Practical machine learning final project – using personal activity data to predict the quality of the exercise

X Guo

June 12, 2016

Summary

This report is the final course project of Practical Machine Learning by JHU. The data source can be found [here] (http://groupware.les.inf.puc-rio.br/har). The dataset were collected from from 6 adults who were performing unilateral dumbbell biceps curl. The measurement includes the data detected from arm, belt, dumbell and forearm. The goal is to make prediction on how well they perfomed the exercise. This was classified as 'classe' in the dataset. The report will be focusing on building machine learning model, performing cross validation, selecting the robust model and making prediction.

- Classfication tree with cross validation
- Boosting with cross validation
- Model evaluation
- Prediction

Results

- In this report we've build 3 model; 2 were classification tree models; 1 was the boosting models.
- In each model we've applied 6-fold cross validation.
- The boosting model we created is evaluated as the best model.
- Prediction was made on the final boosting model.
- The out of sample error estimation is 0.23%.

Loading and cleaning data

We first load and clean the data. We have select several variables in our training set. The selection criteria of predictor variables are:

1. Number of NA is zero
2. Non-zero variance variables

```
library(caret); library(cvTools)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

## Loading required package: robustbase
```

```r
training = read.csv(file = 'pml-training.csv', header = TRUE, sep = ",")
testing = read.csv(file = 'pml-testing.csv', header = TRUE, sep = ",")
dim(training)
dim(testing)
##calculate the NA's in each variables
number.NA = rep(0, dim(training)[2])

for (i in 1:dim(training)[2]) {
    number.NA[i] = sum(is.na(training[, i]))
}
number.NA

##we will not use NA = 601 variables for prediction
good.var = which(number.NA == 0)
good.training = training[, good.var]
dim(good.training)
kurto = which(substr(names(good.training), 1, 8) == 'kurtosis')
ske = which(substr(names(good.training), 1, 4) == 'skew')
maxi = which(substr(names(good.training), 1, 3) == 'max')
mini = which(substr(names(good.training), 1, 3) == 'min')
amp = which(substr(names(good.training), 1, 9) == 'amplitude')
bad = c(kurto, ske, maxi, mini, amp, 1:7)
good.training2 = good.training[, -bad]
```

good.training2 contains the cleaned data we are going to work on. We can use nzv() to varify the variables.

```r
nearZeroVar(good.training2, saveMetrics= TRUE)
```

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt             1.101904     6.7781062   FALSE FALSE
## pitch_belt            1.036082     9.3772296   FALSE FALSE
## yaw_belt              1.058480     9.9734991   FALSE FALSE
## total_accel_belt      1.063160     0.1477933   FALSE FALSE
## gyros_belt_x          1.058651     0.7134849   FALSE FALSE
## gyros_belt_y          1.144000     0.3516461   FALSE FALSE
## gyros_belt_z          1.066214     0.8612782   FALSE FALSE
## accel_belt_x          1.055412     0.8357966   FALSE FALSE
## accel_belt_y          1.113725     0.7287738   FALSE FALSE
## accel_belt_z          1.078767     1.5237998   FALSE FALSE
## magnet_belt_x         1.090141     1.6664968   FALSE FALSE
## magnet_belt_y         1.099688     1.5187035   FALSE FALSE
## magnet_belt_z         1.006369     2.3290184   FALSE FALSE
## roll_arm             52.338462    13.5256345   FALSE FALSE
## pitch_arm            87.256410    15.7323412   FALSE FALSE
## yaw_arm              33.029126    14.6570176   FALSE FALSE
## total_accel_arm       1.024526     0.3363572   FALSE FALSE
## gyros_arm_x           1.015504     3.2769341   FALSE FALSE
## gyros_arm_y           1.454369     1.9162165   FALSE FALSE
## gyros_arm_z           1.110687     1.2638875   FALSE FALSE
## accel_arm_x           1.017341     3.9598410   FALSE FALSE
## accel_arm_y           1.140187     2.7367241   FALSE FALSE
## accel_arm_z           1.128000     4.0362858   FALSE FALSE
## magnet_arm_x          1.000000     6.8239731   FALSE FALSE
```

