Practical Machine Learning Final Project

mt

Saturday, September 27, 2015

This research proposed an efficient method in classification of Human Activity Recognition tasks.

The evaluated tuned models show higher than 99 percent mean accuracy and gained more training and testing accuracy in comparison to previous studies.

Human Activity Recognition(HAR) is a key research area in last 8 years and has broad range of applications in smart human activity recognition.

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

all needed libraries loaded as below:

```
setwd(file.path("D:", "sbu", "Rlearning",
                "practical machine learning", "finalproject"))
library(randomForest); library(gbm);
library(caret); library(doParallel);
require(foreach); library(adabag); library(xtable);
```

Here gbm for Boosting adabag for bagging and randomforest package have been introduced. in this study class A corresponded to specified execution of exercise and other 4 classes showed mistakes in execution of exercise. All patients were between the ages of 20-28 and surveillance have been done by experienced weight lifter[**?**].

First the dataset should be cleaned from not assigned values.

```
set.seed(123)
data <- read.csv("pml-training.csv", na.strings=c("#DIV/0!"))

## Data set cleaning procedure starts from here
CleanedData <- data
for(i in c(8:ncol(CleanedData)-1)) {CleanedData[,i] = as.numeric(as.character(CleanedData[,i]))}

## removing all features with not assigned values
featuresnames <- colnames(CleanedData[colSums(is.na(CleanedData)) == 0])[-(1:7)]
```

75 percent of data asigned for training and others are for testing sets as below.

```
features <- CleanedData[featuresnames]
TrainData <- createDataPartition(y=features$classe, p=3/4, list=FALSE )
training <- features[TrainData,]
testing <- features[-TrainData,]
```

Random forest provides an improvement over bagged trees, each time a split have been considered on a tree, a random selection of m predictors are chosen as split candidates from the full set of p predictors

```
mtry<-11 # Number of features at each splits
treenumber<-50 # Number of trees at each split
fit2 <- foreach(ntree=treenumber, .combine=randomForest::combine, .packages='randomForest') %dopar%
        randomForest(training[-ncol(training)], mtry=mtry, training$classe, ntree=ntree)
```

Out of Bag error is a way of error estimations of test results in bagged models. #The key idea here is that in bootstrap, sampling occurs on two-thirds of observations and another one-thirds that not used in fitting, could be referred as out of bag samples and equivalent error called out of sample error.

Typically best number of evaluated features at each split could be assigned by the value $m \approx \sqrt{p}$. Random forest could be the best and fastest decision tree model in big classification problems like pattern recognition by assigning appropriate tuning parameters.

Generalized Boosted Model(gbm) is another package in R that implements Freund and Schapire's adaboost algorithm.

In contrary to bagging and Random Forest models the big number of trees in Boosting model could cause overfitting.

learning rate in Boosting model known as shrinkage value(λ), this mechanism controls the rate that model could learn, this value depends on the case study.

Number of splits(d) in Boosting model controls the complexity of the boosted ensemble and generally d shows the interaction depth.

```r
library(gbm);
fit3 <-gbm(classe~., data = training, var.monotone = NULL,
           n.trees = treenumber, interaction.depth = 16, n.minobsinnode = 10            ,shrinkage = 0.:
           cv.folds=0, keep.data = TRUE, verbose = "CV",
           class.stratify.cv=NULL, n.cores = NULL)
```

Test and Train Accuracy

Random forest train and test accuracy is as below

```r
TestPred <- predict(fit2, newdata=testing)
TrainPred <- predict(fit2, newdata=training)
RFacctest <- with(testing,mean((classe==TestPred))) ##misclassification
RFacctrain <- with(training,mean((classe==TrainPred)))
confusionMatrix(TestPred, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    2    0    0    0
##          B    1  945    9    0    0
##          C    0    2  844    6    0
##          D    0    0    2  796    1
##          E    0    0    0    2  900
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9925, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9958   0.9871   0.9900   0.9989
## Specificity            0.9994   0.9975   0.9980   0.9993   0.9995
## Pos Pred Value         0.9986   0.9895   0.9906   0.9962   0.9978
## Neg Pred Value         0.9997   0.9990   0.9973   0.9981   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1927   0.1721   0.1623   0.1835
## Detection Prevalence   0.2847   0.1947   0.1737   0.1629   0.1839
## Balanced Accuracy      0.9994   0.9966   0.9926   0.9947   0.9992
```

Boosting model train and test accuracy is as below

```r
TestPred <- predict(fit3, newdata=testing, n.trees=treenumber,type="response")
TrainPred <- predict(fit3, newdata=training, n.trees=treenumber,type="response")
class <- c("A","B","C","D","E")
gbmtestpre<-rep(0, nrow(TestPred))
gbmtrainpre<-rep(0, nrow(TrainPred))
testmaxpre<-apply(TestPred, 1, max)
trainmaxpre<-apply(TrainPred, 1, max)
for (k in 1:nrow(TestPred)){
        gbmtestpre[k] <- class[(TestPred[k,,]==testmaxpre[k])]
}
for (k in 1:nrow(TrainPred)){
        gbmtrainpre[k] <- class[(TrainPred[k,,]==trainmaxpre[k])]
}
boostacctest<-with(testing,mean(classe==gbmtestpre))
boostacctrain<-with(training,mean(classe==gbmtrainpre))
confusionMatrix(gbmtestpre, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1391    1    0    0    0
##          B    3  944    6    0    0
##          C    1    4  845   12    0
##          D    0    0    4  789    2
##          E    0    0    0    3  899
##
## Overall Statistics
##
##                Accuracy : 0.9927
##                  95% CI : (0.9899, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9907
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9971   0.9947   0.9883   0.9813   0.9978
## Specificity            0.9997   0.9977   0.9958   0.9985   0.9993
## Pos Pred Value         0.9993   0.9906   0.9803   0.9925   0.9967
## Neg Pred Value         0.9989   0.9987   0.9975   0.9963   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2836   0.1925   0.1723   0.1609   0.1833
## Detection Prevalence   0.2838   0.1943   0.1758   0.1621   0.1839
## Balanced Accuracy      0.9984   0.9962   0.9921   0.9899   0.9985
```

so finally random forest model shows

```
## [1] 0.9949021
```

test accuracy and

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

train accuracy. Boosting model shows

```
## [1] 0.9926591
```

test accuracy and

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

train accuracy. finally all of 20 extra tests have been evaluated and shows correct answers in both two proposed algorithms.