学校代码： 10255
学　　号： 2202183
中图分类法： O213.9

专业学位硕士学位论文

海量高维数据的分位数回归

学 位 申 请 人：王富雅
指　导　教　师：姜荣副教授
专业学位类别：应用统计
所　在　学　院：理学院
提　交　日　期：2022 年 5 月

University Code: 10255
Student ID: 2202183
CLC Index: O213.9

PROFESSIONAL MASTER DISSERTATION

QUANTILE REGRESSION FOR MASSIVE HIGH-DIMENSIONAL DATA

Author: Fuya Wang

Supervisor: Prof. Rong Jiang

Major: Applied Statistics

College: College of Science

Date of Submit: May, 2022

硕士学位论文答辩委员会名单

姓名	职称	单位	备注
郑明	教授	复旦大学	主席
闫理坦	教授	东华大学	委员
张振中	教授	东华大学	委员
徐琪	教授	东华大学	委员
刘欣	副教授	东华大学	委员
李瑜苗	未评职称（统计人员）	妙盈科技	委员
田琳琳	讲师	东华大学	秘书

答 辩 人：王富雅

答辩地点：

答辩日期：2022 年 05 月 17 日

海量高维数据的分位数回归

摘 要

分位数回归模型因有较强的鲁棒性且可以更全面地反映自变量与响应变量分布之间的关系，在一般规模数据的变量关系研究中被广泛应用。近年来，随研究的数据量与数据维数的增加，学者们开始探索其在海量高维数据中的应用。

分位数回归在海量高维数据中的应用存在以下问题：（1）海量数据的存储与计算超出计算机内存。（2）维数过高导致模型选择的非重要变量增加。本文采用分而治之的方法，将整个数据集拆分为几个小数据集分别进行参数的计算，以减小内存的占用，并在求解的目标函数中增加 L1 正则项，帮助进行变量选择。

同时，在分位数回归的求解中，本文将分位数回归模型目标函数最小化问题转化为误差项服从非对称拉普拉斯分布(ALD)的非线性回归模型的似然函数的最大化问题，从而将非平滑的损失函数转化为可微的二次函数，再通过极大似然的思想进行求解。而后将模型求解转化为含有缺失数据的目标函数求解问题，推导得到 EM 算法求解分位数回归的目标函数。结合分而治之与添加正则项的方法，最终得到求解海量高维数据分位数回归模型的 PQREM 方法，并通过数据模拟与实证检验了该方法的有效性。

关键词：海量高维数据；分位数回归；分而治之；L1 正则项

QUANTILE REGRESSION FOR MASSIVE HIGH-DIMENSIONAL DATA

ABSTRACT

The quantile regression model has strong robustness and can more comprehensively reflect the relationship between the distribution of independent variables and response variables, so it has been widely used in research by various scholars, and its application in massive high-dimensional data also has significance.

The application of quantile regression in massive high-dimensional data has the following problems: (1) The storage and calculation of massive data exceeds the computer memory. (2) Too high dimension leads to the increase of unimportant variables selected by the model. In this paper, we adopt the divide-and-conquer method to divide the entire data set into several small data sets for parameter calculation, then we can reduce the memory usage. We also add the L1 regular term to the objective function to achieve the purpose of variable selection..

At the same time, in the solution of quantile regression, this paper transforms the minimization of the objective function of the quantile regression model into the maximization of the likelihood function of the nonlinear regression model whose error term obeys the asymmetric Laplace distribution (ALD), Thus, the non-smooth loss function is transformed into a smooth convex quadratic function, and then solved by the idea of maximum likelihood. Then, the model solution is transformed into the objective function solution problem with missing data, and the EM algorithm is used to solve the quantile regression model. Finally, the PQREM method for solving the quantile regression model of massive high-level data is obtained, and the effectiveness of the method is tested by data simulation and empirical analysis.

Author：Fuya Wang

Supervised by：Rong Jiang

KEY WORDS: Massive high-dimensional data, quantile regression, divide-and-conquer , L1 regular term

目 录

图录

表录

第 1 章 绪论

1.1 研究背景与意义

1.1.1 海量高维数据计算背景

随着互联网、物联网、传感器网络与人工智能、云计算技术的不断发展，人们产生的用户数据与实验数据量级呈爆炸式增长，大数据逐渐成为驱动科技进步与经济发展的重要动力。

“大数据”的概念最早由著名未来学家阿尔文·托夫勒在其 1980 年发表的著作《第三次浪潮》中提出。我们存储与计算数据的单位逐渐由 MB 与 GB 变为 TB 与 PB，对 PB、EB 甚至 ZB 级别数据的处理将成为未来的趋势。根据全球最大的数据中心 IDC 的调查显示，2025 年全球产生的数据量将达 163 ZB[1]，2019 年中国科学院发布地球大数据共享服务平台，平台存储包括生物生态数据、大气海洋数据，基础地理数据及地面观测数据在内总计 5PB 数据，同时每年将以 3PB 的数据量进行更新。“大数据”的量级已经远远超过单台计算机的内存。

除数据量级更高外，现代研究中涉及到的数据维度也更高，也即影响因变量的自变量维数更高，从几十维到上千维：如证券交易数据、航空航天数据、基因序列数据、语音/图像数据、用户画像数据、用户评价数据等，在分析这些数据的过程中，常面临维数过高带来的计算量指数倍增加，有时甚至很难得到计算结果的问题，即“维度灾难”。

数据产生量的增大与维数的增加，对现在数据的处理与分析能力提出了更高的要求。传统的分析工具一般将数据集放入一台计算机中，在模型中一次性计算，但随数据量级超出单台计算机的内存，传统分析方法逐渐不能胜任处理数据从而得到信息的工作。由此，著名研究机构 Gartner 从对处理能力的要求上定义了大数据：“大数据是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力来适应海量、高增长率和多样化的信息资产。”

在增强数据的处理能力的方向上，学者与研究人员主要在分布式计算框架与数据挖掘算法上进行探索。

目前主流的大数据计算框架有 2004 年 Google 公司研究提出的 MapReduce 分布式计算框架与 2012 年伯克利大学 AMP 实验室发布的 Spark 分布式计算框架，两框架均应用了“分而治之”的思想，即将整个数据集分割成几个小数据集，对每个单独的数据集进行储存与统计推断，最终合并每个小数据集的统计结果得到整个数据集的统计结果。MapReduce 计算框架基于 Hadoop，可并行处理 TB 级数据，其基本工作原理为：将输入的数据进行逻辑切片，每一片对应一个 Map 任务，Map 步以并行的方式处理切片，框架对 Map 步的输出进行排序后发给 Reduce 步，Reduce 步将切片的数据进行汇总、计算。相较于 MapReduce，Spark 框架改进了其每个中间结果都需要从磁盘中读取于存储数据集的计算方式，将数据在第一次访问后存储于内存中，减少了数据读写时间，提高了计算效率，更适用于处理交互式查询、机器学习及图计算等任务。基于 Spark 框架的 API，Spark Streaming 框架扩展了 Spark 处理大规模流式数据的能力，可用于实时流计算，能运行在 100+节点以上，并达到秒级延迟。

然而，目前主流的大数据处理框架集中于解决数据条数海量的问题，对于一些高维数据的处理，由于“维数灾难”，基于目前大数据计算框架中的传统分析模型如逻辑回归、分位数回归等都不再适用。针对高维数据的数据挖掘问题，目前主要的解决方式为机器学习、神经网络中算法的改进。学者与研究者们在高维数据的降维方向提出了一系列经典方法，主要思想为对使用空间变变换法进行降维与增加惩罚项进行变量筛选两类。

Pearson(1901)[2]将主成分分析法(PCA)引入非随机变量，Hotelling(1933)[3]将其扩展至随机变量，其原理是通过正交变换将一组可能存在相关性的变量转换为一组线性不相关的变量，并使新变量尽可能反应原有信息；奇异值矩阵分解(SVD)[4]也常用于机器学习中对矩阵的降维，其原理为将任意 m*n 维矩阵分解为两个酉矩阵与奇异值矩阵相乘，取奇异值矩阵中最大的 k 个的奇异值和对应的左右奇异向量来近似描述矩阵，以达到降维的目的。以上两种方法均属于特征空间变换思想的践行。

变量选择方法也具有较长的历史，一般线性回归中采取的变量选择方法包括 AIC 准则[5]、BIC 准则[6]、RIC 准则[7]，Arthur(1975)[8]为解决传统的最小二乘回归的过拟合问题，提出岭回归，在目标函数中增加了 L2 惩罚项，减小每个特征的权重，以防止具有相关性的变量权重较高带来的过拟合。但以上选择准则在解

释变量的维数增加时，计算代价将大大增加，同时使得参数量增加，模型解释性差。为解决岭回归的以上问题，Tibshirani(1996)[9]提出 LASSO 回归，在目标函数中加入 L1 正则项，使得一些系数变小，甚至还使得一些绝对值较小的系数直接变为 0，以达到重要特征筛选的目的。Efron(2004)[10]提出了最小角回归(LARS)，将目标向量依次分解为若干组特征向量的线性组合，最终使得与所有特征均线性无关的残差向量最小，提供了快速求解 LASSO 回归的方法。

在获取高维数据信息的机器学习方法中，集成学习以其高预测准确率被广泛关注与应用，其主要思想仍是对“分而治之”的应用：把多个学习器通过一定方法进行组合,以达到最终效果的提升。目前主要的集成学习算法有基于 Bagging 的随机森林，基于 Boosting 的 GBDT、AdaBoost 与 XGBoost。除此之外，神经网络算法在高维语义、图像数据的处理中得到广泛关注。但由于集成学习算法与神经网络的决策边界对参数的敏感性，使得其需要复杂的调参过程，且结果在实际应用中具有较低的可解释性。因此，对传统分析模型在海量高维数据中的改进与应用仍有较大研究意义。

1.1.2 分位数回归背景

一般线性回归模型研究解释变量与被解释变量条件期望之间的关系，假设误差项与解释变量相互独立，且误差项服从均值为 0 方差为定值的正态分布，则可用最小二乘法求解预测因子与响应变量条件期望之间的线性关系，其关系通常以如下表达式描述：

$$y_i = X_i^T \beta_0 + \varepsilon_i，\ i = 1，\ 2，\ \ldots，\ n$$

其中，y_i为响应变量，X_i为第i次观测的p维解释变量，β_0为未知的p维回归系数。

然而，在实际问题中，误差项常常不满足以上假设，数据可能呈厚尾、偏态或尖峰分布，此时一般线性回归模型求得的模型将受到较大影响，而解释变量的中位数统计量更能代表其一般水平，且人们对解释变量与被解释变量的其他分位数之间的关系同样感兴趣。为解决上述最小二乘法中的不足，Laplace(1818)[11]提出了最小绝对偏差估计(LAD),研究预测因子对响应变量中位数的影响，Koenker 和 Bassett(1978)[12]提出分位数回归的方法，以研究预测因子对响应变量条件分位数的影响，且对误差项的分布并不做正态分布的假设。在基础线性模型的基础

上，给定分位数τ, $0 < \tau < 1$，假定ε_i的τ分位数为 0，则预测因子对响应变量的影响，即系数β_0的值，可以通过最小化如下分位数损失函数求得：

$$L(\beta) = \frac{1}{n}\sum_{i=1}^{n}\rho_\tau(y_i - X_i^T\beta)$$

其中，$\rho(h) = h\{\tau - I(h < 0)\}$。由于分位数回归的损失函数对高于分位数$y_{i,\tau}$与低于分位数$y_{i,\tau}$的预测值进行了不同的权重的惩罚，且$L(\beta)$的负次梯度方向：

$$M(\beta) = \sum_{i=1}^{n} X_i\{\tau - I(y_i - X_i^T\beta < 0)\}$$

只与残差的符号有关，与其具体数值无关，因此分位数回归相较于一般线性回归模型受异常值影响较小，具有更强的鲁棒性。同时，由于分位数回归可以描述预测因子对响应变量分布的影响，我们将通过模型获得相比一般线性回归更加全面的信息。

由于其稳健性与信息获取的全面性，此后分位数回归广泛应用于社会、经济、金融、医疗等领域的实证研究，如 Boyué 和 Craighhead(2002)[13]将分位数回归用于研究汇率市场中不同汇率之间尾部区域的风险相关性，Whittaker(2005)等[14]将指数加权分位数回归应用于预测超市的日销售情况，苏栝芳和蔡经汉(2010)[15]将分位数回归用于研究影响中国教育回报的因素。

基于海量高维数据的运算要求与分位数的广泛应用，本论文将在前人研究的基础上，依据“分而治之”的思想，改进分位数回归的算法，提高计算效率，使这一经典分析模型适用于海量高维数据的研究。

1.2 国内外研究现状

关于处理海量数据方法的研究：传统的分位数回归模型需要一次性处理整个数据集，而海量高维数据为这一运算方式为计算机的存储与计算带来了负担，在一台计算机上处理海量高维数据显然是不可行的。为了解决单独计算机无法快速求解大数据集问题，Lin 和 Xi 等(2011)[16]将分而治之法应用于线性回归，通过对数据集进行拆分后再聚合计算的估计器，有效地对数据集与计算量进行压缩，减少了计算时间和计算机内存需求， Chen 等(2014)[17]将分而治之法应用于几种计算密集型惩罚回归(惩罚项分别为 L1 正则、SCAD、MCP)，理论证明了其可行性。

Elizabet 等(2015)提[18]供了可在线更新的标准误差公式的分治估计在线性模型(LM)和估计方程(EE)框架，在效率与存储要求上表现出相对于传统算法的优越性，且考虑到了子集设计矩阵中由于罕见事件协变量可能存在的秩不足的问题。Jiang 和 Hu 等(2018)[19]提出了分而治之的 CQR 方法，将整个数据集拆分成若干块，对每一块中的数据应用 CQR 方法，最后通过加权平均将这些回归结果结合起来。模拟数据与实证检验证明，在不影响估计结果的有效性前提下，显著减少了所需的主内存量。Jiang 等(2020)[20]通过分而治之策略，将复合最小二乘法和最小绝对偏差法的稳健方法扩展到海量数据的分析。

然而，分治法有以下缺点：由于他们的理论严重依赖于某些估计量的渐近展开式，因此在每个局部机器上都需要很大的样本量。此外，对于使用惩罚模型的高维回归，Lee 等(2017)[21]注意到，由于不可忽视的偏差大小，对切分数据集计算结果的简单平均效果并不好，为解决这一问题，他们通过平均“去偏差”LASSO 估计量，设计了一种一次性方法来在高维设置中进行分布式稀疏回归，并证明只要数据集没有拆分到太多机器上，该方法以与 LASSO 相同的速率收敛，且可以扩展到广义线性模型。Jordan 等(2019)[22]提出了通信效率代理似然(CSL)程序来解决上述问题，然而，他们的方法要求损失函数是二阶可微的，因此不能应用于非光滑分位数回归损失。因此，Chen 等(2020)[23]通过构造新的响应，将非光滑分位数回归损失转化为光滑分位数回归损失。然而，他们的方法只适用于误差项独立同分布的场景，这是一个有限制性的假设。

