z      1.053        1.06682   FALSE FALSE
## accel_dumbbell_x      1.006        2.21170   FALSE FALSE
## accel_dumbbell_y      1.062        2.41986   FALSE FALSE
## accel_dumbbell_z      1.150        2.12323   FALSE FALSE
## magnet_dumbbell_x     1.094        5.84929   FALSE FALSE
## magnet_dumbbell_y     1.189        4.38177   FALSE FALSE
## magnet_dumbbell_z     1.027        3.51270   FALSE FALSE
## roll_forearm         11.726       11.23543   FALSE FALSE
## pitch_forearm        64.576       15.09679   FALSE FALSE
## yaw_forearm          15.236       10.29871   FALSE FALSE
## gyros_forearm_x       1.050        1.54559   FALSE FALSE
## gyros_forearm_y       1.043        3.84055   FALSE FALSE
## gyros_forearm_z       1.112        1.58201   FALSE FALSE
## accel_forearm_x       1.143        4.13197   FALSE FALSE
## accel_forearm_y       1.050        5.20920   FALSE FALSE
## accel_forearm_z       1.019        3.01311   FALSE FALSE
## magnet_forearm_x      1.013        7.92569   FALSE FALSE
## magnet_forearm_y      1.256        9.72627   FALSE FALSE
## magnet_forearm_z      1.018        8.75833   FALSE FALSE
## classe                1.471        0.02602   FALSE FALSE
```

References

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