

引用格式:陈少东,李志强.海量数据广义线性模型变量选择算法研究[J].北京化工大学学报(自然科学版),2020,47(2):130-136.

CHEN ShaoDong, LI ZhiQiang. A variable selection algorithm for generalized linear models of massive data[J]. Journal of Beijing University of Chemical Technology (Natural Science), 2020,47(2):130-136.

海量数据广义线性模型变量选择算法研究

陈少东 李志强*

(北京化工大学 数理学院, 北京 100029)

摘 要: 首先推导出了用于求解一般广义线性模型变量选择问题的非凸惩罚迭代估计算法,并利用分治思想对算法进行修正,使其能够适用于海量数据情形,以解决海量数据下进行变量选择时可能存在的内存溢出等问题。考虑到当前处理海量数据实际使用的工具,进一步给出了算法在分布式并行下的计算步骤,大幅提高了计算速度。在数值模拟中,通过单机和集群两种方式对算法进行数值计算,结果表明本文方法有效解决了数据存储问题且适用于分布式环境。最后,通过所提算法来完成 Probit 模型的变量选择,并将其用于新闻数据集的分类问题。

关键词: 海量数据; 广义线性模型; 变量选择; 分治算法

中图分类号: O212 **DOI:** 10.13543/j.bhxbzr.2020.02.018

引 言

广义线性模型由 Nelder 等^[1]在研究指数族分布时引入,该模型不仅能刻画连续型数据,还可刻画计数数据、属性数据等离散型数据,因此在医学、经济学以及社会科学等领域得到广泛应用。

随着数据时代的到来,数据呈现海量趋势。数据海量主要体现在两个方面:一是能获取的变量维度越来越高;二是可获取的观测样本越来越多。在数据量不是很大时,面对高维的变量如何进行变量选择的问题已得到了广泛的研究。在以往文献中,有许多研究人员利用惩罚函数进行变量选取,例如 Tibshirani^[2]提出的 least absolute shrinkage and selection operator (Lasso) 惩罚、Fan 等^[3]提出的 smoothly clipped absolute deviation (SCAD) 惩罚以及 Zhang^[4]提出的 minimax concave penalty (MCP) 惩罚等。

为了解决基于惩罚函数的广义线性模型变量选择问题,Breheeny 和 Huang^[5]给出一种基于坐标下降法的计算方法,该算法能有效求出模型局部最优解;

然而,在海量数据环境下,由于该方法使用了全部海量数据进行迭代计算,导致出现计算效率较低甚至计算机存储空间不足等问题。为了克服这些瓶颈,需要探究新的估计算法来解决海量数据下广义线性模型的变量选择问题。

分治算法是在海量数据下估计统计模型参数的一类主流方法。目前已有部分文献利用分治算法研究海量数据下的统计模型。其中,Lin 等^[6]提出了一种解决海量数据非线性估计方程的估计方法;在变量选取方面,方方等^[7]研究了海量数据下 Logistic 模型的平均算法;Chen 等^[8]给出了一种非凸惩罚下具有典则连接的广义线性模型的变量选择估计方法。基于分治算法处理海量数据的详细介绍可参考文献^[9]。

然而,以上研究的相关工作还有待进一步深入,例如采用文献^[7]中的模型平均估计方法难以进行高效的运算;文献^[6]、文献^[8]只研究了海量数据下具有典则连接的广义线性模型。但在实际建模应用中,大部分连接函数都是非典则连接,因此研究一般广义线性模型的惩罚估计具有更重要的现实意义。

本文基于文献^[5]在 Logistic 模型下的估计方法,推导出一般广义线性模型在 SCAD 惩罚和 MCP 惩罚下的迭代公式,并将该方法推广到了海量数据下一般广义线性模型中。该方法避免了在迭代过程中一次性加载全部数据,从而有效地克服了基于原

