Using Machine Learning to predict Human movement

Rebecca Kitching

30/07/2018

Background

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 project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.** These are coded as A-E in a variable labeled 'classe'.

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).

Data Cleaning and Preprocessing

We first want to download the data sets and import them into the session.

```
# Set up working directory wgere information is stored.
setwd('/Users/rek514/Desktop/Coursera/Machine Learning/Assignment')
# Lood the training and test data sets
training <- read.csv('pml-training.csv',na.strings=c("NA","#DIV/0!", ""))
testing <- read.csv('pml-testing.csv',na.strings=c("NA","#DIV/0!", ""))
```

The data in its present state are messy, containing irrelevant variables and missing values, so we need to first clean the data up. Here ee remove the identifying user information as well as the time/data stamps. We also remove any variables which have lots of NA values and those that show zero variance.

```
# Removes variables with NA values and zero variance.
removevar <- matrix(1:7, nrow=1)
for (variable in 8:dim(training)[2]){
  if (sum(is.na(training[,variable]))>100) {
    removevar <- cbind(removevar, c(variable))}
  else if (var(as.numeric(training[,variable]))==0){
    removevar <- cbind(removevar, c(variable))}
}
training <- training[,-removevar]
testing <- testing[,-removevar]
```

Above, we have therefore removed `107` variables in total leaving `54` variables, including the classification variable. The next thing to do is to partition the training data set to give a validation set (20% training data) - this is used to validate the model and provide an out-sample error rate.

```
inTrain <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
training <- training[inTrain,]
validating <- training[-inTrain,]
```

Random Forest Modelling

We now want to perform a random forest model based on the training data set and use the remaining 53 variables to classify the movement type. Random forest was chosen as it is equipt to deal with the large number of remaining variables and will limit overfitting often seen in decision tree models (although calculation speed will be slower). As we can see below, the model is unsuprisingly accurate at classified all the data points into their correct classe.

```
modfit <- randomForest(classe ~., data=training)
table(predict(modfit, training), training$classe)
```

```
##
##        A    B    C    D    E
##   A 4185    0    0    0    0
##   B    0 2848    0    0    0
##   C    0    0 2567    0    0
##   D    0    0    0 2412    0
##   E    0    0    0    0 2706
```

The next thing to do is to use the same model we've just created based on the training data set and apply it to the validation data set. This will then give an out-sample error term to show how accurate the model is. As we can see below, the model has accuratly classified all of the 3692 cases in validating data set with an out-sample accuracy of 1, suggesting the model is significantly better at classifying the exercise types from the fitness band data compared to chance.

```
valid_prediction <- predict(modfit, validating)
confusionMatrix(valid_prediction, validating$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1080    0    0    0    0
##          B    0  664    0    0    0
##          C    0    0  660    0    0
##          D    0    0    0  606    0
##          E    0    0    0    0  682
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.2925
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2925   0.1798   0.1788   0.1641   0.1847
## Detection Rate         0.2925   0.1798   0.1788   0.1641   0.1847
## Detection Prevalence   0.2925   0.1798   0.1788   0.1641   0.1847
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Test predictions

The final part of the analysis is to now predict the exercise class of new data which we do not know the class of. We do this in a similar way to above by using the initial random forest model but on the testing data set instead.

```
predict(modfit, testing)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```