Prediction of Barbell Lifting Technique using Wearable Sensor Data

Alex Van Russelt

28 December 2017

Setup

```
library(caret)
library(dplyr)
library(knitr)
library(parallel)
library(doParallel)
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
set.seed(1234)
```

Introduction

The Human Activity Recognition (HAR) dataset contains data from wearable sensors where the user was performing barbell lifts. The `classe` variable contains a label describing the method used to lift the barbell.

The objective of this document is to develop a model that could be used to predict the `classe` variable from the sensor information, and to apply it to 20 unseen observations.

Exploration

```
training <- read.csv('pml-training.csv', stringsAsFactors = FALSE,
                     na.strings=c('NA', ''))
testing <- read.csv('pml-testing.csv', stringsAsFactors = FALSE)
training$classe <- as.factor(training$classe)
ncol(training)
```

```
## [1] 160
```

After reading in the data, it is apparent that there are 159 features that could be used to predict `classe`. It would be preferable to reduce the number of components, as it would make the training process execute in a shorter time.

All the missing data in concentrated in particular columns. These columns are almost entirely filled with missing data. These columns are removed here, reducing the feature count to 59:

```
na_count <- sapply(training, function(y) sum(length(which(is.na(y)))))
columns_to_keep <- names(na_count[na_count == 0])

training <- select(training, columns_to_keep)
# use head to ignore "classe" target column, as it doesn't exist in testing data
testing <- select(testing, head(columns_to_keep, -1))
```

```
ncol(training)
```

```
## [1] 60
```

The first seven columns contain metadata (such as the user and date) that we do not want to predict from, so those are removed, leaving 52 features:

```
training <- select(training, -(X:num_window))
testing <- select(testing, -(X:num_window))
ncol(training)
```

```
## [1] 53
```

The number of features could be reduced even further by using a principle components analysis (PCA):

```
preProcess(training[,-ncol(training)], method='pca', thresh=0.95)
```

```
## Created from 19622 samples and 52 variables
##
## Pre-processing:
##   - centered (52)
##   - ignored (0)
##   - principal component signal extraction (52)
##   - scaled (52)
##
## PCA needed 25 components to capture 95 percent of the variance
```

This reduces the number of features down to 25, while still capturing 95 % of the variance in the data.

Finally, 15 % of the training data is placed into a validation set that may be used later to estimate the out-of-sample error.

```
inTrain <- createDataPartition(training$classe, p = 0.85)[[1]]
training <- training[inTrain,]
validation <- training[-inTrain,]
```

