Practical Machine Learning Course Project

Mats

5 september 2019

Executive summary

The project goal is to make a prediction of how the people in the data set performed on a specific training exercise. The data set consists of data from six (6) participants. The data comes from accelerometers on the belt, forearm, arm, and dumbbell and is measured when they have performed training exercises during a specific period of time. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Three machine learning algorithms: *Decision tree*, *Generalized boosted regression* and *Random forest*, were evaluated to find the one with the best accuracy for this specific use case. In the training set the variable named *classe* is used to specify how well the participants carried out the exercise, from A (the best) to E (the worst). This was the variable used to model as a function of all the other variables in the algorithms which were evaluated. Random forest was found to have the highest accuracy (over 99 %). This was used to predict how well the training exercises in the test set (20 exercise sessions) were carried out.

Preparation

Load packages

```
library(caret)
library(rpart)
library(lattice)
library(ggplot2)
```

Download and load data set

```
urlTraining <- url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
training <- read.csv(urlTraining, header = TRUE)

urlTest <- url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
test <- read.csv(urlTest, header = TRUE)
```

Clean data set

The training and test sets initially had the following dimensions:

```
dim(training)
```

```
## [1] 19622   160
```

```
dim(test)
```

```
## [1]  20 160
```

However, they contain many variables with NA values which are not of use or could cause misleading results. Also, the first seven columns contain information which is not useful in our model, for example person names (persons doing the exercise) and time stamps.

```
removeColumns <- which(colSums(is.na(training) | training == "")
                        > 0.9 *dim(training)[1])
training <- training[, -removeColumns]
training <- training[, -c(1:7)]

removeColumns <- which(colSums(is.na(test) | test == "")
                        > 0.9 *dim(test)[1])
test <- test[, -removeColumns]
test <- test[, -1]
```

After removing variables with NA values and other unnecessary variables, the training and test sets now have the following dimensions.

```
dim(training)
```

```
## [1] 19622    53
```

```
dim(test)
```

```
## [1] 20 59
```

Machine learning algorithm evaluation

We start with splitting the training data set into two parts: *trainPart* and *testPart*.

```
set.seed(123456)
dataPartition <- createDataPartition(training$classe, p = 0.7, list = FALSE)
trainPart <- training[dataPartition,]
testPart <- training[-dataPartition,]
```

Three commonly used algorithms are chosen for evaluation: *Decision tree*, *Generalized boosted regression* and *Random forest*.

In the training, cross-validation (cv) was used with five (5) folds. After the training, the test part of the initial training data set is used to make the predictions and a confusion matrix is used to control the accuracy when using the different algorithms.

Decision tree

```
rpartModel <- train(classe ~ ., data = trainPart, method = "rpart", trControl = trainControl(method = '
rpartModel
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10991, 10988, 10989, 10991, 10989
## Resampling results across tuning parameters:
##
##   cp         Accuracy   Kappa
##   0.0320415  0.5273342  0.38640038
##   0.0603533  0.4156568  0.20828705
##   0.1156546  0.3322253  0.07292141
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0320415.
```

```
rpartPredTest <- predict(rpartModel, testPart)
rpartAccuracyTest <- confusionMatrix(rpartPredTest, testPart$classe)
rpartAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1517  456  468  429  152
##          B   38  416   50  175  180
##          C  113  267  508  360  264
##          D    0    0    0    0    0
##          E    6    0    0    0  486
##
## Overall Statistics
##
##                Accuracy : 0.4974
##                  95% CI : (0.4845, 0.5102)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3434
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9062  0.36523  0.49513   0.0000  0.44917
## Specificity            0.6426  0.90666  0.79337   1.0000  0.99875
## Pos Pred Value         0.5020  0.48428  0.33598      NaN  0.98780
## Neg Pred Value         0.9452  0.85615  0.88155   0.8362  0.88949
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2578  0.07069  0.08632   0.0000  0.08258
## Detection Prevalence   0.5135  0.14596  0.25692   0.0000  0.08360
## Balanced Accuracy      0.7744  0.63595  0.64425   0.5000  0.72396
```

```
rpartAccuracyTest$overall[1]
```

```
##  Accuracy
## 0.4973662
```

