Advances in Applied Mathematics 应用数学进展, 2024, 13(6), 2780-2790
Published Online June 2024 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2024.136267

Hans汉斯

求解大型线性最小二乘问题的贪婪随机坐标下降法

董　勤

温州大学数理学院，浙江　温州

收稿日期：2024年5月24日；录用日期：2024年6月19日；发布日期：2024年6月26日

摘　要

贪婪随机坐标下降法(GRCD)是求解大型线性最小二乘问题的有效迭代方法之一。本文在GRCD算法中引入松弛因子，构造了一种含参数的贪婪随机坐标下降法。并证明了当线性最小二乘问题的系数矩阵为列满秩时该方法依期望的收敛性。数值实验表明，当选取适当的松弛因子时，该算法在迭代步数和计算时间比GRCD方法更有效。

关键词

最小二乘问题，贪婪随机坐标下降法，松弛因子

A Greedy Randomized Coordinate Descent Method for Solving Large Linear Least Squares Problem

Qin Dong

College of Mathematics and Physics, Wenzhou University, Wenzhou Zhejiang

Received: May 24th, 2024; accepted: Jun. 19th, 2024; published: Jun. 26th, 2024

Abstract

The greedy randomized coordinate descent method (GRCD) is one of the effective iterative methods to solve large linear least squares problem. A greedy randomized coordinate descent method with parameters was constructed by introducing a relaxation parameter in the GRCD algorithm. It is also proved that the method has the expected convergence when the coefficient matrix of the linear least squares problem is of full column rank. Numerical experiments show that the

文章引用: 董勤. 求解大型线性最小二乘问题的贪婪随机坐标下降法[J]. 应用数学进展, 2024, 13(6): 2780-2790.
DOI: 10.12677/aam.2024.136267

proposed algorithm is more effective than the GRCD method in terms of iterative steps and calculation time when the appropriate relaxation parameter is selected.

Keywords

Least Squares Problem, The Greedy Randomized Coordinate Descent Method, Relaxation Parameter

1. 引言

对于大型线性最小二乘问题

$$f(x)=\min_{x\in R}\|b-\boldsymbol{A}x\|_2^2 \tag{1}$$

其中系数矩阵 $\boldsymbol{A}\in\boldsymbol{R}^{m\times n}$ 列满秩，$b\in R^m$。$x_*=\boldsymbol{A}^+b$，为其唯一最小范数最小二乘解，其中 $\boldsymbol{A}^+=\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{\mathrm{T}}$ 表示 $\boldsymbol{A}$ 的 Moore-Penrose 广义逆。众所周知，线性最小二乘问题在数学上等同于正规方程 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}x=\boldsymbol{A}^{\mathrm{T}}b$，也等价于无约束二次优化问题

$$\min_{x\in R^n} f(x):=\frac{1}{2}x^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}x-b^{\mathrm{T}}\boldsymbol{A}x \tag{2}$$

对于矩阵 $\boldsymbol{G}=\left(g_{ij}\right)\in\boldsymbol{R}^{m\times n}$，$\boldsymbol{G}_{(j)}$、$\|\boldsymbol{G}\|_2$ 和 $\|\boldsymbol{G}\|_F$ 分别代表矩阵的第 j 列、2 范数以及 F 范数，e_i 表示第 i 个坐标向量，$(\bullet)^{(j_k)}$ 表示第 k 次迭代向量的第 j 个元素，用 $\lambda_{\min}\left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}\right)$ 表示 $\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}$ 最小的正特征值。

最小二乘问题在岭回归[1]-[3]，计算机断层扫描[4]-[6]，机器学习[7]，生物特征选择[8]及其他领域都有广泛的应用[9]-[12]。对于最小二乘问题(1)，坐标下降法是求解这一类问题的重要方法之一，它将经典的 Gauss-Seidel 迭代方法直接应用于求解(2) [13]。在 2009 年 Strohmer 和 Vershynin [14]最先通过随机选择系数矩阵 $\boldsymbol{A}$ 的一行作为工作行提出随机 Kaczmarz 算法。2010 年 Leventhal 和 Lewis [15]受此启发，通过随机选择系数矩阵 $\boldsymbol{A}$ 的一列作为工作列提出随机坐标下降(RCD)方法。之后，随机坐标下降法被进一步推广或拓展[16]-[18]。2019 年 Bai 和 Wu [19]用了一种更有效的概率准则选取系数矩阵 $\boldsymbol{A}$ 的列，提出了贪婪随机坐标下降法(GRCD)。本文中在 GRCD 的算法分析中引入松弛因子，构造求解系统(1)的含参数的 GRCD 算法，并给出了该算法的收敛性。