关于处理高维数据方法的研究：当数据集的维数过高，除会导致计算负担外，还会影响计算参数的准确性，尤其是当样本量小于样本维数的时候。因此，根据 Chen 等(2020)[23]，可选择在目标函数中加入 L1 正则项进行变量选择。对于带 L1 正则项的损失函数最小化的求解方法， Schmidt.M(2010)[24]提出了次梯度缩放投影算法，将求解非平滑目标函数最小值的问题转化为求解目标函数的次梯度缩放投影问题，并通过理论与模拟数据证明了该算法的收敛性与可行性。Solntsev 等(2015)[25]提出了主动集方法，应用于带 L1 正则项目标函数的求解。

关于分位数回归模型求解的研究：分位数回归模型的损失函数为不可微函数，缺乏强凸性，无法采用一般的最小二乘与极大似然方法求解目标函数。对于此，Yu 等(2001)[26]注意到加入惩罚项前，分位数回归的目标函数最小化，等价于非对称拉普拉斯分布(ALD)的随机误差非线性回归模型的似然函数的最大化，Kozumi

等(2011)[27]在拟合基于 ALD 的位置-尺度混合表示的分位数回归模型过程中证明，ALD 可以表示为正态-方差-均值混合指数分布。因此，我们可以将标准 QR 损失函数转化为可微的二次损失函数。此时虽然确定误差项分布形式，但其具体分布仍然未知，因此可将对目标函数的求解看作对不完全数据的估计。

在针对不完全数据的最大似然估计的研究中，A.P Demsper 等(1977)[28]提出了 EM 算法，并从理论上证明了算法的似然性和收敛性的单调性。Karlis Dimitris(2002)[29]将 EM 算法应用于解决正态-逆高斯分布的直接求解参数的最大似然估计的复杂问题，克服了使用常规方法时出现的数值困难。Tian 等(2014)[30]将 EM 算法应用于求解分位数回归问题，获得了未知参数向量 β 估计量的封闭形式。在将线性回归模型表示为误差项服从指数混合分布的基础上，Zhou 等(2014)[31]分别开发了用于计算线性和非线性分位数回归模型的 EM 算法和广义 EM 算法，且证明了所提出的 EM 算法与分位数回归的 MM(Majorization-Minimization)算法的更新公式相同。杨雪梅(2019)[32]使用 MM 算法求解分位数回归，并将其与内点法进行了对比，数值研究证明了 MM 算法的稳定性。

1.3 本文结构安排

第一章介绍了本文所作研究的背景，梳理了国内外研究现状，并对研究内容以及章节安排进行说明。

第二章为相关理论介绍。从求解分位数回归模型开始，首先介绍了分位数模型的相关定义与优点，后重点介绍了求解分位数回归的单纯形法与 EM 算法，最后介绍了降维使用的 LASSO 方法。

第三章为基于 EM 算法的海量高维数据分位数回归模型。首先将分位数回归的损失函数最小化问题转化为最大化误差项服从非对称拉普拉斯分布(ALD)的似然函数问题，推导得到 EM 算法求解分位数回归的目标函数。而后为对高维数据进行分析，在目标函数中增加 L1 正则项，达到变量选择的目的，提出了 PQREM 方法求解高维数据分位数回归。最后为将该方法应用于海量数据中，采用分而治之的方法，将完整数据集划分为几个小数据集，分别计算目标函数中的参数，最终得到适用于海量高维数据的基于 EM 算法的分位数回归模型求解方法(PQREM)。

第四章为模拟实验部分。首先在模拟生成的一般规模数据集（200 条数据）中检验 PQREM 方法的拟合效果，将 R 语言中自带的 QR-LASSO 程序(PQR)与 PQREM 方法的计算结果进行对比，验证了 PQREM 方法预测的有效性。其次在模拟生成的海量规模数据集（20 万条数据）中检验 PQREM 方法的拟合效果，对比 PQREM 与 Chen 等(2020)年提出的 PQRC 方法的计算结果，验证了 PQREM 预测的有效性。最后分别使用 PQR、PQRC、PQREM 方法对真实的音乐特征与发行年份数据集进行分位数回归，完成 PQREM 方法的实证检验，验证其回归结果的有效性、变量选择的有效性与划分小数据集带来的计算量的减小。

第五章为总结与展望部分。总结论文中使用的方法与结论，分析论文的优点与不足之处，对后续的研究进行展望。论文涉及的代码见附录。

第 2 章 相关理论介绍

2.1 分位数回归

2.1.1 分位数回归模型相关定义

分位数回归模型是一般线性回归模型的延伸，一般线性回归模型研究自变量与因变量的条件均值间的关系，分位数回归研究自变量与因变量的条件分位数之间线性关系的模型。设随机变量X的分布函数为：

$$F(x) = P(X \le x)\,. \tag{2-1}$$

则$F(x)$的逆为：

$$F_X^{-1}(\tau) = inf\{x \in R: F(x) \ge \tau\}(0 < \tau < 1). \tag{2-2}$$

随机变量X的τ分位数可定义为：

$$Q_\tau(X) = \ arginf\{x \in R: F(x) \ge \tau\}\,(0 < \tau < 1) = F_X^{-1}(\tau). \tag{2-3}$$

设有随机变量 (X, Y)，其中Y在给定$X = x$条件下累计分布函数为$F_{Y|X=x}(y|x)$，则将随机变量Y在$X = x$条件下的τ分位数定义为：

$$Q_\tau(Y|X = x) = arginf\{y \in R: F(y|x) \ge \tau\}(0 < \tau < 1). \tag{2-4}$$

分位数回归即研究变量X与Y的分位数$Q_\tau(Y|X = x)$之间的关系，设有样本序列$\{(X_i, Y_i), i = 1, \dots, n\}$，一般分位数回归模型可表示为：

$$y_i = x_i^{\intercal}\beta_{0,\tau} + \varepsilon_i,\ i = 1,2, \dots, n \tag{2-5}$$

其中，$y_i = Q_\tau(Y|X = x_i)$为响应变量，x_i为p维向量第i次的观测值，$\beta_{0,\tau}$为未知p维向量，τ表示τ分位数，$\varepsilon_i\{i = 1, \dots, n\}$为独立同分布的序列，分布情况未知，其$\tau$分位数为 0。

一般分位数回归的求解即为对未知系数向量β_0的求解（此处β_0即为$\beta_{0,\tau}$），基本思想与一般线性回归求解相似，一般线性回归求解遵循残差平方和最小化的思想，一般分位数回归遵循非对称的绝对值残差最小化思想。

一般线性回归的损失函数可表示为：

$$min \sum_{i=1}^{n} (y_i - x_i^{\intercal}\beta)^2. \tag{2-6}$$

可用最小二乘法求解。

分位数回归的损失函数可表示为：

$$min\sum_{i=1}^{n}\rho_\tau(y_i-x_i^\intercal\beta). \tag{2-7}$$

其中，$\rho_\tau(u)=\tau u-uI(u<0)$，$I(\cdot)$为示性函数。因此，分位数回归损失函数为：

$$Q(\beta)=\min\left(\sum_{i:y_i\geq x_i^\intercal\beta}\tau\left|y_i-x_i^\intercal\beta\right|+\sum_{i:y_i<x_i^\intercal\beta}(1-\tau)\left|y_i-x_i^\intercal\beta\right|\right). \tag{2-8}$$

参数β的估计值即为：

$$\hat{\beta}=arg\min\left(\sum_{i:y_i\geq x_i^\intercal\beta}\tau\left|y_i-x_i^\intercal\beta\right|+\sum_{i:y_i<x_i^\intercal\beta}(1-\tau)\left|y_i-x_i^\intercal\beta\right|\right). \tag{2-9}$$

一般线性回归与一般分位数回归损失函数图示如下：

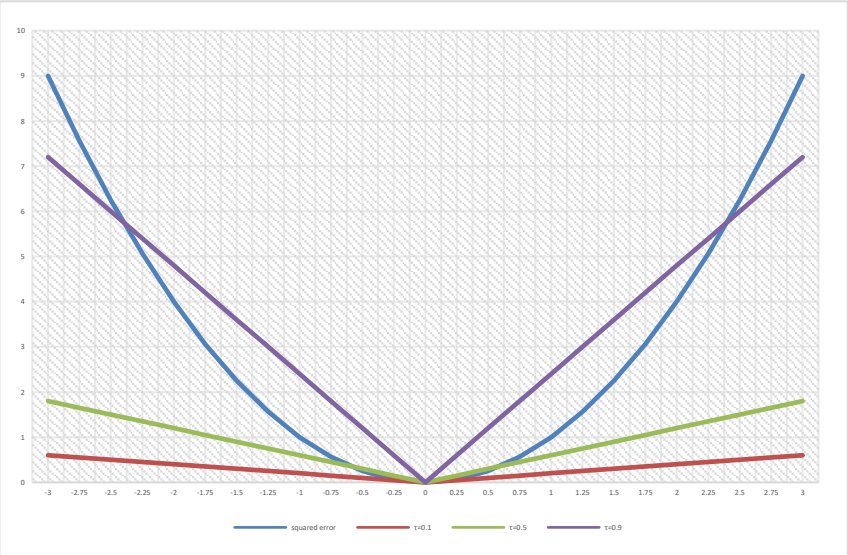

图 2-1 一般线性回归与分位数回归损失函数

Fig2-1 Loss functions of general leaner regression and quantile regression

由图可看出，一般线性回归的损失函数可微，一般分位数回归的损失函数在$y_i-x_i^\intercal\beta$处不可微，因此无法通过最小二乘法求得。

2.1.2 分位数回归模型优点

相对于一般线性回归，分位数回归具有如下优点：

1. 一般线性回归假设随机误差项服从正态分布，分位数回归对误差项的分布不做假设，因此适用范围更广，模型结果更加稳健，尤其当随机误项分布呈厚尾时。

2. 一般线性回归计算响应变量的条件均值，容易受样本点中异常值的影响，分位数回归计算响应变量的条件分位数，对异常值的抵抗性较强。

3. 一般线性回归仅描述自变量对响应变量均值的影响，分位数回归可以描述自变量对响应变量全部分位数的影响，可以更清晰地反映响应变量的各个分布情况。

下图为对家庭收入与食物支出真实数据的一般线性回归与分位数回归的结果对比图，其中蓝色实线代表 0.5 分位回归的回归曲线，红色虚线代表使用最小二乘法拟合的一般线性回归曲线，几条灰色的曲线，分别代表 0.05, 0.1, 0.25, 0.75, 0.9, 0.95 分位回归曲线。可以明显看到，受图 2-2 中红色框中异常值的影响，0.5 分位回归相对于一般线性回归与样本的拟合优度更高，且通过多条分位数回归曲线，可以得到样本的更全面的分布情况。

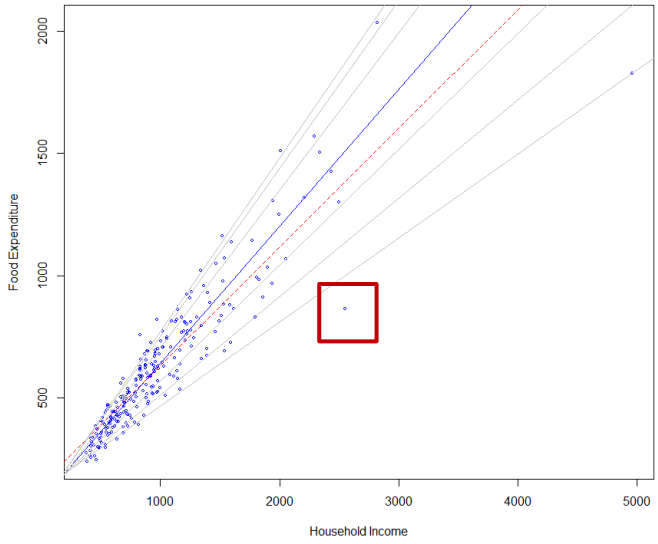

图 2-2 一般线性回归与分位数回归结果对比

Fig2-2 Comparison of general linear regression and quantile regression results

2.1.3 分位数回归模型求解

分位数回归一般采用线性规划法(LP)对回归系数进行求解，常见的方法有 Koenker(1987)[33]优化的单纯性法、Karmarkar(1984)[34]提出的内点法与 Madson 和 Nislsen(1993)[35]提出的平滑法。Chen(2004)[36]结合以上三种方法提出自适应

法，可以根据数据集的大小及特征自动选择求解方法。目前 R 软件中的分位数回归方法默认为单纯性法，因此下文将主要对单纯形法原理进行介绍。

单纯形即为在p维空间R^p中，具有p+1 个顶点的凸多面体。单纯形搜索法的基本思想为，首先确定初始单纯形，计算每个顶点的函数值，并通过排序得到最大值与最小值点，此后通过反射、扩展、压缩等方法求出更好的点代替原来的顶点，形成新的单纯形，从而逼近真实的最小值点。[37]具体步骤如下：

1. 设损失函数为$Q(\beta)$，在p维空间取p+1 个顶点，分别记为$\beta_1,\dots,\beta_{p+1}$，形成初始单纯形，计算相应的损失函数值并排序，不妨设：

$$Q(\beta_1) < Q(\beta_2) < \cdots < Q(\beta_{p+1}). \quad (2-10)$$

2. 得到损失函数最大值点$Q(\beta_{p+1})$，计算除最大值点外其余点的重心$\bar{\beta} = \frac{1}{p}\sum_{i=1}^{p}\beta_i$，连接最大值点与重心点，进行反射，得到反射点$\beta_{p+2}$：

$$\beta_{p+2} = \bar{\beta} + \gamma_1(\bar{\beta} - \beta_{p+1}). \quad (2-11)$$

其中，γ_1为反射系数，一般取$\gamma_1 = 1$。

3. 比较$Q(\beta_{p+2})$与$Q(\beta_1), Q(\beta_p)$的大小，分 3 种情况构造新的单纯形：

1）若$Q(\beta_{p+2}) < Q(\beta_1)$，则方向$d = \beta_{p+2} - \bar{\beta}$有利于损失函数的减小，应继续扩展，有

$$\beta_{p+3} = \bar{\beta} + \gamma_2(\beta_{p+2} - \bar{\beta}). \quad (2-12)$$

其中，$\gamma_2 > 1$为扩展系数，可取$\gamma_2 = 2$，若$Q(\beta_{p+3}) < Q(\beta_{p+2})$，则用$Q(\beta_{p+3})$代替 $Q(\beta_{p+1})$，若$Q(\beta_{p+3}) > Q(\beta_{p+2})$，则用$Q(\beta_{p+2})$代替 $Q(\beta_{p+1})$，以此得到新的单纯形；

