An Introduction to SMLE

Qianxiang Add last name

9/25/2021 Update year

Introduction

This vignette describes how one can use the **SMLE** package to perform Ultra-high dimensional feature screening. Suppose the data $\{(y_i, \boldsymbol{x}_i), i = 1, \dots, n\}$ are collected independently from $(Y, \boldsymbol{x})$, where Y is a response variable and $\boldsymbol{x} = (x_1, \dots, x_p)$ is a p-dimensional covariate (feature) vector.

Under GLM setting:

$$f(y;\theta) = \exp(\theta y - b(\theta) + c(y)), \text{ and } \theta = \boldsymbol{x\beta},$$

where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)^T$ is a p-dimensional regression coefficient.

SMLE iteratively estimate the problem:

$$\hat{\boldsymbol{\beta}_k} = \max_{\beta} \sum_{i=1}^{n} [y_i \cdot \boldsymbol{x}_i\boldsymbol{\beta} - b(\boldsymbol{x}_i\boldsymbol{\beta})] \quad \text{subject to} \quad ||\beta||_0 \le k,$$

The theory and algorithms in this implementation are described in Xu and Chen (2014).

Usage

A demo code for SMLE-screening

First we show how to use `SMLE` to conduct feature screening and post-screening selection via a simulated example. We generate a dataset with $n = 400$ observations and $p = 1000$ features. We generate the feature matrix X from a multivariate normal distribution with an auto-regressive structure, where the adjacent features have a high correlation of $\rho = 0.9$. The response variable Y is generated based on the following logistic model with success rate π and linear predictor:

$$\text{logit}(\pi) = 2x_1 + 3x_3 - 3x_5 + 3x_7 - 4x_9.$$

```
library(SMLE)

set.seed(1)

Data_eg <- Gen_Data(n = 400, p = 1000, family = "binomial",
correlation = "AR", rho = 0.9, pos_truecoef = c(1,3,5,7,9),
effect_truecoef = c(2,3,-3,3,-4))

Data_eg
```

```
## Call:
##  Gen_Data(n = 400, p = 1000, pos_truecoef = c(1, 3, 5, 7, 9),
##     effect_truecoef = c(2, 3, -3, 3, -4), correlation = "AR",
##     rho = 0.9, family = "binomial")
##
## An object of class sdata
##
## Simulated Dataset Properties:
##  Length of response: 400
##  Dim of features: 400 x 1000
##  Correlation: auto regressive
##  Rho: 0.9
##  Index of causal features: 1, 3, 5, 7, 9
##  Model type: binomial
```

In this setup, the feature matrix contains only five features that are causally-related to the response, as indicated in the model. Some features have marginal effects to response due to the correlation structure. From the true model, we know that x_2 is not causally-related to the response. Yet, we can see that the marginal effect of x_2 appears to be pretty high; thus, this irrelevant feature is likely to be retained in the model if the screening is done based on marginal effects only.

```
coef(summary(glm(Data_eg$Y ~ Data_eg$X[,2], family = "binomial")))
```

```
##                  Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)    0.02440072  0.1125979 0.2167067 8.284369e-01
## Data_eg$X[, 2] 1.10465766  0.1337063 8.2618250 1.434619e-16
```

The following code shows the simplest function call to `SMLE()`, where we aim to retain only $k = 10$ important features out of $p = 1000$.

```
fit1 <- SMLE(Y = Data_eg$Y,X = Data_eg$X, k = 10, family = "binomial")

summary(fit1)
```

```
## Call:
##   SMLE(X = Data_eg$X, Y = Data_eg$Y, k = 10, family = "binomial")
##
## An object of class summary.smle
##
## Summary:
##
##   Length of response: 400
##   Dim of features: 400 x 1000
##   Model type: binomial
##   Model size: 10
##   Feature name: 1, 3, 5, 7, 9, 68, 430, 536, 661, 709
##   Feature index: 1, 3, 5, 7, 9, 68, 430, 536, 661, 709
##   Intercept: 0.5075
##   Coefficients estimated by IHT: 1.769, 3.845, -1.355, 1.853, -4.581, -0.693, -0.686,
0.631, 0.589, 0.633
##   Number of IHT iteration steps: 2
```

The function returns a `'smle'` object and `summary()` function confirms that a refined set of 10 features is selected after 59 IHT iterations. We can see that all 5 causal features used to generate the response are retained in the refined set. This indicates that screening is successful; the dimensionality of the feature space is reduced from $p = 1000$ down to $k = 10$ without losing any important information. In this example, `SMLE()` accurately removes x_2 , x_4 , x_6 , x_8 , as its screening naturally incorporates the joint effects among features.

Further selection after screeening

Note that the refined set returned in the model still contains some irrelevant features; this is to be expected (the `k` always chosen to be larger than the actual number of casual features), as the goal of feature screening is merely to remove most irrelevant features before conducting an in-depth analysis. One may conduct an elaborate selection on the refined set to further identify the causal features.