2. 参数的贪婪随机坐标下降算法

Bai 和 Wu 提出了求解大型线性最小二乘问题(1)的 GRCD 方法，该算法如下。

算法 1 GRCD 算法

输入：$\boldsymbol{A},b,l$ 和 x_0

输出：x_l

1：对 $k=0,1,\cdots,l-1$ 做

2：计算

$$\delta_k=\frac{1}{2}\left(\frac{1}{\left\|\boldsymbol{A}^{\mathrm{T}}r_k\right\|_2^2}\right)\max_{1\le j\le n}\left(\frac{\left|\boldsymbol{A}_{(j)}^{\mathrm{T}}r_k\right|^2}{\left\|\boldsymbol{A}_{(j)}\right\|_2^2}+\frac{1}{\|\boldsymbol{A}\|_F^2}\right)$$

3：构造集合

$$\nu_k=\left\{j\left|\left|\boldsymbol{A}_{(j)}^{\mathrm{T}}r_k\right|^2\geq\delta_k\left\|\boldsymbol{A}^{\mathrm{T}}r_k\right\|_2^2\left\|\boldsymbol{A}_{(j)}\right\|_2^2\right.\right\}$$

4：令 $s_k=\boldsymbol{A}^{\mathrm{T}}r_k$，根据如下公式计算 $\tilde{s}_k^{(j_k)}$

$$\tilde{s}_k^{(j_k)}=\begin{cases}s_k^{(j_k)}, & j_k\in\nu_k,\\ 0, & j_k\notin\nu_k,\end{cases}$$

5：依概率 $\Pr\left(column=j_k\right)=\dfrac{\left|\tilde{s}_k^{(j_k)}\right|^2}{\left\|\tilde{s}_k\right\|_2^2}$ 选择 $j_k\in\nu_k$

6：计算 $x_{k+1}=x_k+\dfrac{s_k^{(j_k)}}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}e_{j_k}$

7：结束

本文中含参数的 GRCD 算法(GRCD(ω))是在 GRCD 算法迭代公式中添加一个参数 $\omega\in(0,2)$，见如下算法 2。

算法 2 含参数的贪婪随机坐标下降法

输入：$\boldsymbol{A},b,l,\omega$ 和 x_0

输出：x_l

1：对 $k=0,1,\cdots,l-1$ 做

2：计算

$$\delta_k=\frac{1}{2}\left(\frac{1}{\left\|\boldsymbol{A}^{\mathrm{T}}r_k\right\|_2^2}\right)\max_{1\leq j\leq n}\left(\frac{\left|\boldsymbol{A}_{(j)}^{\mathrm{T}}r_k\right|^2}{\left\|\boldsymbol{A}_{(j)}\right\|_2^2}+\frac{1}{\left\|\boldsymbol{A}\right\|_F^2}\right)$$

3：构造集合

$$\nu_k=\left\{j\left|\left|\boldsymbol{A}_{(j)}^{\mathrm{T}}r_k\right|^2\geq\delta_k\left\|\boldsymbol{A}^{\mathrm{T}}r_k\right\|_2^2\left\|\boldsymbol{A}_{(j)}\right\|_2^2\right.\right\}$$

4：令 $s_k=A^{\mathrm{T}}r_k$，根据如下公式计算 $\tilde{s}_k^{(j_k)}$

$$\tilde{s}_k^{(j_k)}=\begin{cases}s_k^{(j_k)}, & j_k\in\nu_k,\\ 0, & j_k\notin\nu_k,\end{cases}$$