Generalized boosted regression

```
gbmModel <- train(classe ~., data = trainPart, method="gbm", trControl = trainControl(method = 'cv', nu
gbmModel
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10990, 10988, 10990, 10991
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7506757  0.6839550
##   1                  100      0.8204127  0.7727389
##   1                  150      0.8564467  0.8183573
##   2                   50      0.8583392  0.8205262
##   2                  100      0.9088597  0.8846433
##   2                  150      0.9329551  0.9151585
##   3                   50      0.8972840  0.8699461
##   3                  100      0.9433644  0.9283327
##   3                  150      0.9617813  0.9516504
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
gbmPredTest <- predict(gbmModel, testPart)
gbmAccuracyTest <- confusionMatrix(gbmPredTest, testPart$classe)
gbmAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1645   43    0    0    1
##          B   17 1070   34    1    6
##          C    8   23  980   39    7
##          D    3    1   10  922   20
##          E    1    2    2    2 1048
##
## Overall Statistics
##
##                Accuracy : 0.9626
##                  95% CI : (0.9575, 0.9673)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9527
##
##  Mcnemar's Test P-Value : 2.285e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9827   0.9394   0.9552   0.9564   0.9686
## Specificity            0.9896   0.9878   0.9842   0.9931   0.9985
## Pos Pred Value         0.9739   0.9486   0.9272   0.9644   0.9934
## Neg Pred Value         0.9931   0.9855   0.9905   0.9915   0.9930
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2795   0.1818   0.1665   0.1567   0.1781
## Detection Prevalence   0.2870   0.1917   0.1796   0.1624   0.1793
## Balanced Accuracy      0.9861   0.9636   0.9697   0.9748   0.9836
```

```
gbmAccuracyTest$overall[1]
```

```
##  Accuracy
## 0.9626168
```

Random forest

```
randomFModel = train(classe ~., data = trainPart, method = 'rf', trControl = trainControl(method = 'cv'
randomFModel
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10991, 10989, 10988, 10990, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9904639  0.9879366
##   27    0.9904642  0.9879365
##   52    0.9826754  0.9780827
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
rfPredTest <- predict(randomFModel, testPart)
rfAccuracyTest <- confusionMatrix(rfPredTest, testPart$classe)
rfAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674   12    0    0    0
##          B    0 1125    7    0    0
##          C    0    2 1018   10    1
##          D    0    0    1  953    5
##          E    0    0    0    1 1076
##
## 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            1.0000   0.9877   0.9922   0.9886   0.9945
## Specificity            0.9972   0.9985   0.9973   0.9988   0.9998
## Pos Pred Value         0.9929   0.9938   0.9874   0.9937   0.9991
## Neg Pred Value         1.0000   0.9971   0.9984   0.9978   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1912   0.1730   0.1619   0.1828
## Detection Prevalence   0.2865   0.1924   0.1752   0.1630   0.1830
## Balanced Accuracy      0.9986   0.9931   0.9948   0.9937   0.9971
```

```
rfAccuracyTest$overall[1]
```

```
## Accuracy
## 0.993373
```

Conclusion - algorithm selection

When comparing the outcome of the three algorithms evaluated, we clearly see that *Random forest* has the highest performance. It has the highest accuracy, over 99 %. Second out is *Generalized boosted regression* with an accuracy of close to 96 %. *Decision tree* has an outcome which is significantly lower than the other two, approximately 50 %.

Prediction on the test set

The Random forest model is now used for making a prediction on the test set to predict how well the training exercises in the test set (20 exercise points) were carried out.

```
predict(randomFModel, 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
```