Data Science/Practical ML exercise

Nikolay Dobrinov

Oct 18, 2017

Executive Summary

"One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The goal of your project is to predict the manner in which they did the exercise. This is the"classe" variable in the training set. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). The training and testing data is located here:

- https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
- https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv "

The data was pre-processed and four calssification methods were tested. Random forests performs best, with 99.54% accuracy on 100 trees. The most important 5 variables in the random forest model are roll.belt, yaw.belt, magnet.dumbbell.z, magnet.dumbbell.y, pitch.belt.

Data pre-processing

The seed is fixed for the purposes of reproducibility. We split the training data into a train and test data sets to cross validate the models. The testing data set provided for this assignment is used as a validation set.

```r
setwd("/Users/nikolaydobrinov/Documents/work/Courses/R/WorkDirectory/Course8_week4_coding_assignments")

library(dplyr)
library(caret)

set.seed(333) # use a seed for replicability

# load data
traintest <- read.csv("./data/pml-training.csv", na.strings = c("NA", ""))
validate <- read.csv("./data/pml-testing.csv", na.strings = c("NA", ""))

# split train-test
inTrain <- createDataPartition(y=traintest$classe, p=0.7, list=FALSE)
train <- traintest[inTrain,]
test <- traintest[-inTrain,]
```

Remove variables that do not seem useful, or it is not clear what they represent. Note that user_name should not be used in clasification as the prediction algorithm should work regardless of the specific user using the device Remove the first 7 columns

```r
train <- select(train, -(1:7))
test <- select(test, -(1:7))
validate <- select(validate, -(1:7))
```

Remove variables with NAs. naVars below reveals that in all variables where NAs exist, about 98% of the observations are NA. We remove all of these features

```
naVars <- sapply(train, function(x) sum(is.na(x)))/nrow(train)
naVarsExclude <- names(naVars[naVars > 0])
train <- train[, !names(train) %in% naVarsExclude]
test <- test[, !names(test) %in% naVarsExclude]
validate <- validate[, !names(validate) %in% naVarsExclude]
```

Check for variables with low variation and remove them. There are no variables with low or zero variance

```
lowVariance <- nearZeroVar(train, saveMetrics=TRUE)
sum(lowVariance$zeroVar) + sum(lowVariance$nzv)
```

```
## [1] 0
```

Analysis

I fit four classification methods - tree, random forest, generalized boosted regression (gbm), and linear discriminant analysis (lda). The corresponding classification accuracy of the models on the testing sampe is: single tree - 49%; random forest with 100 trees - 99.34%; gbm - 96%; lda - 71%. This section presents the results on the lowest error model, random forests, the rest of the models are presented in the Appendix.

The random forest model with 100 trees produces an expected error on the test sample of 0.66%. The variable importance function reveals that across the 100 trees the most important 5 variables are related to the belt and the position of the dumbbel: roll.belt, yaw.belt, magnet.dumbbell.z, magnet.dumbbell.y, pitch.belt. Optimal number of variables to be randomly sampled as candidates at each split is mtry=2.

```
pred.rf <- predict(modFit.rf,test) # predict on test data
confusionMatrix(pred.rf,test$classe) # measure accuracy on test data
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    3    0    0    0
##          B    0 1135    7    0    0
##          C    1    1 1017   19    0
##          D    0    0    2  944    5
##          E    0    0    0    1 1077
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.991, 0.9953)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9965   0.9912   0.9793   0.9954
## Specificity            0.9993   0.9985   0.9957   0.9986   0.9998
## Pos Pred Value         0.9982   0.9939   0.9798   0.9926   0.9991
## Neg Pred Value         0.9998   0.9992   0.9981   0.9959   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1929   0.1728   0.1604   0.1830
## Detection Prevalence   0.2848   0.1941   0.1764   0.1616   0.1832
## Balanced Accuracy      0.9993   0.9975   0.9935   0.9889   0.9976
```

```
varImp(modFit.rf) # variable importance
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## yaw_belt               73.83
## magnet_dumbbell_z      65.35
## pitch_forearm          59.97
## magnet_dumbbell_y      58.36
## pitch_belt             55.70
## magnet_dumbbell_x      51.32
## roll_forearm           48.04
## accel_belt_z           43.22
## roll_dumbbell          41.91
## accel_dumbbell_y       39.94
## magnet_belt_z          39.88
## accel_dumbbell_z       36.20
## magnet_belt_y          35.33
## roll_arm               33.70
## gyros_belt_z           31.27
## accel_forearm_x        28.88
## magnet_arm_x           28.58
## total_accel_dumbbell   28.42
## accel_arm_x            26.77
```

```
print(modFit.rf) # optimal mtry
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
```

```
##    2    0.9877342  0.9844821
##   27    0.9876307  0.9843518
##   52    0.9743812  0.9675907
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```r
predict(modFit.rf,validate) # predict on validation data
```

```
##  [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
```