2）若$Q(\beta_1) < Q(\beta_{p+2}) < Q(\beta_p)$，则用$Q(\beta_{p+2})$代替 $Q(\beta_{p+1})$，得到新的单纯形；

3）若$Q(\beta_{p+2}) > Q(\beta_p)$，则需进行压缩，令$Q(\beta_{p+4}) = \min\{Q(\beta_{p+1}), Q(\beta_{p+2})\}$，$\beta_{p+4} \in \{\beta_{p+1}, \beta_{p+2}\}$，进行压缩得到：

$$\beta_{p+5} = \bar{\beta} + \gamma_3(\beta_{p+4} - \bar{\beta}). \quad (2-13)$$

其中$\gamma_3 \in (0,1)$，为压缩系数，不妨取$\gamma_3 = 0.5$，若$Q(\beta_{p+5}) < Q(\beta_{p+4})$，则$Q(\beta_{p+5})$代替$Q(\beta_{p+1})$，得到新的单纯形，若$Q(\beta_{p+5}) < Q(\beta_{p+4})$，则应继续压缩，最小值点$\beta_{p+1}$不动，其余点均应向最小值点挪动一半的距离：

$$\beta'_{i+1} = \beta_{i+1} + \frac{1}{2}(\beta_1 - \beta_{i+1}), i = 1,2,\dots,p. \quad (2-14)$$

用β'_{i+1}代替原来的β_{i+1}得到新的单纯形。

4. 重复以上三个步骤，则单纯形的内部范围逐渐缩小，不断逼近损失函数的最小值，当各顶点满足收敛准则：

$$\{\frac{1}{p+1}\sum_{i=1}^{p+1}[Q(\beta_i)-Q(\bar{\beta})]^2\}^{\frac{1}{2}}<\varepsilon. \qquad (2-15)$$

则停止迭代，取最后一次迭代中$Q(\beta_i)$中的最小值即为损失函数的近似值。

单纯形法估计的参数具有很好的稳定性，不易受异常样本点的影响，但由于每次迭代均需要计算多次损失函数的值进行比较，当数据量较大时，会导致计算效率较低。随后续学者的研究，求解方法的效率与准确性与本节初始提到的三种方法中对比得到了显著提升。Yu 和 Moyeed(2001)[26]提出了一种贝叶斯方法，可以较大幅度提高求解效率，但模型中引入较多参数，对先验精确度的要求较高，基于该研究，Tian 等(2014)[30]提出将 EM 算法应用于分位数回归求解，以降低对计算过程的要求，使得求解更加简便。因此本文在后续应用于海量高维数据的算法中，使用 EM 算法进行求解。

2.2 EM 算法

EM 算法由 Dempster 等于 1977 年提出，是一种用于解决含有缺失数据的复杂统计问题的迭代优化策略[28]。其计算方法中每一次迭代一般分为两步：期望步（E 步）与极大步（M 步），因此被称为 EM 算法。其基本思想为：首先根据已经给出的观测数据，对模型中的未知参数进行估计，得到初始估计值；此后根据上一步估计出的参数初始值对缺失数据进行估计，再根据估计得到的缺失数据值与已有的观测数据，应用极大似然思想使得观测数据出现的概率最大化，从而得到新的参数估计值；对上述步骤重复迭代，直至收敛。EM 算法的原理及推导主要涉及极大似然估计与 Jensen 不等式两部分理论基础。

2.2.1 极大似然估计

极大似然估计为给定样本观测值的情况下，估计模型未知参数的常用方法，其基本原理为：假设模型参数为某一定值，计算所有样本观测值出现对应情况的概率，样本观测值出现的概率最大时的模型参数即为未知参数的极大似然估计值。

假定参数为一定值的情况下，求样本观测值发生概率的表达式即为似然函数，假设随机变量X服从包含未知参数θ的某一分布，对其进行随机抽样得到 N 个样本，记为样本集D，$D=\{x_1,x_2,\dots,x_N\}$，其中样本集中的每个样本独立同分布，当θ取一定值时，$X=x$的概率记为$P(x|\theta)$。则联合密度$P(D|\theta)$即为似然函数：

$$L(\theta)=P(D|\theta)=P(x_1,x_2,\dots,x_N|\theta)=\prod_{i=1}^{N}P(x_i|\theta). \quad (2-16)$$

若$\theta=\hat{\theta}$时使得似然函数$L(\theta)$取得最大值，则$\hat{\theta}$即为参数θ的极大似然估计值。因此若要求得θ的估计值，表达式如下：

$$\hat{\theta}=argmax\,L(\theta)=argmax\prod_{i=1}^{N}P(x_i|\theta). \quad (2-17)$$

实际情况中，为方便求解，定义了对数似然函数：

$$H(\theta)=\ln L(\theta). \quad (2-18)$$

由于对数似然函数与似然函数的单调性相同，

$$\hat{\theta}=argmax\,H(\theta)=argmax\,\ln L(\theta)=argmax\prod_{i=1}^{N}ln\,P(x_i|\theta). \quad (2-19)$$

当未知参数只有一个，即θ为标量时，若似然函数满足连续、可微的条件，则求解极大似然估计可通过求导的方式找到对数似然函数导数为 0 的值，即为极大似然估计值。因此极大似然估计值可通过求解以下方程得到：

$$\frac{dL(\theta)}{d\theta}=0 \text{ 或 } \frac{d\ln L(\theta)}{d\theta}=0. \quad (2-20)$$

当未知参数有多个，即θ为向量时，则θ可表示为具有s个分量的未知向量：

$$\theta=[\theta_1,\theta_2,\dots,\theta_s]^{\mathrm{T}}. \quad (2-21)$$

在各分量方向上对θ求偏导，得到梯度算子：

$$\nabla_\theta=[\frac{\partial L(\theta)}{\partial\theta_1},\frac{\partial L(\theta)}{\partial\theta_2},\dots,\frac{\partial L(\theta)}{\partial\theta_s}]^{\mathrm{T}}. \quad (2-22)$$

若似然函数满足在每个方向上连续可微的条件，则θ的极大似然估计值为以下方程的解：

$$\nabla_\theta H(\theta)=\nabla_\theta\,ln\,L(\theta)=\sum_{i=1}^{N}\nabla_\theta\,ln\,P(x_i|\theta)=0. \quad (2-23)$$

当样本量足够多时，方程的解接近于真实值。

2.2.2 Jensen 不等式

定义 1（凸函数）：设函数$f(x)$定义域为实数，对于所有的x均有$f(x)$的二阶导数大于零，那么$f(x)$为凸函数。

定义 2（Jensen 不等式）：假设函数$f(x)$为凸函数，X为随机变量，则有$E[f(X)] \geq f(E[X])$，当且仅当X为常量时，等号取到。其中，$E[X]$表示X的数学期望。

Jensen 不等式证明如下：

假设函数$f(x)$在$x = x_0$处的切线方程为：

$$l(x) = ax + b. \tag{2-24}$$

其中，$a = f'(x_0), b = f(x_0) - ax_0$。

由凸函数性质可得：

$$f(x) \geq f(x_0) + f'(x_0)(x - x_0) = ax + b. \tag{2-25}$$

对 2-25 式两边同时求期望可得：

$$E[f(x)] \geq E(ax + b) = aE[x] + b. \tag{2-26}$$

取$x_0 = E[x]$，相应地$a = f'(x_0), b = f(x_0) - ax_0$，带入式 2-27 则有：

$$E[f(x)] \geq aE[x] + b = ax_o + b = f(x_0) = f(E[x]). \tag{2-27}$$

证毕。

Jensen 不等式举例示意图如下：

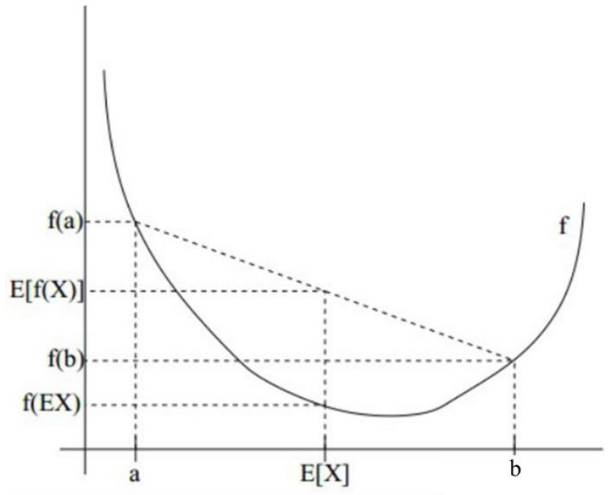

图 2-3 Jensen 不等式举例
Fig2-3 Example of Jensen's inequality

上图中$f(x)$凸函数，X为随机变量，a, b为X的最大值与最小值，$E[X]$为X的期望，由图可明显看出$E[f(X)] \geq f(E[X])$成立。

2.2.3 EM 算法推导流程

设随机变量X服从含有未知参数θ的分布，对 n 个样本数据$x = x_1, x_2, \dots, x_n$，要求未知参数θ，可通过将模型对数似然函数极大化求解：

$$\hat{\theta} = argmax \sum_{i=1}^{n} \log p(x_i; \theta). \quad (2-28)$$

若观测到的样本数据中含有为观测到的隐含数据$z = (z_1, z_2, \dots, z_n)$，即样本分布未知，那么我们将模型分布极大化的对数似然函数如下：

$$\hat{\theta} = argmax \sum_{i=1}^{n} \log p(x_i; \theta) = argmax \sum_{i=1}^{n} log \sum_{z_i} p(x_i, z_i; \theta). \quad (2-29)$$

由于上式根据样本数据x_i的边缘概率计算得来，无法直接求解θ。因此可先引入未知的新分布$Q_i(z_i)$，再使用 Jensen 不等式对上式进行放缩得到：

$$argmax \sum_{i=1}^{n} log \sum_{z_i} p(x_i, z_i; \theta) = argmax \sum_{i=1}^{n} log \sum_{z_i} Q_i(z_i) \frac{p(x_i, z_i; \theta)}{Q_i(z_i)}. \quad (2-30)$$

$$\geq argmax \sum_{i=1}^{n} \sum_{z_i} Q_i(z_i) log \frac{p(x_i, z_i; \theta)}{Q_i(z_i)}. \quad (2-31)$$

其中，$\sum_{z_i} Q_i(z_i) \frac{p(x_i,z_i;\theta)}{Q_i(z_i)}$为$\frac{p(x_i,z_i;\theta)}{Q_i(z_i)}$的期望，且$\log(x)$为凸函数，根据 Jensen 不等式可由式 2-30 得到式 2-31，由此得到$\sum_{i=1}^{n} \log p(x_i; \theta)$的下界。当$\theta$确定时，$\sum_{i=1}^{n} \log p(x_i; \theta)$的值取决于$Q_i(z_i)$和$p(x_i, z_i; \theta)$，可通过调整$Q_i(z_i)$和$p(x_i, z_i; \theta)$使下界不断上升，逼近$\sum_{i=1}^{n} \log p(x_i; \theta)$的值，即调整至式 2-31 中等号取到时，下界值与真实值相等。

由 Jensen 不等式取等条件可得，若式 2-31 等号取到，则有

$$\frac{p(x_i, z_i; \theta)}{Q_i(z_i)} = c, c为常数. \quad (2-32)$$

由于$Q_i(z_i)$为概率分布，所以$\sum_{z_i} Q_i(z_i) = 1$，由此，$\sum_{z_i} p(x_i, z_i; \theta) = c$。因此有

$$Q_i(z_i) = \frac{p(x_i, z_i; \theta)}{\sum_{z_i} p(x_i, z_i; \theta)} = \frac{p(x_i, z_i; \theta)}{p(x_i; \theta)} = p(z_i | x_i; \theta). \quad (2-33)$$

至此，解决了$Q_i(z_i)$选择的问题，可计算z_i, θ联合分布的条件概率期望，即为 EM 算法的 E 步。

同时，当等号取到，即$Q_i(z_i)$为条件概率时，最大化对数似然函数的下界即为最大化对数似然函数，因此对θ的求解即为：

$$\hat{\theta} = argmax \sum_{i=1}^{n} \sum_{z_i} Q_i(z_i) log \frac{p(x_i, z_i; \theta)}{Q_i(z_i)}. \quad (2-34)$$

对上式 2-34 进行最大化求解，即为 EM 算法的 M 步。

EM 算法流程总结如下：

1. 输入：观察到的样本数据$x = (x_1, x_2, \dots, x_n)$，联合分布$p(x, z; \theta)$，条件分布$p(z|x; \theta)$与最大迭代次数 J；
2. 对模型参数θ进行随机初始化赋值得到θ_0；
3. $j = 1,2, \dots, J$，开始进行 EM 算法迭代：

   E步：计算联合分布的条件概率期望：

$$Q_i(z_i) = p(z_i|x_i; \theta_j); \quad (2-33)$$

$$l(\theta, \theta_j) = \sum_{i=1}^{n} \sum_{z_i} Q_i(z_i) log \frac{p(x_i, z_i; \theta)}{Q_i(z_i)}. \quad (2-35)$$

   M 步：极大化$l(\theta, \theta_j)$，得到θ_{j+1}：

$$\theta_{j+1} = argmax l(\theta, \theta_j). \quad (2-36)$$

   直到θ_{j+1}收敛或者达到最大迭代次数，停止迭代；
4. 输出：模型参数θ。

2.3 LASSO 方法

2.3.1 LASSO 方法降维原理

在回归模型中，若特征之间存在高度的相关关系，那么直接最小化目标函数求解可能导致某些负相关的特征均被赋予较高的权重，这样即使加权平均后带来的权重依旧很小，但由于权重较大，将导致过拟合。正则化可以降低模型的复杂度，减少模型过拟合的风险。LASSO 方法即为正则化方法中的一种，即在目标函数中增加 L1 正则项，常见的正则化方法还有在目标函数中增加 L2 正则项。若模型求解的未知参数为β，则 L1 正则项可表示为：

$$\|\beta\|_1 = \sum_{i=1}^{n} |\beta_i|.$$

L2 正则项可表示为下式。

$$\|\beta\|_2^2 = \sum_{i=1}^{n} |\beta_i|^2.$$

L1 正则项主要用于稀疏模型中的特征选择。特征数量较多，而真正对模型有贡献的特征数量较少的模型即为系数模型。考虑一般优化问题：

$$\hat{\beta} = arg\min_{\beta \epsilon R^p} \frac{1}{n}(y - x^{\intercal}\beta)^2 + \lambda\|\beta\|_1. \quad (2-37)$$

其中，λ为正则化参数。由于 L1 正则项为绝对值项，求解模型的目标函数不可微，无法通过求导直接求解。因此可将优化问题等价于约束条件下无 L1 正则项模型求解：

$$\hat{\beta} = arg\min_{\beta \epsilon R^p} \frac{1}{n}(y - x^{\intercal}\beta)^2 \quad s.t.\ \lambda\|\beta\|_1 < t. \quad (2-38)$$

