On the Classification of Qualitative Characteristics of Physical Exercise

Tim Pearce

7 October 2017

Abstract

```
## Loading required package: lattice
## Loading required package: ggplot2
## Warning: package 'randomForest' was built under R version 3.4.1
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

We compare two classifiers to disinguish correctly-performed exercises (barbell lifts) from the same exercise performed incorrectly in a number of ways. We then select a final model based on the error rate for the two models.

The approach to data collection is described in Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. "Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements". Proceedings of 21st Brazilian Symposium on Artificial Intelligence, Advances in Artificial Intelligence - SBIA 2012, in: Lecture Notes in Computer Science, pp. 52-61. Curitiba.

Based on cross-validation of the models on the training data, the best model produced is expected to achieve accuracy of around 99% on unseen data.

Description and pre-processing of the data

The training data consists of 19622 observations of 159 variables. However, some variables have many NA or blank values, which we first eliminate to leave 60 variables (including the class variable). Finally, we remove the first seven columns, which are not meaningful for prediction.

```r
pml_train2 <- pml_train[,colSums(is.na(pml_train))/dim(pml_train)[1] == 0] # Eliminate NA
pml_train3 <- pml_train2[,colSums(pml_train2 == "") == 0] # Eliminate blanks
pml_train_clean <- pml_train3[,8:60]
```

We apply the same transformations to the test data.

Model building and evaluation

We compare a simple classifier (Linear Discriminant Analysis) with a more sophisticated one (random forests) which we expect to be more accurate.

We evaluate both on the training set using five-fold cross-validation. The choice of cross-validation in place of caret's default method of bootstrapping is to improve performance, given the large size of the training set (both in terms of the number of instances and the number of attributes). All other parameters are left at their default values.

Model 1: Linear Discriminant Analysis

We now build the first model.

```
# Separate predictors from class attribute
x <- pml_train_clean[,-53]
y <- pml_train_clean[,53]
# Set up cross-validation
set.seed(1729)
fitControl <- trainControl(method = "cv", number = 5)
# Build and print model
modelLDA <- train(x, y, method="lda", data = pml_train_clean, trControl = fitControl)
print(modelLDA)
```

```
## Linear Discriminant Analysis
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15698, 15699, 15696, 15697, 15698
## Resampling results:
##
##   Accuracy   Kappa
##   0.7013054  0.6220026
```

We see that the accuracy of this model is quite low (70.1% on the training data), so try to improve upon this performance with a more sophisticated model.

Model 2

```
# Set up cross-validation
set.seed(9271)
fitControl <- trainControl(method = "cv", number = 5)
# Build and print model
modelRF <- train(x, y, method="rf", data = pml_train_clean, trControl = fitControl)
print(modelRF)
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15697, 15698, 15698, 15699, 15696
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9941902  0.9926504
##   27    0.9941900  0.9926502
##   52    0.9889403  0.9860082
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Accuracy on the cross-validated training data is much higher here, at over 99%. Here is the associated confusion matrix:

```
confusionMatrix(modelRF)
```

```
## 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  0.3  0.0
##          D  0.0  0.0  0.0 16.1  0.0
##          E  0.0  0.0  0.0  0.0 18.3
##
##  Accuracy (average) : 0.9942
```

However, we may not achieve the same accuracy on unseen data. We can estimate the out-of-bag error rate as follows.

```
mean(modelRF$finalModel$err.rate[,"OOB"])
```

```
## [1] 0.007198186
```

We therefore continue to expect an accuracy rate about 99% on the test set.

Since the random forest is the more accurate of the two models, we take this as our final model and apply it to the test set.

Application of model to test data

We can now predict the class of the test set using the random forest model. The results are output in order (problem IDs 1-20).

```
pml_test_clean$predictedClass <- predict(modelRF, pml_test_clean)
print(pml_test_clean$predictedClass)
```

```
##  [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
```