Practical Machine Learning Project

Ernest Jum

10/24/2016

Background

"Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self-movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways."

Goal

The goal of this project is to predict the manner in which the 6 participants performed the exercise. That is, we try to predict the "classe" variable in the training set.

Models and Method

In order to predict the "classe"" variables, we trained the following four models:

- Artificial Neural Network (ANN)
- Support Vector Machine (SVM)
- Generalized Boosted Regression (GBR)
- Random Forest (RF)

The given training data set had 19622 rows with 160 variables and the test data set had 20 rows with 160 variables. In order to clean this data set, we removed the columns with too many missing values. After cleaning the data set, we noted that the first seven columns were not useful for predicting the "classe" variable. Therefore, we removed these columns.

Since the test data set did not contain the response variable ("classe") we partitioned the given data set into new training (75%) and validation (25%) sets repectively. The new training set was then used to train the aforementioned models and then tested on the validation data set. Here we used cross validation as the control method. These models were trained on one of the randomly selected data sets and tested on the remaining four. In order to reduce variability, 5 rounds of cross-validation were performed using different partitions, and the validation results were averaged over the rounds. That is, for each such split, the model(s) was fit to the training data, and predictive accuracy is assessed using the validation data. The results were then averaged over the splits.

```
## set working directory
setwd("/Users/ernestjum/Documents/practical_mach_learning")

############################################################
############################################################
```

```
## Needed packages: install if not already installed

# install.packages("caret", dependencies = c("Depends", "Suggests"))
# install.packages("kernlab", dependencies = c("Depends", "Suggests"))
# install.packages("ISLR", dependencies = c("Depends", "Suggests"))
# install.packages("neuralnet", dependencies = c("Depends", "Suggests"))
# install.packages("ggplot2", dependencies = c("Depends", "Suggests"))
# install.packages("ksvm", dependencies = c("Depends", "Suggests"))
```

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

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

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
#############################################################
#############################################################
## random seed for replication of results
set.seed(123456)

#############################################################
#############################################################
# The data for this project comes from the following source:
# http://groupware.les.inf.puc-rio.br/har.

## read in training and test data sets
train<-read.csv("pml-training.csv", stringsAsFactors = FALSE,na.strings = c("NA", ""))
```

```
test<-read.csv("pml-testing.csv", stringsAsFactors = FALSE, na.strings = c("NA", ""))

#################################################################
#################################################################
## explore training set

# str(train)
# dim(train)
# summary(train)
```

Clean data set for modeling

A summary of the training data set indicates that there are some columns with too many missing values. These variables will not be useful in our analysis thus we ommit them.

```
#################################################################
#################################################################
## Delete columns with too many missing values
train1 <- train[, colSums(is.na(train)) == 0]
test1 <- test[, colSums(is.na(test)) == 0]
```

The first seven columns cannot be used for predictions so we omit them

```
#################################################################
#################################################################
## ommit first seven columns
train2 <- train1[, -c(1:7)]
test_final <- test1[, -c(1:7)]
```

Data Partitioning

We split the cleaned data set into training (75%) and validation (25%) set, which are then used to train the different model.

```
#################################################################
#################################################################
# Partition data set into training and validation--75% to 25%
inTrain <- createDataPartition(train2$classe, p = 0.75, list = FALSE)
train_final <- train2[inTrain, ]
validation <- train2[-inTrain, ]
```

Control Method is Cross Validation

```
#################################################################
#################################################################
## cross validation--number of samples equals 5
ctrl <- trainControl(method = "cv", number = 5, savePred=T, classProb=T)
```

Artificial Neural Network with 15 hidden layers

```
model_ANN<-train(classe ~., data=train_final, method="nnet", trace=FALSE,
                                    tuneLength=2 ,hidden=15, trControl=ctrl)
```

```
## Loading required package: nnet
```

ANN Statisitcs and Accuracy and Predicted Values

