Practical Machine Learning. Prediction Assignment

Sergey Bogdanov

Mar 12, 2019

Abstract

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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 report data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants are used to predict 5 different ways of perform barbell lifts correctly and incorrectly. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Several different type of prediction algorithm were analyzed and compared each other in report.
In conclusion Random Forest method was used to predict manner of barbell lifts on test dataset.

Read and clean data

At preparing phase data was read and variables that cannot be used for prediction was removed.

```
# load data from files
pml_training <- read_csv("pml-training.csv"); pml_testing <- as.data.table( read_csv("pml-testing.csv")
 )
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Warning: 182 parsing failures.
##  row               col expected  actual               file
## 2231 kurtosis_roll_arm a double #DIV/0! 'pml-training.csv'
## 2231 skewness_roll_arm a double #DIV/0! 'pml-training.csv'
## 2255 kurtosis_roll_arm a double #DIV/0! 'pml-training.csv'
## 2255 skewness_roll_arm a double #DIV/0! 'pml-training.csv'
## 2282 kurtosis_roll_arm a double #DIV/0! 'pml-training.csv'
## .... ................. ........ ....... ..................
## See problems(...) for more details.
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
# remove text "#DIV/0!"
for (i in names(pml_training)) { t<-  pml_training[,i];  t[t == "#DIV/0!"] <- NA;  pml_training[,i] <- t
}
# remove empty varibales of test dataset from train dataset
t<- apply(pml_testing, 2, function(X) mean(!is.na(X)*1) ) > 0.0;
pml_training <- pml_training[t]
# remove row headers and timestamps that can't be used for prediction
pml_training <- pml_training[,-c(1:7)]
# convert target variable to the factor type
pml_training$classe <- factor(pml_training$classe); pml_training <- as.data.table(pml_training)
rm("i","t")
```

Model development

Split data for development and validation datasets

First of all let`s split data.
Training data will be used for model developing and testing part will be used to validate model accuracy.

```
set.seed(123)
# create a partition vector
inTrain <- createDataPartition(y=pml_training$classe, p=0.60, list=FALSE)
# subset data to training
training <- pml_training[inTrain,]
# subset data (the rest) to test
testing <- pml_training[-inTrain,]
```

Let’s try several algorithm, that can be used for classification task and compare results.

Prediction with Trees

```
tree <- train(classe ~ ., data=training, method="rpart")
# save model accuracy
treeAcc <- round(confusionMatrix(testing$classe, predict(tree, newdata=testing))[[3]][[1]], 4)
```

Bagging

```
bag <- train(classe ~ ., data=training, method="treebag")
# save model accuracy
bagAcc <- round(confusionMatrix(testing$classe, predict(bag, newdata=testing))[[3]][[1]], 4)
```

Random Forest

```
rf <- randomForest(classe ~ ., data=training)
# save model accuracy
rfAcc <- round(confusionMatrix(testing$classe, predict(rf, newdata=testing))[[3]][[1]], 4)
```

Boosting

```
gbm <- train(classe ~ ., data=training, method="gbm", verbose=F)
# save model accuracy
gbmAcc <- round(confusionMatrix(testing$classe, predict(gbm, newdata=testing))[[3]][[1]], 4)
```

Linear Discriminant Analysis

```
lda <- train(classe ~ ., data=training, method="lda", verbose=F)
# save model accuracy
ldaAcc <- round(confusionMatrix(testing$classe, predict(lda, newdata=testing))[[3]][[1]], 4)
```

Compare results

```
modelAccuracyCompare <- data.frame( "DecisionTree" =  treeAcc, "Bagging" = bagAcc, "Boosting" = gbmAcc,
                                    "RandomForest" = rfAcc, "LDA" = ldaAcc)
modelAccuracyCompare
```

```
##   DecisionTree Bagging Boosting RandomForest    LDA
## 1       0.5069  0.9847   0.9629       0.9943 0.7056
```

You can see that Random Forest model gives best accuracy on validation dataset

Combining Predictors

Let's try to improve model accuracy by combine classifiers.

```
# predict in training dataset
tree.pred.train <- predict(tree,training)
bag.pred.train <- predict(bag,training)
rf.pred.train <- predict(rf,training)
gbm.pred.train <- predict(gbm,training)
lda.pred.train <- predict(lda,training)
# combine the prediction results and the true results into new data frame
combinedData <- data.frame(tree.pred = tree.pred.train, bag.pred = bag.pred.train, rf.pred = rf.pred.tra
in,
        gbm.pred =gbm.pred.train , lda.pred = lda.pred.train, classe=training$classe)
# run Random Forest model on the combined test data
comb.fit <- train(classe ~ .,method="rf",data=combinedData)
# use the models to predict results on the testing dataset
tree.pred.test <- predict(tree,testing)
bag.pred.test <- predict(bag,testing)
rf.pred.test <- predict(rf,testing)
gbm.pred.test <- predict(gbm,testing)
lda.pred.test <- predict(lda,testing)
# combine the results into data frame for the comb.fit
combinedTestData <- data.frame(tree.pred = tree.pred.test, bag.pred = bag.pred.test, rf.pred = rf.pred.t
est,
        gbm.pred =gbm.pred.test , lda.pred = lda.pred.test, classe=testing$classe)
# run the comb.fit on the combined testing data
comb.pred.test <- predict(comb.fit,combinedTestData)
# save model accuracy
combAcc <- round(confusionMatrix(testing$classe, comb.pred.test)[[3]][[1]], 4)
```

Final model selection and conclusions

```
modelAccuracyCompare <- cbind(modelAccuracyCompare, "CombinedModel"= combAcc)
modelAccuracyCompare
```

```
##   DecisionTree Bagging Boosting RandomForest    LDA CombinedModel
## 1       0.5069  0.9847   0.9629       0.9943 0.7056        0.9908
```

You can see that simple models ensembling do not improve accuracy.

In conclusion we can point that Random Forest model gives best accuracy and we use it to predict on test dataset in "pml_testing" variable.

Prediction on test dataset

```
    result <- cbind( pml_testing[ ,.(problem_id)], "classe"= predict(rf, newdata=pml_testing) )
    result
```

```
##     problem_id classe
##  1:          1      B
##  2:          2      A
##  3:          3      B
##  4:          4      A
##  5:          5      A
##  6:          6      E
##  7:          7      D
##  8:          8      B
##  9:          9      A
## 10:         10      A
## 11:         11      B
## 12:         12      C
## 13:         13      B
## 14:         14      A
## 15:         15      E
## 16:         16      E
## 17:         17      A
## 18:         18      B
## 19:         19      B
## 20:         20      B
```