LASSO Adaptativo e Critérios de Informação para LASSO

Daniel Coutinho

2019-04-11

Em um post anterior, eu falei do LASSO (Least Absolute Shrinkage and Select Operator). Como vamos explorar uma variação do LASSO hoje, eu vou repetir o problema que o LASSO resolvia:

$$\hat{\beta}_{LASSO} \in \arg\min_{\beta} \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda \sum_{k=0}^{p} |\beta_k|$$

(Onde $|.|$ é o valor absoluto do termo). E como eu já disse, o LASSO nos fornece uma maneira de selecionar quais variáveis entram no modelo ou não. Vamos fazer um pequeno teste com o LASSO: eu vou gerar 50 variáveis normais, independentes, e dessas dez - as dez primeiras - eu colocarei $\beta = 1$. As outra vão ser irrelevantes para o problema e vão ter $\beta = 0$. O tamanho da amostra vai ser igual a 100.

Como de praxe, nós podemos ter diversos objetivos ao estimar um modelo. Eu vou comparar 3 coisas: a quantidade de vezes que o LASSO coloca as variáveis relevantes, a quantidade de vezes que ele exclui as variáveis irrelevantes e quando ele obtém o modelo certo - o que exige colocar todo mundo que é relevante **e** excluir todos os irrelevantes. Vou fazer só 500 replicações e usar o Cross Validation para escolher o λ:

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16
```

```
coeficientes <- Matrix(0,ncol=500,nrow=51)

for(i in 1:500){
  x <- matrix(rnorm(50*100),ncol = 50) #gerando x
  betas <- c(rep(1,10),rep(0,40)) #
  y <- x%*%betas + rnorm(100)
  modelo <- cv.glmnet(x,y) #estimando usando LASSO e Cross Validation
  coeficientes[,i] <- coef(modelo)
}

#Fim da simulação

analise <- matrix(0,ncol=3,nrow=500)
colnames(analise)<- c("Não zeros certos","Zeros certos", "Modelo certo?")

for(i in 1:500){
  analise[i,1]<- mean(coeficientes[2:11,i] != 0)
  analise[i,2]<- mean(coeficientes[12:51,i] == 0)
  analise[i,3]<- ifelse(analise[i,1]+analise[i,2] == 2,1,0)
}

tabela_final <- colMeans(analise)*100
knitr::kable(tabela_final,caption = "Os valores estão em porcentagem")
```

Table 1: Os valores estão em porcentagem

	x
Não zeros certos	100.000
Zeros certos	83.825
Modelo certo?	1.000

O LASSO sempre inclui as variáveis relevantes, e exclui as variáveis irrelevantes em 84% das vezes. Mesmo assim, a proporção de vezes que o LASSO consegue recuperar o modelo correto é um por volta de 1%. Isso parece esquisito a primeira vez, mas lembre que recuperar o modelo certo envolve *acertar todas as relevantes e todas as irrelevantes*. Se tivermos 50 variáveis e 500 replicações e em toda replicação o LASSO colocar apenas umas variável irrelevante no modelo, nós teríamos 98% de zeros certos (49/50) e exatamente 0 modelos certos.

Parte do problema é que o LASSO penaliza todos os coeficientes igualmente, usando o λ. Nós esperaríamos que algumas variáveis sejam mais importantes que outras - e isso pode vir a priori ou ser dito pelos dados. O LASSO adaptativo (adaLASSO) adiciona pesos (ω) para cada uma das variáveis na penalidade. Logo o novo problema a ser resolvido é:

$$\hat{\beta}_{adaLASSO} \in \arg\min_{\beta} \sum_{i=1}^{n}(y_i - x_i\beta)^2 + \lambda \sum_{k=0}^{p} \frac{|\beta_k|}{\omega_k}$$

A única exigência desses pesos é que eles sejam positivos. Veja que nós temos um novo parâmetro a escolher, os pesos. Eis um algoritmo muito simples que gera os pesos baseado nos dados e usa o LASSO:

1. Estime o modelo usando LASSO. Guarde os coeficientes, que eu chamarei de β_{LASSO}
2. Defina $\omega_k = \frac{1}{|\beta_{LASSO}|}$
3. Estime o adaLASSO usando os pesos definidos em 2.

A boa notícia é que o *glmnet* já nos oferece uma opção para colocar o peso, via o argumento *penalty.factor*. Nosso trabalho é basicamente reduzido a escrever umas duas linhas de código a mais: uma que faz o LASSO de "primeiro estágio" e outra que define os pesos.

Uma coisa deve ficar evidente: da maneira que os pesos foram estabelecidos no meu algoritmo, coeficientes que são zerados pelo LASSO serão automaticamente excluídos pelo LASSO adaptativo: teremos como peso 1/0 (com perdão aos matemáticos), que no limite é infinito. Como o LASSO não excluiu as variáveis relevantes em nenhum caso, não vamos nos preocupar. Mas é possível imaginar situações em que o LASSO poderia ter problemas (pense em coeficientes altissimamente correlacionados em uma amostra relativamente pequena). Outra coisa que deve ficar clara é que precisamos selecionar o λ e agora duas vezes!

Vamos repetir a simulação ali de cima, mas usando o adaLASSO. Em ambos os estágios eu vou usar o Cross Validation:

