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

Curation and Cross Validation to Reduce Features

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")
```

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
```

And this is our loss function:

```
lossFunMethod = function(yval, Xval, method) {
    hatmat = as.matrix(matrix(predict(method, newdata = Xval, type = "response"),
        ncol = 3, byrow = TRUE))
    ymat = matrix(yval, ncol = 3, byrow = TRUE)

    error = colSums((ymat - hatmat)^2)
    wt = colMeans(ymat)^2

    cat("The coupon error is:  ", sum(error/wt), "\n")
    cat("By column error:      ", error, "\n")
    cat("By column weight:     ", wt, "\n\n")
    return(sum(error/wt))
}
```

The following functions will help me in this process:

```
## I will use cross validation to estimate an error instead of checking the
## validation set
RFxVal = function(CVk, Xrf, yrf, mtry = 8, ntree = 1000, maxnodes = 50) {
    row.order = sample(1:nrow(Xrf))
    CV.bounds = round(seq.int(1, nrow(Xrf), length = (CVk + 1)))
    OOBset = lapply(1:CVk, function(i) row.order[CV.bounds[1]:(CV.bounds[2] -
        1)])

    # Fit a random forest for couponUsed
    cvRF = function(rows) randomForest(Xrf[-rows, ], y = yrf[-rows], ntree = ntree,
        mtry = mtry, replace = TRUE, maxnodes = maxnodes)

    message("Fitting randomForest")
    rf_cvs = lapply(1:CVk, function(i) cvRF(OOBset[[i]]))

    message("Calculating Loss")
    lossFunction = lapply(1:CVk, function(i) lossFunMethod(yrf[OOBset[[i]]],
        Xrf[OOBset[[i]], ], rf_cvs[[i]]))

    print(lossFunction)
    return(rf_cvs)
}

## CV to get the mean importance
RFxVal_imp = function(RFs) {
    RFimp = data.frame(feature = rownames(RFs[[1]]$importance), IncNodePurity1 = RFs[[1]]$importance)
    rownames(RFimp) = NULL
    names(RFimp) = c("feature", "IncNodePurity1")
    for (i in 2:length(RFs)) {
        RFimp.i = data.frame(feature = rownames(RFs[[i]]$importance), IncNodePurity1 = RFs[[i]]$importan
        names(RFimp.i) = c("feature", paste0("IncNodePurity", i))
        rownames(RFimp.i) = NULL
        RFimp = RFimp %>% left_join(RFimp.i, by = "feature")
    }
    RFimplong = RFimp %>% gather(colname, IncNodePurity, -feature) %>% mutate(CViteration = as.numeric(g
        "", colname))) %>% select(feature, CViteration, IncNodePurity) %>% arrange(feature,
        CViteration)
    RFimplongMeans = RFimplong %>% group_by(feature) %>% summarize(meanIncNodePurity = mean(IncNodePuri

    CVimpplotfunc = function(dsn) {
        CVplot = ggplot(data = dsn, aes(x = CViteration, y = IncNodePurity,
            group = feature, color = feature)) + geom_line() + theme(legend.position = "none") +
            geom_text(data = RFimplong[RFimplong$CViteration == 1, ], aes(label = feature),
                hjust = 1, size = 0.1)
        CVplot = direct.label(CVplot + xlim(0, length(RFs) + 10), "last.qp")
        return(CVplot)
    }

    RFimp = list(importance = RFimp, iterations = RFimplong, mean = RFimplongMeans,
        plot = CVimpplotfunc(RFimplong), makeplot = CVimpplotfunc)

    return(RFimp)
}
```

