DMC@ISU: The 2015 Iowa State University Data Mining Cup Team

Weighted Random Forest

Spring 2015, A Team as Strong as Steel

Last Day: May 19, 2015

I am using the following packages:

```
library(magrittr)
library(dplyr)
library(reshape2)
library(tidyr)
library(lubridate)
library(ggplot2)
library(directlabels)
library(rCharts)
library(xtable)
library(foreach)
library(gtools)
library(knitr)
library(utils)
source("~/dmc2015/ian/R/renm.R")
```

My working directory is set to `~/dmc2015/ian/`.

0.1 Curating and Cross Validating

The reason features should be removed from an "large data" approach to a problem is if they are dominated by better features.
However, when you have as many features as we do at the moment it can be difficult to
I am starting with **set 1**

0.2 Load feature matrix

```
## long
f1 = readRDS("../data/featureMatrix/featMat_based-on-HTVset1_LONG_ver0.3.rds")

## wide
d1 = readRDS("../data/featureMatrix/featMat_based-on-HTVset1_WIDE_ver0.3.rds")

# estimate weights from the historical data:
HTVset = readRDS("~/dmc2015/data/featureMatrix/HTVset1.rds")

## Baseline basketValue
sum((d1$validation$y$basketValue - mean(HTVset$H$basketValue))^2)/mean(d1$validation$y$basketValue)^2

## [1] 5442.26

## baseline coupons 1
sum((d1$validation$y$coupon1Used - mean(HTVset$H$coupon1Used))^2)/mean(d1$validation$y$coupon1Used)^2

## [1] 4310.565
## baseline coupons 2
sum((d1$validation$y$coupon2Used - mean(HTVset$H$coupon2Used))^2)/mean(d1$validation$y$coupon2Used)^2

## [1] 6568.962

## baseline coupons 3
sum((d1$validation$y$coupon3Used - mean(HTVset$H$coupon3Used))^2)/mean(d1$validation$y$coupon3Used)^2

## [1] 7157.268
```

0.3 Check the data

```
## isolate the X and y for set 1
Xn = f1$train$X
yn = f1$train$y

## remove the naive columns
Xn = Xn[, !grepl("naive", names(Xn))]

## keep the validation sets
Xv = f1$validation$X
yv = f1$validation$y
```

0.4 How do we estimate the weights? Bayes

We need an estimate of the mean of each couponUsed column. This can be accomplished in a Bayesian sense. I believe that coupons are used about 20% of the time in this data, and that there is less than a 5% chance that coupons are actually used less than 13% of the time or more than 27% of the time. This give me the following choice for α and β:

```
# prior estimates: mean of coupon use at .2, F(.025) = .1337, F(.975) =
# .2758
alpha.est = 24
beta.est = 4 * alpha.est
qbeta(0.025, alpha.est, beta.est)

## [1] 0.1337019

qbeta(0.975, alpha.est, beta.est)

## [1] 0.2757604

alpha.est

## [1] 24

beta.est

## [1] 96

alpha.est/(alpha.est + beta.est)

## [1] 0.2
```

This gives us the following posterior for p_1, p_2, p_3:

```
## Historical estimates of p1,p2,p3:
p1 = (sum(HTVset$H$coupon1Used) + alpha.est)/(alpha.est + beta.est + nrow(HTVset$H))
p2 = (sum(HTVset$H$coupon2Used) + alpha.est)/(alpha.est + beta.est + nrow(HTVset$H))
p3 = (sum(HTVset$H$coupon3Used) + alpha.est)/(alpha.est + beta.est + nrow(HTVset$H))

p1
```

```
## [1] 0.2263798
```

```
p2
```

```
## [1] 0.1884939
```

```
p3
```

```
## [1] 0.1730589
```

We can weight our responses and hopefully use this to get better estimates by unweighting. I am only using similarity columns and the one way loglikelihood columns.

```
## lets try a small random forest
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## get the similarity columns
sim_columns = c(3, grep("sim_", names(Xn)))

## get single way llrs:
llr1_columns = which(grepl("llr", names(Xn)) & !grepl("X", names(Xn)))

# The predictor and response columns
Xrf = Xn[, c(sim_columns, llr1_columns)]
yrf = yn[, "couponUsed"]

# unweighted error
unweighted.rf = randomForest(Xrf, y = yrf, ntree = 5000, mtry = 11, replace = TRUE,
    maxnodes = 100)
```

```
## Warning in randomForest.default(Xrf, y = yrf, ntree = 5000, mtry = 11,
## replace = TRUE, : The response has five or fewer unique values.  Are you
## sure you want to do regression?
```

```
# weighted error
yrfw = yn[, "couponUsed"] * rep(c(1/p1, 1/p2, 1/p3), times = nrow(Xrf)/3)
weighted.rf = randomForest(Xrf, y = yrfw, ntree = 5000, mtry = 11, replace = TRUE,
    maxnodes = 100)
```

```
## Warning in randomForest.default(Xrf, y = yrfw, ntree = 5000, mtry = 11, :
## The response has five or fewer unique values.  Are you sure you want to do
## regression?
```