收稿日期:2019-11-03

第一作者:男,1994年生,硕士生

*通信联系人

E-mail: li-zhiqiang2000@163.com

始海量数据的估计方法带来的计算机存储空间不足等问题。与此同时,考虑到当前在解决海量数据问题时,大部分处理工具采用分布式并行框架,本文结合分治算法与分布式计算的共同特征,给出了算法在 Spark 集群环境的计算步骤。模拟结果表明,估计算法在克服内存瓶颈的同时提高了计算效率,其估计精度要优于随机抽样得到的估计,而且能有效应用于集群环境。

1 广义线性模型及惩罚方法

1.1 广义线性模型

一般广义线性模型满足

$$\begin{cases} f(y_i) = \exp \left\{ \frac{y_i \theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi) \right\} \\ E(y_i) = \mu_i = b'(\theta_i) \\ \text{Var}(y_i) = \sigma_0^2 V(\mu_i) \\ g(\mu_i) = \mathbf{x}_i^T \boldsymbol{\beta} \end{cases} \quad (1)$$

式中, $f(\cdot)$ 为概率密度函数, $y_i \in \mathbb{R}$ 为响应变量, $\mathbf{x}_i \in \mathbb{R}^p$ 为观测变量, $\boldsymbol{\beta} \in \mathbb{R}^p$ 为未知参数, θ 为典则参数, $a(\cdot)$ 、 $b(\cdot)$ 、 $c(\cdot)$ 为已知函数, μ_i 为均值函数, $V(\mu_i)$ 为方差函数, g 为连接函数, σ_0^2 为常数, $i = 1, \dots, N$ 。

为了估计模型的未知参数 $\boldsymbol{\beta}$, 通常采用极大似然估计方法, 其似然函数可表示为

$$l(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^N \left[\frac{y_i \theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi) \right] \quad (2)$$

1.2 基于惩罚的变量选择

惩罚函数方法是一种常用的变量选择方法, 在进行变量选择时, 通过在式(2)上添加惩罚项, 即可得到惩罚目标函数, 现将其转化为极小化问题

$$l_p(\boldsymbol{\beta}) = -l(\boldsymbol{\beta}) + \sum_{j=1}^p p_{\lambda, \gamma}(|\beta_j|) \quad (3)$$

式中, β_j 为 $\boldsymbol{\beta}$ 的第 j 维分量。

SCAD 惩罚和 MCP 惩罚是两种常用的非凸惩罚函数, 与 Lasso 方法相比, 这两种惩罚能保证 Oracle 性质, 因此本文将围绕这两种惩罚方法进行研究。

1.2.1 SCAD 惩罚函数

SCAD 惩罚函数在 $\beta_j \in [0, \infty)$ 上定义为

$$p_{\lambda, \gamma}(\beta_j) = \begin{cases} \lambda \beta_j, & \beta_j \leq \lambda \\ \frac{\gamma \lambda \beta_j - 0.5(\beta_j^2 + \lambda^2)}{\gamma - 1}, & \lambda < \beta_j \leq \gamma \lambda \\ \frac{\lambda^2(\gamma^2 - 1)}{2(\gamma - 1)}, & \beta_j \geq \gamma \lambda \end{cases} \quad (4)$$

式中, $\lambda \geq 0, \gamma > 2$ 。

当 β_j 较小时, SCAD 和 Lasso 具有一样的惩罚力度; 随着 β_j 增大至 λ , SCAD 惩罚力度开始逐渐下降; 当 $\beta_j > \gamma \lambda$ 时, 惩罚力度降为 0。SCAD 惩罚保留了 Lasso 惩罚的稀疏性等优点, 并修正了 Lasso 惩罚过度压缩系数的问题。

1.2.2 MCP 惩罚函数

MCP 惩罚函数在 $\beta_j \in [0, \infty)$ 上定义为

$$p_{\lambda, \gamma}(\beta_j) = \begin{cases} \lambda \beta_j - \frac{\beta_j^2}{2\gamma}, & \beta_j \leq \gamma \lambda \\ \frac{1}{2} \gamma \lambda^2, & \beta_j > \gamma \lambda \end{cases} \quad (5)$$