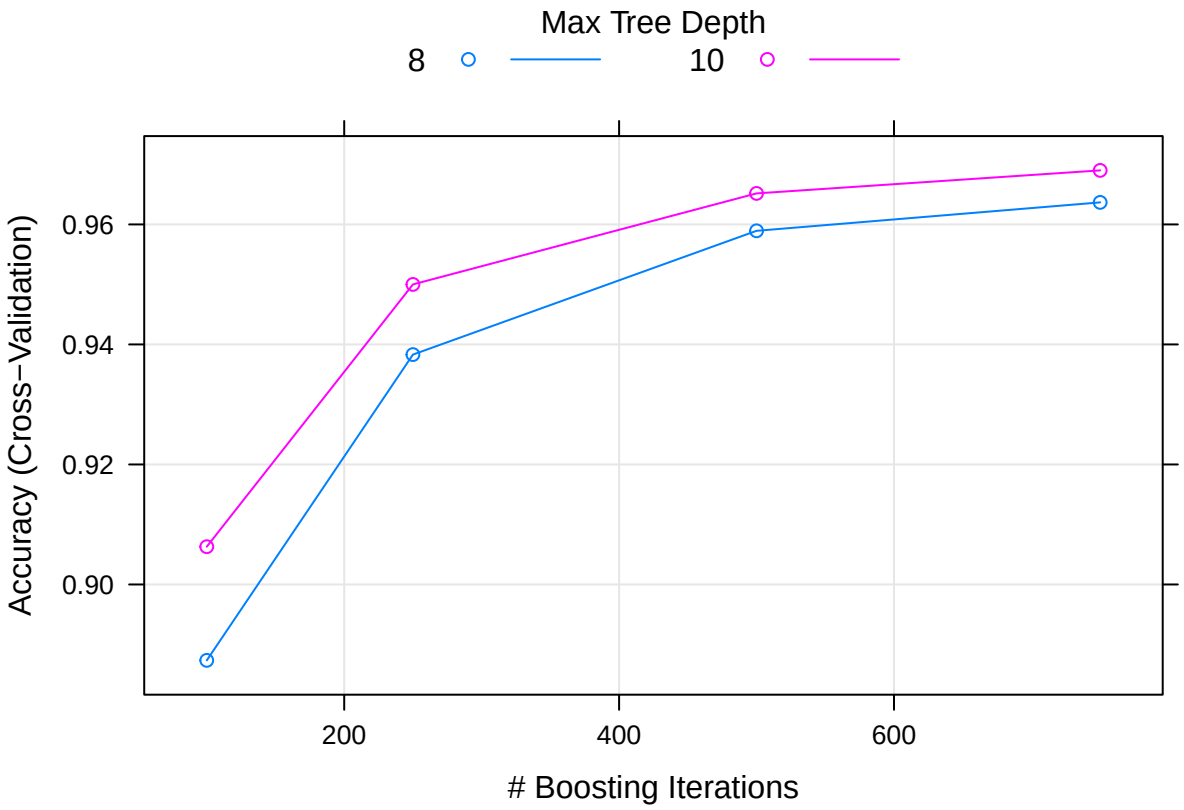
Training

```
options <- expand.grid(interaction.depth = c(8, 10),
                       n.trees = c(100, 250, 500, 750),
                       shrinkage = 0.1,
                       n.minobsinnode = 10)

control <- trainControl(method = "cv",
                        number = 4,
                        allowParallel = TRUE,
                        preProcOptions = list(threshold = 0.95))

model_gbm <- train(classe ~ .,
                   data = training,
                   method = 'gbm',
                   preProcess = 'pca',
                   trControl = control,
                   tuneGrid = options)
```

```
plot(model_gbm)
```



A gradient boosting model was used because of its effectiveness in classification problems. The plot shows that 750 iterations and an interaction depth of 10 is sufficient to provide acceptable accuracy; any further increase is likely to provide diminishing returns.

```
kable(model_gbm$results)
```

	shrinkage	interaction.depth	n.minobsinnode	n.trees	Accuracy	Kappa	AccuracySD	KappaSD
1	0.1	8	10	100	0.8873511	0.8574162	0.0047347	0.0059700
5	0.1	10	10	100	0.9062954	0.8813856	0.0032068	0.0040593
2	0.1	8	10	250	0.9383102	0.9219317	0.0038106	0.0048246
6	0.1	10	10	250	0.9500003	0.9367340	0.0019631	0.0024953
3	0.1	8	10	500	0.9589331	0.9480410	0.0021410	0.0027037
7	0.1	10	10	500	0.9651681	0.9559322	0.0016013	0.0020182
4	0.1	8	10	750	0.9636692	0.9540342	0.0010554	0.0013389
8	0.1	10	10	750	0.9690052	0.9607874	0.0014724	0.0018627

A 4-fold cross-validation was used to estimate the out-of-sample performance. As shown above, the expected out-of-sample accuracy for the optimised model is expected to be approximately 97 %. This can be tested further using the validation data set:

```
validation_predictions <- predict(model_gbm, validation)
conf <- confusionMatrix(validation_predictions, validation$classe)
kable(conf$table)
```

	A	B	C	D	E
A	725	0	0	0	0
B	0	483	0	0	0
C	0	0	414	0	0
D	0	0	0	419	0
E	0	0	0	0	459

In this case, the model achieves 100 % accuracy for the validation set, so the previous estimate does not seem unreasonable.

Prediction

The predicted `classe` outcomes for the test data set are:

```
predict(model_gbm, testing)
```

```
##  [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
```