0.4 I am going to start with a simple random forest

```
set.seed = 1999
## 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)]

## I will use cross validation to estimate an error instead of checking the
## validation set
set.seed = 1999
CVrf = RFxVal(5, Xrf, yn[, "couponUsed"], mtry = 8, ntree = 1000, maxnodes = 50)
```

```
## Fitting randomForest
```

```
## Warning in randomForest.default(Xrf[-rows, ], y = yrf[-rows], ntree =
## ntree, : The response has five or fewer unique values.  Are you sure you
## want to do regression?
```

```
## Warning in randomForest.default(Xrf[-rows, ], y = yrf[-rows], ntree =
## ntree, : The response has five or fewer unique values.  Are you sure you
## want to do regression?
## Warning in randomForest.default(Xrf[-rows, ], y = yrf[-rows], ntree =
## ntree, : The response has five or fewer unique values.  Are you sure you
## want to do regression?
```

```
## Warning in randomForest.default(Xrf[-rows, ], y = yrf[-rows], ntree =
## ntree, : The response has five or fewer unique values.  Are you sure you
## want to do regression?
```

```
## Warning in randomForest.default(Xrf[-rows, ], y = yrf[-rows], ntree =
## ntree, : The response has five or fewer unique values.  Are you sure you
## want to do regression?
```

```
## Calculating Loss
```

```
## The coupon error is:   5471.761
## By column error:       78.29988 81.46362 86.24724
## By column weight:      0.04180429 0.04180429 0.05226918
##
## The coupon error is:   5475.783
## By column error:       78.43724 81.5225 86.21214
## By column weight:      0.04180429 0.04180429 0.05226918
##
## The coupon error is:   5472.216
## By column error:       78.36817 81.3622 86.31244
## By column weight:      0.04180429 0.04180429 0.05226918
##
## The coupon error is:   5477.79
## By column error:       78.34235 81.58994 86.35134
## By column weight:      0.04180429 0.04180429 0.05226918
##
## The coupon error is:   5474.873
## By column error:       78.21605 81.57335 86.37757
## By column weight:      0.04180429 0.04180429 0.05226918
##
## [[1]]
## [1] 5471.761
##
## [[2]]
## [1] 5475.783
##
## [[3]]
## [1] 5472.216
##
## [[4]]
## [1] 5477.79
##
## [[5]]
## [1] 5474.873
```

