EPD894 - Modelos de Regressão Paramétricos e Não-Paramétricos: Teoria e Aplicações Implementação Computacional

José Geraldo Fernandes
Escola de Engenharia
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
josegeraldof@ufmg.br

Este trabalho mostra uma implementação do algoritmo de Newton-Raphson para regressão de Bernoulli. Compara-se o resultado com o obtido a partir da função nativa do R, *glm*, e uma implementação de regularização. Todo o código desenvolvido está disponível em repositório Git~~, apanhei pro markdown :(~~.

I. Conjunto de Dados

Utilizou-se a base de dados *Banknote Authentication*, disponível no repositório UCI. Trata-se de um problema de classificação de cédulas monetárias entre genuínas e fraudulentas. São 1372 amostras e 4 atributos conseguidos a partir da extração de características de imagens das cédulas pela transformada Wavelet. As classes são balanceadas e esta base mostra alta acurácia para a maioria dos modelos de classificação.

II. Algoritmo, glm.r

O método consiste no cálculo de quatro equações: probabilidade estimada; matriz $\boldsymbol{W}$; matriz $\boldsymbol{Z}$; e, resultado dos parâmetros $\boldsymbol{\beta}$. O código é como na Figura 1.

$$\boldsymbol{p} = \frac{1}{1+\exp(-\boldsymbol{X\beta})}$$

$$[\boldsymbol{W}]_{ii} = p_i(1-p_i)$$

$$\boldsymbol{Z} = \boldsymbol{X\beta} + \boldsymbol{W}^{-1}(\boldsymbol{y}-\boldsymbol{p})$$

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{Z}$$

Para o cálculo da inversão de $\boldsymbol{W}$, por problemas de computação numérica, fez-se a inversão dos elementos da diagonal.

$$[\boldsymbol{W}^{-1}]_{ii} = \frac{1}{p_i(1-p_i)}$$

Para critério de parada utilizou-se um número máximo de iterações e uma tolerância mínima de mudança do vetor de parâmetros, $\Delta = \max(|\boldsymbol{\beta} - \boldsymbol{\beta}_{anterior}|)$.

A estimativa do vetor de parâmetros, $\boldsymbol{\beta}$, inicial partiu-se do resultado de um modelo de regressão linear a partir de uma transformação não linear da saída. Por motivo númerio, adicionou-se uma pequena perturbação δ na saída.

```
# model
epochs <- model$iter
print(paste('glm epochs:', epochs))
tol <- 1e-3
epoch <- 0
delta <- 1e6
while (delta > tol & epoch < epochs){
  p <- as.vector(1 / (1 + exp(- X %*% beta)))
  W <- p * (1 - p) * diag(length(p))
  W_inv <- 1 / (p * (1 - p)) * diag(length(p))
  Z <- X %*% beta + W_inv %*% (Y - p)

  beta_old <- beta
  beta <- solve(t(X) %*% W %*% X) %*% t(X) %*% W %*% Z
  delta <- max(abs(beta - beta_old))
  epoch <- epoch + 1
  print(paste('epoca:', epoch, '| delta:', delta))
}
```

Figura 1. Algoritmo implementado.

$$\boldsymbol{X\beta}_0 = -\ln(\tilde{\boldsymbol{y}}^{-1} - 1)$$

$$y_i = 0 \rightarrow \tilde{y}_i = \delta$$

$$y_i = 1 \rightarrow \tilde{y}_i = 1 - \delta$$

```
# first guess
y_num = matrix(0, nrow = nrow(Y), ncol = ncol(Y))
delta <- 1e-3
y_num[Y == 0] = delta
y_num[Y == 1] = 1 - delta
y_lm <- - log(1 / y_num - 1)

data_lm <- data.frame(y = y_lm, x = train)
linear <- lm(y_lm ~ train, data = data_lm)

beta <- linear$coefficients
```

Figura 2. Estimativa inicial dos parâmetros.

Testou-se também com o vetor nulo e o resultado foi similar.

III. Comparação

Para validação, comparou-se com o resultado obtido pela função nativa do R, *glm*. Para isso, limitou-se o número de

épocas do algoritmo para o encontrado na função e imprimiu-se a diferença do vetor de parâmetros.

$$\epsilon = \boldsymbol{\beta}_{glm} - \boldsymbol{\beta}$$

O algoritmo convergiu por tolerância e os parâmetros encontrados foram muito próximos, 1e-7, do resultado pela função nativa.

```
> source("~/Documents/ufmg/mestrado/2022-1/reg/glm.r")
[1] "banknote"
[1] "glm epochs: 12"
[1] "epoca: 1 | delta: 1.61860056044772"
[1] "epoca: 2 | delta: 9.26947076451187"
[1] "epoca: 3 | delta: 12.4157111388206"
[1] "epoca: 4 | delta: 15.17637420402"
[1] "epoca: 5 | delta: 15.4671646323341"
[1] "epoca: 6 | delta: 10.0919066490031"
[1] "epoca: 7 | delta: 2.5570202820153"
[1] "epoca: 8 | delta: 0.12229382687633"
[1] "epoca: 9 | delta: 0.00026448795446754"
[1] "epsilon:"
             [,1]
[1,] 4.806734e-08
[2,] 2.722050e-07
[3,] 2.956573e-07
[4,] 4.040281e-07
[5,] 2.470231e-08
```

Figura 3. Resultado da execução.

