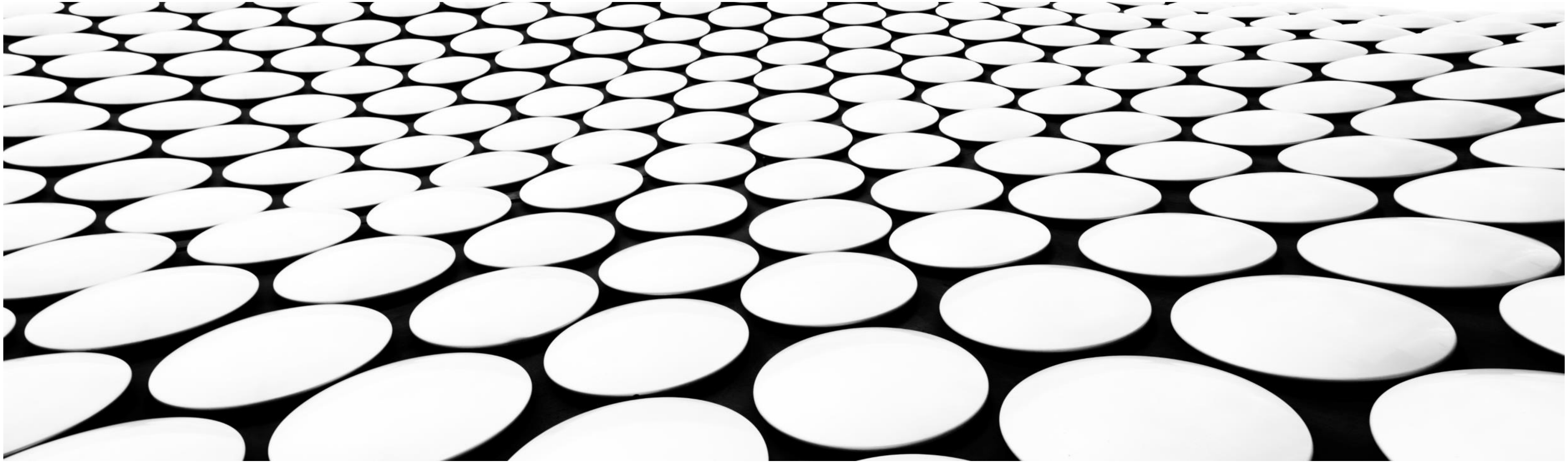
分布式计算

邱怡轩



今天的主题

- 分布式 Logistic 回归
- 梯度下降法与牛顿法

问题的提出：Logistic 回归

Logistic 回归

- 假定 $Y|x \sim Bernoulli(\rho(\beta' x))$
- $\rho(x) = 1/(1 + e^{-x})$，即 Sigmoid 函数
- $\rho(\beta' x)$ 代表 Y 取1的概率

- 给定数据 (y_i, x_i)，$i = 1, \ldots, n$
- 估计 β

目标函数

- 利用极大似然准则

$$L(\beta) = -\sum_{i=1}^{n}\{y_i \log \rho_i + (1-y_i)\log(1-\rho_i)\}$$

- 其中 $\rho_i = \rho(x_i'\beta)$

- 找到一个 β 的取值，使得 $L(\beta)$ 最小

迭代算法

- 与线性回归不同的是，Logistic 回归的系数估计没有显式解
- 需要使用迭代式优化算法来求解

最优化算法

优化问题

- 找到参数 $x \in \mathbf{R}^d$ 的取值，使得函数 $f(x)$ 的取值达到最小
- 记为

$$\min_x f(x)$$

- $f(x)$ 通常具有一些特定的性质，如可导、凸性等

优化算法

- 一阶算法（梯度下降法）
- 二阶算法（牛顿法）

梯度下降法

- $x^{new} = x^{old} - \alpha \cdot \left.\frac{\partial f(x)}{\partial x}\right|_{x=x^{old}}$
- α 称为步长或学习率

牛顿法

- $x^{new} = x^{old} - \alpha \cdot \left(\frac{\partial^2 f(x)}{\partial x \partial x'}\right)^{-1} \cdot \frac{\partial f(x)}{\partial x}\Big|_{x=x^{old}}$
- 牛顿法可以“自适应”步长
- 可以固定 $\alpha = 1$，也可手动调节

Logistic 回归

目标函数

- 极大似然准则

$$L(\beta) = -\sum_{i=1}^{n}\{y_i \log \rho_i + (1-y_i)\log(1-\rho_i)\}$$

- 其中 $\rho_i = \rho(x_i'\beta)$

符号定义

- y：因变量向量，$n \times 1$
- X：自变量矩阵，$n \times (p+1)$
- ρ：$\rho_i = \rho(x_i'\beta)$，$n \times 1$
- W：以 $\rho_i(1-\rho_i)$ 为对角线元素的对角矩阵