5：依概率 $\Pr\left(column=j_k\right)=\dfrac{\left|\tilde{s}_k^{(j_k)}\right|^2}{\left\|\tilde{s}_k\right\|_2^2}$ 选择 $j_k\in\nu_k$

6：计算 $x_{k+1}=(1-\omega)x_k+\omega\left(x_k+\dfrac{s_k^{(j_k)}}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}e_{j_k}\right)$

7：结束

注 1：上述算法中依概率选取工作列时采用 Matlab 的 randsrc 命令。

注 2：当 $\omega=1$ 时，GRCD(ω)算法简化为 GRCD 算法。

3. GRCD(ω)算法收敛性分析

关于算法 2 有如下收敛定理。

定理 1 对于线性最小二乘问题(1)，设 $\omega\in(0,2)$。那么由算法 2 迭代生成的序列 $\{x_k\}_{k=0}^{\infty}$ 依期望指数收敛到 x_*，且解误差依期望满足

$$E_k\left\|x_{k+1}-x_*\right\|_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}^2\le\left(1-\omega(2-\omega)\frac{\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\|\boldsymbol{A}\|_F^2}\right)\left\|x_k-x_*\right\|_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}^2 \tag{3}$$

以及

$$E\left\|x_k-x_*\right\|_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}^2\le\left(1-\omega(2-\omega)\frac{\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\|\boldsymbol{A}\|_F^2}\right)^k\left\|x_0-x_*\right\|_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}^2 \tag{4}$$

这里用 E_k 表示前 k 次迭代的期望值，即 $E_k[\cdot]=E_k\left[\cdot\middle|j_0,j_1,\cdots,j_{k-1}\right]$，$j_l\,(l=0,1,\cdots,k-1)$ 表示算法在第 l 次迭代时选择的第 j_l 列，同时也易知 $E\left[E_k[\cdot]\right]=E[\cdot]$。

证明：令 $\boldsymbol{P}_{j_k}=\dfrac{\boldsymbol{A}_{(j_k)}\boldsymbol{A}_{(j_k)}^{\mathrm{T}}}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}$，则

$$\begin{aligned}\boldsymbol{A}\left(x_{k+1}-x_*\right)&=\boldsymbol{A}\left(x_k-x_*\right)+\omega\boldsymbol{A}\frac{s_k^{(j_k)}}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}e_{j_k}\\&=\boldsymbol{A}\left(x_k-x_*\right)+\omega\frac{\boldsymbol{A}\boldsymbol{A}_{(j_k)}^{\mathrm{T}}r_k}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}e_{j_k}=\boldsymbol{A}\left(x_k-x_*\right)+\omega\frac{\boldsymbol{A}_{(j_k)}^{\mathrm{T}}r_k\boldsymbol{A}_{(j_k)}}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\\&=\boldsymbol{A}\left(x_k-x_*\right)+\omega\frac{\boldsymbol{A}_{(j_k)}^{\mathrm{T}}\left(\boldsymbol{A}x_*-\boldsymbol{A}x_k\right)\boldsymbol{A}_{(j_k)}}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\\&=\boldsymbol{A}\left(x_k-x_*\right)-\omega\frac{\boldsymbol{A}_{(j_k)}\boldsymbol{A}_{(j_k)}^{\mathrm{T}}}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\boldsymbol{A}\left(x_k-x_*\right)\\&=\boldsymbol{A}\left(x_k-x_*\right)-\omega\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\end{aligned}$$

由 $\boldsymbol{P}_{j_k}$ 的定义，可知 $\boldsymbol{P}_{j_k}^{\mathrm{T}}\boldsymbol{P}_{j_k}=\boldsymbol{P}_{j_k}$，因此

$$\begin{aligned}&\left\|\boldsymbol{A}\left(x_{k+1}-x_*\right)\right\|_2^2=\left\|\boldsymbol{A}\left(x_k-x_*\right)-\omega\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2\\&=\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2-2\omega\left[\boldsymbol{A}\left(x_k-x_*\right)\right]^{\mathrm{T}}\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)+\omega^2\left[\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\right]^{\mathrm{T}}\boldsymbol{P}_{j_k}A\left(x_k-x_*\right)\\&=\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2-2\omega\left[\boldsymbol{A}\left(x_k-x_*\right)\right]^{\mathrm{T}}\boldsymbol{P}_{j_k}^{\mathrm{T}}\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)+\omega^2\left[\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\right]^{\mathrm{T}}\boldsymbol{P}_{j_k}A\left(x_k-x_*\right)\\&=\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2-2\omega\left\|\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2+\omega^2\left\|\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2\\&=\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2-\omega(2-\omega)\left\|\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2\end{aligned} \tag{5}$$

