PracticalMachineLearning in Human Activity Recongnition

Hanifa

Sunday, March 22, 2015

Synopsis

The data (http://groupware.les.inf.puc-rio.br/har#literature (http://groupware.les.inf.puc-rio.br/har#literature)) comes from sensors attached to six random users to monitor the quality of the exercises they perform. The idea is that using this sensors, we'd like to predict whether a user has performed his exercises correctly. After cleansing the data (by removing irrelevant variables), we perform PCA to identify the dependant variables which contribute the greatest variance in the dataset. Since linearity couldnt be detected, we stuck to tree based models to predict our outcome and we were able to get a high accuracy.

Data Loading

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

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

```
## Warning: package 'ggplot2' was built under R version 3.1.2
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.1.3
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(ggplot2)

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

```
dim(train.raw)
```

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

```
dim(test.raw)
```

```
## [1]  20 160
```

There is a lot of dependant variables,it is neccesary to remove the irrelevant ones.

Data cleansing

Irrelevant variable and variables having NA are removed. A more elegant solution would be to impute these values for NA columns, however we took the shorter solution of just ignoring these columns.

```
cleanse.records <- function(records.raw){
  #Remove unrelated columns
  records.raw<-records.raw[,!grepl("^X|user_name|timestamp|window",names(records.raw))]
  #Remove columns with NA values. Another alternative might be to impute those values
  #which are NA.We will take the easier alternative
  records.raw<-records.raw[,colSums(is.na(records.raw))==0]
  #Factor variables have to be removed to apply PCA later on.
  records.raw <- records.raw[, sapply(records.raw,is.numeric) |
                               colnames(records.raw)=="classe"]
}

train.cleanse <- cleanse.records(train.raw)
test.cleanse <- cleanse.records(test.raw)

dim(train.cleanse)
```

```
## [1] 19622    53
```

Preprocessing with PCA

We use PCA to identify the top components which give at least 90% of the overall variance to the dataset.90% of the overall variance could be explained with just 19 variables.

```
preProc <- preProcess(train.cleanse[,-53],method="pca",thresh=0.9)
train.PC <- predict(preProc,train.cleanse[,-53])
train.PC$classe<-train.raw$classe
#ignore the problem_id for the PCA
test.PC <-predict(preProc,test.cleanse[,-53])
test.PC$problem_id<-test.raw$problem_id

dim(train.PC)
```

```
## [1] 19622    20
```

Data spliting for validation (testing/training)

```
set.seed(2011)
inTrain <- createDataPartition(train.PC$classe, p=0.70, list=F)
training <- train.PC[inTrain, ]
testing <- train.PC[-inTrain, ]
```

Exploratory analysis

Pls refer to the appendix 1. for the pairwise plots, From the plots the relationships seems quite random (not showing linearity), In this absence, we can use tree based models.

Decision Trees

A single tree based model gives poor accuracy of just over 30%. We will reject this model. The tree model is being visualized in the appendix 2.

```
modRpart<-train(classe~.,data=training,method="rpart")
```

```
## Loading required package: rpart
```

```
predictRpart<-predict(modRpart,testing)
confusionMatrix(predictRpart,testing$classe)$overall[1]
```

```
## Accuracy
## 0.379949
```

Random Forest

We build a random forest with 10 trees and get over 97% accuracy.In appendix 3, we also see the importance ranking of the variables. We can conclude that variable PC1 need not be the most important variable,even though it contributes the highest variablity to the dataset.It is recommended to use a large number of trees(>200), however since it was taking too long,I just used 10 here.