式中, $\lambda \geq 0, \gamma > 1$ 。

MCP 惩罚起初与 Lasso 惩罚有相等的惩罚力度, 然后随着 β_j 的增大, 惩罚力度逐渐减小, 当 $\beta_j > \gamma \lambda$ 时, 惩罚力度降到 0。在高维线性回归中, MCP 惩罚方法是一种几乎无偏且准确的变量选择方法。

1.2.3 Logistic 模型惩罚估计方法

为了解决 Logistic 模型在上述两种惩罚下的估计问题, Breheny 和 Huang^[5] 结合了文献[10]给出的用于求解广义线性模型的迭代加权最小二乘法, 将优化问题转化为线性模型下加权最小二乘的优化问题, 然后采用以下方法进行估计。

设对数似然函数的第 m 次迭代的估计值为 $\hat{\boldsymbol{\beta}}_m$, Logistic 模型在 $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_m$ 处的近似惩罚似然函数为

$$l_p(\boldsymbol{\beta}) \approx \frac{1}{2N} \sum_{i=1}^N (\tilde{y}_{mi} - \mathbf{x}_i^T \boldsymbol{\beta}) w_{mi} (\tilde{y}_{mi} - \mathbf{x}_i^T \boldsymbol{\beta}) + \sum_{j=1}^p p_{\lambda, \gamma}(|\beta_j|) \quad (6)$$

式中, $\tilde{y}_{mi} = \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_m + (\hat{w}_{mi})^{-1} (y_i - \hat{\mu}_{mi})$, $\hat{w}_{mi} = \hat{\mu}_{mi} (1 - \hat{\mu}_{mi})$, $\hat{\mu}_{mi}$ 为 μ_i 在 $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_m$ 时计算得到的值。

记 \mathbf{X}_j 为第 j 个自变量所对应的整体观测值, $\hat{\boldsymbol{\beta}}_{mj}$ 为 $\hat{\boldsymbol{\beta}}_m$ 的第 j 维分量, $\hat{r}_i = (\hat{w}_{mi})^{-1} (y_i - \hat{\mu}_{mi})$, $\hat{\mathbf{r}} = (\hat{r}_1, \dots, \hat{r}_N)^T$, $\mathbf{W} = \text{diag}(\hat{w}_{m1}, \dots, \hat{w}_{mN})$, $\hat{v}_j = \frac{1}{N} \mathbf{X}_j^T \mathbf{W} \mathbf{X}_j$ 。

为了求解式(6)的极小值, 文献[5]给出具体估计步骤如下:

$$(1) \text{ 计算 } z_j = \frac{1}{N} \mathbf{X}_j^T \mathbf{W} \hat{\mathbf{r}} + \hat{v}_j \hat{\boldsymbol{\beta}}_{mj};$$

$$(2) \text{ 更新 } \hat{\boldsymbol{\beta}}_{m+1,j} = \frac{f(z_j, \lambda, \gamma)}{v_j};$$

$$(3) \text{ 更新残差 } \hat{\mathbf{r}};$$

(4) 基于坐标下降法重复步骤(1)~(3)直到收敛。

当惩罚函数为 SCAD 时, $f(z_j, \lambda, \gamma)$ 满足

$$f = f_{\text{SCAD}} = \begin{cases} S(z_j, \lambda), & |z_j| \leq 2\lambda \\ \frac{S(z_j, \lambda\gamma/(\gamma-1))}{1-1/(\gamma-1)}, & 2\lambda < |z_j| \leq \gamma\lambda \\ z_j, & |z_j| > \gamma\lambda \end{cases}$$

当惩罚函数为 MCP 时, $f(z_j, \lambda, \gamma)$ 满足

$$f = f_{\text{MCP}} = \begin{cases} \frac{S(z_j, \lambda)}{1-1/\gamma}, & |z_j| \leq \gamma\lambda \\ z_j, & |z_j| > \gamma\lambda \end{cases}$$

这里,

$$S(z, \lambda) = \begin{cases} z - \lambda, & z > \lambda \\ 0, & -\lambda \leq z \leq \lambda \\ z + \lambda, & z < -\lambda \end{cases}$$

