逻辑回归似然函数用牛顿法迭代寻优过程和程序优化

汪齐
2018310864@email.cufe.edu.cn

关于逻辑回归的对数似然函数用牛顿法寻优的方法，之前在课堂展示时也有人提到过，在循环迭代过程中会出现返回 Inf 或 NaN 的情况，汪齐同学经过排查后认为这是在计算 exp() 时，因为返回值超出双精度浮点型数值存储范围导致，另外他发现一些数值在计算过程中完全可以做近似计算，从而基本上解决了这个问题，而不需要频繁地变更初始值。具体的思路和过程如下。

根据逻辑回归的似然函数以及梯度矩阵和海森矩阵（具体见附录），计算似然函数的程序如下：

```
FIT = X%*%t(BETA)
P = 1/(1+exp(-FIT))
f = t(Y)%*%(log(P))+t(1-Y)%*%log(1-P) # 似然函数值
```

其中 X 是 n*p 的预测变量矩阵，Y 是 n*1 的逻辑值列向量。
计算梯度矩阵的程序如下：

```
dbeta1 = t(X)%*%(Y-P)
```

计算海森矩阵的程序如下：

```
trans = matrix(1, 1, p)
THET = (exp(-FIT)/((1+exp(-FIT))^2))
THET = THET%*%trans
dbeta2 = t(X)%*%(X*THET)
```

但在实际操作中，会出现 f=Inf 而出错，经检查发现，在计算 P,THET 时，会出现返回 Inf 或 NaN 的情况，推测是计算 exp(-FIT)时返回值超出双精度浮点型数值范围导致。
使用如下代码考察使 exp(x)不返回 Inf 的最大值

```
flag = T
x = 0
while(flag)
{
  x = x+1
  y = exp(x)
  if(y == Inf)
  {flag = F}
}
```

得知使 exp(x)不返回 Inf 的最大 x 在 709 至 710 之间
使用代码 max(FIT)考察 FIT 的最大值，返回 1882.916，远大于 710，可以肯定是计算 exp(-FIT)时返回值超出双精度浮点型数值范围导致出错。
解决方法：
由于

```
f = t(Y)%*%(log(P))+t(1-Y)%*%log(1-P)
```

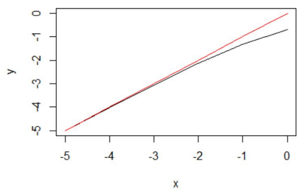
其中

```
P = 1/(1+exp(-FIT))
```

所以

f = t(Y)%*%(-log(1+exp(-FIT)))+t(1-Y)%*%(-FIT-log(1+exp(-FIT)))

显然，在-FIT 较大时，log(p) = -log(1+exp(-FIT))可以用 FIT 近似，如图（黑色为 y=-log(1+exp(-x))红色为 y=x（之前图放错了））



因此，用直接计算

log(P) = -log(1+exp(-FIT))

log(1-P) = -FIT-log(1+exp(-FIT)) = -FIT+log(p)

并带入 f,

并在-FIT 较大时（返回-Inf），用 FIT 中的对应值代替-log(p)中的-Inf 值来代替计算 P 并带入 f

实现代码为：

```
logP = -log(1+exp(-FIT))
logP[logP == -Inf] = FIT[logP == -Inf]
f = t(Y)%*%(-logP)+t(1-Y)%*%(-FIT+logP)
```

由

THET = exp(-FIT)/((1+exp(-FIT))^2)

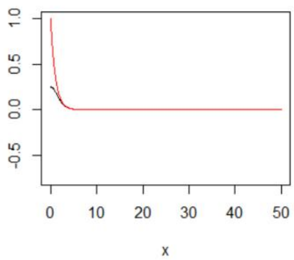
化简得

THET = 1/(exp(-FIT)+exp(FIT)+2)

显然，在 FIT 较大时，THET 可以用

1/exp(abs(FIT))

近似，如图（红色为 y=1/exp(abs(x))，黑色为 y=1/(exp(-x)+exp(x)+2)）：



因此，计算

THET = exp(-FIT)/((1+exp(-FIT))^2)

THETp = 1/exp(abs(FIT))

并在 FIT 较大时（返回 Inf），用 THETp 中的对应值替代 THET 中的 Inf 值

实现代码为

```
THET = (exp(FIT)/((1+exp(FIT))^2))
THETp = 1/exp(abs(FIT))
THET[exp(FIT) == Inf] = THETp[exp(FIT) == Inf]
```

解决上述问题后重新运行程序，因海森矩阵不可逆而出错。

显然，在迭代后期，随着 abs(FIT)的增大，海森矩阵趋向 0 矩阵是不可避免的。解决问题的思路是：海森矩阵非满秩时，改为用梯度下降法继续执行迭代。这样也结合了牛顿法和梯度下降法的优点。

实现代码为：

```
        if(qr(dbeta2)$rank == p)
        {
          BETA = BETA-t(solve(dbeta2)%*%dbeta1)
        }else
        {
          BETA = BETA+(step*t(dbeta1))
          step = step*at
        }
```

解决上述问题后的程序见文件” 逻辑回归似然函数牛顿迭代寻优函数.R”（由于同时使用了牛顿寻根和梯度下降，函数名改为 hibrid）：

使用数据集”iris”运行函数并可视化结果的程序见文件” 逻辑回归似然函数牛顿迭代寻优函数实操.R”

附录：

逻辑回归的对数似然函数

$$L(\beta) = \sum_{i=1}^{n} [y_i \log P_i + (1 - yi) \log (1 - p_i)]$$

逻辑回归对数似然函数对β的梯度矩阵：

$$\frac{\partial L}{\partial \beta} = \begin{vmatrix} \sum_{i=1}^{n} (y_i - P_i) x_{1i} \\ \vdots \\ \sum_{i=1}^{n} (y_i - P_i) x_{pi} \end{vmatrix}$$

逻辑回归对数似然函数对β的海森矩阵：

$$\frac{\partial L}{\partial \beta \partial \beta^{'}} = \begin{pmatrix} -\sum_{1=1}^{n} x_{1i} x_{1i} \theta_i & \cdots & -\sum_{1=1}^{n} x_{1i} x_{pi} \theta_i \\ \vdots & \ddots & \vdots \\ -\sum_{1=1}^{n} x_{pi} x_{1i} \theta_i & \cdots & -\sum_{1=1}^{n} x_{pi} x_{pi} \theta_i \end{pmatrix}$$

其中

$$\theta_i = \frac{\mathrm{e}^{-\varepsilon_i}}{(1 + \mathrm{e}^{-\varepsilon_i})^2}$$

其中

$$\varepsilon_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{iP}$$