Machine Prediction of Exercise Results

John W. Hoggard

7/4/2017

Problem

We have data regarding a variety of measures taken from sensors on a person's body while they performed an exercise, and a classification of whether the activity was performed correctly or featuring one of four common errors. The data was analyzed in a 2013 paper[1] and is available at http://groupware.les.inf.puc-rio.br/har. The data includes a variable `classe` which contains a classification code (`A` through `E`) for how the exercise was performed. Our task is to predict `classe` based off of the other available data.

Importing and Cleaning Data

Data for training was provided in a file `pml-training.csv`. To begin the analysis, we import the data:

```r
train.data <- read.table("pml-training.csv", header=TRUE, sep=",", stringsAsFactors=FALSE)
```

We note that a large number of columns here are mostly blank or mostly filled with NAs. Blank columns have imported as character type, so we eliminate these, and the columns that are mostly NA, but keep character values which we might use. We also convert the variable `classe` (which contains the classification of how the exercise was completed) to a factor variable:

```r
keep <- !(sapply(train.data, class)=="character")
keep[c("user_name", "cvtd_timestamp", "new_window", "classe")]<- TRUE
train.data <- train.data[, keep]
train.data <- train.data[,(sapply(train.data, function(x) {sum(is.na(x))}) < 19000)]
train.data$classe<-as.factor(train.data$classe)
```

For cross-validation, we will select a subset of this data for training, and retain the remainder for testing at the end:

```r
library(caret)
inTrain<-createDataPartition(train.data$classe, p=.7, list=FALSE)
trTrain <- train.data[inTrain,]
trTest <- train.data[-inTrain,]
```

Discriminant Analysis

For predicting a factor variable like `classe`, we could attempt linear or quadratic discriminant analysis. On the training data `trTrain`, linear discriminant analysis is able to predict on all remaining variables with about 70% accuracy (in-sample error), but the more flexible Quadratic Discriminant Analysis is able to reach about 89% accuracy in-sample:

```r
modQDAS <- train(classe ~., method="qda", data=trTrain[,-c(1:7)])
```

```
## Loading required package: MASS
```

[1]Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. **Proceedings of 4th International Conference in Cooperation with SIGCHI** (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

```
confusionMatrix(predict(modQDAS, trTest), trTest$classe)$overall[1]
```

```
##  Accuracy
## 0.8892099
```

(Here we exclude the first seven variables from consideration, as these include details such as the test case and the name of the subject which should not be used.) This is fairly good, despite the fact that examination of some of the variables suggests that the data is probably not normally distributed.

Tree Models

Given the potentially non-linear nature of the prediction required, we consider a classification tree, and use `rpart` to create one:

```
modTreeFit <- train(classe ~ ., method="rpart", data=trTrain[,-c(1:7)])
```

```
## Loading required package: rpart
```

```
confusionMatrix(predict(modTreeFit, trTrain), trTrain$classe)$overall[1]
```

```
##  Accuracy
## 0.4969062
```

Here, in-sample accuracty is only about 50%, so we choose to expand the tree model to a random forest. We move to bootstrapping via random forests to improve the predictive power of the model.

