Weight Lifting Activity Machine Learning

Ravi Addanki

12/28/2019

Overview

The purpose of this document is to generate a machine learning algorithm that predicts if the Weight Lifting Excercises are done correctly provided certain body movement measurements are collected through sensors. For this a collection of weightlifting samples of measurements was used from the groupware website http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). This data was classified by trainers using the variable classe. According to the final ML algorithm developed, the weight lifting excercises were performed pefectly (classe=A) in 7 out of 20 test cases (validation cases) while 8 others were good enough (Classe=B).

Data set

The training and testing datasets are obtained from specified sources. Per documentation the variables of interest are raw accelerometer, gyroscope and magnetometer readings and Euler angles roll, pitch, and yaw. There seems to be no missing data.

```
buildFile <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.
csv'
validFile <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.c
sv'
if (!file.exists("data/buildData.csv")) download.file(buildFile,method="curl",d
estfile= "data/buildData.csv")
if (!file.exists("data/validData.csv")) download.file(validFile,method="curl",d
estfile= "data/validData.csv")
buildData <-read.csv("data/buildData.csv")
validData <-read.csv("data/validData.csv")
colNames1 <-colnames(buildData)
colNames2 <- colNames1[grep("^accel_|^gyros_|^magnet_|^pitch_|^roll_|^yaw_",col
Names1)]
colNames3 <- colNames2[grep("^pitch_|^roll_|^yaw_",colNames2)]
buildX <- buildData[,colNames2];validX <- validData[,colNames2];buildY <-buildD
ata$classe
table(complete.cases(buildX))
```

```
##
##   TRUE
## 19622
```

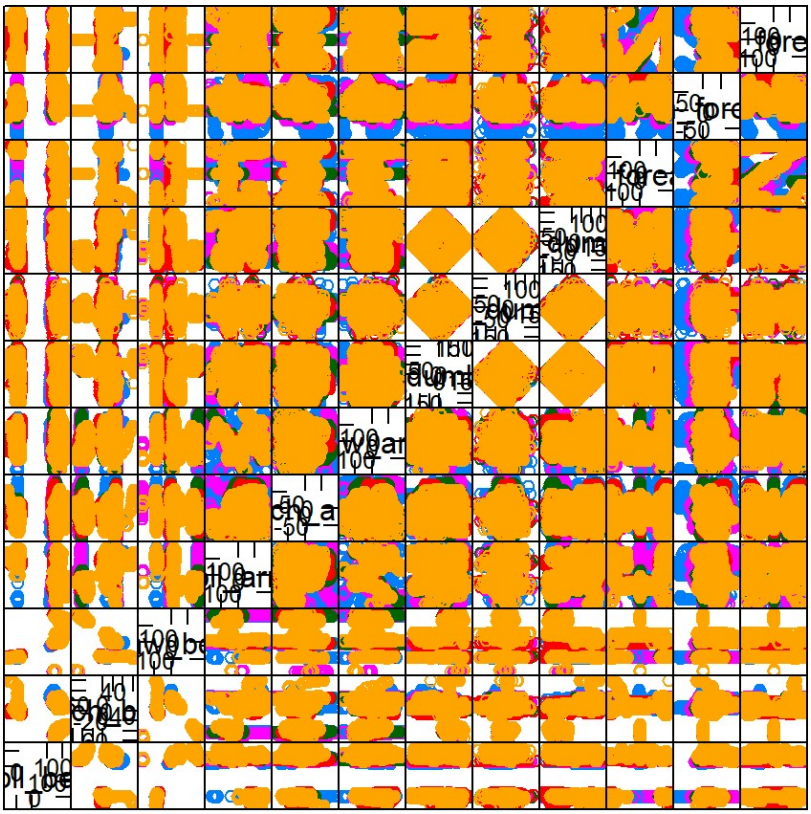
```
table(complete.cases(validX))
```

```
##
## TRUE
##   20
```

Data Exploration

Data exploration did not reveal any obvious trends.

```
buildX1 <- buildX[,grep("^pitch_|^roll_|^yaw_",colNames2)]
featurePlot(buildX1,y=buildY,plot="pairs")
```



Scatter Plot Matrix

Preprocesing and Covariate Creation

The data is normalized for uniformity before using different methods for analysis. Since the feature creation is computationally intensive and require knowledge of the subjet, I did not create any features here. Since all predictors seems to be continuous variables, no dummy variables are created here. Also observed no zero covariates here.

Cross Validation

As the size of the data is big enough, a K-fold cross validation is planned with a K value of 2. This means the Build set is divided into Training set and Testing set (split equally).

```
preObj <- preProcess(x=buildX,method=c("center","scale"))
buildX1 <- as.data.frame(predict(preObj,buildX))
validX1 <- as.data.frame(predict(preObj,validX))
nzvbuild <- nearZeroVar(buildX1,saveMetrics = TRUE)
set.seed(2019-12-29)
sampTrain <- createDataPartition(y=buildY,p=.5,list=F)
trainX2 <- cbind(buildX1[sampTrain,],classe=buildY[sampTrain])
trainX3 <- cbind(buildX1[sampTrain,colNames3],classe=buildY[sampTrain])
testX2 <- cbind(buildX1[-sampTrain,],classe=buildY[-sampTrain])
testX3 <- cbind(buildX1[-sampTrain,colNames3],classe=buildY[-sampTrain])
validX3 <- validX1[,colNames3]
```