根据线性规划知识，一般模型$\hat{\beta} = arg\min_{\beta \epsilon R^p} \frac{1}{n}(y - x^{\intercal}\beta)^2$与约束条件$\lambda\|\beta\|_1 < t$均有可行域，两个可行域的切点即为 L1 正则化模型的解。为更直观地理解其几何意义，在二维模型中，我们画图表示如下：

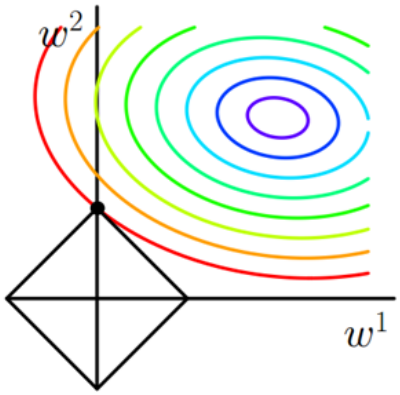

图 2-4 L1 正则项求解的几何示意图
Fig2-4 The geometric diagram of L1 regular term solution

图中正方形部分即为 L1 约束条件的可行域，等高线部分即为$\hat{\beta} = arg\min_{\beta \epsilon R^p} \frac{1}{n}(y - x^{\intercal}\beta)^2$的可行域，两个可行域的切点有较大的概率落在坐标轴上，以使得其中一个特征的权重为 0，从而达到特征选择的目的，高维数据中同理。同时，当正则化参数λ越大时，正方形越小，即可行域越小，相应的其顶点其距离远点更接近，最终使每个特征的权重越小，从而一定程度上防止过拟合。

L2 正则项主要用于特征的权值衰减，使得相关特征之间的权重分布均匀。考虑一般优化问题：

$$J(X, y; \beta) = L(X, y; \beta) + \frac{1}{2}\lambda\|\beta\|_2^2. \quad (2-39)$$

采用梯度下降法求解时，可得：

$$\beta^{t+1} = \beta^t - \eta\nabla J(X, y; \beta)$$

$$= \beta^t - \eta(\nabla L(X, y; \beta) + \lambda\beta^t)$$
$$= (1 - \eta\lambda)\beta^t - \eta\nabla L(X, y; \beta) \quad (2-40)$$

由此，第 t+1 步的参数在第 t 步的参数之前乘了$1 - \eta\lambda$，多次迭代下可使得每个特征的权重趋近于 0，从而达到均衡权重分布，防止过拟合的效果。同样，可将 L2 正则化模型的求解问题转化为带约束条件的求解问题：

$$\hat{\beta} = arg\min_{\beta \epsilon R^p} \frac{1}{n}(y - x^{\mathrm{T}}\beta)^2 \quad s.t.\ \lambda\|\beta\|_2^2 < t. \quad (2-41)$$

在二维模型中，我们可画图表示如下：

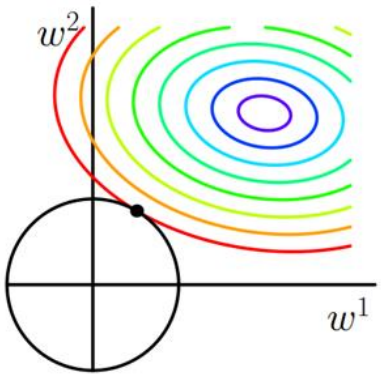

图 2-5 L2 正则项求解的几何示意图

Fig2-5 The geometric diagram of L2 regular term solution

图中黑色圆形部分为 L2 约束条件的可行域，彩色等高线部分为$\hat{\beta} = arg\min_{\beta \epsilon R^p} \frac{1}{n}(y - x^{\mathrm{T}}\beta)^2$的可行域，两个可行域的切线一般落在象限区域内而非坐标轴上，不容易使某一特征的权重为 0，达不到特征选择的效果。

因此在高维数据中，为达到特征选择的目的，选择 LASSO 方法正则化较合适。

2.3.2 求解 LASSO 的方法

由于 LASSO 方法中目标函数加入的正则化项不可微，因此传统的牛顿法与梯度下降法等需要求导的算法不再适用。

Efron (2004)[10]提出了最小角回归算法(LARS)求解 LASSO，该方法在找到与因变量相关性最高的自变量后，在该方向上前进找到与之残差相关性相同的方向选择下一个变量，此后不断根据残差相关性相同的规制进行变量选择，直至模型的残差和足够小。该方法计算步骤较少，有效降低了计算的复杂度，但对样本噪声较敏感。

Wu 和 Lange(2008)[38]提出了坐标轴下降法(CD)求解 LASSO，该方法在 p 维未知参数的 p 个坐标轴上进行迭代，当所有坐标轴上都达到收敛时，即求得目标函数的最优解。该方法的优点是简单、计算速度快。随着处理模型数据量的增大，后来学者提出了较多 LASSO 求解的改进方法。Schmidt.M(2010)提出了次梯度缩放投影算法(PSSsp)，该方法使用次梯度替代近似梯度，并使用了拟牛顿算法(L-BFGS)以优化空间内存[31]，适合求解海量数据的 LASSO 模型。本文对 LASSO 方法的求解即使用 PSSsp 算法。

2.4 本章小结

本章主要介绍了分位数回归的求解方法与高维数据降维方法的相关理论，为第三章的模型建立与求解打下理论基础。首先从一般分位数回归模型入手，介绍了模型的相关定义与优点，阐明了在海量高维数据中研究分位数回归模型的意义。在分位数回归的求解方法中，主要介绍了 R 语言自带程序包 QR 中使用的单纯形法与 Tian 等(2014)[28]使用的 EM 算法，为下文两方法的对比进行了理论铺垫。最后，由于对高维数据的分位数回归需进行变量选择，本章还介绍了常用的变量选择方法 LASSO 方法的降维原理与其求解方法。在下一章中，本文将给出求解海量高维数据分位数回归模型的方法。

第 3 章 基于 EM 算法的海量高维数据的分位数回归

3.1 EM 算法求解分位数回归模型

在分位数回归模型式 2-5 中，Yu 和 Moyeed(2001)[26]指出，对于目标函数$\sum_{i=1}^{n}\rho_\tau(y_i-x_i^\intercal\beta)$，最小化该式等价于最大化误差项服从非对称拉普拉斯分布(ALD)的非线性回归模型的似然函数。因此我们可以假设，模型 2-5 中的ε_i服从非对称拉普拉斯分布，其概率密度函数如下：

$$f_\tau(\varepsilon_i)=\tau(1-\tau)\exp\{-\rho_\tau(\varepsilon_i)\}. \qquad (3-1)$$

对于以上模型中的误差项，Kozumi 和 Kobayashi(2011)[27]指出，非对称拉普拉斯分布(ALD)可表示为一个正态分布与一个指数分布的混合分布，我们可将$\varepsilon=(\varepsilon_1,\dots,\varepsilon_n)$表示为由常量构成的位置-尺度混合物如下：

$$\varepsilon=\theta_1 z+\theta_2\sqrt{z}u. \qquad (3-2)$$

其中，$\theta_1=\frac{1-2\tau}{\tau(1-\tau)},\theta_2^2=\frac{2}{\tau(1-\tau)}$，z 服从均值为 1 的指数分布，u 服从标准正态分布，且 z 与 u 相互独立。由此，模型 2-5 可表示为如下形式：

$$y_i=x_i^\intercal\beta_0+\theta_1 z_i+\theta_2\sqrt{z_i}u_i,\qquad i=1,\dots,n. \qquad (3-3)$$

由式 3-3 可得，当z_i给定时，y_i的条件分布为正态分布，其均值为$x_i^\intercal\beta_0+\theta_1 z_i$，方差为$\theta_2^2 z_i$。由此，Tian 等(2014)[30]和 Zhou 等(2014)[31]指出，若将z_i视为已经给定的缺失数据，模型 3-3 可视作普通线性回归模型。求解包含缺失数据的模型中的未知参数，常用 EM 算法。因此可通过 EM 算法求解模型中的未知参数β_0。

完整的数据由观测数据$Y=(y_1,y_2,\dots,y_n)$与缺失数据$Z=(z_1,z_2,\dots,z_n)$构成，而$X=(x_1,x_2,\dots,x_n)$在推导过程中被控制为固定的协变量。求解模型 3-3 的 EM 算法步骤如下：

E 步：给定初始估计值$\hat{\beta}^0$，将 t-1 次迭代值记为$\hat{\beta}^{t-1}$，$Q(\cdot)$函数可表示为：

$$Q\left(\beta\middle|\hat{\beta}^{t-1}\right)=E\left(L_\beta\middle|Y,\hat{\beta}^{t-1}\right). \qquad (3-4)$$

其中，期望部分是给定Y和$\hat{\beta}^{t-1}$时 Z 的条件期望，L_β是完整数据(Y,Z)的对数似然函数。

M 步：通过最大化$Q\left(\beta\middle|\hat{\beta}^{t-1}\right)$，更新$\hat{\beta}^{t-1}$至$\hat{\beta}^t$。

重复 E 步与 M 步，直到β收敛。

在模型 3-3 中，由于$y_1,y_2,\dots,y_n$相互独立，对于给定的 z 与已知的β，Y 的

条件概率密度函数可表示为：

$$f(Y|Z,\beta)=\prod_{i=1}^{n}\left[\frac{1}{\sqrt{2\pi}\cdot\theta_2\sqrt{z_i}}\exp\left\{-\frac{(y_i-x_i^{\intercal}\beta-\theta_1 z_i)^2}{2\theta_2^2 z_i}\right\}\right]. \quad (3-5)$$

又 Y 与 Z 相互独立，$z_1,z_2,\dots,z_n$相互独立，完整数据(Y,Z)的似然函数可表示为：

$$f(Y,Z|\beta)=\prod_{i=1}^{n}\left[\frac{1}{\sqrt{2\pi}\cdot\theta_2\sqrt{z_i}}\exp\left\{-\frac{(y_i-x_i^{\intercal}\beta-\theta_1 z_i)^2}{2\theta_2^2 z_i}\right\}\right]\cdot\prod_{i=1}^{n}\exp\{-z_i\}. \quad (3-6)$$

因此，我们可以得到完整数据(Y,Z)的对数似然函数L_β表达式如下：

$$\begin{aligned}L_\beta&=\log\big(f(Y,Z|\beta)\big)\\&=-nlog\big(\sqrt{2\pi}\theta_2\big)-\frac{1}{2}\sum_{i=1}^{n}\log(z_i)-\frac{1}{2\theta_2^2}\sum_{i=1}^{n}\frac{(y_i-x_i^{\intercal}\beta-\theta_1 z_i)^2}{z_i}-\sum_{i=1}^{n}z_i.\end{aligned} \quad (3-7)$$

进而，我们可得到$Q\big(\beta\big|\hat{\beta}^{t-1}\big)$的表达式如下：

$$\begin{aligned}Q\big(\beta\big|\hat{\beta}^{t-1}\big)&=E\big(L_\beta\big|Y,\hat{\beta}^{t-1}\big)\\&=-nlog\big(\sqrt{2\pi}\theta_2\big)-\frac{1}{2}\sum_{i=1}^{n}E(\log(z_i))-\frac{1}{2\theta_2^2}\sum_{i=1}^{n}\{E(\frac{(y_i-x_i^{\intercal}\beta)^2}{z_i})+E(\theta_1^2 z_i)-E[2\\&\quad\cdot\theta_1(y_i-x_i^{\intercal}\beta)]\}-\sum_{i=1}^{n}E(z_i)\\&=\frac{\theta_1}{\theta_2^2}\sum_{i=1}^{n}(y_i-x_i^{\intercal}\beta)-\frac{1}{2\theta_2^2}\sum_{i=1}^{n}(y_i-x_i^{\intercal}\beta)^2E\big(z_i^{-1}\big|Y,\hat{\beta}^{t-1}\big)+C.\end{aligned} \quad (3-8)$$

其中，所有与β无关的项记为C。在上式 3-8 中，我们需要计算$E\big(z_i^{-1}\big|Y,\hat{\beta}^{t-1}\big)$。

由模型 3-3 可得，给定y_i和$\hat{\beta}^{t-1}$的条件下，z_i的边际条件概率密度正比于：

$$\begin{aligned}&z_i^{-\frac{1}{2}}\exp\left[-\frac{1}{2}\left\{\frac{\big(y_i-x_i^{\intercal}\hat{\beta}^{t-1}\big)^2}{\theta_2^2}z_i^{-1}+\left(\frac{\theta_1^2}{\theta_2^2}+2\right)z_i\right\}\right] \quad (3-9)\\&\sim GIG\left(\frac{1}{2},\frac{\big(y_i-x_i^{\intercal}\hat{\beta}^{t-1}\big)^2}{\theta_2^2},\left(\frac{\theta_1^2}{\theta_2^2}+2\right)\right).\end{aligned}$$

$GIG(\alpha,\delta,\gamma)$为广义逆高斯分布，其概率密度函数为：

$$f(v|\alpha,\delta,\gamma)=\frac{(\gamma/\delta)^\alpha}{2K_\alpha(\delta\gamma)}v^{\alpha-1}\exp\left\{-\frac{1}{2}\left(\frac{\delta^2}{v}+\gamma^2 v\right)\right\}. \quad (3-10)$$

其中，$v>0,-\infty<\alpha<\infty,\delta>0,\gamma>0$，且$K_\alpha(\cdot)$是第三类改进的巴塞尔函数。Karlis(2002)[27]指出，广义逆高斯分布的高阶矩的期望如下：

$$E(Z^r)=\left(\frac{\delta}{\gamma}\right)^r\frac{K_{\alpha+r}(\delta\gamma)}{K_\alpha(\delta\gamma)}. \quad (3-11)$$

根据 Abramowitz 和 Stegun(1972)，$K_\alpha(\cdot)$具有重要性质：

$$K_\alpha(\cdot)=K_{-\alpha}(\cdot). \quad (3-12)$$

取$r=-1,\alpha=1/2$，则有如下等式成立：

$$E(Z^{-1})=\left(\frac{\delta}{\gamma}\right)^{-1}\frac{K_{-1/2}(\delta\gamma)}{K_{1/2}(\delta\gamma)}=\frac{\gamma}{\delta}. \tag{3-13}$$

由此，我们可以得到：

$$E\left(z_i^{-1}\middle|Y,\hat{\beta}^{t-1}\right)=\frac{\sqrt{\theta_1^2+2\theta_2^2}}{\left|y_i-x_i^{\mathrm{T}}\hat{\beta}^{t-1}\right|}. \tag{3-14}$$

