Human Activity Recognition

S. Stefansen

27 January 2016

Summary

A Random Forest model was fit to training data and validated on testing data, yielding an accuracy of .993. Because the Random Forest method minimises bias, the high accuracy achieved on the test data is indicative of a very low out of sample error rate.

Design

A *Random Forest* model was used to classify activity performed by six participants. The classifation was fit to 5 classes of physical exercise *[A, B, C, D, E]*. The training data set consisted of 19622 rows of data with 160 variables. The *Random Forest* model was chosen because it is a good fit for prediting multiple classes - in comparison to linear models which tend to be better at predicting continous variables or binary classification. **It also has the advantage of automatically performing cross-validation.** Training and validation data sets were produced from the original *training.csv* file. The fit was made on the training set, and the results reported are from the fit applied to the validation set.

Method

This section first describes the steps performed on the data for this report. This is followed by the actual code used to achieve those steps.

- The data was downloaded from the server specified in the assignment. It was loaded into R using *read.csv* with *stringsAsFactors* argument set to *FALSE*.
- The following pre-processing was performed:

1. The data set was pruned for variables containing empty values. This was achieved through a custom function, though using caret's *nearZeroVar* function would have yielded a similar result.
2. Rows with *new_window* set to *yes* were removed, as the final test set does not contain such rows.
3. Variables pertaining to time, windows and row-id were removed. This was done as no time-sensitive analysis was performed.
4. The *classe* and *user_name* variables were cast as a factor variables.
5. The training data was split into two data sets. These were used to train and validate the fit. Since *Random Forests* was used, this step was not technically necessary since this method yields an unbiased result. It was done to enable the testing of the fit on fresh data, before applying it to the final test set.

- A fit was made on the training set using the *randomForest* function of the package of the same name.
- **A separate cross-validation step was not needed since the random forest method inherently performs many folds of the data while building its trees.**

Load necessary packages

```
library(caret)
library(dplyr)
library(randomForest)
```

Load the data from local drive:

```
set.seed(500)
setwd("~/Projects/CDS/Course8/Assignment")
dat <- read.csv("training.csv", stringsAsFactors = FALSE)
```

Pre-processing steps 1-3:

```
isMissing <- function(x) {
    if (is.na(x)) return (TRUE)
    if (x=="") return (TRUE)
    return (FALSE)
}

isComplete <- c()
for (field in names(dat)) {
    missingCount <- sum(sapply(dat[, field], function(x) isMissing(x)))
    if (missingCount==0) isComplete <- c(isComplete, field)
}

dat <- dat[dat$new_window=="no", ]
dat <- dat[, isComplete]
dat <- dat[, -c(1, 3:7)] #remove id, time and window vars
```

Pre-processing step 4: Set factor variables

```
dat$user_name <- as.factor(dat$user_name)
dat$classe <- as.factor(dat$classe)
```

Pre-processing step 5: Split into training and validation data sets

```
inTrain <- createDataPartition(y=dat$classe, p=3/4, list = FALSE)
training <- dat[inTrain,]
testing <- dat[-inTrain,]
```

Perform fit and prediction

```
modelFit <- randomForest(x=training[,-54], y=training[, 54], ntree=1000)
pred <- predict(modelFit, newdata = testing[, -54])
```

Results

Fitting the data on the random forest algorithm yielded the following results.

```
confusionMatrix(testing$classe, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1367    0    0    0    0
##          B    8  921    0    0    0
##          C    0    6  831    1    0
##          D    0    0   17  769    0
##          E    0    0    0    2  880
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9901, 0.9951)
##     No Information Rate : 0.2863
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9942   0.9935   0.9800   0.9961   1.0000
## Specificity            1.0000   0.9979   0.9982   0.9958   0.9995
## Pos Pred Value         1.0000   0.9914   0.9916   0.9784   0.9977
## Neg Pred Value         0.9977   0.9985   0.9957   0.9993   1.0000
## Prevalence             0.2863   0.1930   0.1766   0.1608   0.1833
## Detection Rate         0.2847   0.1918   0.1731   0.1601   0.1833
## Detection Prevalence   0.2847   0.1935   0.1745   0.1637   0.1837
## Balanced Accuracy      0.9971   0.9957   0.9891   0.9959   0.9997
```

The error rate is very low on the validation data set, indicating the fit will do well on out of sample instances drawn from the same population.