导数

- 一阶导数（梯度）向量化表示

$$\frac{\partial L(\beta)}{\partial \beta} = X'(\rho - y)$$

导数

- 一阶导数（梯度）向量化表示

$$\frac{\partial L(\beta)}{\partial \beta} = \underbrace{\boxed{X}}_{n \times (p+1)}{}'(\overbrace{\boxed{p}}^{n \times 1} - \underbrace{\boxed{y}}_{n \times 1})$$

导数

- 二阶导数（Hessian 矩阵）向量化表示

$$\frac{\partial^2 L(\beta)}{\partial\beta\partial\beta'} = X'WX$$

- $W = \mathbf{diag}(\rho_1(1-\rho_1), \ldots, \rho_n(1-\rho_n))$

导数

- 二阶导数（Hessian 矩阵）向量化表示

$$\frac{\partial^2 L(\beta)}{\partial\beta\partial\beta'} = \underbrace{\overbrace{\boxed{X}}^{n\times(p+1)}{}'\,\boxed{W}}_{n\times n} X$$

- $W = \mathbf{diag}(\rho_1(1-\rho_1), \ldots, \rho_n(1-\rho_n))$

牛顿法

牛顿法

- 参考 https://online.stat.psu.edu/stat508/lesson/9/9.1/9.1.2

迭代公式

- $\beta^{new} = \beta^{old} - \left(\frac{\partial^2 L(\beta)}{\partial\beta\partial\beta'}\right)^{-1} \cdot \frac{\partial L(\beta)}{\partial\beta}\bigg|_{\beta=\beta^{old}}$

迭代公式

The Newton-Raphson step is:

$$\begin{aligned}\beta^{new} &= \beta^{old} + (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y}-\mathbf{p})\\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}(\mathbf{X}\beta^{old}+\mathbf{W}^{-1}(\mathbf{y}-\mathbf{p}))\\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z}\end{aligned}$$

where $\mathbf{z} \stackrel{\triangle}{=} \mathbf{X}\beta^{old} + \mathbf{W}^{-1}(\mathbf{y}-\mathbf{p})$

分块计算

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{pmatrix} \quad X \in \mathbb{R}^{n \times p}, \quad X_i \in \mathbb{R}^{n_i \times p}$$

$$W = \begin{pmatrix} W_1 & & & \\ & W_2 & & \\ & & \ddots & \\ & & & W_m \end{pmatrix} \quad W \in \mathbb{R}^{n \times n}, \quad W_i \in \mathbb{R}^{n_i \times n_i}$$

$$Z = \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_m \end{pmatrix} \quad Z \in \mathbb{R}^{n \times 1}, \quad Z_i \in \mathbb{R}^{n_i \times 1}$$

$$X'WX = (X_1' \cdots X_m') \begin{pmatrix} W_1 & & \\ & \ddots & \\ & & W_m \end{pmatrix} \begin{pmatrix} X_1 \\ \vdots \\ X_m \end{pmatrix}$$

$$= X_1' W_1 X_1 + \cdots + X_m' W_m X_m$$

$$X'WZ = (X_1' \cdots X_m') \begin{pmatrix} W_1 & & \\ & \ddots & \\ & & W_m \end{pmatrix} \begin{pmatrix} Z_1 \\ \vdots \\ Z_m \end{pmatrix}$$

$$= X_1' W Z_1 + \cdots + X_m' W_m Z_m$$

实现方法

1. 从原始数据生成 RDD（与线性回归步骤相同）
2. 将 β^{old} 发送至每个分区
3. 在每个分区上计算 p 和 W 的对角线
4. 在每个分区上计算 $X'WX$ 和 $X'Wz$
5. 汇总分区结果，计算完整的 $X'WX$ 和 $X'Wz$
6. 更新 β
7. 反复迭代直至收敛

梯度下降法

迭代公式

- $\beta^{new} = \beta^{old} - \alpha \cdot \left.\frac{\partial L(\beta)}{\partial \beta}\right|_{\beta=\beta^{old}}$
- 注意，$\partial L(\beta)/\partial \beta$ 通常会随着 n 而增长，建议使用 $n^{-1} \cdot \partial L(\beta)/\partial \beta$
- α 的选取需进行一些尝试

实现方法

1. 从原始数据生成 RDD（与线性回归步骤相同）
2. 在每一个分区上计算 $\rho = \rho(X\beta)$
3. 分布式地计算 $X'(\rho - y)$
4. 计算梯度并更新 β
5. 反复迭代直至收敛

实现

- lec9-logistic-regression.ipynb

方法对比

- 梯度下降法
  - 计算简单，只需求一阶导数
  - 适合高维问题，存储 $p \times 1$ 向量
  - 收敛较慢
  - 需人工设置步长
- 牛顿法
  - 需要求二阶导数（Hessian 矩阵）
  - 存储 $p \times p$ 矩阵，计算 $O(p^3)$
  - 通常收敛很快
  - 自适应步长

小结

小结

- 通过分布式计算方法得到精确解
- 对原问题没有进行近似或抽样
- 已实现方法
  - 通过显式解计算（线性回归）
  - 共轭梯度法（岭回归）
  - 梯度下降法（Logistic 回归）
  - 牛顿法（Logistic 回归）
- 核心都是利用分块矩阵的计算规则