R 语言编程：基于 tidyverse

第 22 讲（选讲） 梯度下降法, 广义线性模型

张敬信

2022 年 3 月 8 日

哈尔滨商业大学

一. 梯度下降法

正规方程法求解多元线性回归，简单、容易实现，但有其缺点：

- 若 X^TX 不可逆，则正规方程法失效
- 若样本量非常大（$n > 10000$），矩阵求逆会非常慢

梯度下降法是广泛用于机器学习，其核心思想是迭代地调整参数，使得损失函数达到最小值。

梯度下降法，就好比在浓雾笼罩的山上下山，每次只能看到前方一步远，那么就 360° 每个方向迈一步的话下降的最多，那就往哪个方向迈一步，重复该过程，逐步到达较低点（不一定是最低点）。

根据数学知识，一步下降最快的方向就是梯度方向！

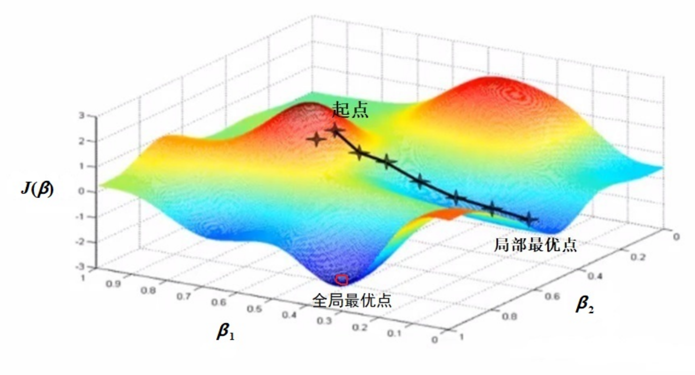


图 1: 梯度下降法示意图

线性回归问题，就是计算损失函数 $J(\beta)$ 关于参数向量 β 的局部梯度，同时它沿着梯度下降的方向进行下一次迭代。当梯度值为零的时候，就达到了损失函数最小值。

开始需要选定一个随机的 β（初始值），然后逐渐去改进它，每一次变化一小步，每一步都试着降低损失函数 $J(\beta)$，直到算法收敛到一个极小值。

该极小值不一定是全局最小值，若损失函数是凸函数（线性回归损失函数是凸函数），则极小值就是唯一的全局最小值。

梯度下降法的重要参数是每一步的步长，叫作**学习率**。一个好的策略是，开始的学习率大一些以更快速趋于收敛，让学习率慢慢减小，最后阶段要足够小以稳定地到达收敛点。

注：梯度下降法对自变量取值的量级是敏感的，若所有自变量的数量级基本相当，则能更快地收敛到最小值。所以，在用梯度下降法训练模型时，有必要对数据做归一化（放缩），以加速训练。

线性回归模型的损失函数为（除以 2 可抵消求偏导的系数）:

$$J(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(x^{(i)}\beta - y_i)^2$$

在梯度下降法过程中，需要计算每一个 β_j（维度）下损失函数的梯度。即当 β_j 变化一点点时，损失函数改变了多少，这就是偏导数:

$$\frac{\partial}{\partial \beta_j}J(\beta) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)}\beta - y_i)x_{ij}, \quad j = 1, \cdots, m$$

改为向量化表示，得到损失函数的梯度向量：

$$\nabla J(\beta) = \left[\frac{\partial}{\partial \beta_1} J(\beta), \cdots, \frac{\partial}{\partial \beta_m} J(\beta)\right] = \frac{1}{n} X^T (X\beta - y)$$

注意，梯度下降法每一步梯度向量的计算，都是基于整个训练集，故称为批量梯度下降：每一次训练过程都使用所有的训练数据。因此，在大数据集上，训练速度也会变得很慢[1]，但其复杂度是 $O(n)$，比正规方程法 $O(n^3)$ 快的多。

梯度向量有了，只需要每步以学习率 η 调整参数即可：

$$\beta^{\text{next}} = \beta \nabla J(\beta)$$

[1]进一步提速，还有随机梯度下降算法，每次只用一个随机样本计算梯度向量，但是收敛过程不够稳定，折衷的做法是小批量梯度下降算法.

- 定义函数实现梯度下降法求解线性回归

```
gd = function(X, y, init, eta = 1e-3, err = 1e-3,
              maxit = 1000, adapt = FALSE) {
  ## X 为自变量数据矩阵, y 为因变量向量, init 为参数初始值
  ## eta 为学习率, err 为误差限, maxit 为最大迭代次数,
  ## adapt 是否自适应修改学习率
  ## 返回回归系数估计, 损失向量, 迭代次数, 拟合值, RMSE
  # 初始化
  X = cbind(Intercept = 1, X)
  beta = init
  names(beta) = colnames(X)
  loss = crossprod(X %*% beta - y)
  tol = 1
  iter = 1
```