将式 3-14 代入式 3-8 可得：

$$\begin{aligned}Q\left(\beta\middle|\hat{\beta}^{t-1}\right)&=\frac{\theta_1}{\theta_2^2}\sum_{i=1}^{n}(y_i-x_i^{\mathrm{T}}\beta)-\frac{1}{2\theta_2^2}\sum_{i=1}^{n}(y_i-x_i^{\mathrm{T}}\beta)^2\cdot\frac{\sqrt{\theta_1^2+2\theta_2^2}}{\left|y_i-x_i^{\mathrm{T}}\hat{\beta}^{t-1}\right|}+C\\&=\frac{\theta_1}{\theta_2^2}\sum_{i=1}^{n}(y_i-x_i^{\mathrm{T}}\beta)-\frac{\sqrt{\theta_1^2+2\theta_2^2}}{2\theta_2^2}\sum_{i=1}^{n}\frac{(y_i-x_i^{\mathrm{T}}\beta)^2}{\left|y_i-x_i^{\mathrm{T}}\hat{\beta}^{t-1}\right|}+C\\&=\frac{1-2\tau}{2}\sum_{i=1}^{n}(y_i-x_i^{\mathrm{T}}\beta)-\frac{1}{4}\sum_{i=1}^{n}\frac{(y_i-x_i^{\mathrm{T}}\beta)^2}{\left|y_i-x_i^{\mathrm{T}}\hat{\beta}^{t-1}\right|}+C.\end{aligned} \tag{3-15}$$

当给定初始值$\hat{\beta}^0$(即式 3-15 中$t=1$)时，通过 EM 算法，我们可在 E 步中求得对数似然函数的条件期望$Q\left(\beta\middle|\hat{\beta}^0\right)$，在 M 步，通过使$Q\left(\beta\middle|\hat{\beta}^0\right)$最大化求得$\beta$即完成一次迭代，求得的$\beta$记为$\hat{\beta}^1$：

$$\begin{aligned}\hat{\beta}^1&=arg\max_{\beta\epsilon R^p}Q\left(\beta\middle|\hat{\beta}^0\right)\\&=arg\max_{\beta\epsilon R^p}\left\{\frac{1-2\tau}{2}\sum_{i=1}^{n}(y_i-x_i^{\mathrm{T}}\beta)-\frac{1}{4}\sum_{i=1}^{n}\frac{(y_i-x_i^{\mathrm{T}}\beta)^2}{\left|y_i-x_i^{\mathrm{T}}\hat{\beta}^{t-1}\right|}+C\right\}\\&=arg\min_{\beta\epsilon R^p}\left\{\frac{2\tau-1}{2}\sum_{i=1}^{n}(y_i-x_i^{\mathrm{T}}\beta)+\frac{1}{4}\sum_{i=1}^{n}\frac{(y_i-x_i^{\mathrm{T}}\beta)^2}{\left|y_i-x_i^{\mathrm{T}}\hat{\beta}^{t-1}\right|}\right\}\\&=arg\min_{\beta\epsilon R^p}\frac{1}{n}\sum_{i=1}^{n}\left\{\frac{(y_i-x_i^{\mathrm{T}}\beta)^2}{2\left|y_i-x_i^{\mathrm{T}}\hat{\beta}^0\right|}+(2\tau-1)\cdot(y_i-x_i^{\mathrm{T}}\beta)\right\}\\&=arg\min_{\beta\epsilon R^p}\{\frac{1}{2}\beta^{\mathrm{T}}D_n\beta-\beta^{\mathrm{T}}G_n\}.\end{aligned} \tag{3-16}$$

其中，D_n与G_n均只包含已知的观测值，$D_n=\frac{1}{n}\sum_{i=1}^{n}x_ix_i^{\mathrm{T}}/\left|y_i-x_i^{\mathrm{T}}\hat{\beta}^0\right|,G_n=\frac{1}{n}\sum_{i=1}^{n}x_i(\frac{y_i}{\left|y_i-x_i^{\mathrm{T}}\hat{\beta}^0\right|}-1+2\tau)$。

至此，我们可得到在 EM 算法中，分位数回归模型未知参数β的迭代公式，目标函数由开始的不可微函数变为可微的二次函数，方便直接计算。

3.2 PQREM 方法求解高维数据的分位数回归模型

在对高维数据进行分位数回归时，需要面对维数过高带来的计算量大，选择

的特征过多而真正对因变量有影响的特征只有少数几个的问题。因此，本节提出针对高维数据的 PQREM 方法，在式 3-16 的基础上，对变量进行降维。本文在降维方法上选择较常见的 LASSO 方法。下面介绍 PQREM 的具体方法：

根据第 2 章，在一般分位数回归模型中，求解未知参数的目标函数为：

$$arg\ \min_{\beta \epsilon R^p} \sum_{i=1}^{n} \rho_\tau(y_i - x_i^\intercal \beta). \tag{3-17}$$

在分位数回归的 LASSO 方法中，求解未知参数的目标函数为：

$$arg\ \min_{\beta \epsilon R^p} \sum_{i=1}^{n} \rho_\tau(y_i - x_i^\intercal \beta) + \lambda \|\beta\|_1. \tag{3-18}$$

其中λ为正则化参数，可通过 BIC 准则计算。

由此，将 LASSO 方法应用于 3.1 节求解分位数回归的 EM 算法中，参数β的迭代公式可表示为：

$$\begin{aligned}\hat{\beta}^1 &= arg \min_{\beta \epsilon R^p} \frac{1}{n} \sum_{i=1}^{n} \left\{ \frac{(y_i - x_i^\intercal \beta)^2}{2|y_i - x_i^\intercal \hat{\beta}^0|} + (2\tau - 1) \cdot (y_i - x_i^\intercal \beta) \right\} + \lambda \|\beta\|_1 \\ &= arg \min_{\beta \epsilon R^p} \left\{ \frac{1}{2} \beta^\intercal D_n \beta - \beta^\intercal G_n \right\} + \lambda \|\beta\|_1. \end{aligned} \tag{3-19}$$

其中，$D_n = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\intercal / |y_i - x_i^\intercal \hat{\beta}^0|, G_n = \frac{1}{n} \sum_{i=1}^{n} x_i (\frac{y_i}{|y_i - x_i^\intercal \hat{\beta}^0|} - 1 + 2\tau)$。在上式中，高维稀疏$\beta^0$的估计可通过求解类似 L1 惩罚加权的最小二乘优化来实现，而非通过求解 L1 惩罚加权的分位数回归模型实现，从而较大地减少了计算量。

以上，PQREM 方法的求解重点在于对式 3-19 进行求解，本文采用 PSSsp 算法：首先给定参数的初始估计值$\hat{\beta}^0$，计算伪梯度，并结合投影缩放方法找到梯度下降的方向，在该方向上移动设定的步长，从而完成参数估计值的一次迭代，重复进行该步骤直至参数估计值收敛，最终得到的估计值即逼近参数真实值。具体步骤如下：

1. 首先输入类似似然函数形式的不含正则项的优化目标函数$L(x)$，正则化参数λ，参数初始估计值$\hat{\beta}^0$，最高迭代次数 T，学习率η和容忍度ε。
2. 根据初始估计值$\hat{\beta}^0$，计算含有正则项的目标函数的值：

$$J(\hat{\beta}^0) = L(\hat{\beta}^0) + \lambda \|\beta\|_1. \tag{3-20}$$

并根据$\hat{\beta}^0$计算伪梯度g_k^0，第 t 次迭代对应的伪梯度计算公式为：

$$g_t = \tilde{\nabla}_t J(\hat{\beta}^t) = \begin{cases} \nabla_t L(\beta), & \lambda = 0 \\ \nabla_t L(\beta) + \lambda_t sign(\hat{\beta}^t), & \lambda > 0, |\hat{\beta}^t| > 0 \\ \nabla_t L(\beta) + \lambda_t, & \lambda > 0, |\hat{\beta}^t| = 0, \nabla_t L(\beta) < -\lambda_t \\ \nabla_t L(\beta) - \lambda_t, & \lambda > 0, |\hat{\beta}^t| = 0, \nabla_t L(\beta) > \lambda_t \\ 0, & \lambda > 0, |\hat{\beta}^t| = 0, |\nabla_t L(\beta)| \le \lambda_t. \end{cases} \quad (3-21)$$

3. 计算伪梯度中最大系数的绝对值：

$$\|g_t\|_\infty = \max|g_i|, i = 1,2,\dots,n. \quad (3-22)$$

当$\|g_t\|_\infty > \varepsilon$时，进入循环：

1) 根据投影缩放公式，计算伪梯度下降方向，记为d_t：

$$d_t = -\sigma_t \cdot g_t. \quad (3-23)$$

其中，

$$\sigma_t = \begin{cases} \min\left(1, \frac{1}{\|g_t\|_1}\right), t = 0 \\ \sigma_t \quad , t \ge 1 \end{cases}. \quad (3-24)$$

2) 根据d_t计算$\hat{\beta}$的更新公式：

$$\hat{\beta}^{t+1} = \begin{cases} 0, & \hat{\beta}^t(\hat{\beta}^t + \alpha d_t) < 0 \\ \hat{\beta}^t + \alpha d_t, & else \end{cases}. \quad (3-25)$$

其中，步长$\alpha = 1$。

3) 根据$\hat{\beta}^{t+1}$计算 $J(\hat{\beta}^{t+1})$与伪梯度g_{t+1}，检验$J(\hat{\beta}^{t+1})$与$J(\hat{\beta}^t)$是否满足：

$$J(\hat{\beta}^{t+1}) > J(\hat{\beta}^t) + \eta g_t^{\mathrm{T}}(\hat{\beta}^{t+1} - \hat{\beta}^t). \quad (3-26)$$

若满足式 3-26，则调整$\alpha = \alpha - 0.1$，重新计算$\hat{\beta}^{t+1}$，直至式 3-26 不满足，完成$\hat{\beta}$的一次更新。

4) 更新缩放投影公式中的σ_t：

$$\sigma_t = \begin{cases} 1, s_t = 0 \ or \ y_t = 0 \\ \min\left(\frac{s_t^{\mathrm{T}} s_t}{y_t^{\mathrm{T}} s_t}, \frac{s_t^{\mathrm{T}} y_t}{y_t^{\mathrm{T}} y_t}\right), t \ge 1 \end{cases}. \quad (3-27)$$

其中，$s_t = \hat{\beta}^{t+1} - \hat{\beta}^t, y_t = g_{t+1} - g_t$

4. 重复步骤 3，直至伪梯度$g_t = 0$，即$\|g_t\|_\infty < \varepsilon$，终止循环。最终所求的$\hat{\beta}^t$即为模型中未知参数的估计值。

3.3 海量数据下的 PQREM 方法

当处理海量数据时，将所有数据一次输入计算式 3-19 可能导致计算机内存不足与耗时过长的问题，根据 Lin 和 Xi(2011)[16]，本可采用分而治之的方法，将

完整数据分成 K 个小数据集，分别在每个小数据集中计算式$3-19$中的参数，以减少对内存与时间的消耗。最终我们得到分而治之的 PQREM 方法以求解海量高维数据分位数回归，步骤如下：

1. 将数据量为 n 的完整数据集划分为数据量相等的 K 个小数据集：$S_1, S_2, \ldots, S_K$，其中第 K 个小数据集包含n_k个观测值：$(x_{k,i}, y_{k,i}), i = 1,2, \ldots, n_k$；

2. 根据S_1计算$\hat{\beta}_\lambda^0$：

$$\hat{\beta}_\lambda^0 = arg\min_{\beta \in R^p} \sum_{i \in S_1} \rho_\tau (y_i - x_i^\top \beta) + \lambda \|\beta\|_1 \qquad (3-28)$$

3. 计算每个小数据集中的D_k和G_k，$k = 1,2, \ldots, K$，其中

$$D_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \frac{x_{k,i} x_{k,i}^\top}{\left| y_{k,i} - x_{k,i}^\top \hat{\beta}_\lambda^{t-1} \right|}, G_k = \frac{1}{n_k} \sum_{i=1}^{n_k} x_{k,i} \left(\frac{y_{k,i}}{\left| y_{k,i} - x_{k,i}^\top \hat{\beta}_\lambda^{t-1} \right|} - 1 + 2\tau \right)$$

4. 根据D_k和G_k计算$D_n = \frac{1}{n} \sum_{k=1}^K n_k D_k$，$G_n = \frac{1}{n} \sum_{k=1}^K n_k G_k$，并使用 PSSsp 算法求解带 L1 正则项的目标函数，进而得到$\hat{\beta}_\lambda^1$：

$$\hat{\beta}_\lambda^1 = arg\min_{\beta \in R^p} \left\{ \frac{1}{2} \beta^\top D_n \beta - \beta^\top G_n + \lambda \|\beta\|_1 \right\} \qquad (3-29)$$

5. 重复迭代步骤 3 与 4，直至$\hat{\beta}$收敛。

计算过程实现见表 3-1：

表 3-1 PQREM 方法的计算过程

Table3-1 Calculation steps of the PQREM method

PQREM 方法
输入：分块数据S_k，其中$k = 1, \ldots, K$， 分位数τ，迭代次数T，初始正则化参数λ
输出：回归系数$\widehat{\beta}$
1: 在S_1中，利用 R 语言中自带的 PQR 程序，计算初始估计量$\hat{\beta}^0$， $$\hat{\beta}^0 = arg\min_{\beta \in R^p} \sum_{i \in S_1} \rho_\tau (y_i - x_i^\top \beta) + \lambda \|\beta\|_1. \qquad (3-30)$$
2: for t = 1, …, T do:
3: for k = 1, …, K do:

表 3-1 PQREM 方法的计算过程（续）
Table3-1 Calculation steps of the PQREM method

4: 计算 $D_k = \frac{1}{n_k}\sum_{i=1}^{n_k}\frac{x_{k,i}x_{k,i}^{\intercal}}{\left|y_{k,i}-x_{k,i}^{\intercal}\hat{\beta}^{t-1}\right|}$,

$$G_k = \frac{1}{n_k}\sum_{i=1}^{n_k} x_{k,i}\left(\frac{y_{k,i}}{\left|y_{k,i}-x_{k,i}^{\intercal}\hat{\beta}^{t-1}\right|} - 1 + 2\tau\right).$$

5: 计算 $D_n = \frac{1}{n}\sum_{k=1}^{K} n_k D_k$, $G_n = \frac{1}{n}\sum_{k=1}^{K} n_k G_k$ ， 用 PSSsp 方法求解：

$$\hat{\beta}_\lambda^t = arg\min_{\beta\in R^p}\left\{\frac{1}{2}\beta^{\intercal}D_n\beta - \beta^{\intercal}G_n + \lambda\|\beta\|_1\right\}. \qquad (3-31)$$

6: 直至$\hat{\beta}_\lambda^t - \hat{\beta}_\lambda^{t-1} \le \varepsilon$或 t = T

7: end for

3.4 本章小结

在一般分位数回归模型的求解中，首先根据 Yu 和 Moyeed(2001)[26]，将分位数回归的误差项分布近似表示为相互独立的指数分布与正态分布的组合成的混合分布，从而将分位数回归求解转化为含缺失数据 z 的具有异方差的普通最小二乘优化求解。写出完整数据的似然函数后，利用广义逆高斯分布的性质，可求得含有缺失数据部分的期望，最终推导得到 EM 算法求解一般分位数回归的参数β的迭代公式。