又

$$
\begin{aligned}
E_k\left\|\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2 &= E_k\left\|\frac{\boldsymbol{A}_{(j_k)}\boldsymbol{A}_{(j_k)}^{\mathrm{T}}}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2 \\
&= E_k\left\|\frac{\boldsymbol{A}_{(j_k)}}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\left|\boldsymbol{A}_{(j_k)}^{\mathrm{T}}\boldsymbol{A}\left(x_k-x_*\right)\right|\right\|_2^2 \\
&= E_k\left(\left(\frac{\left|\boldsymbol{A}_{(j_k)}^{\mathrm{T}}\boldsymbol{A}\left(x_k-x_*\right)\right|}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}A_{(j_k)}\right)^{\mathrm{T}}\left(\frac{\left|\boldsymbol{A}_{(j_k)}^{\mathrm{T}}\boldsymbol{A}\left(x_k-x_*\right)\right|}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\boldsymbol{A}_{(j_k)}\right)\right) \\
&= E_k\left(\frac{1}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\left|\boldsymbol{A}_{(j_k)}^{\mathrm{T}}\boldsymbol{A}\left(x_k-x_*\right)\right|^2\right) \\
&= \sum_{j_k=1}^{n}\frac{\left|\tilde{s}_k^{(j_k)}\right|^2}{\sum_{j\in\nu_k}\left|\tilde{s}_k^{(j)}\right|^2}\left(\frac{1}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\left|\boldsymbol{A}_{(j_k)}^{\mathrm{T}}\boldsymbol{A}\left(x_k-x_*\right)\right|^2\right) \\
&= \sum_{j_k\in\nu_k}\frac{\left|s_k^{(j_k)}\right|^2}{\sum_{j\in\nu_k}\left|s_k^{(j)}\right|^2}\left(\frac{1}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\left|\boldsymbol{A}_{(j_k)}^{\mathrm{T}}\boldsymbol{A}\left(x_k-x_*\right)\right|^2\right) \\
&= \sum_{j_k\in\nu_k}\frac{\left|s_k^{(j_k)}\right|^2}{\sum_{j\in\nu_k}\left|s_k^{(j)}\right|^2}\frac{\left|s_k^{(j_k)}\right|^2}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}
\end{aligned}
$$

由 ν_k 的定义可知，若 $j_k\in\nu_k$，则 $\left|s_k^{(j_k)}\right|^2\geq\delta_k\left\|s_k\right\|_2^2\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2$，即 $\frac{\left|s_k^{(j_k)}\right|^2}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\geq\delta_k\left\|s_k\right\|_2^2$

因此 $\sum_{j_k\in\nu_k}\frac{\left|s_k^{(j_k)}\right|^2}{\sum_{j\in\nu_k}\left|s_k^{(j)}\right|^2}\frac{\left|s_k^{(j_k)}\right|^2}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}\geq\sum_{j_k\in\nu_k}\frac{\left|s_k^{(j_k)}\right|^2}{\sum_{j\in\nu_k}\left|s_k^{(j)}\right|^2}\delta_k\left\|s_k\right\|_2^2=\delta_k\left\|s_k\right\|_2^2$

即 $E_k\left\|\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2\geq\delta_k\left\|s_k\right\|_2^2=\delta_k\left\|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2$

对(5)式左右两边求期望可得

$$
\begin{aligned}
&E_k\left\|\boldsymbol{A}\left(x_{k+1}-x_*\right)\right\|_2^2 \\
&=\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2-\omega\left(2-\omega\right)E_k\left\|\boldsymbol{P}_{j_k}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2 \\
&\leq\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2-\omega\left(2-\omega\right)\delta_k\left\|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2 \\
&\leq\left(1-\omega\left(2-\omega\right)\delta_k\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)\right)\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2
\end{aligned}
\tag{6}
$$