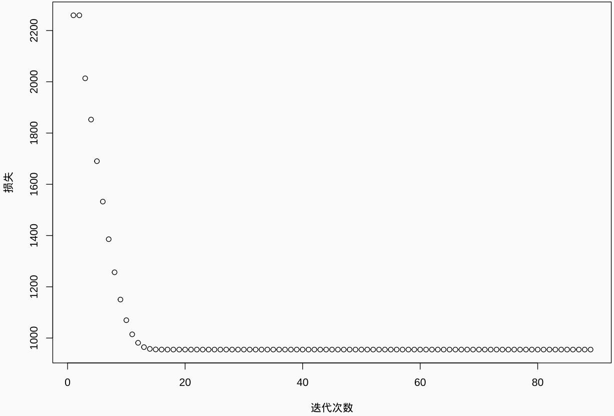
```
  while(tol > err && iter < maxit) {       # 迭代
    LP = X %*% beta
    grad = t(X) %*% (LP - y)
    betaC = beta - eta * grad
    tol  = max(abs(betaC - beta))
    beta = betaC
    loss = append(loss, crossprod(LP - y))
    iter = iter + 1
    if(adapt)
      eta = ifelse(loss[iter] < loss[iter-1], eta * 1.2,
                   eta * 0.8)
  }
  list(beta = beta, loss = loss, iter = iter, fitted = LP,
    RMSE = sqrt(crossprod(LP - y) / (nrow(X) - ncol(X))))
}
```

- 用随机生成数据的二元线性回归来测试函数

```r
n = 1000
set.seed(123)
x1 = rnorm(n)
x2 = rnorm(n)
y = 1 + 0.6*x1 - 0.2*x2 + rnorm(n)
X = cbind(x1, x2)
```

```r
gd_rlt = gd(X, y, rep(0,3), err = 1e-8, eta = 1e-4,
            adapt = TRUE)
rbind(gd = round(gd_rlt$beta[, 1], 5),
      lm = coef(lm(y ~ x1 + x2)))      # 与 lm 结果对比
#>    Intercept    x1     x2
#> gd     0.979 0.579 -0.172
#> lm     0.979 0.579 -0.172
gd_rlt$iter                            # 迭代次数
#> [1] 89
```

```r
plot(gd_rlt$loss, xlab = " 迭代次数", ylab = " 损失")
```



可见，算法收敛速度非常快，迭代 14 步损失函数基本就不再减小。

二. 广义线性模型

线性回归是回归家族的基本模型，从不同角度进行扩展可以衍生出几十种回归模型。

线性回归要求残差满足正态性：$\varepsilon = y - X\beta \sim N(0,\, \sigma^2)$, 则 $y \sim N(X\beta,\, \sigma^2)$. 这说明线性回归通常要求因变量 y 是近似服从正态分布的连续数据。

但实际中，因变量数据可能会是类别型、计数型等，可以考虑对 y 做变换，或直接考虑广义线性模型。

要让线性回归也适用于因变量非正态连续情形，就需要推广到广义线性模型。Logistic 回归、softmax 回归、泊松回归、Probit 回归、二项回归、负二项回归、最大熵模型等都是广义线性模型的特例。

广义线性模型，相当于是复合函数。先做线性回归，再接一个变换：

$$\mathbf{w}^T X + \mathbf{b} = \ u \sim \text{正态分布}$$
$$\downarrow$$
$$g(u) = y$$

经过变换后到达非正态分布的因变量数据。

一般更习惯反过来写：即对因变量 y 做一个变换，就是正态分布，从而就可以做线性回归：

$$\sigma(y) = \mathbf{w}^T X + \mathbf{b}$$

其中，$\sigma(\cdot)$ 称为连接函数。

回归模型	变换	连接函数	逆连接函数	误差
线性回归	恒等	$\mu_Y = X^T\beta$	$\mu_Y = X^T\beta$	正态分布
Logistic 回归	Logit	$\text{Logit}\,\mu_Y = \mathrm{X}^{\mathrm{T}}\beta$	$\mu_Y = \frac{\exp(X^T\beta)}{1+\exp(X^T\beta)}$	二项分布
泊松回归	对数	$\ln \mu_Y = X^T\beta$	$\mu_Y = \exp(X^T\beta)$	泊松分布
负二项回归	对数	$\ln \mu_Y = X^T\beta$	$\mu_Y = \exp(X^T\beta)$	负二项分布
Gamma 回归	逆	$\frac{1}{\mu_Y} = X^T\beta$	$\mu_Y = \frac{1}{X^T\beta}$	Gamma 分布

图 2: 常见连接函数与误差函数

注 1： 因变量数据只要服从指数族分布：正态分布、伯努利分布、泊松分布、指数分布、Gamma 分布、卡方分布、Beta 分布、狄里克雷分布、Categorical 分布、Wishart 分布、逆 Wishart 分布等，就可以使用对应的广义线性模型。

注 2： 泊松回归和负二项回归都是针对因变量是计数数据，区别是泊松回归一般用于个体之间独立的情形；负二项回归则可用于个体之间不独立的情形。

本篇主要参阅 (张敬信, 2022), Michael Clark: gradient_descent，模板感谢 (黄湘云, 2021), (谢益辉, 2021).

参考文献

张敬信 (2022). *R* 语言编程：基于 *tidyverse*. 人民邮电出版社, 北京.

谢益辉 (2021). *rmarkdown: Dynamic Documents for R*.

黄湘云 (2021). *Github: R-Markdown-Template*.