其次，为在高维数据中进行变量选择，本章在分位数回归损失函数的基础上添加 L1 正则项，得到求解高维数据分位数回归模型的 PQREM 方法。在 PQREM 方法的迭代过程中使用 Schmidt(2010)提出的次梯度缩放投影算法(PSSsp)对带有 L1 正则项的目标函数求解。

最后，由于需要处理海量数据，为降低计算量，减少内存消耗，本章将完整数据集划分为几个小数据集，在每个小数据集中分别计算迭代公式中β的系数，将每个小数据集的计算结果按照数据量占比加权平均即为完整数据集中β的系数。最终得到适用于海量高维数据的基于 EM 算法的分位数回归求解方法(PQREM)。

第 4 章 模拟实验

为对第三章中提出的 PQREM 方法进行评估，在将其应用于真实数据集之前，我们首先通过蒙特卡罗方法生成模拟数据集，分别在模拟生成的一般规模高维数据与海量规模高维数据中应用该算法，并将 PQREM 方法与已有方法进行对比，检验其拟合效果。

4.1 一般数据规模的模拟研究

PQREM 方法涉及分而治之的原理，本质上是将几个小数据集的计算结果进行综合，因此为检验该方法用于分位数回归模型的有效性，本节首先检验在一次输入一般规模的高维数据集时，PQREM 方法的有效性。为使得检验结果更有说服力，我们将 PQREM 方法与 R 语言中用于高维数据的分位数回归模型包“quantreg”中的 LASSO 方法进行对比。为便于表述，我们将对比的方法记为 PQR 方法。

4.1.1 模型参数设置

1. 确定数据条数$n = 200$；
2. 随机构造p维回归系数：

$$\beta = (1,1,1,0,0,1,0,1,1,1,0,0,0,0,1,0,0,1,1,0,\dots,0)^{\intercal}, \quad p = 100$$

3. 稀疏系数$s = 10$，即有 10 个非零的回归系数；
4. 为验证不同分位数模型中的回归效果，分别取$\tau = 0.2,0.5,0.8$；
5. 根据如下线性模型构造数据：

$$Y_i = X_i^{\intercal}\beta_0 + \sigma\{\varepsilon_i - F_{\varepsilon_i}^{-1}(\tau)\}, \ i = 1,2,\dots,n. \qquad (4-1)$$

其中，$X_i = (1, X_{i1}, X_{i2}, \dots, X_{ip})^{\intercal}$为$p+1$维向量，$(X_{i1}, X_{i2}, \dots, X_{ip})$由$[0,2]^p$区间上的多变量均匀分布与协差阵$\Sigma$构成，$\Sigma_{ij} = 0.8^{|i-j|}, 1 \le i,j \le 100$。这样可以保证自变量之间的弱相关关系，更接近真实情况。由于分位数回归模型对方差并无某一特定分布的要求，且真实数据中方差多呈现异方差的情况，因此随机误差项分别按照如下两种情况构造，分别记为 case1 与 case2：

1）case1：$\sigma = 0.2$，ε服从标准正态分布$N(0,1)$，此时误差项呈对称分布；

2）case2：$\sigma(X) = 0.1(2 + \sin(X^{\mathrm{T}}\beta_0))$，$\varepsilon$服从自由度为 5 的$t$分布$t(5)$，此时误差项呈异方差的偏态分布。

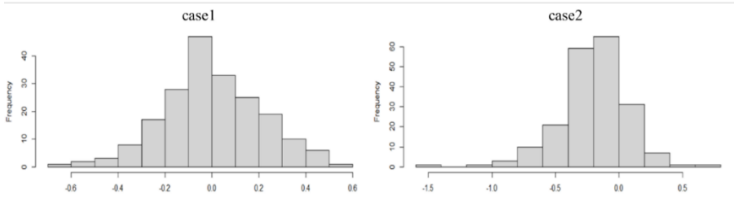

图 4-1 一般规模数据集模拟实验随机误差项分布

Fig4-1 Distribution of random error terms in simulation experiments on general-scale datasets

4.1.2 模型评价指标

模型的评价主要针对回归系数估计的准确性，选取均方根差(RMSE)指标，可反映模型估计的回归系数$\hat{\beta}$与真实值β之间的差距：

$$RMSE = \sqrt{\left\|\hat{\beta} - \beta\right\|_2} = \sqrt{\sum_{i=1}^{p}(\beta_i - \widehat{\beta}_i)^2}. \qquad (4-2)$$

RMSE 越小，参数估计值与真实值的差距越小，模型效果越好。

4.1.3 模拟实验结果

根据以上参数设置，我们在 500 次模拟实验中计算了 RMSE 指标的均值与方差，结果如下：

表 4-1 一般规模数据集中 PQR 与 PQREM 方法 RMSE 指标对比

Table4-1 Comparison of RMSE indicators between PQR and PQREM methods in general-scale datasets

Case	τ	RMSE_mean		RMSE_sd	
		PQR	PQREM	PQR	PQREM
Case1	0.5	0.108	0.069	0.014	0.021
	0.2	0.106	0.071	0.015	0.016
	0.8	0.105	0.081	0.015	0.018
Case2	0.5	0.335	0.224	0.308	0.219
	0.2	0.646	0.454	0.102	0.098
	0.8	0.641	0.492	0.096	0.126

根据表 4-1 中的数据对比可得到如下结论：

无论误差项服从什么分布，分位数τ如何，PQREM 方法的 MSE 均值均明显小于 PQR 方法，且两方法 RMSE 的方差差距并不大，RMSE 的稳定性相当。因此，PQREM 方法对回归系数估计的准确性更优，尤其当误差项服从异方差的偏态分布时(case2)，这一优势更为明显。

4.2 海量数据规模的模拟研究

在一般规模数据集中验证 PQREM 方法在分位数回归模型的有效性后，因为该方法是针对海量高维数据的分位数回归提出，所以我们将使用蒙特卡罗生成的海量高维数据集对该方法的有效性进行检验。为使检验结果更有说服力，我们选择 Chen 等(2020)[23]提出的同样适用于海量高维数据分位数回归的 PQRC 方法进行对比，PQRC 同样运用了分而治之与增加 L1 正则项的方法，与 PQREM 的区别在于未选择使用 EM 算法进行分位数回归求解。同时本节检验了 PQREM 方法在划分数据集个数增加时计算量是否有减小。

4.2.1 模型参数设置与评价指标选择

模型参数设置：

1. 确定数据条数$n = 2 * 10\text{^}5$；

2. 划分数据集的个数$K = 1,10,50,100$;

3. p维回归系数、稀疏系数、线性模型的构造与以上在一般规模高维数据中使用的参数相同，仅在最终的误差项中存在以下变化：

1）case1：$\sigma = 0.2$，ε服从标准正态分布$N(0,1)$，此时误差项呈对称分布；

2）case3：$\sigma = 0.5$，ε服从自由度为 3 的卡方分布$\chi^2(3)$，此时误差项呈偏态分布。

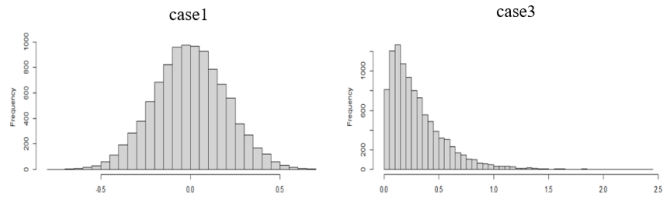

图 4-2 海量高维数据集模拟实验随机误差项分布
Fig4-2 Distribution of random error terms in simulation experiments of massive high-dimensional datasets

模型有效性评价指标仍为 RMSE，计算量评价指标为所用的时间t。

4.2.2 模拟实验结果

按照以上参数设置，在 100 次模拟实验中计算了 RMSE 指标的均值与方差，结果如下：

表 4-2 海量高维数据集中 PQRC 与 PQREM 方法 RMSE 指标对比
Table4-2 Comparison of RMSE indicators between PQRC and PQREM methods in massive high-dimensional datasets

Case	τ	K	RMSE_mean		RMSE_sd	
			PQRC	PQREM	PQRC	PQREM
Case1	0.5	1	0.027	0.027	0.004	0.010
		10	0.060	0.076	0.017	0.029
		50	0.295	0.271	0.050	0.054
		100	0.441	0.405	0.054	0.073
	0.2	1	0.033	0.034	0.006	0.008
		10	0.081	0.114	0.025	0.036
		50	0.372	0.299	0.062	0.081
		100	0.531	0.463	0.046	0.141
	0.8	1	0.030	0.026	0.006	0.005
		10	0.073	0.094	0.023	0.037
		50	0.334	0.266	0.066	0.111
		100	0.527	0.499	0.064	0.059
Case3	0.5	1	0.065	0.060	0.010	0.022
		10	0.136	0.188	0.038	0.073
		50	0.656	0.588	0.060	0.164
		100	0.942	0.938	0.090	0.124
	0.2	1	0.035	0.036	0.005	0.009
		10	0.088	0.108	0.017	0.043
		50	0.430	0.384	0.065	0.083
		100	0.614	0.602	0.050	0.079
	0.8	1	0.133	0.090	0.019	0.020

表 4-2 海量高维数据集中 PQRC 与 PQREM 方法 RMSE 指标对比
Table4-2 Comparison of RMSE indicators between PQRC and PQREM methods in massive high-dimensional datasets

Case3	0.8	10	0.382	0.398	0.066	0.160
		50	1.151	1.059	0.091	0.289
		100	1.690	1.657	0.216	0.181

由表 4-2 数据可得：

无论分位数τ如何，误差服从什么分布，在划分数据集个数大于等于 50 的情况下，PQREM 方法的 RMSE 均值较低，此时 PQREM 估计的回归系数更加准确；在划分数据集个数小于 50 的情况下，PQRC 方法的 RMSE 均值较低，此时 PQRC 估计的回归系数更加准确。因此，随划分数据集个数增加，PQREM 方法相较于 PQRC 方法更加准确。这一优势同样在误差分布呈偏态时(case3)更加明显。

表 4-3 海量高维数据集中 PQREM 方法计算时间
Table4-3 Calculation time of PQREM method in massive high-dimensional datasets

τ	K	Case1	Case3
0.5	1	74.168	87.710
	10	52.084	38.885
	50	27.576	29.528
	100	26.621	24.427
0.2	1	54.670	78.488
	10	19.891	31.037
	50	24.026	25.870
	100	22.249	27.094
0.8	1	65.189	93.877
	10	35.413	33.172
	50	27.927	21.178
	100	33.115	18.234

由表 4-3 可得：

无论分位数τ如何，误差项服从什么分布，随划分数据集个数增加，PQREM 方法的计算用时均不断减小，因此 PQREM 采用的分而治之方法可以有效减少内存占用。

综上，在海量高维数据集的模拟实验中，无论分位数τ如何，误差分布与划分数据集个数如何，当划分数据集数量大于等于 50 时，PQREM 方法的准确性更优，当划分数据集数量小于 50 时，PQRC 方法的准确性更优。因此，PQREM 方法更适用于数据量大，将数据集划分为个数更多的情况，且随划分数据集划分个数增加，计算时间逐渐减小。

4.3 实证分析

本节中我们检验提出的海量高维数据的分位数回归方法(PQREM)在真实数据中的准确性，为使得检验结果更有说服力，我们将 PQREM 方法与上述提到的 PQR 方法与 PQRC 方法进行对比。同时检验了 PQREM 方法在划分数据集个数增加时计算量的减小与在变量选择上的有效性。

4.3.1 数据集描述

YearPredictionMSD 数据集由公开的机器学习数据集网站获得(https://archive.ics.uci.edu/ ml/datasets/YearPredictionMSD)，该数据集是 1922 年至 2011 年当代流行音乐曲目的音频特征集合。数据集中记录了约 515345 个观测值，包含 91 个变量，其中有歌曲的年份，12 个音色平均值和 78 个音色协方差。人们猜测受到发型设备与流行趋势的影响，同一年份发行的音乐可能具有相同的某类特征，因此以往的研究中，学者们希望通过音色特征预测歌曲的发行年份。所以本数据集中歌曲的年份为因变量，90 个音色特征为自变量，数据集满足海量高维的条件，且可以通过分位数回归模型进行预测。为方便对模型估计效果的检验，我们将前 500000 条数据划分为训练集，后 15345 条数据划分为测试集。

4.3.2 模型评价指标

在真实数据集中依旧从模型的预测准确性、变量选择与运行时间三方面选择指标进行模型评价：

1. 平均绝对预测误差(MAPE)：可反映测试集中模型估计的发行年份预测值与真实值的误差：

$$MAPE = \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\left|Y_i - \hat{Y}_i\right|. \quad (4-6)$$

2. 变量选择个数(Vars)：可反映模型对于高维数据集的降维效果：

$$\text{Vars} = \text{I}\left(\hat{\beta}_j \neq 0\right). \quad (4-7)$$

3. 算法运行时间 t。

4.3.3 实验结果

对 YearPredictionMSD 数据集进行分位数回归拟合后，得到相关指标与结论如下：

表 4-4 真实数据集中三种方法 MAPE 指标对比
Table4-4 Comparison of MAPE indicators of three methods in real data

τ	K	PQR	PQRC	PQREM
0.5	1	6.694	6.948	6.934
	10	—	6.857	6.853
	50	—	6.900	6.891
	100	—	7.048	7.040
0.2	1	9.015	9.582	9.556
	10	—	9.253	9.211
	50	—	9.496	9.381
	100	—	9.502	9.259
0.8	1	8.325	7.627	7.628
	10	—	7.981	7.985
	50	—	7.801	7.802
	100	—	7.870	7.867

由表 4-4 可得：

在 0.2 与 0.5 分位数回归的拟合中，数据一次计算时，PQRC 方法的 MAPE 略高于 PQREM 方法，两方法的 RMSE 均高于 PQR 方法，此时 PQR 的拟合效果最好，PQREM 方法次之，PQRC 方法最末，将数据集划分为多个小数据集时，PQREM 方法优于 PQRC 方法。在 0.8 分位数回归中，PQRC 与 PQREM 的拟合效果相当，PQR 方法的效果最末。

同时，我们发现，无论使用什么方法，三种分位数回归模型中 0.5 分位数回归的 RMSE 小于 0.2 与 0.8 分位数回归，因此，0.5 分位数回归模型用于预测音乐发行年份更为合适。

表 4-5 真实数据集中三种方法计算时间对比
Table4-5 Comparison of calculation time of three methods in real data

τ	K	PQR	PQRC	PQREM
0.5	1	53.126	95.370	101.940
	10	—	16.790	23.040
	50	—	7.980	15.270
	100	—	8.110	14.660
0.2	1	89.400	111.250	121.300
	10	—	18.830	28.420
	50	—	11.020	17.750
	100	—	9.010	13.090
0.8	1	71.400	99.050	117.890
	10	—	19.990	32.200
	50	—	13.260	28.280
	100	—	17.390	25.810

