Practical Machine Learning - Human Activity Recognition from sensor data

Vincent

13 janvier 2019

Summary

This report uses data from http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. This is sensor data from wearable devices and the goal is to predict a class of human behaviour from this sensor data. After loading and cleaning the data we evaluate several models with a 5-fold cross validation, select the models with the best accuracy and combine their predictions with a majority vote to make our final prediction on the test data.

Getting, Reading and cleaning data

The data is available in two files, one for the training set and the other for the testing set.

```
if (!file.exists("pml-training.csv")) {download.file("https://d396qusza40orc.cloudfront.net/predmachlea
if (!file.exists("pml-testing.csv")) {download.file("https://d396qusza40orc.cloudfront.net/predmachlearn
```

```
train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")
```

This data contains a lot of missing values. We could be tempted to keep only records without missing values, but we would lose most of the data.

```
print(paste(sum(complete.cases(train)), '/', nrow(train)))
```

```
## [1] "406 / 19622"
```

We could also impute values, using the mean of the column, to fill those missing values. However in theses columns values are missing for the vast majority of the observations, so it makes more sense to remove them altogether.

```
sum(is.na(train$max_roll_belt)) / nrow(train)
```

```
## [1] 0.9793089
```

```
missingtrain <- apply(train, 2, function(col) {sum(is.na(col))})
missingtest <- apply(test, 2, function(col) {sum(is.na(col))})
train <- train[,missingtrain == 0 & missingtest == 0]
test <- test[,missingtrain == 0 & missingtest == 0]
```

We also remove the first seven columns because they do not contain sensor data and should not be used as predictors. We transform the columns that have been loaded as factors into numerical values.

```
train <- train[,8:ncol(train)]
for(i in 1:(ncol(train)-1)) {
  train[, i] <- as.numeric(train[,i])
}
test <- test[,8:ncol(test)]
for(i in 1:(ncol(test)-1)) {
  test[, i] <- as.numeric(test[,i])
}
```

At this point we could split our train data to reserve a cross validation dataset in order to evaluate our models without using the testing data. However, we will let the caret package do the cross validation for us. We will also enable parallel computation because building the model can be very time comsuming, particularly with random forests.

Enable parallel computation

```
library(caret)
library(parallel)
library(doParallel)
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
fitControl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)
```

Random Forest

```
fit_rf <- train(classe ~ ., data = train, method = "rf", trControl = fitControl)
predictTest <- predict(fit_rf, test)
```

```
confusionMatrix.train(fit_rf)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.4  0.1  0.0  0.0  0.0
##          B  0.0 19.2  0.1  0.0  0.0
##          C  0.0  0.0 17.3  0.2  0.0
##          D  0.0  0.0  0.0 16.1  0.0
##          E  0.0  0.0  0.0  0.0 18.4
##
##  Accuracy (average) : 0.9946
```

The Accuracy of the random forest algorithm is excellent.

lda

```
fit_lda <- train(classe ~ ., data = train, method = "lda", trControl = fitControl)
predictlda <- predict(fit_lda, test)
```

```
confusionMatrix.train(fit_lda)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 23.3  3.0  1.7  1.0  0.7
##          B  0.6 12.4  1.7  0.7  3.1
##          C  2.3  2.3 11.4  1.9  1.7
##          D  2.2  0.7  2.2 12.1  1.8
##          E  0.1  0.9  0.4  0.7 11.1
##
##  Accuracy (average) : 0.7019
```

LDA has a much lower accuracy with only 70% compared to more than 99% for Random Forest. Since Random Forest seems wery well fitted to the problem, we can explore other implemantation of the same algorithm available in the caret package.

Random Forest 2

```
fit_rf2 <- train(classe ~ ., data = train, method = "Rborist", trControl = fitControl)
predictTest2 <- predict(fit_rf2, test)
```

```
confusionMatrix.train(fit_rf2)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 27.6  0.0  0.0  0.0  0.0
##          B  0.3 19.2  0.0  0.0  0.0
##          C  0.1  0.1 17.3  0.1  0.0
##          D  0.1  0.0  0.1 16.3  0.0
##          E  0.4  0.0  0.0  0.0 18.3
##
##  Accuracy (average) : 0.9874
```

Random Forest 3

```
fit_rf3 <- train(classe ~ ., data = train, method = "ranger", trControl = fitControl)
1
```

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

```
predictTest3 <- predict(fit_rf3, test)
```

```
confusionMatrix.train(fit_rf3)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 28.4  0.1  0.0  0.0  0.0
##          B  0.0 19.3  0.1  0.0  0.0
##          C  0.0  0.0 17.3  0.1  0.0
##          D  0.0  0.0  0.0 16.2  0.0
##          E  0.0  0.0  0.0  0.0 18.4
##
##  Accuracy (average) : 0.9962
```

Stop parallel computation

```
stopCluster(cluster)
registerDoSEQ()
```

Predictions

Each of our 3 Ramdom Forest models has an accuracy of 99%. We will base our final prediction on a majority vote from our 3 Random Forest models and we can expect a near perfect accuracy.

```
predictions <- data.frame(predictTest, predictTest2, predictTest3)
library(prettyR)
result <- apply(predictions, 1, Mode)
names(result) <- 1:20
result
```

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

Conclusion

The random forest algorithm proved to be very effective on this dataset with an accuracy estimated from a 5-fold cross validation around 99%. By combining the predictions of 3 different implementations of random forests with a majority vote, we aim to improve the global accuracy. This approach led to a 100% accuracy on the test set.