Classification using Generalized Partial Least Squares

Beijing Ding
Robert Gentleman
October 14, 2013

Introduction

The gpls package includes functions for classification using generalized partial least squares approaches. Both two-group and multi-group (more than 2 groups) classifications can be done. The basic functionalities are based on and extended from the Iteratively ReWeighted Least Squares (IRWPLS) by Marx (1996). Additionally, Firth’s bias reduction procedure (Firth [1992a,b,1993]) is incorporated to remedy the nonconvergence problem frequently encountered in logistic regression. For more detailed description of classification using generalized partial least squares, refer to [Ding and Gentleman (2005)].

The glpls1a function

The glpls1a function carries out two-group classification via IRWPLS(F). Whether or not to use Firth’s bias reduction is an option (br=T). The X matrix shouldn’t include an intercept term.

```R
> library(gpls)
> set.seed(123)
> x <- matrix(rnorm(20),ncol=2)
> y <- sample(0:1,10,TRUE)
> ## no bias reduction
> glpls1a(x,y,br=FALSE)

Call:
NULL

Coefficients:
Intercept      X:1      X:2
  -2.3122   -0.5069   -0.4529

> ## no bias reduction and 1 PLS component
> glpls1a(x,y,K.prov=1,br=FALSE)
```

1
The `glpls1a.cv.error` and `glpls1a.train.test.error` functions

The `glpls1a.cv.error` calculates leave-one-out classification error rate for two-group classification and `glpls1a.train.test.error` calculates test set error where the model is fit using the training set.

```r
> ## training set
> x <- matrix(rnorm(20), ncol=2)
> y <- sample(0:1, 10, TRUE)
> ## test set
> x1 <- matrix(rnorm(10), ncol=2)
> y1 <- sample(0:1, 5, TRUE)
> ## no bias reduction
> glpls1a.cv.error(x, y, br=FALSE)

$error
[1] 0.6

$error.obs
[1] 1 2 4 7 9 10

> glpls1a.train.test.error(x, y, x1, y1, br=FALSE)
```
The `glpls1a.mlogit` and `glpls1a.logit.all` functions

The `glpls1a.mlogit` carries out multi-group classification using MIRWPLS(F) where the baseline logit model is used as counterpart to `glpls1a` for two group case. `glpls1a.logit.all` carries out multi-group classification by separately fitting $C$ two-group classification using `glpls1a` separately.
for $C$ group vs the same baseline class (i.e. altogether $C + 1$ classes). This separate fitting of logit is known to be less efficient but has been used in practice due to its more straightforward implementation.

Note that when using `glpls1a.mlogit`, the X matrix needs to have a column of one, i.e. intercept term.

```r
> x <- matrix(rnorm(20),ncol=2)
> y <- sample(1:3,10,TRUE)
> ## no bias reduction and 1 PLS component
> glpls1a.mlogit(cbind(rep(1,10),x),y,K.prov=1,br=FALSE)

$coefficients
 [,1]      [,2]
[1,] -0.7689747 -0.0370627
[2,]  2.1926005 -4.2015007
[3,]  0.5442713 -0.9331431

$convergence
[1] FALSE

$niter
[1] 100

$bias.reduction
[1] FALSE

> glpls1a.logit.all(x,y,K.prov=1,br=FALSE)

$coefficients
 [,1]      [,2]
[1,] -2.339152  1.029019
[2,]  2.906929 -9.000959
[3,]  0.579930 -1.387408

>$bias.reduction
[1] FALSE

> glpls1a.mlogit(cbind(rep(1,10),x),y,br=TRUE)

$coefficients
 [,1]      [,2]
[1,] -1.0327659  0.41635681
[2,]  1.2298647 -2.58869374
[3,]  0.4357512 -0.08656436

$convergence
[1] TRUE
```
The glpls1a.mlogit.cv.error function

The glpls1a.mlogit.cv.error calculates leave-one-out error for multi-group classification using (M)IRWPLS(F). When the mlogit option is set to be true, then glpls1a.mlogit is used, else glpls1a.logit.all is used for fitting.
$error
[1] 0.6

$error.obs
[1] 3 4 5 7 9 10

> glpls1a.mlogit.cv.error(x,y,mlogit=FALSE,br=TRUE)

$error
[1] 0.5

$error.obs
[1] 3 4 5 7 10

> 0.1 Fitting Models to data
Here we demonstrate the use of gpls on some standard machine learning examples. We first make
use of the Pima Indian data from the MASS package.

> library(MASS)
> m1 = gpls(type~., Pima.tr)
> p1 = predict(m1, Pima.te[,,-8])
> ##when we get to the multi-response problems
> data(iris3)
> Iris <- data.frame(rbind(iris3[,1], iris3[,2], iris3[,3]),
+ Sp = rep(c("s","c","v"), rep(50,3)))
> train <- sample(1:150, 75)
> table(Iris$Sp[train])

      c   s   v
23 27 25

> ## your answer may differ
> ## c s v
> ## 22 23 30
> z <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3, subset = train)
> predict(z, Iris[-train, ])$class

[1] s s s s s s s s s s s s s s c c c c c c c c c c c c c c c c c c
[39] c c c c c c c c c c c c v v v v v v v v v v v v v v v v c v v v v v v v v v v
Levels: c s v

>
References