上式最后一步是由下面的不等式得到

$$
\left\|\boldsymbol{A}^{\mathrm{T}}u\right\|_2^2\geq\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)\left\|u\right\|_2^2,u\in R\left(\boldsymbol{A}^{\mathrm{T}}\right)
$$

又由于

$$\delta_k\|\boldsymbol{A}\|_F^2=\frac{1}{2}\left(\frac{1}{\left\|\boldsymbol{A}^{\mathrm{T}}r_k\right\|_2^2}\max_{1\le j_k\le n}\frac{\left|\boldsymbol{A}_{(j_k)}^{\mathrm{T}}r_k\right|^2}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}+\frac{1}{\|\boldsymbol{A}\|_F^2}\right)\|\boldsymbol{A}\|_F^2$$

$$=\frac{\max\limits_{1\le j_k\le n}\dfrac{\left|s_k^{(j_k)}\right|^2}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}}{2\sum\limits_{j_k=1}^{n}\dfrac{\left|s_k^{(j_k)}\right|^2}{\|\boldsymbol{A}\|_F^2}}+\frac{1}{2}=\frac{\max\limits_{1\le j_k\le n}\dfrac{\left|s_k^{(j_k)}\right|^2}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}}{2\sum\limits_{j_k=1}^{n}\dfrac{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}{\|\boldsymbol{A}\|_F^2}\dfrac{\left|s_k^{(j_k)}\right|^2}{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}}+\frac{1}{2}$$

$$\ge\frac{\max\limits_{1\le j_k\le n}\dfrac{\left|s_k^{(j_k)}\right|^2}{\left\|\boldsymbol{A}_{(j)}\right\|_2^2}}{2\sum\limits_{j_k=1}^{n}\dfrac{\left\|\boldsymbol{A}_{(j_k)}\right\|_2^2}{\|\boldsymbol{A}\|_F^2}\max\limits_{1\le i\le n}\dfrac{\left|s_k^{(i)}\right|^2}{\left\|\boldsymbol{A}_{(i)}\right\|_2^2}}+\frac{1}{2}=\frac{1}{2}+\frac{1}{2}=1$$

因此 $\delta_k\ge\dfrac{1}{\|\boldsymbol{A}\|_F^2}$， $k=0,1,\cdots$，

对(6)式进一步估计，可得到

$$E_k\left\|\boldsymbol{A}\left(x_{k+1}-x_*\right)\right\|_2^2\le\left(1-\omega\left(2-\omega\right)\delta_k\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)\right)\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2$$

$$\le\left(1-\omega\left(2-\omega\right)\frac{\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\|\boldsymbol{A}\|_F^2}\right)\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2$$

因此(3)式得证。

对上式两边求全期望可得

$$E\left\|\boldsymbol{A}\left(x_{k+1}-x_*\right)\right\|_2^2\le\left(1-\omega\left(2-\omega\right)\frac{\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\|\boldsymbol{A}\|_F^2}\right)E\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2$$

又由上式可得

$$E\left\|\boldsymbol{A}\left(x_k-x_*\right)\right\|_2^2\le\left(1-\omega\left(2-\omega\right)\frac{\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\|\boldsymbol{A}\|_F^2}\right)E\left\|\boldsymbol{A}\left(x_{k-1}-x_*\right)\right\|_2^2$$

$$\le\left(1-\omega(2-\omega)\frac{\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\|\boldsymbol{A}\|_F^2}\right)^2E\left\|\boldsymbol{A}\left(x_{k-2}-x_*\right)\right\|_2^2$$

$$\le\cdots\le\left(1-\omega\left(2-\omega\right)\frac{\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\|\boldsymbol{A}\|_F^2}\right)^kE\left\|\boldsymbol{A}\left(x_0-x_*\right)\right\|_2^2$$