```
 print(model_ANN, digits = 4)
```

```
## Neural Network
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11774, 11775, 11775, 11773
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy  Kappa
##   1     0.0    0.3204    0.1151
##   1     0.1    0.3581    0.1727
##   3     0.0    0.3988    0.2519
##   3     0.1    0.3604    0.1949
##
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were size = 3 and decay = 0.
```

```
  pred_ANN <- predict(model_ANN, validation)
  (conf_mat_ANN <- confusionMatrix(validation$classe, pred_ANN))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 934   2   0   0 459
##          B  83 195   0   0 671
##          C 205  14   0   0 636
##          D 145  34   0   0 625
##          E  65  81   0   0 755
##
## Overall Statistics
##
##                Accuracy : 0.3842
##                  95% CI : (0.3705, 0.398)
##     No Information Rate : 0.6415
```

```
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2167
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6522  0.59816       NA       NA   0.2400
## Specificity            0.8672  0.83530   0.8257   0.8361   0.9170
## Pos Pred Value         0.6695  0.20548       NA       NA   0.8380
## Neg Pred Value         0.8581  0.96688       NA       NA   0.4027
## Prevalence             0.2920  0.06648   0.0000   0.0000   0.6415
## Detection Rate         0.1905  0.03976   0.0000   0.0000   0.1540
## Detection Prevalence   0.2845  0.19352   0.1743   0.1639   0.1837
## Balanced Accuracy      0.7597  0.71673       NA       NA   0.5785
```

```
  (acc_ANN <- conf_mat_ANN$overall[1])
```

```
##  Accuracy
## 0.3841762
```

```
  ## output prediction on test data set
  (pred_test_ANN<-predict(model_ANN, test_final))
```

```
##  [1] A E E A E E E E A A E A E A E E A B E E
## Levels: A B C D E
```

Support Vector Machine

```
#################################################################
#################################################################
## SVM
model_SVM <- train(classe~., data=train_final, method = "svmRadial",
                                                    trControl = ctrl)
```

SVM Statisitcs and Accuracy and Predicted Values

```
## SVM Accuracy, Confusion Matrix and Prediction
 print(model_SVM, digits = 4)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11776, 11774, 11775, 11774, 11773
## Resampling results across tuning parameters:
##
##   C     Accuracy  Kappa
##   0.25  0.7786    0.7177
##   0.50  0.7983    0.7432
##   1.00  0.8074    0.7552
##
## Tuning parameter 'sigma' was held constant at a value of 0.01400298
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were sigma = 0.014 and C = 1.
```

```
  pred_SVM <- predict(model_SVM, validation)
  (conf_mat_SVM <- confusionMatrix(validation$classe, pred_SVM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1369   11   12    0    3
##          B   34  875   39    0    1
##          C   27   63  662  102    1
##          D   68   75  121  357  183
##          E   11   59   83   39  709
##
## Overall Statistics
##
##                Accuracy : 0.81
##                  95% CI : (0.7987, 0.8208)
##     No Information Rate : 0.3077
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7585
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9072   0.8079   0.7219   0.7169   0.7904
## Specificity            0.9923   0.9806   0.9516   0.8985   0.9521
## Pos Pred Value         0.9814   0.9220   0.7743   0.4440   0.7869
## Neg Pred Value         0.9601   0.9474   0.9370   0.9656   0.9530
## Prevalence             0.3077   0.2208   0.1870   0.1015   0.1829
## Detection Rate         0.2792   0.1784   0.1350   0.0728   0.1446
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9498   0.8943   0.8368   0.8077   0.8712
```

```
  (acc_SVM <- conf_mat_SVM$overall[1])
```

```
##  Accuracy
## 0.8099511
```

```
  ## output prediction on test data set
  (pred_test_SVM<-predict(model_SVM, test_final))
```

```
##  [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
```

Generalized Boosted Regression

```
#################################################################
#################################################################
model_gbm <- train(classe ~ ., data=train_final, method = "gbm",
                                  trControl = ctrl, verbose = FALSE)
```