As can be seen below, `smle_select()` returns a `'selection'` object `"fit1_s"`, which exactly identifies the five features in the true data generating model.

```
fit1_s <- smle_select(fit1, criterion = "ebic")

summary(fit1_s)
```

```
## Call:
##   smle_select(object = fit1, criterion = "ebic")
##
## An object of class summary.selection
##
## Summary:
##
##   Length of response: 400
##   Dim of features: 400 x 1000
##   Model type: binomial
##   Selected model size: 5
##   Selected feature index: 1, 3, 5, 7, 9
##   Selection criterion: ebic
##   Gamma for ebic: 0.5
```

An example for categorical features.

Categorical features fed in the package will be convert to `'factor'` and dummy coded during the iterations. In this example, we generate a dataset with causal categorical features and separate it into training and testing groups in order to perform a prediction task.

```
set.seed(1)
Data_sim2 <- Gen_Data(n = 420, p = 1000, family = "gaussian", num_ctgidx = 5,
                      pos_ctgidx = c(1,3,5,7,9), effect_truecoef= c(1,2,3,-4,-5),
                      pos_truecoef = c(1,3,5,7,8), level_ctgidx = c(3,3,3,4,5))
```

```
train_X <- Data_sim2$X[1:400,]; test_X <- Data_sim2$X[401:420,]
train_Y <- Data_sim2$Y[1:400]; test_Y <- Data_sim2$Y[401:420]

test_X[1:5,1:10]
```

```
##     C1          X2 C3         X4 C5         X6 C7         X8 C9         X10
## 401  C -0.17405549  B  0.3337833  B -1.8054836  B  0.9696772  C -0.88066391
## 402  C  0.96129056  C -0.2113226  A -0.6780407  B -2.1994065  C -0.48558301
## 403  B  0.29382666  A -0.5510979  B -0.4733581  C  1.9480938  B  0.22743281
## 404  B  0.08099936  B  0.2583611  A  1.0274171  B  0.1798532  B -0.06646135
## 405  B  0.18366184  A -1.3752104  B -0.5973876  C  0.4150568  B  0.35161359
```

Users may specify whether to treat those dummy covariates as a single group feature or as individual features, and which type of dummy coding is used by arguments: `gourp` and `codyingtype`. Note that the number of features retained in the model may less than the `k` specified when `group` is `FALSE` since one categorical feature may be chose several times by its covariates. More details see the package manual.

```
fit_1 <- SMLE(Y = train_Y, X = train_X, family = "gaussian", group = TRUE, codingtype =
        "standard", k = 10)
fit_1
```

```
## Call:
##   SMLE(X = train_X, Y = train_Y, k = 10, family = "gaussian", group = TRUE,
##     codingtype = "standard")
##
## An object of class smle
##
## Subset:
##   Model size: 10
##   Feature name: C1, C3, C5, C7, X8, X28, X297, X327, X671, X727
##   Feature index: 1, 3, 5, 7, 8, 28, 297, 327, 671, 727
```

```
predict(fit_1, newdata = test_X)
```

```
##         401         402         403         404         405         406
##  -1.4095163  12.8553088 -15.4013607  -3.4599702  -7.0396945  -9.3402396
##         407         408         409         410         411         412
##   9.1283576   3.9155024   4.5107214  -2.9755387  -5.5625988   4.9382166
##         413         414         415         416         417         418
##  -6.2496557   8.4245219  -4.8765143   0.6937084   7.5703692  -3.3885818
##         419         420
##  -9.3472568   6.7578661
```

```
fit_2 <- SMLE( Y = train_Y, X = train_X, family = "gaussian", group = FALSE, codingtype =
        "all", k = 10)
fit_2
```

```
## Call:
##   SMLE(X = train_X, Y = train_Y, k = 10, family = "gaussian", group = FALSE,
##     codingtype = "all")
##
## An object of class smle
##
## Subset:
##   Model size: 6
##   Feature name: C1, C3, C5, C7, X8, X671
##   Feature index: 1, 3, 5, 7, 8, 671
```

```
predict(fit_2, newdata = test_X)
```

```
##       401       402       403       404       405       406       407
## -1.474629 12.869259 -14.487680 -3.056318 -7.737236 -9.980981  8.660751
##       408       409       410       411       412       413       414
##  3.615428  4.202304 -2.721383 -6.143631  6.157237 -6.907233  8.092940
##       415       416       417       418       419       420
## -4.850294  1.206388  7.685806 -3.486999 -9.214226  5.059840
```

Formula interface

SMLE always works in low dimensional as a selection method. Although it is not recommend, interface to 'formula' object provides user a better understanding to the package in high dimension.

```
library(datasets)
data("attitude")
SMLE(rating ~ complaints + complaints:privileges + learning + raises*advance, data =
        attitude)
```

```
## Call:
##   SMLE(formula = rating ~ complaints + complaints:privileges +
##     learning + raises * advance, data = attitude)
##
## An object of class smle
##
## Subset:
##   Model size: 5
##   Feature name: complaints, privileges, learning, raises, advance
##   Feature index: 2, 3, 4, 5, 7
```

Xu, Chen, and Jiahua Chen. 2014. "The Sparse MLE for Ultrahigh-Dimensional Feature Screening." *Journal of the American Statistical Association* 109 (507): 1257–69.