Machine Learning Project

Chieh-Ju Lin

12/26/2017

Outline

This is an R Markdown document for Coursera Data Science Specialization Course - Applied Machine Learning. The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. In this report, the machine learning algorithms I used including SVM, Decision Tree, Random Forest, Boosting, Neural Network, and K-Nearest Neighbors. The 20 test cases were predicted using the top 3 best models I built.

Data Preparation

Let's read the data and explore it.

```
### Explore the data
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
# A 5580, B 3797, C 3422, D 3216, E 3607
```

Seems like there's no class imbalance problem in our training data. However, I noticed that both data sets have many missing values. There are 100 variables with no data in test set. I decided to drop them in both data sets since they are not helpful in prediction.

```
### Drop variables with no data in testing set
names <- colnames(testing)[colSums(is.na(testing)) > 0]
var <- names(training) %in% c(names)

training_cleaned <- training[!var]
testing_cleaned <- testing[!var]
```

Then I scaled the data and split the training data into 70% training and 30% validation set. The final data set looks like the following:

training set

```
dim(training)
```

```
## [1] 13735    58
```

validation set

```
dim(validation)
```

```
## [1] 5887   58
```

testing set

```
dim(testing)
```

```
## [1] 20 57
```

Model Building

Support Vector Machine

In this part of analysis, I didn't tune the SVM models since they took too long to run in my laptop. For non-linear kernel, default cost and gamma value were used. Table for SVM results, and the confusion matrix for validation set using the best SVM model were shown in the following:

```
##                svm_model training_accuracy validation_accuracy
## 1                 linear         0.9104478           0.9098013
## 2                 radial         0.9566072           0.9563445
## 3  polynomial_degree = 3         0.9444485           0.9432648
## 4  polynomial_degree = 5         0.9110302           0.9030066
## 5 polynomial_degree = 10         0.5155442           0.5073892
```

```
##
## svm_pred_validation_2    A    B    C    D    E
##                     A 1650   80    1    0    0
##                     B    3 1012   44    1    0
##                     C    4   19 1006   72    0
##                     D    0    0    5  889   28
##                     E    0    0    0    0 1073
```

Tree Based Model

For tree based models, I built decision trees, pruned trees, random forests, and boosting. I found that use information gain to split the tree have better result than use gini. But pruning didn't change the classification results in both cases. Random forest and boosting gave us very high accuracy, with 100% in training set and 99% in validation set. Table for tree based model results, and confusion matrix for validation set using random forest and boosting are shown as follows:

```
##                        tree_models training.accuracy validation.accuracy
## 1             Decision Tree (gini)         0.8688751           0.8603703
## 2 Decision Tree (information gain)         0.9238442           0.9150671
## 3     Random Forest with 100 trees         1.0000000           0.9991507
## 4     Random Forest with 500 trees         1.0000000           0.9993205
## 5      Boosting with caret package         1.0000000           0.9974520
## 6        Boosting with gbm package         1.0000000           0.9979616
```

Confusion matrix for random forest with 500 trees

```
##
## forest_pred_validation_2    A    B    C    D    E
##                         A 1657    1    0    0    0
##                         B    0 1110    1    0    0
##                         C    0    0 1055    2    0
##                         D    0    0    0  960    0
##                         E    0    0    0    0 1101
```

Confusion matrix for boosting with caret package (use 500 trees and 10 folds cross validation)

```
##
## boosting_pred_validation_1    A    B    C    D    E
##                           A 1657    1    0    0    0
##                           B    0 1106    2    0    0
##                           C    0    4 1053    0    0
##                           D    0    0    1  956    1
```

```
##                          E    0    0    0    6 1100
```

Neural Network

Since Neural Network can only handle quantitative variables, we need to modify our data. I first drop time variables, and turn window variable and our decision variable - classe into dummy variables.

```
dim(training_nn)
```

```
## [1] 13735    62
```

```
head(training_nn[58:62])
```

```
##       classe_A classe_B classe_C classe_D classe_E
## 5643         0        1        0        0        0
## 15468        0        0        0        1        0
## 8025         0        1        0        0        0
## 17324        0        0        0        0        1
## 18451        0        0        0        0        1
## 894          1        0        0        0        0
```

Since the package "neuralnet" took too long to run in R markdown, I will exclude the code here and only show my results using h2o package. The activation function I used was "tanh". I noticed that large number of neurons didn't give me better result. The result table is as follows: (I used all of the available data with 10 folds cross validation in h2o package for nerual network, so there's no validation accuracy)

```
##                 nn_model training_accuracy validation_accuracy
## 1 nn with h2o (200, 200)         0.7559882                    -
## 2       nn with h2o (10)         0.8983794                    -
## 3   nn with h2o (10, 10)         0.9563755                    -
## 4        nn with h2o (5)         0.9522475                    -
## 5     nn with h2o (5, 3)         0.9576496                    -
```

K Nearest Neighbors

For K Nearest Neighbors models, I noticed that smaller number of K gave me better results.

```
##     knn_model training_accuracy validation_accuracy
## 1 knn, n = 50                 -           0.7757771
## 2 knn, n = 10                 -           0.9079327
## 3  knn, n = 5                 -           0.9505691
## 4  knn, n = 3                 -           0.9707831
## 5  knn, n = 1                 -           0.9879395
```

```
##
## knn_model_5    A    B    C    D    E
##           A 1653    6    1    8    0
##           B    2 1094    2    3    6
##           C    0    3 1041   10    3
##           D    1    4   10  939    3
##           E    1    4    2    2 1089
```

Prediction

Finally, I used my top 3 models (Random Forest with 500 trees, Boosting, and K Nearest Neighbors with K = 1) to predict the test cases.

Prediction results for the top 3 models

```
##    forest_pred_test boosting_pred_test_1 knn_pred_test_1
## 1                 B                    B              B
## 2                 A                    A              A
## 3                 B                    B              B
## 4                 A                    A              A
## 5                 A                    A              A
## 6                 E                    B              E
## 7                 D                    D              D
## 8                 B                    B              B
## 9                 A                    A              A
## 10                A                    A              A
## 11                B                    C              B
## 12                C                    C              C
## 13                B                    B              B
## 14                A                    A              A
## 15                E                    D              E
## 16                E                    E              E
## 17                A                    A              A
## 18                B                    B              B
## 19                B                    B              B
## 20                B                    B              B
```

Using majority vote rule, my final prediction is:

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
## "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B"
##  19  20
## "B" "B"
```