```
## Loading required package: gbm
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

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

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:Hmisc':
##
##     is.discrete, summarize
```

GBR Statisitcs, Accuracy and Prediction

```
print(model_gbm, digits = 4)
```

```
## Stochastic Gradient Boosting
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11774, 11775, 11774, 11773, 11776
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa
##   1                   50      0.7497    0.6826
```

```
##   1                  100      0.8211    0.7735
##   1                  150      0.8528    0.8138
##   2                   50      0.8539    0.8148
##   2                  100      0.9026    0.8767
##   2                  150      0.9287    0.9098
##   3                   50      0.8956    0.8678
##   3                  100      0.9424    0.9271
##   3                  150      0.9613    0.9511
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## 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, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
pred_gbm <- predict(model_gbm, validation)
(conf_mat_gbm <- confusionMatrix(validation$classe, pred_gbm))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1367   15    9    3    1
##          B   26  893   29    0    1
##          C    0   21  823    8    3
##          D    1    2   30  763    8
##          E    1    9   16   17  858
##
## Overall Statistics
##
##                Accuracy : 0.9592
##                  95% CI : (0.9533, 0.9646)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9484
##  Mcnemar's Test P-Value : 7.648e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9799   0.9500   0.9074   0.9646   0.9851
## Specificity            0.9920   0.9859   0.9920   0.9900   0.9893
## Pos Pred Value         0.9799   0.9410   0.9626   0.9490   0.9523
## Neg Pred Value         0.9920   0.9881   0.9793   0.9932   0.9968
## Prevalence             0.2845   0.1917   0.1850   0.1613   0.1776
## Detection Rate         0.2788   0.1821   0.1678   0.1556   0.1750
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9860   0.9679   0.9497   0.9773   0.9872
```

```
(acc_gbm <- conf_mat_gbm$overall[1])
```

```
## Accuracy
## 0.959217
```

```
  ## output prediction on test data set
  (pred_test_gbm<-predict(model_gbm, test_final))
```

```
##  [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
```

Random Forest

```
#################################################################
#################################################################
model_rf <- train(classe ~ ., data = train_final, method = "rf",
                                                    trControl = ctrl)
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:Hmisc':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

RFSummary Statistics, Accuracy and Prediction

```
print(model_rf, digits = 4)
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11773, 11775, 11774, 11775, 11775
```

```
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9925    0.9905
##   27    0.9914    0.9891
##   52    0.9835    0.9791
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
  pred_rf <- predict(model_rf, validation)
  (conf_mat_rf <- confusionMatrix(validation$classe, pred_rf))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    3  944    2    0    0
##          C    0    5  850    0    0
##          D    0    0   12  791    1
##          E    0    0    3    2  896
##
## Overall Statistics
##
##                Accuracy : 0.9943
##                  95% CI : (0.9918, 0.9962)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9928
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9979   0.9947   0.9804   0.9975   0.9989
## Specificity            1.0000   0.9987   0.9988   0.9968   0.9988
## Pos Pred Value         1.0000   0.9947   0.9942   0.9838   0.9945
## Neg Pred Value         0.9991   0.9987   0.9958   0.9995   0.9998
## Prevalence             0.2851   0.1935   0.1768   0.1617   0.1829
## Detection Rate         0.2845   0.1925   0.1733   0.1613   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9989   0.9967   0.9896   0.9972   0.9988
```

```
  (acc_rf <- conf_mat_rf$overall[1])
```

```
##  Accuracy
## 0.9942904
```

```
## output prediction on test data set
  (pred_test_rf<-predict(model_rf, test_final))
```

```
##  [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
```

Conclusion

Based on the accuracy from the validation data set, the random generalized boosted regression and the random forest have the best performance. But the random forest has a better over all accuracy, thus we use it for predicting the "classe" variable. The random forest has an accuracy of 0.9942904 with the following predictions on the test set:

```
pred_test_rf
```

```
##  [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
```