Assignment: Human Activity Recognition Analysis

Wataru Miura

In this analysis, we try to apply some machine learning algorithms on the Human Activity Recognition dataset provided by Groupware. Out of 4 ML algorithms we apply, which is bagging with classification trees, neural network, logistic regression, and support vector machines, bagging with classification trees yields the highest accuracy rate, over 95%. All the model except neural network is ensembled to give a better prediction.

1. Data preprocessing

After loading the data into memory, preprocess the data, there's some blank cells, NA, NaN, and Inf. These data point actually should be 0, thus enabling us to analyze the data. In addition, preprocess the testing data in order to have correct output too.

```
train <- read.csv("pml-training.csv", stringsAsFactors = FALSE)
validation <- read.csv("pml-testing.csv")
temp <- train[, c(1:7, 160)]
temp.a <- train[, -c(1:7, 160)]
for (i in 1:ncol(temp.a)) {if (class(temp.a[, i]) == "character") {
suppressWarnings(temp.a[, i] <- as.numeric(temp.a[, i])) }}
train <- cbind(temp, temp.a)
train[is.na(train)] <- 0
train[is.nan(unlist(train))] <- 0
train[is.infinite(unlist(train))] <- 0
temp <- validation[, c(1:7, 160)]
temp.a <- validation[, -c(1:7, 160)]
for (i in 1:ncol(temp.a)) {if (class(temp.a[, i]) == "character") {temp.a[, i] <-
as.numeric(temp.a[, i])}}
validation <- cbind(temp, temp.a)
validation[is.na(validation)] <- 0
validation[is.nan(unlist(validation))] <- 0
validation[is.infinite(unlist(validation))] <- 0
```

2. Data Analysis

I. Split the data into training & testing data

Notice that the ratio of training data to testing data is 0.2, not typical 0.75, that's because my old-fashioned Mac cannot train data on large scale. Before applying machine learning algorithms to train our model, let us first tune the cross-validation parameters. We expect our out-of-sample error to be low because 5-fold CV should take its effect and avoid overfitting.

```
library(caret)
set.seed(32323)
inTrain <- createDataPartition(y = train$classe, p = 0.2, list = FALSE)
training <- train[inTrain, ]
testing <- train[-inTrain, ]
p <- training[, -1:-7]
p[, 1] <- as.factor(p[, 1])
fitControl <- trainControl(method = "cv", number = 5, returnResamp = "all")
```

II. Train models with the training data

Train our model by applying some machine learning techniques using “Bagging with trees”. In the result, this model shows very high accuracy rate, 96.64%. In fact, this one is sufficient for predicting our testing data, however try some more models and mix them up in order to improve our prediction ability. Applied “logistic regression with boosting”. Boosting is required in order to make our model better at prediction and lower the variance. This model gives 88.34% accuracy rate, a very efficient model as well. let's try support vector machines with boosting. This model gives 76.34% accuracy rate. All models are evalued, now we turn to ensemble them.

```
set.seed(32323)
model.treebag <- train(classe ~ ., method = "treebag", data = p, trControl =
fitControl)
## Loading required package: ipred
## Loading required package: plyr
result2 <- predict(model.treebag, newdata = testing)
confusionMatrix(result2, testing$classe)
## Confusion Matrix and Statistics
##           Reference
## Prediction    A    B    C    D    E
##          A 4375  113    6    3    6
##          B   52 2846   57    9   22
##          C   31   63 2641   61   25
##          D    1    3   28 2498   25
##          E    5   12    5    1 2807
## Overall Statistics
##                Accuracy : 0.966
##                  95% CI : (0.963, 0.969)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.957
##  Mcnemar's Test P-Value : 6.84e-16
## Statistics by Class:
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.980    0.937    0.965    0.971    0.973
## Specificity             0.989    0.989    0.986    0.996    0.998
## Pos Pred Value          0.972    0.953    0.936    0.978    0.992
## Neg Pred Value          0.992    0.985    0.993    0.994    0.994
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.279    0.181    0.168    0.159    0.179
## Detection Prevalence    0.287    0.190    0.180    0.163    0.180
## Balanced Accuracy       0.984    0.963    0.976    0.983    0.986
set.seed(32323)
model.logit <- train(classe ~ ., method = "LogitBoost", data = p, trControl = 
fitControl)
## Loading required package: caTools
result3 <- predict(model.logit, newdata = testing)
confusionMatrix(result3, testing$classe)
## Confusion Matrix and Statistics
##           Reference
## Prediction    A    B    C    D    E
##          A 3958  283   62  100   18
##          B   69 1916  130   35   43
##          C   50  159 1809  121   53
##          D   25   32   98 1743   58
##          E   17   62   44  101 2396
## Overall Statistics
##                Accuracy : 0.883
##                95% CI : (0.878, 0.889)
##    No Information Rate : 0.308
##    P-Value [Acc > NIR] : <2e-16
##                 Kappa : 0.851
##  Mcnemar's Test P-Value : <2e-16
## Statistics by Class:
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.961    0.781    0.844    0.830    0.933
## Specificity             0.950    0.975    0.966    0.981    0.979
## Pos Pred Value          0.895    0.874    0.825    0.891    0.915
## Neg Pred Value          0.982    0.952    0.970    0.969    0.984
## Prevalence              0.308    0.183    0.160    0.157    0.192
## Detection Rate          0.296    0.143    0.135    0.130    0.179
## Detection Prevalence    0.330    0.164    0.164    0.146    0.196
## Balanced Accuracy       0.955    0.878    0.905    0.906    0.956
library(kernlab)
set.seed(32323)
model.svm <- suppressWarnings(train(classe ~ ., data = p, method =
"svmRadialCost",
   trControl = fitControl))
result4 <- predict(model.svm, newdata = testing)
confusionMatrix(result4, testing$classe)
## Confusion Matrix and Statistics
##           Reference
## Prediction    A    B    C    D    E
##          A 3951  380  155  117  125
##          B  100 2153  254  111  339
##          C  174  274 2003  337  271
##          D  211   91  251 1920  196
##          E   28  139   74   87 1954
## Overall Statistics
##              Accuracy : 0.763
##                95% CI : (0.757, 0.77)
##    No Information Rate : 0.284
##    P-Value [Acc > NIR] : <2e-16
##                 Kappa : 0.7
##  Mcnemar's Test P-Value : <2e-16
## Statistics by Class:
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.885    0.709    0.732    0.747    0.677
## Specificity             0.931    0.936    0.919    0.943    0.974
## Pos Pred Value          0.836    0.728    0.655    0.719    0.856
## Neg Pred Value          0.953    0.931    0.942    0.950    0.931
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.252    0.137    0.128    0.122    0.124
## Detection Prevalence    0.301    0.188    0.195    0.170    0.145
## Balanced Accuracy       0.908    0.823    0.825    0.845    0.826
```

III. Ensemble learning algorithm & predict

```
pred1 <- predict(model.treebag, validation)
pred2 <- predict(model.logit, validation)
pred3 <- predict(model.svm, validation)
ensemble <- data.frame(pred1, pred2, pred3)
ensemble
```

Results

So here get reportedly 95% accuracy rate on the final outcome.

```
answers = c("B", "A", "B", "A", "A", "E", "D", "B", "A", "A", "A", "C", "B",
    "A", "E", "E", "A", "B", "B", "B")
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)
    }
}
setwd("answer")
pml_write_files(answers)
```