Machine Learning Course Project

Cameron

October 18, 2018

Setting Up

The code in this report assumes we are in the same directory as our data source .csv files, and that a number of packages are loaded (available in the raw .Rmd file for this report).

Executive Summary

For this project, we analyzed a set of data from wearable activity monitors worn by test subjects performing a weighlifting task either correctly or with one of 4 common mistakes in technique. We developed a model to predict the class ('classe' in the data) of an exercise (i.e. correct form 'A' or one of the 4 common mistakes 'B','C','D','E') based on accelerometer data from the wearable monitors. Using a random forest model with 3-fold cross-validation on a 70/30 training/testing split of the data, we achieved an accuracy of 99.5% (out-of-sample error 0.5%) in classifying each weightlifting task. We correctly predicted 20 of 20 test cases outside our original dataset.

Data Processing

For this project, we are analyzing data from the Groupware@LES Weight Lifting Exercises dataset, from their 2013 paper (Velloso et al. 2013). We begin by loading both the training and testing (more accurately, quiz, since it will only be used to predict quiz answers) datasets.

```
rawtraining<-read.csv2('pml-training.csv',header=TRUE,sep=',',as.is=T)
rawtesting<-read.csv2('pml-testing.csv',header=TRUE,sep=',',as.is=T)
```

Taking a quick look at the data, we see a LOT of NA values. We'll clean these up in both datasets by dropping any columns with more than 95% NA values. After this step, we notice that the test set is now much cleaner, but the training set still contains lots of empty or mostly empty columns, so we'll apply another cleaning step. We manually identify and filter out these problematic columns, in addition to removing the columns containing time information (since each row is a complete repetition of the exercise, there's no reason to keep the time data). We also notice that all of our data columns have come in as character vectors due to the way we implemented read.csv2, and we'd like to convert them to numeric values (or, for one column, factors). This gives us our final training and test (really, quiz) datasets.

```
reducedtraining<-Filter(function(x) sum(!is.na(x))>0.95*length(x),rawtraining)
reducedtesting<-Filter(function(x) sum(!is.na(x))>0.95*length(x),rawtesting)
finaltraining<-reducedtraining[,c(8:11,21:42,49:51,61:73,83:93)]
finaltraining[,1:52]<-sapply(finaltraining[,1:52],as.numeric)
finaltraining[,53]<-as.factor(finaltraining[,53])
finalquiz<-reducedtesting[,8:60]
finalquiz[,1:52]<-sapply(finalquiz[,1:52],as.numeric)
```

Predictive Model Building

Now that the data is nicely cleaned up, we can get started building our predictive model. We will first split our "training" dataset into training and testing subsets, using a 70/30 split.

```
set.seed(3054)
inTrain<-createDataPartition(finaltraining$classe,p=0.7,list=FALSE)
training<-finaltraining[inTrain,]
testing<-finaltraining[-inTrain,]
```

Now, we'll set up parallel processing for building our model fit, which will save us some computing time.

```
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
```

We can now build in some parameters to control our model fitting process. Here, we'll set up for 3-fold cross-validation.

```
fitparams <- trainControl(method = "cv", number = 3,allowParallel = TRUE)
```

Finally, we're ready to train our model. Since we're trying to classify our outcomes into a small number of categories and the data is relatively noisy (as one might expect for data tracking human movements), we'll use a random forest model. We'll cache this particular code chunk to avoid rebuilding the model from scratch every time we knit this markdown document. The 'caret' package in R will handle the model-building process as follows:

```
modRF<-train(classe~.,data=training,method='rf',verbose=FALSE,trControl=fitparams)
modRF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.72%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    4    0    0    2 0.001536098
## B   20 2631    7    0    0 0.010158014
## C    0   13 2374    9    0 0.009181970
## D    0    1   24 2225    2 0.011989343
## E    0    0    5   12 2508 0.006732673
```

Results

We can see from the model summary above that our model fit does quite well at predicting the data it was trained on... of course, the real test will be to use this model on new data and see how accurate it is. We will now use the model to predict on the remaining 30% of the training set that we set aside for testing earlier.

```
predictRF<-predict(modRF,testing)
confusionMatrix(predictRF,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    6    0    0    0
##          B    5 1131    1    0    1
##          C    0    2 1024    9    2
```

```
##          D    0    0    1  955    2
##          E    0    0    0    0 1077
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9929, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9930   0.9981   0.9907   0.9954
## Specificity            0.9986   0.9985   0.9973   0.9994   1.0000
## Pos Pred Value         0.9964   0.9938   0.9875   0.9969   1.0000
## Neg Pred Value         0.9988   0.9983   0.9996   0.9982   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1922   0.1740   0.1623   0.1830
## Detection Prevalence   0.2846   0.1934   0.1762   0.1628   0.1830
## Balanced Accuracy      0.9978   0.9958   0.9977   0.9950   0.9977
```

We see that on 'new' data, we were able to achieve an accuracy of 99.5%, or an out-of-sample error rate of 0.5%. That should be more than adequate to predict the 20 cases given as the quiz for this project. We'll quickly generate predictions for those cases:

```
quizpredict<-predict(modRF,finalquiz)
quizpredict
```

```
##  [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
```

Plugging these predictions directly into the quiz on Coursera, we are able to correctly classify all 20 test cases.

References

Velloso, E, A Bulling, H Gellersen, W Ugulino, and H Fuks. 2013. "Qualitative Activity Recognition of Weight Lifting Exercises." *Proceedings of 4th Augmented Human (AH) International Conference in Cooperation with ACM SIGCHI.* http://groupware.les.inf.puc-rio.br/work.jsf?p1=11201.