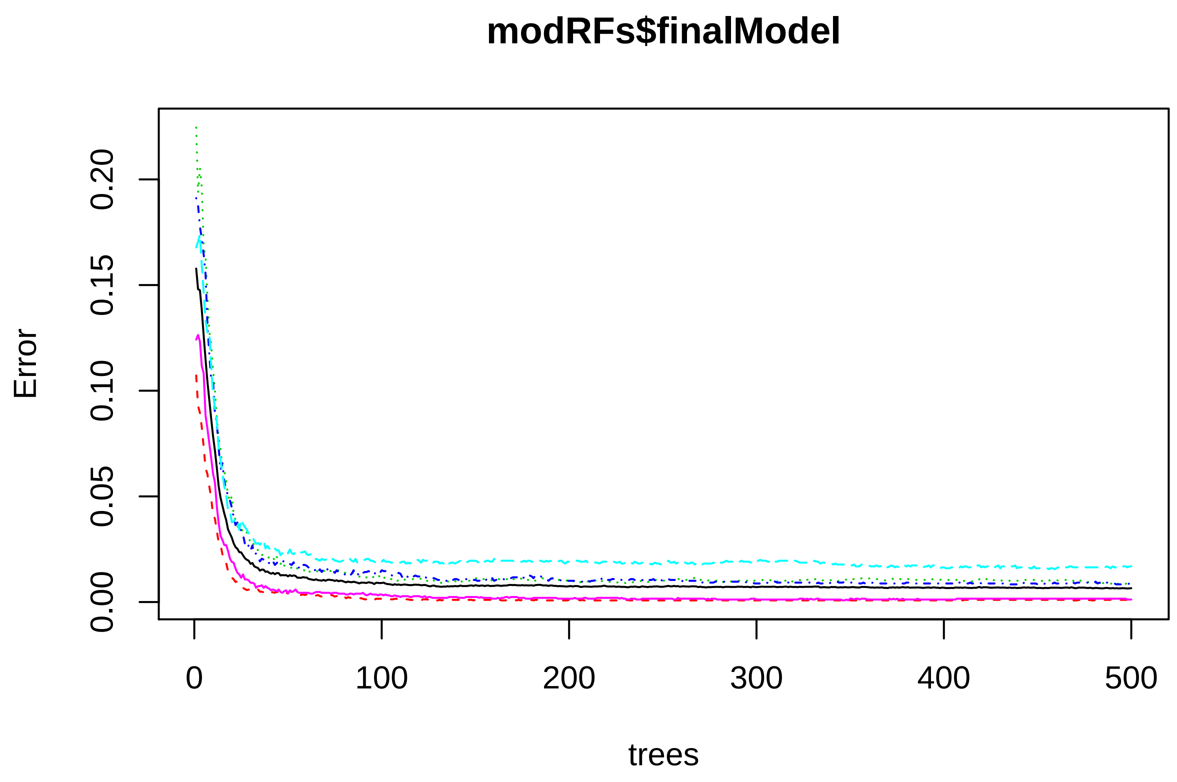
An initial attempt achieves almost perfect accuracy on the training data, although it requires a very long running time:

```
modRFs <- train(classe~., method="rf", data=trTrain[,-c(1:7)])
predRFs <- predict(modRFs, trTrain)
confusionMatrix(predRFs, trTrain$classe)$accuracy[1]
```

```
## NULL
```

The model defaults to using 500 trees. However, checking the error rates versus number of trees, it appears we achieve most of the accuracy by the time we reach 50 trees:

```
plot(modRFs$finalModel)
```



Reducing to 50 trees (to help avoid overfitting), and using repeated cross-validation with 10 folds, we repeat fitting of `classe` to other variables via a random forest, and still achieve about 99% accuracy on the training data. The estimate of error based on out-of-bag error rate is 82%, but this is typically a low estimate. The in-sample accuracy is very high again (about 99%):

```
control <- trainControl(method="repeatedcv", number=10, repeats=3)
mod50TreeRFs <- train(as.factor(classe)~., method="rf", data=trTrain[,-c(1:7)],
                      ntree=50, metric="Accuracy", trControl=control)
mod50TreeRFs$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 50, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.88%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3898    6    1    0    1 0.002048131
## B   24 2614   14    3    3 0.016553800
## C    1   14 2374    7    0 0.009181970
## D    0    1   27 2222    2 0.013321492
## E    0    4    5    8 2508 0.006732673
```

```
confusionMatrix(predict(mod50TreeRFs, trTest), trTest$classe)$overall[1]
```

```
##  Accuracy
## 0.9949023
```

Conclusion and Accuracy

Although the quadratic discriminant analysis appears to show high enough accuracy to be useful (at least in terms of in-sample error), the random forest shows more promise. (Also, the random forest does not require any assumptions about the form of the data.)

We finally check the out-of-sample error rate for our random forest model developed above using the test data we set aside at the start of the analysis:

```r
confusionMatrix(predict(mod50TreeRFs, trTest), trTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    4    0    0    0
##          B    3 1132    4    0    0
##          C    1    3 1019   10    0
##          D    0    0    3  954    2
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9939   0.9932   0.9896   0.9982
## Specificity            0.9991   0.9985   0.9971   0.9990   1.0000
## Pos Pred Value         0.9976   0.9939   0.9864   0.9948   1.0000
## Neg Pred Value         0.9991   0.9985   0.9986   0.9980   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1924   0.1732   0.1621   0.1835
## Detection Prevalence   0.2845   0.1935   0.1755   0.1630   0.1835
## Balanced Accuracy      0.9983   0.9962   0.9951   0.9943   0.9991
```

Indeed, the out-of-sample error rate seems to be very small, with about 99% accuracy of the random forest model on the test data, giving an error rate of 1% or less. This seems to have been a good choice for the model.

Quiz Prediction

For the quiz, we also import the needed test data, and process (converting answers from numeric to letters A-E for convenience.) Output supressed here.

```r
test.data <- read.table("pml-testing.csv", header=TRUE, sep=",", stringsAsFactors = FALSE)
data.frame(ANS = c("A", "B", "C", "D", "E")[predict(mod50TreeRFs, test.data)])
```

The model correctly predicted all answers.