IV. Regularização, glm_l2.r

Para a implementação da regularização, incorporou-se uma restrição do tipo L2 na equação do cálculo dos parâmetros, como na Figura 4.

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{Z}$$

```r
# epochs <- model$iter
epochs <- 100
tol <- 0.001
epoch <- 0
delta <- 1e6
while (delta > tol & epoch < epochs){
  p <- as.vector(1 / (1 + exp(- X %*% beta)))
  W <- p * (1 - p) * diag(length(p))
  W_inv <- 1 / (p * (1 - p)) * diag(length(p))
  Z <- X %*% beta + W_inv %*% (Y - p)

  beta_old <- beta
  beta <- solve(t(X) %*% W %*% X + lambda * diag(ncol(X))) %*% t(X) %*% W %*% Z
  delta <- max(abs(beta - beta_old))
  epoch <- epoch + 1
  # print(c(epoch, delta))
}
```

Figura 4. Modificação do algoritmo para regularização.

Para estimar o parâmetro λ, fez-se uma varredura em *grid* de dois parâmetros: R^2_{pred}; e, erro quadrático médio (mse) em um conjunto separado de teste. Para isso, calculou-se uma matriz $\boldsymbol{H}$ de forma semelhante ao caso da regressão linear. O código é como na Figura 5.

$$R^2_{pred} = 1 - \frac{PRESS}{SQT}$$

$$PRESS = \sum_i (\frac{r_i}{1 - h_{ii}})^2$$

$$\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{W}$$

$$r_i = y_i - \hat{y}_i$$

$$SQT = \sum_i (y_i - \bar{\boldsymbol{y}})^2$$

```r
# press
y_hat <- as.vector(1 / (1 + exp(- X %*% beta)))
r <- Y - y_hat
H <- X %*% solve(t(X) %*% W %*% X + lambda * diag(ncol(X))) %*% t(X) %*% W

y_bar <- mean(Y)

press <- 0
sqt <- 0
for (i in 1:nrow(r)){
  press <- press + (r[i] / (1 - H[i, i]))^2
  sqt <- sqt + (Y[i] - y_bar)^2
}
R2_pred <- 1 - press / sqt

# cv
X_test <- cbind(1, test)
y_hat <- as.vector(1 / (1 + exp(- X_test %*% beta)))
mse <- sum((class_test - y_hat)^2) / length(y_hat)
```

Figura 5. Cálculo das funções de validação.

As Figuras 6 e 7 mostram o resultado deste dois parâmetros. Há uma aproximação dos pontos ótimos, o que valida o resultado para este conjunto de dados.

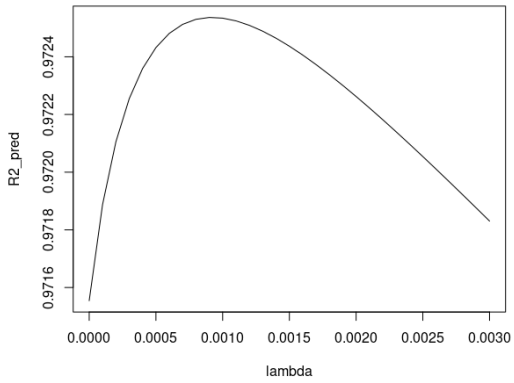

Figura 6. R2 preditivo.

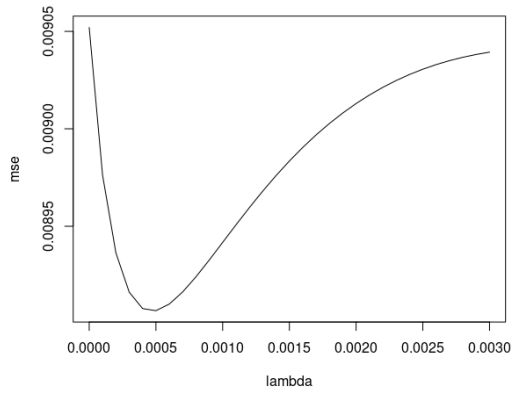

Figura 7. Erro quadrático médio.