Developing ML model

Since target variable is a factor variable, we need to use classification to solve the clustering problem. Random Forests , boosting, Naive Bayes are explored here. Considering all the variables was time consuming and was better than just running only the euler variables (roll,pitch, and yaw). The accuracy of Random Forests was same (0.983) in both cases while gbm accuracy varied (0.96 vs 0.93). Due to time constraints for reproducing this document, only the code for ML model based on features is presented here (just substitute trainX2 in place of trainX3 if you want all variables to be considered).

```
  set.seed(2019-12-29)
start_time <- Sys.time()
modRFs <-train(classe~.,method="rf",data=trainX3,trControl=trainControl(method
="cv",allowParallel=TRUE),number=3)
end_time <-Sys.time()
end_time - start_time
```

```
## Time difference of 4.078367 mins
```

```
start_time <- Sys.time()
modGBMs <-train(classe~.,method="gbm",data=trainX3,trControl=trainControl(metho
d="cv",allowParallel=TRUE),verbose=FALSE)
end_time <-Sys.time()
end_time - start_time
```

```
## Time difference of 2.692039 mins
```

```
start_time <- Sys.time()
suppressWarnings(modNBs <-train(classe~.,method="nb",data=trainX3))
end_time <-Sys.time()
end_time - start_time
```

```
## Time difference of 5.181652 mins
```

```
modRFs$results
```

```
##   mtry  Accuracy     Kappa  AccuracySD     KappaSD
## 1    2 0.9804314 0.9752468 0.003494964 0.004419687
## 2    7 0.9814504 0.9765389 0.003221339 0.004072846
## 3   12 0.9767616 0.9706116 0.002679768 0.003392369
```

```
modGBMs$results
```

```
##   shrinkage interaction.depth n.minobsinnode n.trees  Accuracy     Kappa
## 1       0.1                 1             10      50 0.6776391 0.5913433
## 4       0.1                 2             10      50 0.7901541 0.7352266
## 7       0.1                 3             10      50 0.8513047 0.8122819
## 2       0.1                 1             10     100 0.7346104 0.6650522
## 5       0.1                 2             10     100 0.8529336 0.8144573
## 8       0.1                 3             10     100 0.9019551 0.8761522
## 3       0.1                 1             10     150 0.7667148 0.7055895
## 6       0.1                 2             10     150 0.8847321 0.8545110
## 9       0.1                 3             10     150 0.9286584 0.9098332
##    AccuracySD     KappaSD
## 1 0.014063763 0.017916663
## 4 0.007014493 0.008759577
## 7 0.009466729 0.011868839
## 2 0.009074564 0.011338417
## 5 0.007031165 0.008837973
## 8 0.007180059 0.009054149
## 3 0.009199557 0.011711494
## 6 0.008813089 0.011064951
## 9 0.003049818 0.003826574
```

```
modNBs$results
```

```
##   usekernel fL adjust  Accuracy     Kappa  AccuracySD     KappaSD
## 1     FALSE  0      1 0.4283459 0.2806538 0.005429375 0.007537963
## 2      TRUE  0      1 0.6117743 0.5106865 0.010622335 0.013407825
```

Accuracy of Random Forests and Boosting methods seems to be very good and the ensembling method with majority voting may replace Random Forests in case it does not agree with majority vote. Though the accuracy of Combined model (Model Ensembling) may be similar to that of Random Forests(0.98), it is expected to be a superior method for new data.

We expect the accuracy levels to hold up against cross validation set and expecting around 0.98 accuracy for combined model.

```
predRFs <- predict(modRFs,testX3)
predGBMs <-predict(modGBMs,testX3)
suppressWarnings(predNBs <-predict(modNBs,testX3))
mean(predRFs == testX3$classe)
```

```
## [1] 0.983894
```

```
 mean(predGBMs == testX3$classe)
```

```
## [1] 0.9333333
```

```
  mean(predNBs == testX3$classe)
```

```
## [1] 0.6082569
```

The out of sample accuracy of Random Forests and Boosting methods are good because they have cross validation in their evaluation.

```
predCombs <-predRFs
predCombs[predGBMs == predNBs] <- predGBMs[predGBMs == predNBs]
mean(predCombs==testX3$classe)
```

```
## [1] 0.9586137
```

Though the accuracy of Combined model (Model Ensembling) seems to be less to that of Random Forests(0.98), it is robust and gives consistent results and is the chosen model.

Applying the ML model to Test Validation Data

The combined model is applied to validation data to predict the results of new data.

```
predRF_Vals <- predict(modRFs,validX3)
predGBM_vals <-predict(modGBMs,validX3)
predNB_Vals <-predict(modNBs,validX3)
predComb_Vals <- predRF_Vals
predComb_Vals[predGBM_vals == predNB_Vals ] = predGBM_vals[predGBM_vals == pred
NB_Vals ]
table(predComb_Vals)
```

```
## predComb_Vals
## A B C D E
## 7 8 1 1 3
```

Conclusion

According to the final ML algorithm developed, the weight lifting excercises were performed pefectly (classe=A) in 7 out of 20 test cases (validation cases) while 8 others were good enough (Classe=B).