Practical Machine Learning project

Abstract

In this project, a predictive model that determines the manner an exercise is done is built. Data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, who were asked to perform barbell lifts correctly and incorrectly in 5 different ways, is employed to predict whether a particular exercise is performed correctly or not.

Data retrieval and exploration

First, I downloaded the training dataset from the provided link and divided it into two subsets: one for training and building the model and the second subset to access out of sample error by cross-validation.

According to the rules of thumb for prediction study design, I partitioned the data into 60% training and 40% testing. Then I put aside the testing data and finish the model selection process using the training data. Before building the model, I made exploration on the training data. I also tried to understand the covariates (features) and their associations. The paper by Velloso et al. (2013) explains the covariates. The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Downloading data

download data which is to be used for training and testing

```
data1<-read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",na.strings=c("NA",""))


dim(data1)

names(data1)
```

Then, I divided the dataset into training and testing subsets.

```
for_training<-createDataPartition(y=data1$classe,p=0.6,list=FALSE)

training <-data1[for_training,]

testing <-data1[-for_training,]
```

Explore the training data

```
dim(training)
```

```
11776   160
```

```
names(training)
```

The training data has 160 columns (variables) and 11776 rows (observations). The predictand (dependent) variable is “classe”, which is the right most column.

Then, I removed the covariates which are not important for the machine learning algorithm development. These are the row number(X), user_name, raw_timestamp_part1, raw_timestamp_part2, cvtd_timestamp, new_window and num_window.

It is vital to keep in mind that the testing data is not used in any way for model development. The model building is done entirely based on the training data.

Remove data not important for model development

```
training<-training[,8:160]
```

Many of the covariates have lots of missing data. Those with missing data are removed.

```
non_missing_training<-apply(!is.na(training),2,sum)>=dim(training)[1]
training<-training[,non_missing_training]
```

```
dim(training)
 11776    53
```

After removing the variables with missing data, I am left with 52 features and the dependent variable (53 columns in total).

Predictive model building

Random forest, along with Boosting, is one of the most accurate classifiers. On many problems, the performance of random forests is very similar to boosting (Hastie et al., 2005)

I started with Random Forest and since the performance of random Forest is found to be very good, as I will show below, I did not proceed to Boosting. Random Forest is suitable for this project dataset because:

(a) It handles non-linearity

(b) It handles large inputs whose interactions is not understood

(c) It handles unscaled inputs and categorical data.

(d) It helps to visually examine how the predictors are contributing into the final model by extracting the model output trees.

set.seed(3333) this is set to ensure the reproducibility of the results

load necessary packages

library(caret)
library(ggplot2)

Build a Random Forest model:

rf_fit<-train(classe~.,data=training,method="rf", allowParallel=TRUE,prox=TRUE, trControl=trainControl(method="cv",number=10))

save(rf_fit, file="randomForest_model.RData_cv10")

I used random forest with 10 fold cross validation

The model run was performed in a high performance computer cluster and took hours.

Let us see the importance of the covariates.

importance <- varImp(rf_fit)

plot(importance, main = "Variable Importance of features")

or we can have a look at the top important features:

```
importance
```

```
 only 20 most important variables shown (out of 52)

                     Overall
roll_belt              100.0
pitch_forearm           57.3
yaw_belt                56.4
pitch_belt              45.4
magnet_dumbbell_z       43.8
roll_forearm            43.6
magnet_dumbbell_y       43.5
accel_dumbbell_y        21.4
accel_forearm_x         17.6
roll_dumbbell           17.5
magnet_dumbbell_x       17.5
accel_dumbbell_z        14.6
```

```
magnet_belt_z            14.3
accel_belt_z             14.2
magnet_forearm_z         14.0
total_accel_dumbbell     13.0
gyros_belt_z             11.7
magnet_belt_y            11.3
yaw_arm                  11.3
magnet_belt_x            10.2
```

Importance of covariates from the Random Forest.

Then let us see the confusion matrix to understand how the model performs:

```
options(digits=2)
```

```
rf_fit$finalModel$confusion
```

	A	B	C	D	E	class.error
A	3344	3	1	0	0	0.00119
B	19	2248	10	2	0	0.01360
C	0	15	2030	9	0	0.01168
D	0	1	32	1894	3	0.01865
E	0	1	3	5	2156	0.00416

The confusion matrix shows that the model is doing a good job in fitting the training data set.

The summary of the model is below:

```
rf_fit
```

Random Forest

11776 samples
52 predictors
5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 10598, 10600, 10598, 10600, 10598, 10598, ...

Resampling results across tuning parameters:

mtry	Accuracy	Kappa	Accuracy SD	Kappa SD
2	0.989	0.987	0.00231	0.00292
27	0.990	0.987	0.00325	0.00412
52	0.977	0.971	0.00487	0.00617

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 27.

Cross-validation

Similar to what was done to the training data, the first eight columns and those with missing covariates are removed from the testing data and the rest covariates are used for prediction.

```
testing <-testing[,8:160]
```

Similar to what was done in the training data, remove covariates with NAs

```
non_missing_testing<-apply(!is.na(testing),2,sum)==dim(testing)[1]

testing<-testing[,non_missing_testing]

dim(testing)
```

predict using the model built in with the training data

```
prediction<-predict(rf_fit,testing)

save(prediction, file="randomForest_model_cv10_p60_testing.RData")

accuracy = sum(prediction==testing$classe)/length(testing$classe)

accuracy
```

0.9945195

we see that the accuracy of the testing dataset is very good, with error of only 0.55%
which shows that Random Forest is a good model for the project and hence I did not need to try other models.

To predict with the provided test data:

Now, let us download the data provided for prediction submission

download test data to predict using the model built above Similar to the training dataset,
remove the first seven columns which are not covariates for the model and also remove covariates with missing values

```
test_given<-read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",na.strings=c("NA",""))

test_given<-test_given[,8:160]

non_missing_test<-apply(!is.na(test),2,sum)>=dim(test)[1]

test_given<-test_given[,non_missing_test]

 predict for the test data and save the results

prediction = predict(rf_fit,test_given)
```

As the feedback from the online submisssion shows, the model predicts the test data 100% accurate.

saving prediction results

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(prediction)
```

.

References

Hastie, T., R. Tibshirani, J. Friedman, and J. Franklin, 2005: The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, **27,** 83-85.

Velloso, E., A. Bulling, H. Gellersen, W. Ugulino, and H. Fuks, 2013: Qualitative activity recognition of weight lifting exercises. *Proceedings of the 4th Augmented Human International Conference*, ACM, 116-123.