2 海量数据下一般广义线性模型的惩罚估计方法

2.1 一般广义线性模型下的惩罚估计

Logistic 模型仅仅是广义线性模型中的一个特例,在实际问题中,因变量除了服从两点分布外,还可能服从多项分布、泊松分布等,且在建模时,大部分连接函数都是非典则连接。因此有必要进一步研究一般情形下的广义线性模型的估计方法。

在研究一般情形下的广义线性模型的惩罚估计方法时,为了降低非线性计算的复杂度,需要将广义线性模型线性化。根据文献[10]可知,对于广义线性模型对数似然函数,令 $\tilde{y}_{mi} = \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_m + (y_i - \hat{\mu}_{mi}) \times g'(\hat{\mu}_{mi})$, 此时 $\text{Var}(\tilde{y}_{mi}) = V(\hat{\mu}_{mi}) [g'(\hat{\mu}_{mi})]^2$, 进一步令 $\hat{w}_{mi} = \frac{1}{V(\hat{\mu}_{mi}) [g'(\hat{\mu}_{mi})]^2}$, 则有

$$l(\boldsymbol{\beta}) \approx \frac{1}{2N} \sum_{i=1}^N (\tilde{y}_{mi} - \mathbf{x}_i^T \boldsymbol{\beta}) \hat{w}_{mi} (\tilde{y}_{mi} - \mathbf{x}_i^T \boldsymbol{\beta}) \quad (7)$$

因此结合式(6)的惩罚函数和式(7),可以利用与式(6)类似的计算步骤得到一般广义线性模型的惩罚估计。

上述方法在求解过程中需要把所有的观测数据一次性加载到计算机内存当中,当观测数据量 N 很大时,可能会带来计算机内存不足等问题。因此,需要对该算法进行改进,以适应海量数据情形。

2.2 海量数据下一般广义线性模型的惩罚估计

为了解决海量数据下具有典则连接的广义线性模型的变量选择问题,Chen 等^[8]采用分治思想,给出了一种具有典则连接的广义线性模型的变量选择方法,其基本思想是:首先将全部数据划分为 K 个互不相交的子数据集;接着分别求解每块子数据集下的变量选择及参数估计结果;然后通过投票的方式解决不同子数据集下选取的变量可能不同的问题,确定最终选取的变量;最后通过加权方式对 K 个估计结果进行聚合,将其作为最终估计结果。本文延续上述思路,给出了具有一般连接函数的广义线性模型变量选择的估计方法。

设全部观测数据为 $\mathbf{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, 将数据集 \mathbf{D} 分为 K 块小数据集,不失一般性,设每一个子数据集有相同样本数 n , 每块子数据集记为 $\mathbf{D}_k = \{\mathbf{x}_{ki}, y_{ki}\}_{i=1}^n, k=1, \dots, K$ 。

为了求解第 k 块数据集 \mathbf{D}_k 下的压缩统计量 $\hat{\boldsymbol{\beta}}_k$, 记 \mathbf{X}_{kj} 为 \mathbf{D}_k 下第 j 个自变量所对应的观测值, $\hat{\boldsymbol{\beta}}_{mk}$ 为第 m 次迭代的估计值, $\hat{\boldsymbol{\beta}}_{mkj}$ 为 $\hat{\boldsymbol{\beta}}_{mk}$ 的第 j 维分量, $\hat{r}_{ki} = g'(\hat{\mu}_{mki}) (y_{ki} - \hat{\mu}_{mki})$, $\hat{\mathbf{r}}_k = (\hat{r}_{k1}, \dots, \hat{r}_{kn})^T$, $\hat{w}_{mki} = \frac{1}{V(\hat{\mu}_{mki}) [g'(\hat{\mu}_{mki})]^2}$, $\hat{v}_{kj} = \frac{1}{n} \mathbf{X}_{kj}^T \mathbf{W}_k \mathbf{X}_{kj}$, $\mathbf{W}_k = \text{diag}(\hat{w}_{mk1}, \dots, \hat{w}_{mkn})$, 通过 1.2.3 节方法进行估计:

$$(1) \text{ 计算 } z_{kj} = \frac{1}{n} \mathbf{X}_{kj}^T \mathbf{W}_k \hat{\mathbf{r}}_k + \hat{v}_{kj} \hat{\boldsymbol{\beta}}_{mkj};$$

$$(2) \text{ 更新 } \hat{\boldsymbol{\beta}}_{m+1, kj} = \frac{f_k(z_{kj}, \lambda, \gamma)}{v_{kj}};$$

$$(3) \text{ 更新残差 } \hat{\mathbf{r}}_k;$$

(4) 基于坐标下降法重复步骤(1)~(3)直到收敛。这里, $\hat{\mathbf{r}}_k, \hat{w}_{mki}, \hat{v}_{kj}$ 均为 $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_{mk}$ 时代入相应公式计算得到的值。

因为不同的子数据集选取的变量可能不同,所以可先采用投票方式确定选取的变量

$$v_j = \begin{cases} 1, & \sum_{k=1}^K I(\hat{\boldsymbol{\beta}}_{mkj} \neq 0) \geq w \\ 0, & \text{其他} \end{cases} \quad (8)$$

当 $v_j = 1$ 表明第 j 个变量被选中,否则剔除该变量, $I(\cdot)$ 为示性函数, $w \in (0, K]$ 为阈值。设 $A = \{j | v_j = 1, j=1, \dots, p\}$ 为被选中变量集。令 $\hat{\boldsymbol{\beta}}_{Ak}$ 表示由 $\hat{\boldsymbol{\beta}}_k$ 的第 j 维分量组成的子向量,其中 $j \in A$ 。

为了对 K 个数据集上的估计结果 $\hat{\boldsymbol{\beta}}_{Ak}$ 进行聚

合,将似然函数的二阶近似导数 $S_k = \sum_{i=1}^n \mathbf{x}_{ki} \hat{\mathbf{w}}_{ki} \mathbf{x}_{ki}^T$, $k = 1, \dots, K$ 作为权重矩阵进行加权,得到最终的聚合估计结果

$$\hat{\boldsymbol{\beta}}_{ANK} = \left(\sum_{k=1}^K S_{Ak} \right)^{-1} \sum_{k=1}^K S_{Ak} \hat{\boldsymbol{\beta}}_{Ak} \tag{9}$$

式中 $S_{Ak} = \mathbf{A}^T S_k \mathbf{A}$ 。令 $\mathbf{v} = \text{diag}(v_1, \dots, v_p)$, 则 \mathbf{A} 为由 \mathbf{v} 的第 $\{j|j \in A\}$ 列元素组成的子矩阵。

由式(9)可知,在变量选择及参数估计的实现过程中,仅需保留每个子数据集 \mathbf{D}_k 上的压缩变量 $\hat{\boldsymbol{\beta}}_{Ak}$ 、 S_{Ak} 即可得到整块数据集的近似估计。因为每个小数据集上的估计是独立的,所以对计算机性能的依赖大大降低,只需要能保证每个小数据集的估计能正常进行即可。

2.3 基于 Spark 集群的并行实现

当前处理海量数据的主流工具是基于分布式搭建的。分布式计算通过将一个大任务拆分为多个子任务,然后将这些子任务分发给集群中的多个计算机节点进行计算,以此来完成并行化计算。

在本文所提算法的估计过程中,数据压缩是耗时最久、占用计算资源最多的步骤,然而基于分治算法,每个子数据集在进行数据压缩过程中是独立的,该特性使其适合以分布式方式实现。

Spark 和 Hadoop 是目前处理海量数据的主流工具,借助 Hadoop 的存储框架 HDFS 对数据按块进行分布式存储,并采用适合处理迭代式计算的 Spark 计算框架,便能将该算法以并行的方式实现,从而提高计算效率。

并行计算流程如图 1 所示,每个计算节点从 HDFS 中读取不同的数据块,然后以并行的方式对每一个子数据块上数据进行压缩,接着将所有数据块上的结果广播给主节点,让主节点完成聚合步骤,最后输出结果。