即 $E\left\|x_k-x_*\right\|_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}^2\le\left(1-\omega\left(2-\omega\right)\dfrac{\lambda_{\min}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\|\boldsymbol{A}\|_F^2}\right)^kE\left\|x_0-x_*\right\|_{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}^2$

最后(4)式得证。

4. 数值实验

在本节中，我们通过数值实验来验证 GRCD(ω)方法的有效性，所有的数值结果均是使用软件MATLAB(版本 R2022b)完成的，所使用的计算机搭载 Windows10 操作系统和 16 G 内存以及 3.00 GHz 中央处理器(Intel I CoITM) i5-12500 CPU。

用 GRCD 方法和 GRCD(ω)方法求解线性最小二乘系统(1)的算法算例是由 Matlab 函数 randn 服从标准正态分布 $n(0,\ 1)$随机生成的矩阵，以及从文献中选取的 5 个具有应用背景的稀疏矩阵[20]。分别是 Worldcities、divorce、cage5、Cities、abtaha1。

GRCD(ω)和 GRCD 方法的迭代步骤表示为 IT，运行时间以秒为单位表示为 CPU。表格中 IT 和 CPU 是重复运行 50 次所需迭代步骤数和 CPU 时间的中位数，为了直观地展示 GRCD(ω)方法的优势，我们将 GRCD(ω)方法相对于 GRCD 方法的加速定义为：$\text{Speed-up}=\dfrac{\text{CPU of GRCD}}{\text{CPU of GRCD}(\omega)}$，以此表示 GRCD 相对于 GRCD($\omega$)在计算时间这一指标上的加速程度，其数值越大表明 GRCD(ω)对同一测试矩阵加速得越快。对于选取的稀疏矩阵，表 7 也给出了测试矩阵的一些基本信息，例如矩阵的条件数 cond($\boldsymbol{A}$)，稠密性 $\text{density}=\dfrac{m\times n\text{ 矩阵 }\boldsymbol{A}\text{ 中非零元素的个数}}{m\times n}$。在数值实验中，解向量 x_* 是利用 MATLAB 函数 randn(n, 1)随机生成的。当线性系统相容时，令 $b=\boldsymbol{A}x_*$。当线性系统不相容时取 $b=\boldsymbol{A}x_*-r_0$，其中 r_0 是属于 $\boldsymbol{A}^{\mathrm{T}}$ 的零空间中的非零向量，null ($\boldsymbol{A}^{\mathrm{T}}$)是利用 MATLAB 函数 null 生成的。在迭代运算中选取的初始向量为 $x_0=0$，当相对误差 $\text{RSE}=\dfrac{\|x_k-x_*\|_2^2}{\|x_*\|_2^2}\le 10^{-6}$ 或者迭代步数超过 10 万步迭代停止。

对于随机生成的矩阵，我们将线性系统相容时 GRCD 和 GRCD(ω)方法的迭代步数和计算时间列于表 1~3，线性系统不相容时列于表 4~6。从这些表中，线性系统是相容的还是不相容的，GRCD(ω)方法在迭代步数和 CPU 时间方面几乎是优于 GRCD 方法。我们观察到，当线性系统相容时，加速最大达到 1.28；当线性系统不相容时，加速最大为 1.26。

对于 Davis 和 Hu 的稀疏全秩矩阵，在表 7 和表 8 中分别列出了线性系统相容时和线性系统不相容时 GRCD 和 GRCD(ω)方法的迭代步数和计算时间。数值实验结果表明 GRCD(ω)方法在迭代步数和 CPU 时间方面都明显优于 GRCD 方法。在表 7 中，加速至少为 2.55，最大达到 5.34；在表 8 中，加速度至少为 1.98，最大达到 3.91。

Table 1. The matrix $\boldsymbol{A}$ is randomly generated, when n = 50, m is different, the numerical result of the compatible system is solved

