Practical Machine Learning Course Project

Vincent Alexander Saulys

July 27, 2014

So assuming the data is in the directory, it will be imported here. The requisite R packages will also be called up. These packages are caret and gbm. The caret package is necesary for training the prediction model and the gbm package will be necesary as it contains the main prediction model algorithm to be used, which is called stochastic gradient boosting.

```
# All packages are assumed to be installed
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
library(gbm)
```

```
## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
## Loaded gbm 2.1
```

```
pml_training_set <- read.csv('pml-training.csv')
pml_testing <- read.csv('pml-testing.csv')
```

Now it becomes necesary to begin whittling down excess variables. At 159, there are far too many present to make efficient code. So to start, these excess variables will first have to be removed.

To begin with, the first seven variables will be removed. They contain miscellaneous information such as time stamps.

```
pml_training_set <- pml_training_set[8:160]
```

Following that, variables whose entries are respsented by more than 10% NA values will be deleted.

```
tmn <- names(colSums(is.na(pml_training_set))[  colSums(is.na(pml_training_set))  > (nrow (
pml_training_set) * 0.1) ])
k <- setdiff (names(pml_training_set), tmn)
pml_training_set <- pml_training_set[, k]
```

Lastly, nix all the variables with little variance are nixed. As these variables do not vary much, they will not help in prediction, they simply do not change enough to make a difference in prediction.

```
pml_training_set <- pml_training_set[, -nearZeroVar(pml_training_set)]
```

This gives 52 variables for prediction, which is a marked decrease from the 159 at the outset.

Next comes the paritioning of the data. Given the large number of entries, it can be safely split into 60% training and 40% validation.

```
dataPartition <- createDataPartition (pml_training_set$classe, p = 0.60, list = FALSE)[, 1]
training <- pml_training_set[dataPartition, ]
validate <- pml_training_set[-dataPartition, ]
```

Then the prediction model is trained. There are several types of models one can make, but gbm was used. The following list are the ones that were cut. - **nnet**, also known as neural nets, was cut as its computation time took more than 300 minutes. - **ada** resulted in strange nonsencial errors from the R console. - **rf** also resulted in strange errors after many hours of computation.

Therefore **gbm** was used, which is a stochastic gradient boosting method. Its profile is displayed below.

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    52 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
##
## Summary of sample sizes: 10599, 10600, 10597, 10598, 10599, 10598, ...
##
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa  Accuracy SD  Kappa SD
##   1                  50       0.8       0.7    0.02         0.02
##   1                  100      0.8       0.8    0.02         0.02
##   1                  200      0.9       0.8    0.01         0.02
##   2                  50       0.9       0.8    0.009        0.01
##   2                  100      0.9       0.9    0.008        0.01
##   2                  200      0.9       0.9    0.007        0.009
##   3                  50       0.9       0.9    0.008        0.009
##   3                  100      0.9       0.9    0.008        0.01
##   3                  200      1         0.9    0.004        0.005
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3 and shrinkage = 0.1.
```

Cross-validation was used here to check for the out-of-sample error. We expect the out-of-sample accuracy to be the same as the in-sample accuracy.

```
zeTrain <- trainControl(method='cv',number=10,repeats=1)
modelFit_gbm <- train(classe ~ ., data = training, method = "gbm", trControl = zeTrain)
```

```
## Loading required package: plyr
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1267
##      2        1.5237             nan     0.1000    0.0860
##      3        1.4666             nan     0.1000    0.0703
##      4        1.4207             nan     0.1000    0.0541
##      5        1.3845             nan     0.1000    0.0456
##      6        1.3553             nan     0.1000    0.0425
##      7        1.3272             nan     0.1000    0.0412
##      8        1.3014             nan     0.1000    0.0354
##      9        1.2786             nan     0.1000    0.0321
##     10        1.2581             nan     0.1000    0.0296
##     20        1.1034             nan     0.1000    0.0178
##     40        0.9347             nan     0.1000    0.0096
##     60        0.8254             nan     0.1000    0.0055
##     80        0.7427             nan     0.1000    0.0056
##    100        0.6798             nan     0.1000    0.0040
##    120        0.6292             nan     0.1000    0.0038
##    140        0.5852             nan     0.1000    0.0019
##    150        0.5662             nan     0.1000    0.0032
##
##      The process was truncated here for the sake of making
##      this easier to read.
##
```

Then the validation data is predicted on. The predicted classe is appended to the validation data frame.

```
validate["predicted_gbm"] <- predict(modelFit_gbm, newdata=validate)
```

Next we'll print the data we see as a confusion matrix, which is a convenient way to view the performance of the prediction model on the validation data set.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2187   30   10    4    1
##          B   47 1436   33    2    0
##          C    0   48 1301   15    4
##          D    0    2   40 1238    6
##          E    4   34   17   23 1364
##
## Overall Statistics
##
##                Accuracy : 0.959
##                  95% CI : (0.955, 0.963)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.948
##  Mcnemar's Test P-Value : 3.78e-14
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.977    0.926    0.929    0.966    0.992
## Specificity             0.992    0.987    0.990    0.993    0.988
## Pos Pred Value          0.980    0.946    0.951    0.963    0.946
## Neg Pred Value          0.991    0.982    0.985    0.993    0.998
## Prevalence              0.285    0.198    0.179    0.163    0.175
## Detection Rate          0.279    0.183    0.166    0.158    0.174
## Detection Prevalence    0.284    0.193    0.174    0.164    0.184
## Balanced Accuracy       0.985    0.957    0.959    0.979    0.990
```

The most important variables the model found are displayed below.

```
## gbm variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                    Overall
## roll_belt           100.00
## pitch_forearm        48.89
## yaw_belt             36.55
## magnet_dumbbell_z    29.80
## magnet_dumbbell_y    24.90
## roll_forearm         21.35
## magnet_belt_z        18.91
## roll_dumbbell        15.56
## gyros_belt_z         14.72
## accel_forearm_x      11.81
## pitch_belt           11.48
## gyros_dumbbell_y     10.02
## yaw_arm               8.79
## accel_dumbbell_y      8.68
## magnet_forearm_z      8.54
## accel_forearm_z       7.39
## magnet_arm_z          7.11
## accel_dumbbell_x      6.63
## magnet_belt_x         4.87
## magnet_belt_y         4.66
```

To finish, all the testing data is predicted upon and outputted into text files for upload to Coursera.

```
pml_testing["predicted_gbm"] <- predict(modelFit_gbm, newdata=pml_testing)
answers <- pml_testing$predicted_gbm
dir.create ('solutions')
n = length(answers)
for(i in 1:n)
{
  filename = paste("problem_id_",i,".txt")
  filename = file.path('solutions', filename)
  write.table (answers[i], file=filename, quote=FALSE, row.names=FALSE, col.names=FALSE)
}
```