3 数值模拟

本节对算法进行数值模拟,以验证算法的有效性。有效性通过估计精度和时间消耗两个角度来衡量,模拟通过普通单机和 Spark 集群两种方式完成。选择常用的两点分布,即 $y_i | \mathbf{x}_i \sim B(1, \mu_i)$, 其中 μ_i 通过以下两种方式来建模。

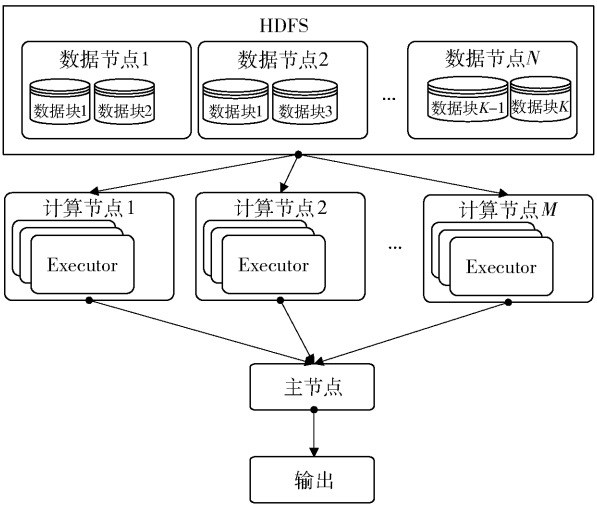


图 1 Spark 并行流程

Fig. 1 Parallel process of Spark

(1)在典则连接下,即为常用的 Logistic 模型

$$\mu(\mathbf{x}_i^T \boldsymbol{\beta}_0) = \frac{\exp(\mathbf{x}_i^T \boldsymbol{\beta}_0)}{1 + \exp(\mathbf{x}_i^T \boldsymbol{\beta}_0)}, i = 1, \dots, N$$

(2)在非典则连接下,选择 Probit 模型

$$\mu(\mathbf{x}_i^T \boldsymbol{\beta}_0) = \Phi(\mathbf{x}_i^T \boldsymbol{\beta}_0), i = 1, \dots, N$$

式中, $\Phi(\cdot)$ 为标准正态分布的累积函数。

模拟时选取的数据量 $N = 10^6$, $\boldsymbol{\beta}_0$ 的维数为 200, 其中非零个数为 24, 自变量独立同分布, 服从标准正态分布。

3.1 单机计算

本次实验在 Ubuntu 16.0.4 操作系统下,使用 Python 3.7.0 完成,计算机配置为 CPU 3.40 GHz、内存 8 G。所有实验重复 50 次取平均值作为最终结果。

为了检验估计效果,重新随机生成了 100 万个样本点,并将这些样本下的模型分类准确率作为估计效果的评价指标,结果如表 1 所示。在所有的实

表 1 分类准确率与分块数的关系

Table 1 Relationship between classification accuracy and number of blocks

K	准确率			
	SCAD ^(a)	MCP ^(a)	SCAD ^(b)	MCP ^(b)
10	0.933 2	0.933 5	0.959 6	0.959 9
20	0.933 0	0.933 2	0.959 7	0.959 7
50	0.932 9	0.928 0	0.959 3	0.959 4
100	0.932 8	0.932 8	0.959 0	0.959 4
200	0.932 8	0.932 6	0.958 6	0.958 9

a—Logistic;b—Probit.

验中,变量选取结果均是正确的非零变量。由表 1 可以看出,Logistic 模型的分类准确率可达 0.928 以上,Probit 模型的分类准确率可达 0.958 以上,并且经过分块后,两个模型的分类准确率下降基本控制在 0.1% 内,这说明该估计的精确度较高且稳定。

表 2 给出了单机环境下参数估计消耗的时间与分块数的关系。由表 2 可知,随着分块数的增加,两个模型的参数估计过程所消耗的时间逐渐减少,这说明了分块方法在一定程度上可以提高模型参数的估计效率。

表 2 单机环境下计算时间与分块数关系
Table 2 Relationship between calculation time and number of blocks in a single machine environment