```
## magnet_arm_y          1.056818     4.4439914   FALSE FALSE
## magnet_arm_z          1.036364     6.4468454   FALSE FALSE
## roll_dumbbell         1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell        2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell          1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell  1.072634     0.2191418   FALSE FALSE
## gyros_dumbbell_x      1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y      1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z      1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x      1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y      1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z      1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x     1.098266     5.7486495   FALSE FALSE
## magnet_dumbbell_y     1.197740     4.3012945   FALSE FALSE
## magnet_dumbbell_z     1.020833     3.4451126   FALSE FALSE
## roll_forearm         11.589286    11.0895933   FALSE FALSE
## pitch_forearm        65.983051    14.8557741   FALSE FALSE
## yaw_forearm          15.322835    10.1467740   FALSE FALSE
## total_accel_forearm   1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x       1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y       1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z       1.122917     1.5645704   FALSE FALSE
## accel_forearm_x       1.126437     4.0464784   FALSE FALSE
## accel_forearm_y       1.059406     5.1116094   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x      1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y      1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z      1.000000     8.5771073   FALSE FALSE
## classe                1.469581     0.0254816   FALSE FALSE
```

Classification tree: rpart.cv

Here we use 'rpart' and 'rpart2' methods to train the models.

- Cross validation is done by 'cv' method in the trainControl; we choose 6-fold cross validation.

```
set.seed(999)
trCtrl <- trainControl(method = 'cv', number = 6, summaryFunction=defaultSummary)
Grid <- expand.grid(cp = seq(0, 0.05, 0.005))
fit.cp <- train(classe~., data = good.training2, method = 'rpart', trControl = trCtrl,tuneGrid = Grid)
plot(fit.cp)
#plot(varImp(fit.cp))

###

set.seed(999)
Grid2 <- expand.grid(.maxdepth = seq(10, 30, 5))
fit.rpart2 <- train(classe~., data = good.training2, method = 'rpart2', trControl = trCtrl,  tuneGrid =
plot(fit.rpart2)
#plot(varImp(fit.rpart2))
```

Here we see that the best complexity paramter is 0.00 for the first rpart model.

Here we see that the optimal tree depth is about 30 for the rpart2 model.

Boosting: gbm and cross validation

We use gbm method to build the boosting model.

- We set the parameter of cross validation to 6-fold.
- We set the maximum iteration to 100.

```
set.seed(888)
trCtl <- trainControl(method = 'cv', number = 6, summaryFunction=defaultSummary)
Grid <- expand.grid( n.trees = seq(5, 100, 3), interaction.depth = c(10), shrinkage = c(0.1), n.minobsi
fit.gbm <- train(classe~., data = good.training2, method = 'gbm', trControl = trCtl, tuneGrid = Grid)
plot(fit.gbm)
#plot(varImp(fit.gbm))
```

We find that the accuracy has reached to a pleatau as the iteration increases to 100.

Model evaluation

```
compare = resamples(list(rtree1 = fit.cp, rtree2 = fit.rpart2, boost = fit.gbm))
bwplot(compare)
```

The figure shows that boost model has the best accuracy compared with other two tree models.

Prediction

Based on the comparison, we choose the boosting model (with 98 iterations, 10 interaction depth, 0.1 shrinkage and 20 minimal number of observation) as our final model.

- We use 6-fold cross validation to get several test sets from the original training dataset.
- CV samples generated by createFolds() are used for evaluation the out of sample error

```
pred1 <- predict(fit.gbm, testing, n.trees = 98)
pred1

set.seed(888)


fld = createFolds(good.training2$classe, k = 6)
Est.error = rep(0, 6)
##outOfSampleError.accuracy <- sum(predictions == testValidateData$classe)/length(predictions)
for (k in 1:6) {
    test = good.training2[unlist(fld[k]), ]
    a.pred = predict(fit.gbm, n.trees = 98, test)
    accur = sum(as.numeric(a.pred) == as.numeric(test$classe))/length(a.pred)
    error = 1 - accur
    Est.error[k] = error
}
Estimation = mean(Est.error)*100
paste0("Out of sample error estimation: ", round(Estimation, digits = 2), "%")
```

```
## [1] "Result: B A B A A E D B A A B C B A E E A B B B"
```

```
## [1] "Out of sample error estimation: 0.23%"
```