由表 4-5 可得：

在所有数据一次输入模型计算时，PQR 方法更有优势，但当数据集划分为多个小数据集时，PQRC 方法与 PQREM 方法的计算时间明显缩短，由此可证明分而治之方法在减小计算量，缩短计算时间中的有效性。

表 4-6 真实数据集中 PQREM 方法变量选择个数
Table4-6 The number of variables selected by the PQREM method in real data

τ	K=1	K=10	K=50	K=100
0.5	51	50	58	58
0.2	65	65	68	69
0.8	33	33	43	46

由表 4-6 可得，PQREM 方法中选择的变量个数远少于 90 个，因此 PQREM 方法起到了变量选择的作用。

综上，通过真实数据，验证了 PQREM 方法预测的有效性、减少计算时间的

有效性与变量选择的有效性。

4.4 本章小结

在本章中，我们采用数值模拟与实证检验的方式，从预测准确性、计算时间与变量选择三个方面，验证提出的 PQREM 方法对于海量高维数据的分位数回归模型的有效性。

在数值模拟中，我们首先通过蒙特卡罗方法生成 200 条 100 维数据作为自变量，根据随机生成的回归系数与选定的误差项得到响应变量，最终得到符合已知分位数回归模型的一般规模高维数据集。在一般规模数据集上对比 R 语言自带的分位数回归方法 PQR 与 PQREM 方法对于回归系数的估计准确性，验证得到 PQREM 方法效果更优。而后将数据集的样本量增加为 20 万条，按照相同的方法得到符合分位数回归模型的海量高维数据集。将海量数据集划分为 1，10，50，100 个小数据集，对比 Chen 等(2020)提出的 PQRC 方法与 PQREM 方法的回归系数准确性，验证得到 PQREM 方法更适用于数据量大，将数据集划分为个数更多（50 个及以上）的情况，且随划分数据集划分个数增加，计算时间逐渐减小。

在实证检验中，我们用分位数回归模型拟合真实的音色特征数据集 YearPredictionMSD 以预测音乐发行的年份，对比不同分位数回归模型，最终得到 0.5 分位数回归的情况下，拟合效果最好。此时 PQREM 方法的预测准确性优于 PQRC 方法，计算时间优于 PQR 方法，且最终模型选择的变量个数远小于 90 个，证明了模型具有变量选择的效果。

第 5 章 总结与展望

5.1 总结

对于海量高维数据分析，本文提出了一种分位数回归估计方法。首先通过理论推导得到求解高维数据分位数回归的目标函数，后将海量数据集划分为几个小数据集分别进行目标函数中参数的计算，从而减少对计算机内存的占用，最终得到 PQREM 方法。该方法解决了数据量大带来的计算量超出内存与维数高带来的模型过于复杂的问题，使得分位数回归模型可以应用于海量高维数据的分析。

论文在求解分位数回归模型的过程中，首先将分位数回归模型目标函数最小化转化为误差项服从非对称拉普拉斯分布(ALD)的非线性回归模型的似然函数的最大化，从而将非平滑的损失函数转化为平滑的二次函数，再通过极大似然的思想进行求解。该转化方法对求解其他非平滑目标函数的问题也具有参考意义。而后将模型求解看做含有缺失数据的目标函数求解问题，从而可以使用已有的 EM 算法求解。在计算过程中，为达到变量选择的目的，本文在目标函数中增加 L1 正则项，此方法对于其他机器学习模型的高维数据应用也具有参考意义。

论文最后通过数值模拟与实证检验，对比已有的分位数回归模型的计算方法(PQR 与 PQRC)，验证了 PQREM 方法在海量高维分位数回归中对回归系数估计与预测的有效性、减少计算内存的有效性与变量选择的有效性。

5.2 不足与展望

文本对海量高维数据，给出了一种分位数回归估计方法。然而未能在理论上证明估计方法的收敛性和估计量的大样本性质。在后续的工作中，我们将对估计方法的理论性质进行研究。

参考文献

[1] 托夫勒, 步茗. 第三次浪潮[J]. 现代情报, 1984(00):35-36.

[2] Pearson K. LIII. On lines and planes of closest fit to systems of points in space[J]. The London, Edinburgh, and Dublin philosophical magazine and journal of science, 1901, 2(11): 559-572.

[3] Hotelling, H. Analysis of a complex of statistical variables in principal components[J]. Journal of Educational Psychology, 1933, 24(7):498-520.

[4] Golub G H. Singular value decomposition and least squares solutions[J]. Numerische Mathematik, 1970, 14(5):403-420.

[5] Akaike H. Information theory and an extension of the maximum likelihood principle[M]//Selected papers of hirotugu akaike. Springer, New York, NY, 1998: 199-213..

[6] Schwarz G. Estimating the dimension of a model[J]. The annals of statistics,1978: 461-464.

[7] Foster D P, George E I. The Risk Inflation Criterion for Multiple Regression[J]. The Annals of Statistics, 1994, 22(4):1947-1975.

[8] Hoerl A E, Kannard R W, Baldwin K F. Ridge regression: some simulations[J]. Communications in Statistics-Theory and Methods, 1975, 4(2): 105-123.

[9] Tibshirani R. Regression shrinkage and selection via the lasso: a retrospective[J]. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2011, 73(3):267-288.

[10] Efron B, Hastie T, Tibshirani J R. Least Angle Regression[J]. Annals of Statistics, 2004, 32(2):407-451.

[11] Powell J L. Least absolute deviations estimation for the censored regression model[J]. Journal of econometrics, 1984, 25(3): 303-325.

[12] Koenker R, Bassett G W. Regression quantiles[J]. Econometrica, 1978, 46(1):211-244.

[13] Bouyé E, Gaussel N, Salmon M. Investigating dynamic dependence using copulae[J]. WP01-03, 2002.

[14] Whittaker J, Somers W M. The Neglog Transformation and Quantile Regression for the Analysis of a Large Credit Scoring Database[J]. Journal of the Royal Stati-stical Society. Series C (Applied Statistics), 2005, 54(5):863-878.

[15]苏栝芳,蔡经汉.影响我国居民教育回报因素的分位数回归分析[J].哈尔滨商业大学学报(社会科学版),2010(03):97-100+109.

[16]Lin N, Xi R. Aggregated estimating equation estimation[J]. Statistics and its Interface, 2011, 4(1): 73-83.

[17]Chen X. Analysis of big data by split-and-conquer and penalized regressions: New methods and theories[J]. Dissertations & Theses - Gradworks, 2013.

[18]Schifano E D, Wu J, Wang C, et al. Online updating of statistical inference in the big data setting[J]. Technometrics, 2016, 58(3): 393-403.

[19]Jiang R, Hu X, Yu K, et al. Composite quantile regression for massive datasets[J]. Statistics, 2018, 52(5): 980-1004.

[20]Jiang R, Guo M F, Liu X. Composite quasi-likelihood for single-index models with massive datasets[J]. Communications in Statistics-Simulation and Computation, 2020: 1-17.

[21]Lee J D, Sun Y, Liu Q, et al. Communication-efficient sparse regression: a one-shot approach[J]. arXiv preprint arXiv:1503.04337, 2015.

[22]Michael I. Jordan, Jason D. Lee, Yun Yang. Communication-Efficient Distributed Statistical Inference[J]. Journal of the American Statistical Association, 2018, 114(526).

[23]Chen Xi, Liu Weidong, Mao Xiaojun, Yang Zhuoyi. Distributed High-dimensional Regression Under a Quantile Loss Function[J]. Journal of Machine Learning Research, 2020,21.

[24]Schmidt M. Graphical model structure learning using L_1-regularization[J]. 2010.

[25]Stefan Solntsev, Jorge Nocedal, Richard H. Byrd. An algorithm for quadratic ℓ1-regularized optimization with a flexible active-set strategy[J]. Optimization Methods and Software,2015,30(6).

[26]Yu K, Moyeed R A. Bayesian quantile regression[J]. Statistics & Probability Letters, 2001, 54.

[27]Hideo Kozumi, Genya Kobayashi. Gibbs sampling methods for Bayesian quantile regression[J]. Journal of Statistical Computation and Simulation,2011,81(11).

[28]Dempster A P. Maximum likelihood from incomplete data via the EM algorithm[J]. Journal of the Royal Statistical Society, 1977, 39.

[29]Karlis D. An EM type algorithm for maximum likelihood estimation of the normal-inverse Gaussian distribution[J]. Statistics & Probability Letters, 2002, 57(1):43-52.

[30] Yuzhu Tian, Maozai Tian, Qianqian Zhu. Linear Quantile Regression Based on EM Algorithm[J]. Communications in Statistics - Theory and Methods,2014,43(16).

[31] Ying-hui Zhou, Zhong-xin Ni, Yong Li. Quantile Regression via the EM Algorithm[J]. Communications in Statistics - Simulation and Computation, 2014,43(10).

[32] 杨雪梅. 基于 MM 算法的部分线性分位数回归[D].华北电力大学(北京), 2019. DOI:10.27140/d.cnki.ghbbu.2019.000798.

[33] Roger W. Koenker, Vasco D'Orey. Computing Regression Quantiles[J]. Journal of the Royal Statistical Society: Series C (Applied Statistics),1987,36(3).

[34] Karmarkar N. A new polynomial-time algorithm for linear programming[C]// Proceedings of the sixteenth annual ACM symposium on Theory of computing. 1984: 302-311.

[35] Madsen K, Nielsen H B. A finite smoothing algorithm for linear L1 estimation[J]. SIAM Journal on Optimization, 1993, 3(2): 223-235.

[36] Chen C. An adaptive algorithm for quantile regression[M]//Theory and applications of recent robust methods. Birkhäuser, Basel, 2004: 39-48..

[37] 陈粮阳. 基于整合分析的部分线性单指标分位数回归模型[D]. 浙江工商大学, 2018.

[38] Wu T T, Lange K. Coordinate descent algorithms for lasso penalized regression[J]. Annals of Applied Stats, 2008, 2(1):224-244.

