Prediction Assignment Writeup

Victor

February 14, 2016

Practical Machine Learning Assignment Writeup

Abstract

In this assignment, I built a predictive model to determine whether a particular form of exercise (barbell lifting) is performed correctly, using accelerometer data.

Data Retrieval

The dataset is downloaded from the Internet:

```
if (! file.exists('./pml-training.csv')) {
    download.file('http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv', destfile = './p
}
if (! file.exists('./pml-testing.csv')) {
    download.file('http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv', destfile = './pm
}
```

Loading CSV data:

```
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-training.csv")
```

Exploratory Analysis

The training data has 19622 observations and 160 features:

```
dim(training)
```

```
## [1] 19622   160
```

Inspection of the data set indicates that many of the 159 predictors are missing in most of the observations:

```
sum(complete.cases(training))
```

```
## [1] 406
```

I've divided test into 80%/20% for training/validation set. The I cleaned the data set by extracting necessary features:

```
library(caret)
```

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

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

```
trainset <- createDataPartition(training$classe, p = 0.8, list = FALSE)
Training <- training[trainset, ]
Validation <- training[-trainset, ]
nzvcol <- nearZeroVar(Training)
Training <- Training[, -nzvcol]
cntlength <- sapply(Training, function(x) {
  sum(!(is.na(x) | x == ""))
})
nullcol <- names(cntlength[cntlength < 0.6 * length(Training$classe)])
descriptcol <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
                 "cvtd_timestamp", "new_window", "num_window")
excludecols <- c(descriptcol, nullcol)
Training <- Training[, !names(Training) %in% excludecols]
```

Predictive Model

For my initial attempt at building a predictive model I chose the random forest algorithm. Random forests have several nice theoretical properties:

1. They deal naturally with non-linearity, and assuming linearity in this case would be imprudent.
2. There's no parameter selection involved. While random forest may overfit a given data set, just as any other machine learning algorithm, it has been shown by Breiman that classifier variance does not grow with the number of trees used (unlike with Adaboosted decision trees, for example). Therefore, it's always better to use more trees, memory and computational power allowing.
3. The algorithm allows for good in-training estimates of variable importance and generalization error, which largely eliminates the need for a separate validation stage, though obtaining a proper generalization error estimate on a testing set would still be prudent.
4. The algorithm is generally robust to outliers and correlated covariates, which seems like a nice property to have when there are known interactions between variables and no data on presence of outliers in the data set.

Given that the problem at hand is a high-dimensional classification problem with number of observations much exceeding the number of predictors, random forest seems like a sound choice.

Let's train a classifier without cross validation. CV tests on such big datasets cause increase in training time significantly and actually unnecessary because of great results without cross validation.

```
set.seed(555)
library(randomForest)
```

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

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

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

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

```
rfModel <- randomForest(classe ~ ., data = Training, importance = TRUE, ntrees = 5)
```

Training set accuracy (In-Sample)

I expect error rate to be less than 10%. Let's see our in-sample results:

```
ptraining <- predict(rfModel, Training)
print(confusionMatrix(ptraining, Training$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4464    0    0    0    0
##          B    0 3038    0    0    0
##          C    0    0 2738    0    0
##          D    0    0    0 2573    0
##          E    0    0    0    0 2886
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Validation set accuracy (Out-of-Sample)

Let's see our out-of-sample results:

```
pvalidation <- predict(rfModel, Validation)
print(confusionMatrix(pvalidation, Validation$classe))
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    3    0    0    0
##          B    0  755    2    0    0
##          C    0    1  682   10    1
##          D    0    0    0  633    1
##          E    0    0    0    0  719
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9928, 0.9973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9971   0.9844   0.9972
## Specificity            0.9989   0.9994   0.9963   0.9997   1.0000
## Pos Pred Value         0.9973   0.9974   0.9827   0.9984   1.0000
## Neg Pred Value         1.0000   0.9987   0.9994   0.9970   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1925   0.1738   0.1614   0.1833
## Detection Prevalence   0.2852   0.1930   0.1769   0.1616   0.1833
## Balanced Accuracy      0.9995   0.9970   0.9967   0.9921   0.9986
```

Test Set Prediction

Prediction of the test set:

```
ptest <- predict(rfModel, testing)
print(confusionMatrix(ptest, testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    3    0    0    0
##          B    0 3793    2    0    0
##          C    0    1 3420   10    1
##          D    0    0    0 3206    1
##          E    0    0    0    0 3605
##
## Overall Statistics
##
##                Accuracy : 0.9991
##                  95% CI : (0.9986, 0.9995)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9988
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   0.9994   0.9969   0.9994
## Specificity            0.9998   0.9999   0.9993   0.9999   1.0000
## Pos Pred Value         0.9995   0.9995   0.9965   0.9997   1.0000
## Neg Pred Value         1.0000   0.9997   0.9999   0.9994   0.9999
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1933   0.1743   0.1634   0.1837
## Detection Prevalence   0.2845   0.1934   0.1749   0.1634   0.1837
## Balanced Accuracy      0.9999   0.9994   0.9993   0.9984   0.9997
```

I obtained the error rate less than 0.001% on the test set. That's very good result.

Conclusion

Given that the model obtained using the initial approach appears to be highly successful, further exploration of the matter does not seem to be necessary.