Demonstration of the use of R Package ncpen

This example demonstrates the use of **R** library **ncpen** which estimates generalized linear models (GLM) with various nonconvex penalties. We use mortgage loan performance (prepayment or performing) example. Mortgage loan performance data is from Fannie Mae (`www.fanniemae.com`) combined with average mortgage rate from Freddie Mac (`www.freddiemac.com`). The response variable is prepaid which is binomial: prepaid (1) or performing (0). Active loans as of beginning of 2011 and 2012 are collected. Data includes original 16 variables with over 142,000 rows.

First, load **ncpen** library.

```
library("ncpen")
```

The data (csv) can be downloaded from the following address. File size is over 20MB and it may take some time to complete the download.

```
# ~20MB file. This may take a couple of minutes depending on network speed.
prepay.data = read.csv(file =
    "https://raw.githubusercontent.com/zeemkr/data/master/mtg_term_2011_2012.csv")
head(prepay.data)
dim(prepay.data)
```

```
[1] 142343     18
```

The data has 18 columns and 142343 observations. The first two columns of `ncpen.data` are `BOY` (beginning of the year), `prepaid` (y variable), and the rest of the columns are `X` variables. Assign the data to another variable for the manipulation of data to be used in **ncpen** package.

```
# Data manipulation
ncpen.data = prepay.data
```

The data set includes one categorical (non-numeric) variable `CHANNEL`. `CHANNEL` has the values of "B", "C" and "R". This variable needs to be converted to a series of indicator variables. Value "B" will be excluded as a base case and other values will be turned in to indicator variables (`CHANNEL_C` and `CHANNEL_R`). Use `to.indicators` function to convert `CHANNEL` to indicator variables then bind them to the original data.

```
# Convert a categorical variables to multiple indicator variables.
ncpen.data = cbind(ncpen.data, to.indicators(subset(ncpen.data, select = "CHANNEL"),
                    type = "exclude.base", base = "B"))

# Now remove the categorical variable.
ncpen.data$CHANNEL = NULL
```

Then, we include all possible interaction combinations in the explanatory variables set using `interact.data` function. However, the columns `CHANNEL_C` and `CHANNEL_R`, that we just have created, are not supposed to be interacted. Also, the data includes `loan_age_sq` which is the square of `loan_age` variable. We also do not want those variables to be interacted. By passing `base.cols = c("CHANNEL", "loan_age")` to `interact.data` function, those interactions can be avoided.

In addition, there are three binary variables: `FTHB`, `Purchase` and `Primary`. When `FTHB` is 1, `Purchase` is always 1 and `Primary` is 1 as well. So, we want exclude interaction pairs of (`FTHB`, `Purchase`) and (`FTHB`, `Primary`). It can be done by including `exclude.pair = list(c("FTHB", "Purchase"), c("FTHB", "Primary"))`.

```
# Include all possible interactions in X.
# The first column is the beginning of year (BOY) and
# the second column is prepaid indicator (y variable).
# So, only interact X, ncpen.data[, -c(1,2)].
# Then, bind to the original data set.
ncpen.data = cbind(ncpen.data, interact.data(ncpen.data[, -c(1,2)],
                    base.cols = c("CHANNEL", "loan_age"),
                    exclude.pair = list(c("FTHB", "Purchase"), c("FTHB", "Primary"))))
head(ncpen.data)
```

We use BOY 2011 data for training (model estimation) and predict the probability of prepayment of BOY 2012 data. The second column is prepaid indicator (0 or 1). All `X` variables follow after `prepaid`.

```
# Train data set, BOY == 2011 and test data set, BOY == 2012
# The second column is y (prepaid = 0 or 1)
# X starts from the third column
y.vec.train = ncpen.data[ncpen.data$BOY == 2011, 2]
x.mat.train = ncpen.data[ncpen.data$BOY == 2011, -c(1,2)]
```

From the train set, random sample 5000 rows to estimate the model.

```
set.seed(123)
sample.idx = sample(1:length(y.vec.train), 5000)
y.vec.train = y.vec.train[sample.idx]
x.mat.train = x.mat.train[sample.idx,]
```

Construct test data set.

```
y.vec.test = ncpen.data[ncpen.data$BOY == 2012, 2]
x.mat.test = ncpen.data[ncpen.data$BOY == 2012, -c(1,2)]
```

First, test the data with glm binomial regression.

```
# 1. GLM test
train.df = as.data.frame(cbind(y.vec.train, x.mat.train))
glm.fit = glm(y.vec.train~., data=train.df,family="binomial")
summary(glm.fit)

Call:
glm(formula = y.vec.train ~ ., family = "binomial", data = train.df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5792  -0.5768  -0.4391  -0.2402   3.1311

Coefficients: (1 not defined because of singularities)
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.412e+00  7.032e+00  -1.196 0.231623
int_spread               -5.048e+00  2.509e+00  -2.012 0.044195 *
......
[omitted to save space]
......
'Big_Seller:CHANNEL_R'    5.561e-01  2.679e-01   2.076 0.037933 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4475.4  on 4999  degrees of freedom
Residual deviance: 3761.4  on 4851  degrees of freedom
AIC: 4059.4

Number of Fisher Scoring iterations: 15

# number of coefficients
sum(!is.na(coef(glm.fit))

[1] 149
```

Calculate, mean absolute error (MAE).

```
# MAE
glm.fit.coef = coef(glm.fit)
glm.fit.coef[is.na(glm.fit.coef)] = 0
exb.vec = exp(drop(as.matrix(cbind(1, x.mat.test))%*%glm.fit.coef))
ph.vec = exb.vec/(1+exb.vec)
nyh.vec = ph.vec > 0.5
mean(abs(y.vec.test - nyh.vec))

[1] 0.396893
```

Now test with **ncpen** packages cross validation estimation `cv.ncpen` (ignore warnings).

```
# 2. ncpen test
cv.ncpen.fit = cv.ncpen(y.vec.train, as.matrix(x.mat.train),
                        family = "binomial", penalty = "scad")
# This may take a couple of minutes...
cv.ncpen.coef = as.matrix(cv.ncpen.fit$opt.ebeta)
rownames(cv.ncpen.coef) = c("Intercept", colnames(x.mat.train))
cv.ncpen.coef

                                [,1]
Intercept               -2.280013e+00
```

```
int_spread              0.000000e+00
......
[omitted to save space]
......
FTHB:CHANNEL_C          4.694868e-01
FTHB:CHANNEL_R          0.000000e+00
Big_Seller:CHANNEL_C    0.000000e+00
Big_Seller:CHANNEL_R    0.000000e+00
```

This is the number of coefficients selected (non-zero).

```
# number of coefficients selected
sum(cv.ncpen.coef!=0)
```

```
[1] 32
```

Finally, MAE for `cv.ncpen`.

```
# MAE
exb.vec = exp(drop(as.matrix(cbind(1, x.mat.test))%*%cv.ncpen.fit$opt.ebeta))
ph.vec = exb.vec/(1+exb.vec)
nyh.vec = ph.vec > 0.5
mean(abs(y.vec.test - nyh.vec))
```

```
[1] 0.3856678
```