附录 代码部分

```
#模拟实验 1
library(quantreg)
library(SparseM)
library(KernSmooth)
library(MASS)
library(stats)
library(MultiRNG)
setwd('C:/Users/JR/Desktop/R code')
source('Func_pss.R')

####### ximple begin#########
N=200 ###全样本数
nite=500 ###模拟迭代数
p=100
b1=c(1,1,1,0,0,1,0,1,1,1,0,0,0,0,1,0,0,1,1,0)
s=length(b1)
b2=matrix(0,nrow=p-s,ncol=1)
beta_true=matrix(c(1,b1,b2),nrow=p+1,ncol=1)
#tauall=c(0.5,0.2,0.8)
tau=0.8
t_ball=matrix(0,nrow=nite,ncol=1)
t_DS=matrix(0,nrow=nite,ncol=1)
RMSE_ball=matrix(0,nrow=nite,ncol=1)
RMSE_DS=matrix(0,nrow=nite,ncol=1)

#Func_pss 方法求解 L1 正则项目标函数
Func_pss=function(X1, X2, x0, lambda)
#fx=t(x0)%*%X2%*%x0+t(x0)%*%X1+lambda*sum(abs(x0)) return x1
{
  F_Lx=t(x0)%*%X2%*%x0+t(x0)%*%X1
  fk=F_Lx+lambda*sum(abs(x0))
```

```
dF_Lx=X1+2*X2%*%x0
gk=(dF_Lx+lambda*xign(x0))*(abs(x0)>10^(-
3))+(dF_Lx+lambda)*(abs(x0)<10^(-3))*(dF_Lx<(-lambda))+(dF_Lx-
lambda)*(abs(x0)<10^(-3))*(dF_Lx>lambda)
d=-min(1,1/sum(abs(gk)))*gk
it=0
while ((it<10)*(max(abs(gk))>10^(-3)))
{
  a=1
  x1=(x0+a*d)*((x0*(x0+a*d))>=0)
  F_Lx1=t(x1)%*%X2%*%x1+t(x1)%*%X1
  fk1=F_Lx1+lambda*sum(abs(x1))
  dF_Lx1=X1+2*X2%*%x1
  gk1=(dF_Lx1+lambda*xign(x1))*(abs(x1)>10^(-
3))+(dF_Lx1+lambda)*(abs(x1)<10^(-3))*(dF_Lx1<(-lambda))+(dF_Lx1-
lambda)*(abs(x1)<10^(-3))*(dF_Lx1>lambda)
  itt=0
  while ((itt<10)*(fk1>(fk+0.1*t(gk)%*%(x1-x0))))
  {
    a=a-0.1
    x1=(x0+a*d)*((x0*(x0+a*d))>=0)
    F_Lx1=t(x1)%*%X2%*%x1+t(x1)%*%X1
    fk1=F_Lx1+lambda*sum(abs(x1))
    dF_Lx1=X1+2*X2%*%x1
    gk1=(dF_Lx1+lambda*xign(x1))*(abs(x1)>10^(-
3))+(dF_Lx1+lambda)*(abs(x1)<10^(-3))*(dF_Lx1<(-lambda))+(dF_Lx1-
lambda)*(abs(x1)<10^(-3))*(dF_Lx1>lambda)
    itt=itt+1
  }
  sk=x1-x0
  yk=gk1-gk
  if ((sum(abs(sk))==0)|(sum(abs(yk))==0))
  {xigma=1}else
  {xigmX1=t(sk)%*%sk/(t(yk)%*%sk)
```

```
      xigmX2=t(sk)%*%yk/(t(yk)%*%yk)
      xigma=min(100,xigmX1,xigmX2)
      }
      d=-xigma*gk1*((xigma*gk1*gk1)>0)
      gk=gk1
      fk=fk1
      x0=x1*(abs(x1)>10^(-3))
      it=it+1
   }
   return(x0)
}

ss=matrix(0,nrow=p,ncol=p)
for (xi in 1:p)
{
   for (xj in 1:p)
   {
      ss[xi,xj]=0.8^(abs(xi-xj))
   }
}

      ##############sample################
      for (nii in 1:nite)
         {
            x_alg=draw.d.variate.uniform(no.row=N,d=p,cov.mat=ss)*2
            x_al=cbind(1,x_alg)
         ####################################
         #err=0.2*rnorm(N,0,1)
         #y_al=x_al%*%beta_true+(err-quantile(err,tau))
         ####################################
         err=rt(N,5)
         err1=0.2*rnorm(N,0,1)
         err2=0.1*(2+xin(x_al%*%beta_true))*(err-quantile(err,tau))
         hist(err1,breaks = function(x) length(x)/20)
```

```
hist(err2,breaks = function(x) length(x)/20)
y_al=x_al%*%beta_true+0.1*(2+xin(x_al%*%beta_true))*(err-
quantile(err,tau))

#######1- ball1    ###################
ball=rq(y_al ~ x_alg, method="lasso",tau=tau)$coef
ball1=ball
ball1[ball1!=0]=1
RMSE_ball[nii]=sqrt(sum((ball-beta_true)^2))
#######err 1- ball1    ################
#######2- PQREM    #################
lambda=0.5*sqrt(p/N)
func_bd=ginv(t(x_al)%*%x_al)%*%t(x_al)%*%y_al
it=0
RMSE=1
while ((it<20)*(RMSE>10^(-4)))
{
    xguo=matrix(0,nrow=N,ncol=(p+1))
    for(pi in 1:(p+1))
    {
      xguo[,pi]=x_al[,pi]/abs(y_al-x_al%*%func_bd)
    }
    D=t(x_al)%*%xguo
    G=t(xguo)%*%y_al+(2*tau-1)*colSums(x_al)
  X1=-G/N
  X2=D/N/2
  func_bd1=Func_pss(X1, X2, func_bd, lambda)
  #fx=t(x0)%*%X2%*%x0+t(x0)%*%X1+lambda*sum(abs(x0)) return x1
  it=it+1
  RMSE=sqrt(sum((func_bd1-func_bd)^2))
  func_bd=func_bd1
}
b_DS=func_bd
RMSE_DS[nii]=sqrt(sum((b_DS-beta_true)^2))
```

```
      #######err 2-PQREM    ############
    }

  RMSE_mean=cbind(mean(RMSE_ball),mean(RMSE_DS))
  RMSE_sd=cbind(sd(RMSE_ball),sd(RMSE_DS))
  print(round(RMSE_mean,3))
  print(round(RMSE_sd,3))

#模拟实验 2
library(quantreg)
library(SparseM)
library(KernSmooth)
library(MASS)
library(stats)
library(MultiRNG)
library(cqrReg)

####### ximple begin#########
N=2*10^5 ###全样本数
nite=100 ###模拟迭代数
p=100
b1=c(1,1,1,0,0,1,0,1,1,1,0,0,0,0,1,0,0,1,1,0)
#b1=c(1,1,1,1)
s=length(b1)
b2=matrix(0,nrow=p-s,ncol=1)
beta_true=matrix(c(1,b1,b2),nrow=p+1,ncol=1)
Kall=c(1,10,50,100)
KL=length(Kall)
tauall=c(0.5,0.2,0.8)
t_DS=matrix(0,nrow=nite,ncol=KL)
RMSE_C=matrix(0,nrow=nite,ncol=KL)
RMSE_DS=matrix(0,nrow=nite,ncol=KL)
ss=matrix(0,nrow=p,ncol=p)
```

```
for (xi in 1:p)
{
   for (xj in 1:p)
   {
      ss[xi,xj]=0.8^(abs(xi-xj))
   }
}
#########sample#######
t11=Sys.time()
for (casei in 1:2)
{
   print(casei)
   for (tauq in 1:3)
   {
      tau=tauall[tauq]
      print(tau)
      for (nii in 1:nite)
      {
         #print(nii)
         x_alg=draw.d.variate.uniform(no.row=N,d=p,cov.mat=ss)*2
         x_al=cbind(1,x_alg)
         ####################
         err1=0.2*rnorm(N,0,1)
         y_al1=x_al%*%beta_true+(err1-quantile(err1,tau))
         ####################
         err2=0.5*rchisq(N,3)
         y_al2=x_al%*%beta_true+(err2-quantile(err2,tau))
         y_alcase=cbind(y_al1,y_al2)
         y_al=y_alcase[,casei]

         for (Kq in 1:KL)
         {
```

```
K=Kall[Kq] ###分块数
#print(K)
n=N/K
###### 1- b0    ######
t1=Sys.time()
y0=y_al[1:n]
x0=x_al[1:n,]
x0g=x_alg[1:n,]
b0=rq(y0 ~ x0g, method="lasso",tau=tau,lambda=0.5*sqrt(p/n))$coef
t2=Sys.time()
t_b0=t2-t1
######err 1- b0    ########

lambda=0.5*sqrt(p/N)
#######2- chen    #########
func_bd=b0
n1=n
tit=max(1,ceiling(log(n1/N)/log((s+1)^2*log(N)/n1)))
fk=matrix(0,nrow=1,ncol=K)
zk=matrix(0,nrow=(p+1),ncol=K)
xigmak=matrix(0,nrow=(p+1),ncol=K)
for(qi in 1:tit)
{
   h=max(((s+1)*log(N)/N)^{1/2},(s+1)^{qi-0.5}*(log(N)/n1)^{qi/2})
   for(ki in 1:K)
   {
      y=y_al[((ki-1)*n+1):(ki*n)]
      x=x_al[((ki-1)*n+1):(ki*n),]
      u=(y-x%*%func_bd)/h
      Ka=(105/64-525/64*u^2+735/64*u^4-315/64*u^6)*(abs(u)<=1)
      fk[ki]=sum(Ka)/n/h
   }
   f=max(mean(fk),0.001)
```

```
   for(ki in 1:K)
   {
      y=y_al[((ki-1)*n+1):(ki*n)]
      x=x_al[((ki-1)*n+1):(ki*n),]
      zk[,ki]=t(x)%*%(x%*%func_bd-((y<=x%*%func_bd)-tau)/f)
      xigmak[,ki]=t(x)%*%x%*%func_bd
   }
   znn=colSums(t(zk))/N
   snn=colSums(t(xigmak))/N
   X1=snn-t(x0)%*%(x0)%*%func_bd/n1-znn
   X2=0.5*t(x0)%*%(x0)/n1
   func_bd=Func_pss(X1, X2, func_bd, lambda)
}
b_C=func_bd
######err 2- Chen    #######
######3- PQREM    ########
t7=Sys.time()
func_bd=b0
it=0
RMSE=1
while ((it<10)*(RMSE>10^(-3)))
{
   D=matrix(0,nrow=(p+1),ncol=(p+1))
   G=matrix(0,nrow=(p+1),ncol=1)
   for(ki in 1:K)
   {
      y=y_al[((ki-1)*n+1):(ki*n)]
      x=x_al[((ki-1)*n+1):(ki*n),]
      xguo=matrix(0,nrow=n,ncol=(p+1))
      absxy=matrix(0,nrow=n,ncol=1)
      abse=abs(y-x%*%func_bd)
      for(i in 1:n)
      {
         absxy[i]=max(abse[i],0.001)
```

```
            }
            for(pi in 1:(p+1))
            {
               xguo[,pi]=x[,pi]/absxy
            }
            D=t(x)%*%xguo+D
            G=t(xguo)%*%y+(2*tau-1)*colSums(x)+G
         }
         X1=-G/N
         X2=D/N/2
         func_bd1=Func_pss(X1, X2, func_bd, lambda)
         #fx=t(x0)%*%X2%*%x0+t(x0)%*%X1+lambda*sum(abs(x0)) return x1
         it=it+1
         RMSE=sqrt(sum((func_bd1-func_bd)^2))
         func_bd=func_bd1
      }
      b_DS=func_bd
      t8=Sys.time()
      t_DS[nii,Kq]=t8-t7+t_b0
      #####err 3-PQREM    #######

      RMSE_C[nii,Kq]=sqrt(sum((b_C-beta_true)^2))
      RMSE_DS[nii,Kq]=sqrt(sum((b_DS-beta_true)^2)
   }
}

for (Kq in 1:KL)
{
   K=Kall[Kq] ###分块数
   print(K)
   RMSE_mean=cbind(mean(RMSE_C[,Kq]),mean(RMSE_DS[,Kq]))
   RMSE_sd=cbind(sd(RMSE_C[,Kq]),sd(RMSE_DS[,Kq]))
   t=mean(t_DS[,Kq])
```

```
        print(round(RMSE_mean,3))
        print(round(RMSE_sd,3))
        print(round(t,3))
      }
    }
}
t12=Sys.time()
t12-t11

#实证检验
library(quantreg)
library(SparseM)
library(KernSmooth)
library(MASS)
library(stats)
library(MultiRNG)
#library(hqreg)
library(cqrReg)
library(readxl)
#setwd('C:/Users/JR/Desktop/R code')
#source('Func_pss.R')
dataallg<-read.csv("C:/Users/DELL/Desktop/YearPredictionMSD.csv")
dataallg<-na.omit(dataallg)
dataallg<- as.matrix(dataallg)
dataallg[dataallg==123456789]<-NA
dataallg<-na.omit(dataallg)
dataall<- as.matrix(dataallg)

p=90
y1<-dataall[,1]
Nall<-length(y1)
y_al<-dataall[1:500000,1]
x_alg<-dataall[1:500000,2:91]
```

```
x_al=cbind(1,x_alg)
N<-length(y_al)
ytest<-dataall[500001:Nall,1]
xtest<-cbind(1,dataall[500001:Nall,2:91])

####################
tauall=c(0.5,0.2,0.8)
Kall=c(1,10,50,100)
for (tauq in 1:3)
{
   tau=tauall[tauq]
   print(tau)
   t1=Sys.time()
   ball=rq(y_al ~ x_alg, method="lasso",tau=tau)$coef
   t2=Sys.time()
   t_ball=t2-t1
   print(t_ball)
   print(mean(abs(ytest-xtest%*%ball)))
   print(sum(abs(ball[2:(p+1)])>10^(-3)))
   for (tk in 1:4)
   {
      K=Kall[tk]
      print(K)
      n=N/K
      ####### 2- b0-L1#######
      t3=Sys.time()
      y0=y_al[1:n]
      x0=x_al[1:n,]
      x0g=x_alg[1:n,]
      b0=rq(y0 ~ x0g, method="lasso",tau=tau,lambda=0.5*sqrt(p/n))$coef
      t4=Sys.time()
      t_b0=t4-t3
      #######err 2- b0-L1#####
lambda=0.5*sqrt(p/N)
```

```
    n1=n
s=p
    #######3- chen #########
    t9=Sys.time()
    func_bd=b0
    tit=max(1,ceiling(log(n1/N)/log((s+1)^2*log(N)/n1)))
    fk=matrix(0,nrow=1,ncol=K)
    zk=matrix(0,nrow=(p+1),ncol=K)
    xigmak=matrix(0,nrow=(p+1),ncol=K)
    for(qi in 1:tit)
    {
      h=max(((s+1)*log(N)/N)^{1/2},(s+1)^{qi-0.5}*(log(N)/n1)^{qi/2})
      for(ki in 1:K)
      {
        y=y_al[((ki-1)*n+1):(ki*n)]
        x=x_al[((ki-1)*n+1):(ki*n),]
        u=(y-x%*%func_bd)/h
        Ka=(105/64-525/64*u^2+735/64*u^4-315/64*u^6)*(abs(u)<=1)
        fk[ki]=sum(Ka)/n/h
      }
      f=max(mean(fk),0.001)
      for(ki in 1:K)
      {
        y=y_al[((ki-1)*n+1):(ki*n)]
        x=x_al[((ki-1)*n+1):(ki*n),]
        zk[,ki]=t(x)%*%(x%*%func_bd-((y<=x%*%func_bd)-tau)/f)
        xigmak[,ki]=t(x)%*%x%*%func_bd
      }
      znn=colSums(t(zk))/N
      snn=colSums(t(xigmak))/N
      X1=snn-t(x0)%*%(x0)%*%func_bd/n1-znn
      X2=0.5*t(x0)%*%(x0)/n1
      func_bd=Func_pss(X1, X2, func_bd, lambda)
    }
```

```
b_C=func_bd
t10=Sys.time()
t_C=t10-t9+t_b0
######err 3- Chen ######
######4- PQREM    #####
t7=Sys.time()
func_bd=b0
it=0
RMSE=1
while ((it<10)*(RMSE>10^(-3)))
{
   D=matrix(0,nrow=(p+1),ncol=(p+1))
   G=matrix(0,nrow=(p+1),ncol=1)
   for(ki in 1:K)
   {
      y=y_al[((ki-1)*n+1):(ki*n)]
      x=x_al[((ki-1)*n+1):(ki*n),]
      xguo=matrix(0,nrow=n,ncol=(p+1))
      absxy=matrix(0,nrow=n,ncol=1)
      abse=abs(y-x%*%func_bd)
      for(i in 1:n)
      {
         absxy[i]=max(abse[i],0.001)
      }
      for(pi in 1:(p+1))
      {
         xguo[,pi]=x[,pi]/absxy
      }
      D=t(x)%*%xguo+D
      G=t(xguo)%*%y+(2*tau-1)*colSums(x)+G
   }
   X1=-G/N
   X2=D/N/2
   func_bd1=Func_pss(X1, X2, func_bd, lambda)
```

```
        #fx=t(x0)%*%X2%*%x0+t(x0)%*%X1+lambda*sum(abs(x0)) return x1
        it=it+1
        RMSE=sqrt(sum((func_bd1-func_bd)^2))
        func_bd=func_bd1
      }
b_DS=func_bd
      t8=Sys.time()
t_DS=t8-t7+t_b0
      #####err 4-PQREM    #####
      BC=mean(abs(ytest-xtest%*%b_C))
      BDS=mean(abs(ytest-xtest%*%b_DS))
      print(round(cbind(BC,BDS),3))
      print(round(cbind(t_C,t_DS),2))
      print(sum(abs(b_DS[2:(p+1)])>10^(-3)))
      print(sum(abs(b_C[2:(p+1)])>10^(-3)))
   }
}
```

致谢

窗间过马，居诸不息，距离收到研究生录取通知书已过将近两年，两年的知识积累与实践运用，让我得以在最终毕业的时候交出一份答卷。在此，要感谢两年来帮助我成长的老师与同学。

首先，感谢我的导师姜荣老师。如果说研究生的生涯是拆解问题与研究问题能力培养的重要时期，那么选择姜老师作为这段时间的引路人即是我人生很重要的幸运选择。课堂上，姜老师严谨而不失活泼，对于复杂的知识点他也可以举重若轻，生动讲解。课后，他关心我们的论文与实践情况，为我们提供好的方向与宝贵的建议。本文的完成历时六个月，这六个月离不开他的帮助与督促。

其次，感谢东华大学理学院应用统计专业的授课老师们。感谢刘晓强老师与宋晖老师的计算机基础课程带我们入门 sql 和 python 语言，为我之后的社会实践打下了硬技能的基础。感谢姜荣老师的属性数据分析课程与童金英老师的多元统计分析课程，为我解决实践问题奠定了方法论基础。感谢吴笑千老师的统计计算与软件，本文程序编写所使用的 R 语言即为您上课教授，希望毕业之后也能学习您课上对待每一个细节都认真的做事态度。感谢张振中老师的随机过程课程，虽然这门课的研究过程比较艰辛，但您对课堂的热情与风趣的讲解使我觉得再难的知识也都不是枯燥的。如果我能够在未来的工作中表现出得心应手，这离不开老师们的教导。

最后，感谢为我们的生活提供各种帮助的辅导员吕铀与张佳老师，感谢一次次论文完成节点中对论文进行审核或听我们答辩的老师们，感谢你们付出的时间与精力。愿未来的路上，我们在各自的领域都能有所成就。