K	计算时间/s			
	SCAD ^{a)}	MCP ^{a)}	SCAD ^{b)}	MCP ^{b)}
10	604	621	1 719	1 718
20	600	622	1 649	1 658
50	589	604	1 605	1 621
100	503	516	1 553	1 559
200	465	465	1 436	1 454

a—Logistic;b—Probit。

结合表 1 和表 2 的实验结果,分块方法能够在保证分类准确率微小下降的情况下提升模型的参数估计效率。

表 3 和表 4 给出了本文的聚合估计方法与随机抽样方法的估计结果对比。选取的评价指标为估计偏差,其计算公式为:设 $\hat{\beta}$ 为估计结果, β_0 为真实值,则估计偏差定义为 $\text{bias}(\beta) = \frac{\|\beta_0 - \hat{\beta}\|_1}{\|\beta_0\|_1}$,这里 $\|\cdot\|_1$ 为一范数。因模拟所用数据是随机产生

表 3 Logisite 下聚合方法与抽样方法偏差比较
Table 3 Comparison of aggregation method and sampling method with Logistic

K	偏差			
	聚合方法 ^{a)}	抽样方法 ^{a)}	聚合方法 ^{b)}	抽样方法 ^{b)}
10	0.038 7	0.048 1	0.023 1	0.033 0
20	0.039 7	0.053 0	0.024 0	0.036 9
50	0.042 5	0.059 7	0.026 5	0.040 0
100	0.048 3	0.070 8	0.030 9	0.048 8
200	0.060 6	0.077 5	0.040 1	0.058 7

a—SCAD;b—MCP。

表 4 Probit 下聚合方法与抽样方法偏差比较
Table 4 Comparison of aggregation method and sampling method with Probit

K	偏差			
	聚合方法 ^{a)}	抽样方法 ^{a)}	聚合方法 ^{b)}	抽样方法 ^{b)}
10	0.004 7	0.017 9	0.004 7	0.011 2
20	0.005 5	0.023 7	0.005 5	0.015 5
50	0.012 5	0.036 8	0.008 3	0.030 4
100	0.014 0	0.042 0	0.012 5	0.036 3
200	0.021 8	0.058 4	0.022 5	0.056 1

a—SCAD;b—MCP。

的,所以可将分块后的每一块数据集下的估计结果视为一次以 $\frac{1}{K}$ 的比例进行随机抽样得到的估计结果。由表中结果可知,两个模型的结果是一致的:固定分块数 K 的情况下,因为聚合方法使用了全部数据进行估计,所以其估计偏差要比基于随机抽样得到的估计偏差小。该结论表明:在计算机内存有限的情况下,采用聚合的估计方法总要优于随机抽样方法。

抽样方法展示的是以 $\frac{1}{K}$ 比例随机抽样得到的结果。

3.2 Spark 集群计算

在海量数据环境下,实际上数据多数是以分布式的方式存储在多台计算机上。Spark 是目前用于海量数据计算的主流分布式计算框架之一,本文通过 Spark 计算框架来完成模拟。实验在 Ubuntu 16.0.4 系统下,通过 4 台普通计算机组成的 Spark 集群来完成,计算环境为 Python 3.7.0 及 Spark 2.3.2。其中,计算节点的配置为 CPU 3.40 GHz、内存 4 G。

表 5 给出的是 Spark 集群下,模型参数估计消耗的时间与分块数的关系。由表 5 可知,相比于单机环境,采用集群方法可以提高模型参数估计的效率,这也说明了分块方法在集群环境中的可行性。

4 实证分析

选取搜狗实验室(<https://www.sogou.com/labs>)公开的 2012 年 6 月—7 月科技板块和股票板块的新闻数据集建立 Probit 模型,以此来检验本文算法在实际应用中的可行性。其中随机抽取 40 138

表 5 Spark 集群下计算时间与分块数的关系
Table 5 Relationship between calculation time and
number of blocks in a Spark cluster

