Activity Correctness Prediction

Tomas Gogorza

December 8th, 2015

About the Data Set

The data used in this project was provided by the Human Activity Recognition project and contains information about weight lifting exercises performed by 6 different individuals. Each performance was measured by 4 different sensors. The labels in the data set indicate whether the exercise was completed according to the specifications or if there was any kind of incorrect movement.

Load and Clean Data Set

After loading the training and testing data sets and performing some exploratory analysis, there are many properties with missing data which can be removed to enhance our predictive model. Given the large number of NA values, it was decided not impute any missing data, and instead just remove the variables. The first 7 columns containing information not relevant to our prediction purposes are also removed.

```
trainData <- read.csv("pml-training.csv")
testData <- read.csv("pml-testing.csv")

#Remove fields with more than 19000 NAs and empty data
classe <- trainData$classe
trainData <- select(trainData,-classe)
trainData <- trainData[ , apply(trainData,2,function(x){ sum(is.na(x)) < 19000 })]
trainData <- trainData[ , apply(trainData,2,function(x){ !("#DIV/0!" %in% x) })]
trainData <- trainData[ , -c(1:7)]

testData <- select(testData,-problem_id)
testData <- testData[ , apply(testData,2,function(x){ sum(is.na(x)) != 20 })]
testData <- testData[ , apply(testData,2,function(x){ !("#DIV/0!" %in% x) })]
testData <- testData[ , -c(1:7)]
```

After removing irrelevant features, we are only left with raw sensors data, with a total of 52 input variables.

Pre-Processing / Normalization

After removing unwanted parameters, we center and scale all our variables to prevent our predictive model from being affected by skewness of the data as well as high variability. After normalizing the training data, some dimensionality reduction is performed by means of Principal Component Analysis to reduce the number of predictors and also the noise that could be added by non relevant features.

```
set.seed(321456)
#Features centering and scaling, and principal component analysis
preProc <- c("center","scale","pca")

#Create Cross Validation set
```

```
trainInd <- createDataPartition(classe, p = 0.8, list = FALSE)
validationData <- trainData[-trainInd,]
validationLabels <- classe[-trainInd]
trainData <- trainData[trainInd,]
trainLabels <- classe[trainInd]
```

After applying some pre-processing we slice the training set and create a small validation set so we can compare performance of the trained models.

Predictive Models

We'll be training 2 predictive models to compare performance and we'll take the most accurate one to predict the test cases. The first model is a Gradient Boosting Machine and the second one will be a Random Forest. In both cases we'll use repeated cross validation with 5 repetitions as training control. For speedup purposes, we'll be using 4 CPU cores to train the models.

```
#Parallel Training
cl <- makeCluster(4, type = "SOCK")
registerDoSNOW(cl)
#Perform cross validation on training
fitControl <- trainControl(method="repeatedcv", repeats = 5, allowParallel = TRUE)
#Gradient Boosting model
gbmFit <- train(x=trainData, y=trainLabels,
                method = "gbm",
                trControl = fitControl,
                preProcess= preProc,
                verbose = FALSE)
save(gbmFit,file = "gbmFit.Rmodel")
#Random Forest model
rfFit <- train(x=trainData, y=trainLabels,
                method = "rf",
                trControl = fitControl,
                preProcess= preProc,
                verbose = FALSE)
save(rfFit,file = "rfFit.Rmodel")

stopCluster(cl)
```

Prediction Performance

After training the models we use the validation data we took out of the training set to measure the accuracy of the GBM and RF models.

```
load(file = "gbmFit.Rmodel")
predvgbm <- predict(gbmFit,newdata = validationData)
cmgbm <- confusionMatrix(predvgbm,validationLabels)
cmgbm
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   A   B   C   D   E
##          A 999  86  29  19  15
##          B  23 572  48  12  49
##          C  36  63 566  68  51
##          D  49  16  28 527  30
##          E   9  22  13  17 576
##
## Overall Statistics
##
##                Accuracy : 0.8259
##                  95% CI : (0.8137, 0.8376)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7797
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8952   0.7536   0.8275   0.8196   0.7989
## Specificity            0.9469   0.9583   0.9327   0.9625   0.9809
## Pos Pred Value         0.8702   0.8125   0.7219   0.8108   0.9042
## Neg Pred Value         0.9578   0.9419   0.9624   0.9646   0.9559
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2547   0.1458   0.1443   0.1343   0.1468
## Detection Prevalence   0.2926   0.1795   0.1998   0.1657   0.1624
## Balanced Accuracy      0.9210   0.8560   0.8801   0.8910   0.8899
```

```
load(file = "rfFit.Rmodel")
predvrf <- predict(rfFit,newdata = validationData)
cmrf <- confusionMatrix(predvrf,validationLabels)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114   12    0    1    0
##          B    0  738    8    0    0
##          C    2    8  668   21    5
##          D    0    0    6  619    5
##          E    0    1    2    2  711
##
## Overall Statistics
##
##                Accuracy : 0.9814
##                  95% CI : (0.9767, 0.9854)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9765
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9723   0.9766   0.9627   0.9861
## Specificity            0.9954   0.9975   0.9889   0.9966   0.9984
## Pos Pred Value         0.9885   0.9893   0.9489   0.9825   0.9930
## Neg Pred Value         0.9993   0.9934   0.9950   0.9927   0.9969
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1881   0.1703   0.1578   0.1812
## Detection Prevalence   0.2873   0.1902   0.1795   0.1606   0.1825
## Balanced Accuracy      0.9968   0.9849   0.9827   0.9797   0.9923
```

We can see that the Random Forest turns out to be a more accurate predictor with a very high accuracy of **98.14%** against a still pretty nice **82.59%** given by the GBM predictor. After this comparison we select the Random Forest model for our final prediction, so this means our expected **out of sample accuracy** should be under **98.14%**, since out of sample error is always larger than in sample error.

Final Prediction

Finally we use our trained Random Forest to predict the test set. This is the final result:

```
testPred <- predict(rfFit,newdata = testData)
pml_write_files(testPred)
testPred
```

```
##  [1] B A B A A E D B A A A C B A E E A B B B
## Levels: A B C D E
```

We can compare the predicted results to the actual classes:

```
actualLabels <- factor(c("B","A","B","A","A","E","D","B","A","A",
                         "B","C","B","A","E","E","A","B","B","B"))
actualLabels
```

```
##  [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
```

```
#Get prediction accuracy
precision <- sum(testPred == actualLabels) / length(actualLabels)
precision
```

```
## [1] 0.95
```

Our predictive model manages to predict 19 out of 20 classes correctly, giving out a **95%** precision, which is very close to the expected out of sample prediction of **98.14%** we found when doing cross validation.