Weight Lifting Exercise Prediciton

ray

2/5/2021

Summary

In this project, I used data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, and predict the manner in which they did the exercise. I preprocessed data with PCA, trained 3 random forests on training set and chose the best one with cross_validation. My best model gives a 95% accuracy on test set.

Data Preprocessing

First, read data. The test set contains only 20 examples, which will not be used until the end of this project.

```
library(caret)
library(dplyr)
library(randomForest)
trainSet <- read.csv("pml-training.csv", na.strings=c("","NA"))
testSet <- read.csv("pml-testing.csv", na.strings=c("","NA"))
```

In the training data set, there are rows with new_window variable being yes. These are processed data summarizing each time window. As we are interested in raw data, we get rid of them. As we do this, there are columns without any entry, so we can delete them. Also, variables such as use_name and time stamp seem irrelevant here, we discard them as well.

```
trainSet <- filter(trainSet, new_window == 'no')
trainSet <- select(trainSet, -(1:6))
trainSet <- Filter(function(x)!all(is.na(x)), trainSet)
trainSet$classe <- factor(trainSet$classe)
testSet <- select(testSet, -(1:6))
testSet <- Filter(function(x)!all(is.na(x)), testSet)
```

Then we can slice the training set into 3 groups – training data, validating data and testing data.

```
set.seed(33833)
inTrain <- createDataPartition(trainSet$classe, p=0.8, list=FALSE)
training <- trainSet[inTrain, ]
testing <- trainSet[-inTrain, ]
inTrain <- createDataPartition(training$classe, p=0.8, list=FALSE)
validating <- training[-inTrain,]
training <- training[inTrain,]
```

The data set contains more than 80 variables, all of which are meant to measure body activity. Naturally, they are highly correlated. This motivates us to processing the data with PCA.

```
preProc <- preProcess(training[,-54], method='pca', thresh=0.8)
trainingPC <- predict(preProc, training[, -54])
validatingPC <- predict(preProc, validating[, -54])
testingPC <- predict(preProc, testing[, -54])
```

PCA reduces the dimension of predictors to 13, while retaining 80% variation.

Training and Cross-validation

As it seems highly unlikely that our data are linearly separable, I choose to use random forest algorithm. The idea is to train 3 random forests with different tree numbers, and use validating data to choose the best-performing one.

```
set.seed(1234)
rf1 <- randomForest(training$classe ~ ., data = trainingPC, ntree = 10)
rf2 <- randomForest(training$classe ~ ., data = trainingPC, ntree = 20)
rf3 <- randomForest(training$classe ~ ., data = trainingPC, ntree = 40)
```

Now make predictions with these models on validating data and compare accuracy.

```
cfmat1 <- confusionMatrix(validating$classe, predict(rf1, validatingPC))
cfmat2 <- confusionMatrix(validating$classe, predict(rf2, validatingPC))
cfmat3 <- confusionMatrix(validating$classe, predict(rf3, validatingPC))
cfmat1$table
```

```
##          Reference
## Prediction   A   B   C   D   E
##          A 837  13  12  12   1
##          B  17 539  26   4   9
##          C  12  13 492  17   2
##          D  13  20  24 443   3
##          E   3   8  10  11 532
```

```
cfmat1$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##    0.925154572    0.905293750    0.915277399    0.934212866    0.287015945
## AccuracyPValue  McnemarPValue
##    0.000000000    0.001992376
```

```
cfmat2$table
```

```
##          Reference
## Prediction   A   B   C   D   E
##          A 847   5  10  10   3
##          B  13 551  24   1   6
##          C   4   7 508  15   2
##          D   9  12  27 452   3
##          E   2   2   7   3 550
```

```
cfmat2$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   0.9463065408   0.9320826124   0.9377394054   0.9540108691   0.2847380410
## AccuracyPValue  McnemarPValue
##   0.0000000000   0.0002505037
```

```
cfmat3$table
```

```
##          Reference
## Prediction   A   B   C   D   E
##          A 850   7   6  10   2
##          B   9 561  23   1   1
##          C   4   6 508  17   1
##          D   5  11  27 459   1
##          E   0   3   6   7 548
```

```
cfmat3$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   0.9521640091   0.9395117216   0.9440138774   0.9594387476   0.2824601367
## AccuracyPValue  McnemarPValue
##   0.0000000000   0.0001876256
```

Model 3 with the most trees has the best performance. So I choose model 3 as the final model.

Testing

Using testing data, I now estimate the out-of-example accuracy of my model.

```
testPred <- predict(rf3, testingPC)
testCf <- confusionMatrix(testing$classe, testPred)
testCf$table
```

```
##          Reference
## Prediction    A    B    C    D    E
##          A 1056    7   14   14    3
##          B   18  699   19    2    5
##          C    8    9  641   10    2
##          D    8    6   37  573    5
##          E    2    7    3    8  685
```

```
testCf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   0.9513147618   0.9384204246   0.9440273766   0.9579051795   0.2843009633
## AccuracyPValue  McnemarPValue
##   0.0000000000   0.0006776532
```

The model gives 95% accuracy on testing data, which is expected out-of-sample accuracy.

Applying on Test Cases

The 20 test cases do not have labels. Therefore, there is no way to decide accuracy on this prediction.

```
testPC <- predict(preProc, testSet[, -54])
predict(rf3, testPC)
```

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