K	计算时间/s			
	SCAD ^{a)}	MCP ^{a)}	SCAD ^{b)}	MCP ^{b)}
8	300	307	712	715
12	261	266	591	591
28	232	234	552	550
50	227	229	537	547
100	185	188	501	502
200	160	163	491	494

a—Logistic; b—Probit。

条科技新闻和 39 971 条股票新闻用于估计模型参数,将剩余的 9 844 条科技新闻和 4 437 条股票新闻用于测试,将其分类准确度作为模型分类效果评价指标。

整个建模流程为:首先对新闻文本进行分词并过滤掉无用的停用词,然后通过 term frequency - inverse document frequency (TF - IDF) 方法将每个新闻文本转化为 1 000 维的词向量,最后通过本文算法完成 Probit 模型的变量选择。结果表明,采用 MCP 惩罚在测试集上得到的准确率为 0. 955 8,采用 SCAD 惩罚在测试集上得到的准确率为 0. 951 4。由以上结果可知,两种惩罚方法在实证数据集上都有较好的表现,对测试集的新闻分类准确度可达 0. 95 以上,这也进一步证明了本文给出的方法对于解决实际问题可行的。

5 结 束 语

本文研究了海量数据下一般广义线性模型的变量选择估计问题,结合分治思想与坐标下降法,给出了基于 MCP 惩罚和 SCAD 惩罚的估计方法,该方法有效降低了估计时对计算机内存的依赖,克服了计算时内存不足的瓶颈。数值模拟表明该方法能进一步提高计算效率,并通过实证分析证明了本文估计

方法在解决实际问题时的可行性。

参考文献:

[1] NELDER J A, WEDDERBURN R W M. Generalized linear models [J]. Journal of the Royal Statistical Society. Series A (General), 1972, 135(3): 370-384.

[2] TIBSHIRANI R. Regression shrinkage and selection via the Lasso [J]. Journal of the Royal Statistical Society. Series B (Methodological), 1996, 58(1): 267-288.

[3] FAN J Q, LI R Z. Variable selection via nonconvex penalized likelihood and its oracle properties [J]. Journal of the American Statistical Association, 2001, 96(456): 1348-1360.

[4] ZHANG C H. Nearly unbiased variable selection under minimax concave penalty[J]. The Annals of Statistics, 2010, 38(2): 894-942.

[5] BREHENY P, HUANG J. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection[J]. The Annals of Applied Statistics, 2011, 5(1): 232-253.

[6] LIN N, XI R B. Aggregated estimating equation estimation [J]. Statistics and Its Interface, 2011, 4(1): 73-83.

[7] 方方,尹相菊,张强. 海量数据下模型平均的分治算法[J]. 系统科学与数学, 2018, 38(7): 764-776. FANG F, YIN X J, ZHANG Q. Divide and conquer algorithms for model averaging with massive data [J]. Journal of System Science and Mathematical Science, 2018, 38(7): 764-776. (in Chinese)

[8] CHEN X Y, XIE M G. A split-and-conquer approach for analysis of extraordinarily large data [J]. Statistica Sinica, 2014, 24(4): 1655-1684.

[9] WANG C, CHEN M H, SCHIFANO E, et al. Statistical methods and computing for big data [J]. Statistics and Its Interface, 2016, 9(4): 399-414.

[10] MCCULLAGH P, NELDER J A. Generalized linear models [M]. 2nd ed. Berlin: Springer-Science + Business Media, 1989.

A variable selection algorithm for generalized linear models of massive data

CHEN ShaoDong LI ZhiQiang*

(College of Mathematics and Science, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract: We derive a non-convex penalty iterative estimation algorithm for solving the generalized linear model variable selection problem, and then use the divide-and-conquer principle to modify the algorithm so that it can be applied to massive data problems and solve the problem of memory overflow bottlenecks which are common with massive data. Compared with the tools currently used to process massive amounts of data, our algorithm's computation steps utilize distributed parallelism, which greatly improves the calculation speed. In subsequent numerical simulations, the algorithm was numerically calculated in two ways: stand-alone and cluster. The results show that our method effectively solves the data storage problem and is suitable for distributed environments. Finally, this algorithm was used to complete the variable selection of the Probit model and used for classification of news datasets.

Key words: massive data; generalized linear model; variable selection; divide-and-conquer