We already fixed the tuning parameter ntree=100, but we can try another search for the optimal tuning parameter mtry. Below we use a different resampling method - 10 fold cross-validation repeated 3 times. This setup runs much longer, because of the three repeats, and provides a marginal improvement to the accuracy. The expected error is reduced to about 0.46%. The most important variables are the same, however the optimal mtry tuning parameter changes to more than 2 variables.

```r
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")
mtry <- sqrt(ncol(train))
```

```r
pred.rf_random <- predict(modFit.rf_random,test) # predict on test data
confusionMatrix(pred.rf_random,test$classe) # measure accuracy on test data
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    1 1135    4    0    0
##          C    0    0 1020    9    3
##          D    0    0    2  955    5
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9952
##                  95% CI : (0.9931, 0.9968)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.994
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9965   0.9942   0.9907   0.9926
## Specificity            0.9991   0.9989   0.9975   0.9986   1.0000
## Pos Pred Value         0.9976   0.9956   0.9884   0.9927   1.0000
## Neg Pred Value         0.9998   0.9992   0.9988   0.9982   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
```

```
## Detection Rate         0.2843   0.1929   0.1733   0.1623   0.1825
## Detection Prevalence   0.2850   0.1937   0.1754   0.1635   0.1825
## Balanced Accuracy      0.9992   0.9977   0.9958   0.9946   0.9963
```

```r
varImp(modFit.rf_random) # variable importance
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## yaw_belt               67.45
## pitch_forearm          63.01
## magnet_dumbbell_z      53.04
## magnet_dumbbell_y      51.49
## pitch_belt             49.78
## roll_forearm           45.20
## roll_dumbbell          27.98
## magnet_dumbbell_x      26.68
## accel_dumbbell_y       25.26
## magnet_belt_z          23.56
## accel_belt_z           23.25
## magnet_belt_y          21.37
## accel_dumbbell_z       19.29
## accel_forearm_x        19.26
## magnet_forearm_z       18.31
## roll_arm               16.00
## gyros_belt_z           15.16
## total_accel_dumbbell   15.05
## yaw_dumbbell           13.28
```

```r
print(modFit.rf_random) # optimal mtry
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 12364, 12362, 12364, 12363, 12361, 12363, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    8    0.9929389  0.9910675
##    9    0.9929631  0.9910979
##   12    0.9930359  0.9911903
##   25    0.9910460  0.9886728
##   27    0.9905121  0.9879977
##   28    0.9905606  0.9880589
##   29    0.9904391  0.9879053
```

```
##   38    0.9881584  0.9850198
##   41    0.9873817  0.9840372
##   42    0.9873812  0.9840359
##   44    0.9862410  0.9825941
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 12.
```

```
predict(modFit.rf_random,validate) # predict on validation data
```

```
##  [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

The results from the lower performing models for this data set are presented below.

- Decision tree:

```
pred.tree <- predict(modFit.tree,test); confusionMatrix(pred.tree,test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1524  463  488  440  147
##          B   27  376   33  169  156
##          C  119  300  505  355  302
##          D    0    0    0    0    0
##          E    4    0    0    0  477
##
## Overall Statistics
##
##                Accuracy : 0.4897
##                  95% CI : (0.4769, 0.5026)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3331
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9104  0.33011  0.49220   0.0000  0.44085
## Specificity            0.6348  0.91888  0.77856   1.0000  0.99917
## Pos Pred Value         0.4977  0.49409  0.31942      NaN  0.99168
## Neg Pred Value         0.9469  0.85109  0.87895   0.8362  0.88805
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2590  0.06389  0.08581   0.0000  0.08105
## Detection Prevalence   0.5203  0.12931  0.26865   0.0000  0.08173
## Balanced Accuracy      0.7726  0.62450  0.63538   0.5000  0.72001
```

- Generalized boosted regression (gbm):

```
pred.gbm <- predict(modFit.gbm,test); confusionMatrix(pred.gbm,test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1647   37    0    1    2
##          B   16 1067   28    1   19
##          C    7   32  980   27    9
##          D    4    3   16  933   24
##          E    0    0    2    2 1028
##
## Overall Statistics
##
##                Accuracy : 0.9609
##                  95% CI : (0.9556, 0.9657)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9506
##  Mcnemar's Test P-Value : 3.597e-10
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9368   0.9552   0.9678   0.9501
## Specificity            0.9905   0.9865   0.9846   0.9904   0.9992
## Pos Pred Value         0.9763   0.9434   0.9289   0.9520   0.9961
## Neg Pred Value         0.9936   0.9849   0.9905   0.9937   0.9889
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1813   0.1665   0.1585   0.1747
## Detection Prevalence   0.2867   0.1922   0.1793   0.1665   0.1754
## Balanced Accuracy      0.9872   0.9617   0.9699   0.9791   0.9746
```

- Linear discrimination analysis:

```
pred.lda <- predict(modFit.lda,test); confusionMatrix(pred.lda,test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1386  173  106   42   29
##          B   41  728   99   34  200
##          C  118  136  664  112   80
##          D  125   52  128  733  121
##          E    4   50   29   43  652
##
## Overall Statistics
##
```

```
##                Accuracy : 0.7074
##                  95% CI : (0.6956, 0.719)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6298
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8280   0.6392   0.6472   0.7604   0.6026
## Specificity            0.9169   0.9212   0.9082   0.9134   0.9738
## Pos Pred Value         0.7984   0.6606   0.5982   0.6324   0.8380
## Neg Pred Value         0.9306   0.9141   0.9242   0.9511   0.9158
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2355   0.1237   0.1128   0.1246   0.1108
## Detection Prevalence   0.2950   0.1873   0.1886   0.1969   0.1322
## Balanced Accuracy      0.8724   0.7802   0.7777   0.8369   0.7882
```