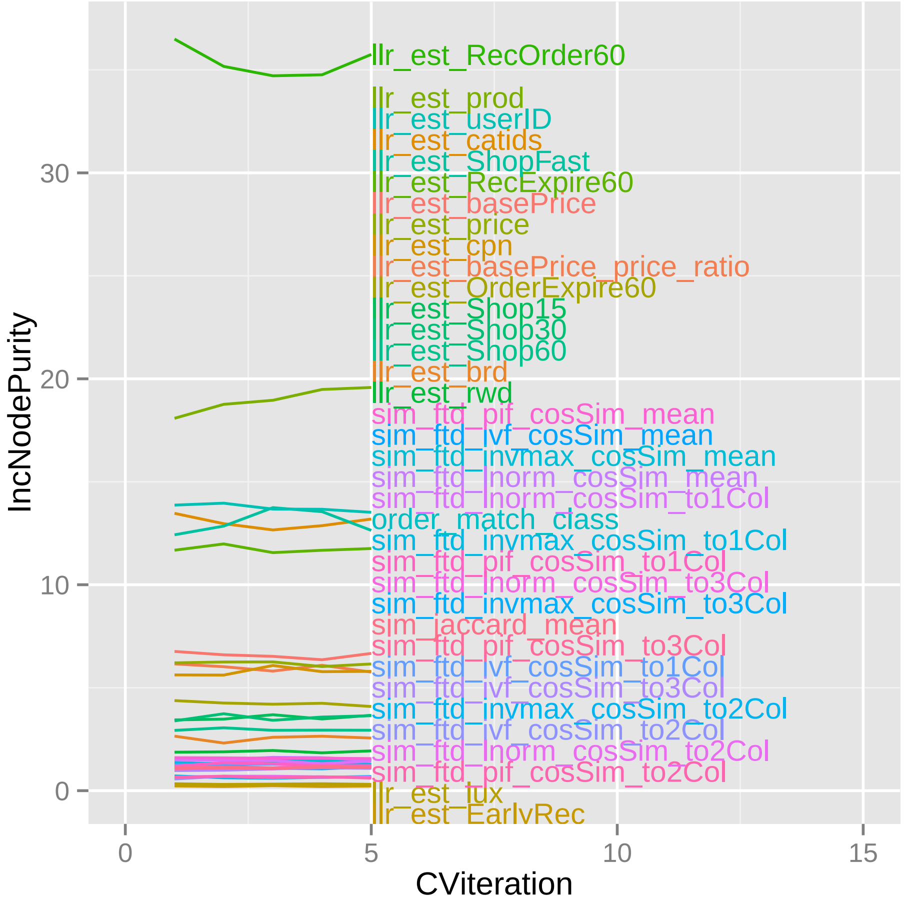
```
CVimp = RFxVal_imp(CVrf)
```

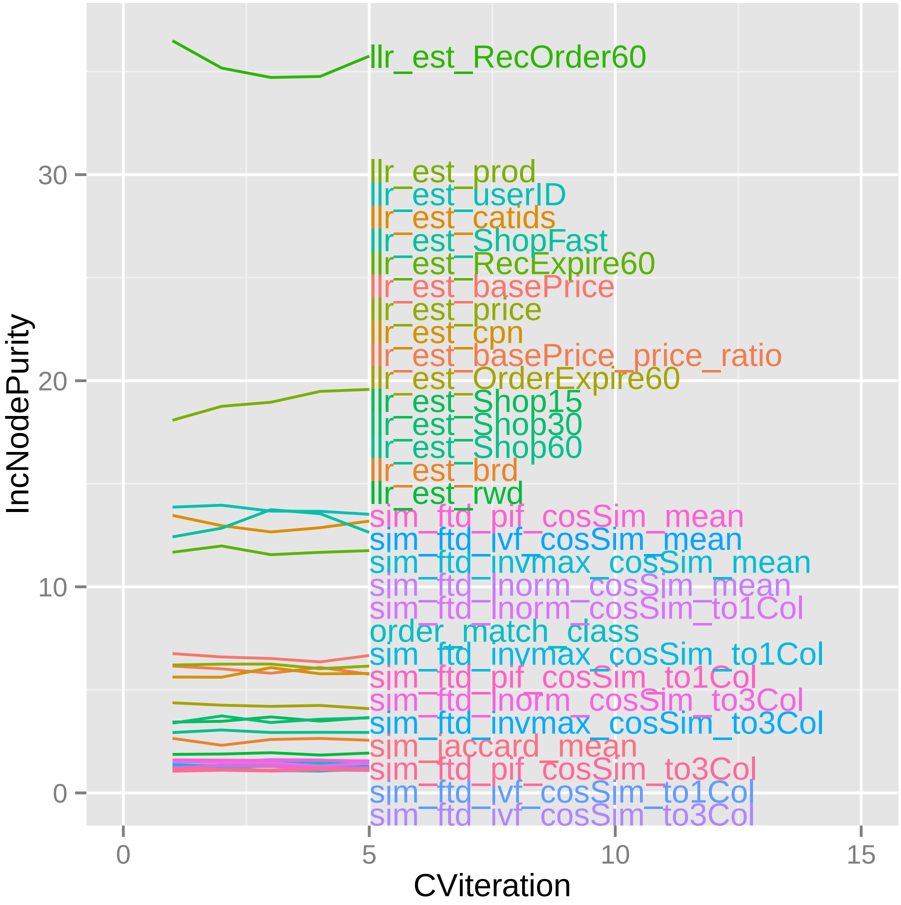
```
## Loading required package: proto
```

```
CVimp$plot
```

```
CVimp$makeplot(CVimp$iteration[which(CVimp$iteration$IncNodePurity > 1), ])
```

```
# fit full data
rf1 = randomForest(Xrf, y = yn$couponUsed, ntree = 1000, mtry = 8, replace = TRUE,
    maxnodes = 50)

## Warning in randomForest.default(Xrf, y = yn$couponUsed, ntree = 1000, mtry
## = 8, : The response has five or fewer unique values.  Are you sure you
## want to do regression?

# get the output
lossFunMethod(yv$couponUsed, Xv[, which(names(Xv) %in% names(Xrf))], rf1)

## The coupon error is:    16557.93
## By column error:        221.7057 174.9989 165.6331
## By column weight:       0.05660508 0.02898854 0.02507926

## [1] 16557.93
```




llr_est_RecOrder60
llr_est_prod
llr_est_userID
llr_est_catids
llr_est_ShopFast
llr_est_RecExpire60
llr_est_basePrice
llr_est_price
llr_est_cpn
llr_est_basePrice_price_ratio
llr_est_OrderExpire60
llr_est_Shop15
llr_est_Shop30
llr_est_Shop60
llr_est_brd
llr_est_rwd
sim_ftd_pif_cosSim_mean
sim_ftd_lvf_cosSim_mean
sim_ftd_invmax_cosSim_mean
sim_ftd_lnorm_cosSim_mean
sim_ftd_lnorm_cosSim_to1Col
order_match_class
sim_ftd_invmax_cosSim_to1Col
sim_ftd_pif_cosSim_to1Col
sim_ftd_lnorm_cosSim_to3Col
sim_ftd_invmax_cosSim_to3Col
sim_jaccard_mean
sim_ftd_pif_cosSim_to3Col
sim_ftd_lvf_cosSim_to1Col
sim_ftd_lvf_cosSim_to3Col
IncNodePurity
30
20
10
0
0
5
10
15
CViteration