0.4.1 Comparing Errors:

Training Set Results

```
# Unweighted Loss Aspects
uw.fitted = predict(unweighted.rf, newdata = Xrf, type = "response")
uw.fitted.loss = colSums(matrix((yrf - uw.fitted)^2, ncol = 3, byrow = TRUE))
uw.scaled.fitted.loss = uw.fitted.loss/(colMeans(matrix(yrf, ncol = 3, byrow = TRUE))^2)

# Weighted Loss Aspects
w.fitted = predict(weighted.rf, newdata = Xrf, type = "response") * rep(c(p1,
    p2, p3), times = nrow(Xrf)/3)
w.fitted.loss = colSums(matrix((yrf - w.fitted)^2, ncol = 3, byrow = TRUE))
w.scaled.fitted.loss = w.fitted.loss/(colMeans(matrix(yrf, ncol = 3, byrow = TRUE))^2)

message("loss unweighted rf (no col weights):", paste(c(uw.fitted.loss, sum(uw.fitted.loss)),
    collapse = " | "))
```

```
## loss unweighted rf (no col weights):381.548741744449 | 324.549008499592 | 292.502649479028 | 998.600
```

```
message("loss unweighted rf (col weights):", paste(c(uw.scaled.fitted.loss,
    sum(uw.scaled.fitted.loss)), collapse = " | "))
```

```
## loss unweighted rf (col weights):6376.80003449942 | 8651.8583427113 | 10498.8488246348 | 25527.5072
```

```
message("loss weighted rf (no col weights):", paste(c(w.fitted.loss, sum(w.fitted.loss)),
    collapse = " | "))
```

```
## loss weighted rf (no col weights):388.758212952329 | 325.727555326448 | 291.297057392399 | 1005.7828
```

```
message("loss weighted rf (col weights):", paste(c(w.scaled.fitted.loss, sum(w.scaled.fitted.loss)),
    collapse = " | "))
```

```
## loss weighted rf (col weights):6497.29147167974 | 8683.276156136 | 10455.5762967304 | 25636.1439245
```

Validation Set Results

```
## validation sets
yv = yv$couponUsed
Xv = Xv[, which(names(Xv) %in% names(Xrf))]

message("VALIDATION SET")
```

```
## VALIDATION SET
```

```
# Unweighted Loss Aspects
uw.predicted = predict(unweighted.rf, newdata = Xv, type = "response")
uw.predicted.loss = colSums(matrix((yv - uw.predicted)^2, ncol = 3, byrow = TRUE))
uw.scaled.predicted.loss = uw.predicted.loss/(colMeans(matrix(yv, ncol = 3,
    byrow = TRUE))^2)

# Weighted Loss Aspects
w.predicted = predict(weighted.rf, newdata = Xv, type = "response") * rep(c(p1,
    p2, p3), times = nrow(Xv)/3)
w.predicted.loss = colSums(matrix((yv - w.predicted)^2, ncol = 3, byrow = TRUE))
w.scaled.predicted.loss = w.predicted.loss/(colMeans(matrix(yv, ncol = 3, byrow = TRUE))^2)

message("loss unweighted rf (no col weights):", paste(c(uw.predicted.loss, sum(uw.predicted.loss)),
    collapse = " | "))
```

```
## loss unweighted rf (no col weights):220.949118250527 | 174.825179372017 | 164.790932402203 | 560.565

message("loss unweighted rf (col weights):", paste(c(uw.scaled.predicted.loss,
    sum(uw.scaled.predicted.loss)), collapse = " | "))

## loss unweighted rf (col weights):3903.3445179996 | 6030.83694272542 | 6570.80642043898 | 16504.9878

message("loss weighted rf (no col weights):", paste(c(w.predicted.loss, sum(w.predicted.loss)),
    collapse = " | "))

## loss weighted rf (no col weights):220.13419662128 | 174.707346812765 | 164.114698235298 | 558.956241

message("loss weighted rf (col weights):", paste(c(w.scaled.predicted.loss,
    sum(w.scaled.predicted.loss)), collapse = " | "))

## loss weighted rf (col weights):3888.9479008087 | 6026.77214522917 | 6543.84253510349 | 16459.562581
```