```
modelRf <- train(classe ~ ., data=training, method="rf", ntree=10)
```

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

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
predictRf<-predict(modelRf,testing)
confusionMatrix(predictRf,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1622   37   12    8   10
##          B   16 1049   29   12   16
##          C   17   29  962   38   24
##          D   12    4   18  899   11
##          E    7   20    5    7 1021
##
## Overall Statistics
##
##                Accuracy : 0.9436
##                  95% CI : (0.9374, 0.9493)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9286
##  Mcnemar's Test P-Value : 0.0001051
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9689   0.9210   0.9376   0.9326   0.9436
## Specificity            0.9841   0.9846   0.9778   0.9909   0.9919
## Pos Pred Value         0.9603   0.9349   0.8991   0.9523   0.9632
## Neg Pred Value         0.9876   0.9811   0.9867   0.9868   0.9874
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2756   0.1782   0.1635   0.1528   0.1735
## Detection Prevalence   0.2870   0.1907   0.1818   0.1604   0.1801
## Balanced Accuracy      0.9765   0.9528   0.9577   0.9617   0.9678
```

Predicting the test set

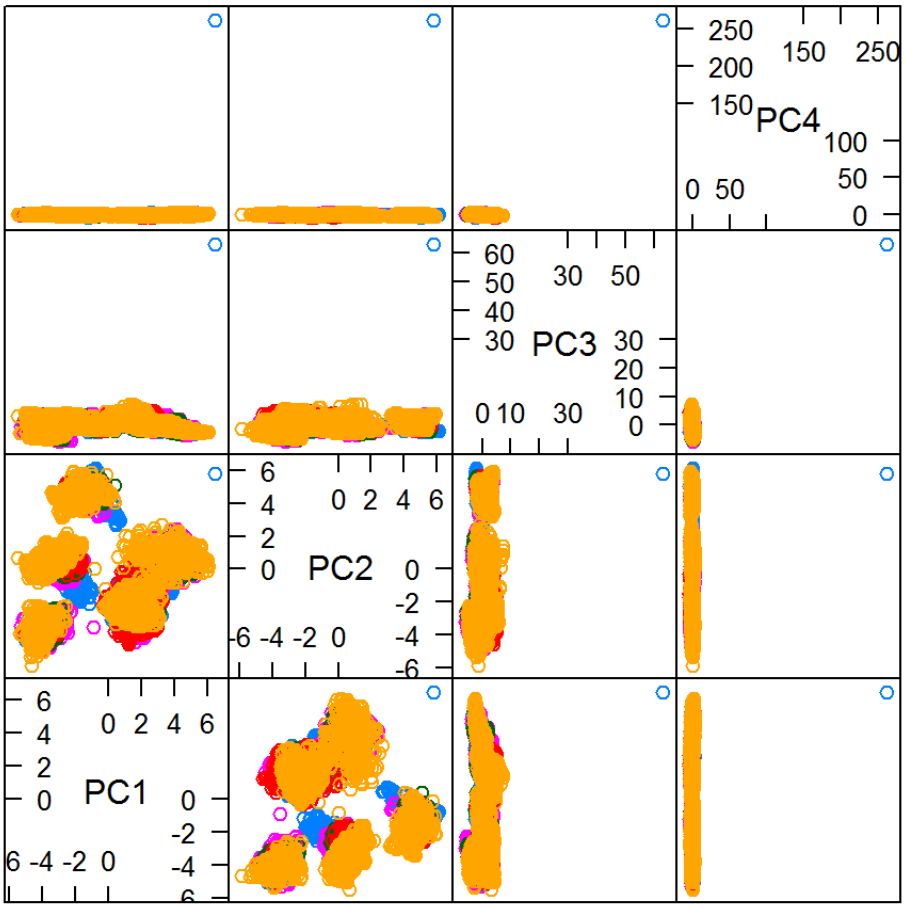
```
test.PC$classe<-predict(modelRf,test.PC[-20])
test.PC[,c("problem_id","classe")]
```

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

Appendix

Appendix 1

```
featurePlot(x=training[,c(1,2,3,4)],y=training$classe,plot="pairs")
```



Scatter Plot Matrix

Appendix 2

```
#fancyRpartPlot(modRpart$finalModel)
```

Appendix 3

```
#varImpPlot(modelRf$finalModel)
```