```
coeficientes_adalasso <- Matrix(0,ncol=500,nrow=51)

for(i in 1:500){
  x <- matrix(rnorm(50*100),ncol = 50) #gerando x
  betas <- c(rep(1,10),rep(0,40)) #
  y <- x%*%betas + rnorm(100)
  primeiro_estagio <- cv.glmnet(x,y) #estimando usando LASSO e Cross Validation
  pesos <- 1/abs(coef(primeiro_estagio)[-1,]) #veja que eu tenho que jogar fora o intercepto
  adalasso <- cv.glmnet(x,y,penalty.factor = pesos)
  coeficientes_adalasso[,i] <- coef(adalasso)
}

#Fim da simulação
```

```r
analise <- matrix(0,ncol=3,nrow=500)
colnames(analise)<- c("Não zeros certos","Zeros certos", "Modelo certo?")

for(i in 1:500){
  analise[i,1]<- mean(coeficientes_adalasso[2:11,i] != 0)
  analise[i,2]<- mean(coeficientes_adalasso[12:51,i] == 0)
  analise[i,3]<- ifelse(analise[i,1]+analise[i,2] == 2,1,0)
}

tabela_final <- colMeans(analise)*100
knitr::kable(tabela_final,caption = "Os valores estão em porcentagem")
```

Table 2: Os valores estão em porcentagem

	x
Não zeros certos	100.00
Zeros certos	96.56
Modelo certo?	57.40

A performance do LASSO adaptativo é muito melhor que a do LASSO: em aproximadamente 60% dos casos agora recuperamos o modelo correto, contra um pouco mais de 1% dos casos para o LASSO. O adaLASSO é uma modificação extremamente simples do algoritmo do LASSO que gera excelente resultados. Veja que a escolha do Cross Validation aqui é apenas pela conveniência do glmnet já trazer o Cross Validation. Veja que implementar um critério de informação é extremamente simples. Eu vou descrever o algoritmo para o LASSO, e para o adaLASSO basta juntar os dois procedimentos:

1. Estime o modelo usando LASSO via o glmnet (não o cv.glmnet!). Isso vai devolver uma matriz de coeficientes, de dimensão $p \times L$, onde L é a quantidade de lambdas que a função usou.
2. Calcule o resíduo para cada modelo estimado $u_l = y - X\beta_l$
3. Calcule a soma do quadrado dos erros para cada modelo $SSR = \sum_{i=1}^{n} u_i^2$.
4. Crie um vetor que tem a quantidade de coeficientes não zeros para cada modelo estimado. Vamos nos referir a cada entrada desse modelo como s.
5. Calcule $IC = n\ln(SSR) + cs$ para cada modelo, onde c é o critério de informação escolhido
6. Escolha o lambda que minimiza o valor de IC

Vamos escrever uma função que faz cada um dos passos. Eu vou permitir que o usuário escolha qualquer um dos 3 critérios, e para isso eu usarei uns if:

```r
ic_glmnet <- function(x,y,penalty.factor,ic){
  require(glmnet)
  n <- length(y)
  modelo <- glmnet(x,y) #passo 1
  #passo 2
  x_aux <- cbind(1,x)
  u <- y - x_aux%*%coef(modelo)
  #passo 3
  ssr <- colSums(u^2)
  #passo 4
  conj_ativo <- colSums(coef(modelo) !=0)
  #vamos permitir o usuário escolha qual critério de informação vai ser usado
  if(ic == "aic"|ic=="AIC"){
    ic_val <- 2} else if(ic=="bic"|ic=="BIC"){
      ic_val <- log(n)
```

```r
    } else if(ic == "hqc"|ic == "HQC"){
      ic_val <- 2* log(log(n))
    } else{
      stop("IC not implemented")
    }
  #passo 5
  val <- n*log(ssr)+ic_val*conj_ativo
  #passo 6
  selecionado <- which.min(val)
  return(list("modelo_selecionado" = coef(modelo)[,selecionado],"modelo_glmnet" = modelo, "lambda_selec
}
```

Veja que eu faço a função retorna os coeficientes escolhidos, o modelo inteiro escolhido e o λ escolhido (o motivo para isso vai ficar claro). Vamos fazer um pequeno teste da nossa função:

```r
x <- matrix(rnorm(50*100),ncol = 50) #gerando x
betas <- c(rep(1,10),rep(0,40)) #
y <- x%*%betas + rnorm(100)
teste <- ic_glmnet(x,y,ic="BIC")
teste$modelo_selecionado
```

```
## (Intercept)          V1          V2          V3          V4          V5
##  0.16119226  0.78411957  0.86038404  1.09768532  1.07426857  0.71551084
##          V6          V7          V8          V9         V10         V11
##  0.80153921  0.96048905  0.83488429  1.08804089  0.97039455  0.00000000
##         V12         V13         V14         V15         V16         V17
## -0.11141625 -0.07557362  0.00000000  0.00000000  0.00000000  0.10736482
##         V18         V19         V20         V21         V22         V23
##  0.00000000  0.00000000 -0.08784322  0.00000000 -0.01297664 -0.21257744
##         V24         V25         V26         V27         V28         V29
##  0.00000000  0.00000000  0.00000000  0.00000000 -0.04838874 -0.04755889
##         V30         V31         V32         V33         V34         V35
##  0.00000000 -0.08244718  0.00000000  0.00000000 -0.13655546  0.00000000
##         V36         V37         V38         V39         V40         V41
##  0.00000000  0.00000000  0.03670344  0.00000000 -0.13452340  0.13106194
##         V42         V43         V44         V45         V46         V47
##  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.10010310
##         V48         V49         V50
##  0.04545107  0.00000000  0.00000000
```

E uma simulação, com uma mudança: nós não vamos escolher o λ duas vezes. Nós vamos repetir o λ escolhido no primeiro estágio para o segundo estágio. Isso tem dois objetivos:

1. Isola o efeito da seleção do lambda dos efeitos da pessagem do LASSO adaptativo
2. Ao invés de termos de selecionar os paramêtros duas vezes, selecionamos apenas uma vez.

Vamos a simulação:

```r
coeficientes_adalasso_bic <- Matrix(0,ncol=500,nrow=51)

for(i in 1:500){
  x <- matrix(rnorm(50*100),ncol = 50) #gerando x
  betas <- c(rep(1,10),rep(0,40)) #
  y <- x%*%betas + rnorm(100)
  primeiro_estagio <- ic_glmnet(x,y,ic = "BIC") #estimando usando LASSO e BIC
  pesos <- 1/abs(primeiro_estagio$modelo_selecionado[-1]) #veja que eu tenho que jogar fora o intercept
```

```r
  adalasso <- glmnet(x,y,lambda = primeiro_estagio$lambda_selecionado,penalty.factor = pesos)
  coeficientes_adalasso_bic[,i] <- coef(adalasso)
}

#Fim da simulação

analise <- matrix(0,ncol=3,nrow=500)
colnames(analise)<- c("Não zeros certos","Zeros certos", "Modelo certo?")

for(i in 1:500){
  analise[i,1]<- mean(coeficientes_adalasso_bic[2:11,i] != 0)
  analise[i,2]<- mean(coeficientes_adalasso_bic[12:51,i] == 0)
  analise[i,3]<- ifelse(analise[i,1]+analise[i,2] == 2,1,0)
}

tabela_final <- colMeans(analise)*100
knitr::kable(tabela_final,caption = "Os valores estão em porcentagem")
```

Table 3: Os valores estão em porcentagem

	x
Não zeros certos	100.00
Zeros certos	95.36
Modelo certo?	46.60

Veja que o BIC é bastante bem sucedido em recuperar o modelo certo. Veja que como mantivemos a penalidade parada, todo o ganho vem de acertar os pesos melhor, ilustrando bem qual é o segredo do adaLASSO. Apesar disso, ele é pior do que o Cross Validation. Isso vem com duas observações:

1. CV é um processo computacionalmente intensivo: quebre em blocos os seus dados, estime o modelo repetidas vezes.
2. Mais importante, Cross Validation supõe *independência* entre as observações, ou seja, não podemos usar ele para dados de séries temporais

O BIC não sofre de nenhum desses problemas, o que faz dele uma alternativa atraente para selecionar modelos com LASSO adaptativo.

Esse é um post que aborda duas coisas: LASSO adaptativo e seleção do parâmetro de penalidade via critérios de informação. Ambos tem aplicações interessantes e não precisam ser empregadas em conjunto.

Mostramos como o adaLASSO é bastante superior ao LASSO em seleção de modelos. Uma pergunta justa é se a diferença de previsão dos modelos são tão diferentes assim…