表 1. 随机生成矩阵 $\boldsymbol{A}$ 当 n = 50，m 不同时求解相容系统的数值结果

$m\times n$		1000 × 50	2000 × 50	3000 × 50	4000 × 50	5000 × 50
ω		1.04	1.03	1.01	1.01	1.01
GRCD	IT	130.5	114	100	98	103
	CPU	0.002578	0.002676	0.002805	0.003731	0.005030
GRCD(ω)	IT	120	108	99	97	100
	CPU	0.002013	0.002365	0.002564	0.003694	0.004784
Speed-up		1.28	1.13	1.09	1.03	1.05

Table 2. The matrix $\boldsymbol{A}$ is randomly generated, when n = 100, m is different, the numerical result of the compatible system is solved
表 2. 随机生成矩阵 $\boldsymbol{A}$ 当 n = 100，m 不同时求解相容系统的数值结果

$m\times n$		1000 × 100	2000 × 100	3000 × 100	4000 × 100	5000 × 100
ω		1.09	1.03	1.03	1.01	1.02
GRCD	IT	348.5	246	239	213	216
	CPU	0.006807	0.006185	0.008076	0.010942	0.015263
GRCD(ω)	IT	277.5	236	226	210	210
	CPU	0.005704	0.005671	0.007313	0.010639	0.014583
Speed-up		1.20	1.09	1.10	1.03	1.05

Table 3. The matrix $\boldsymbol{A}$ is randomly generated, and when n = 150, m is different, the numerical result of the compatible system is solved
表 3. 随机生成矩阵 $\boldsymbol{A}$ 当 n = 150，m 不同时求解相容系统的数值结果

$m\times n$		1000 × 150	2000 × 150	3000 × 150	4000 × 150	5000 × 150
ω		1.15	1.07	1.04	1.03	1.02
GRCD	IT	612.5	423	371	365.5	338
	CPU	0.013390	0.013721	0.018007	0.029131	0.032732
GRCD(ω)	IT	476	389	354	353	325
	CPU	0.012004	0.012360	0.015663	0.027459	0.031827
Speed-up		1.12	1.11	1.15	1.06	1.03

Table 4. The matrix $\boldsymbol{A}$ is randomly generated, and when n = 50, m is different, the numerical result of the incompatible system is solved
表 4. 随机生成矩阵 $\boldsymbol{A}$ 当 n = 50，m 不同时求解不相容系统的数值结果

$m\times n$		1000 × 50	2000 × 50	3000 × 50	4000 × 50	5000 × 50
ω		1.03	1.01	1.01	1.01	1.02
GRCD	IT	122	106	105	99	100
	CPU	0.001985	0.002142	0.002708	0.003863	0.004740
GRCD(ω)	IT	118	105	103	97	96
	CPU	0.001877	0.002094	0.002662	0.003615	0.004498
Speed-up		1.06	1.02	1.02	1.07	1.05

Table 5. The matrix $\boldsymbol{A}$ is randomly generated, and when n = 100, m is different, the numerical result of the incompatible system is solved
表 5. 随机生成矩阵 $\boldsymbol{A}$ 当 n = 100，m 不同时求解不相容系统的数值结果

$m\times n$		1000 × 100	2000 × 100	3000 × 100	4000 × 100	5000 × 100
ω		1.05	1.03	1.03	1.01	1.01
GRCD	IT	325	268	242	205	210
	CPU	0.005651	0.006633	0.008443	0.010361	0.014445

续表

GRCD(ω)	IT CPU	283 0.005175	241 0.006070	226 0.007721	201 0.010205	207 0.013776
Speed-up		1.09	1.09	1.09	1.02	1.05

Table 6. The matrix $\boldsymbol{A}$ is randomly generated, and when n = 150, m is different, the numerical result of the incompatible system is solved
表 6. 随机生成矩阵 × 当 n = 150，m 不同时求解不相容系统的数值结果

$m\times n$		1000 × 150	2000 × 150	3000 × 150	4000 × 150	5000 × 150
ω		1.1	1.08	1.04	1.03	1.02
GRCD	IT CPU	602 0.012388	457 0.013413	386 0.016221	358 0.024975	333 0.322180
GRCD(ω)	IT CPU	475 0.009764	400 0.011613	353 0.015425	340 0.023833	322 0.030974
Speed-up		1.26	1.16	1.05	1.05	1.04

Table 7. The GRCD(ω) and GRCD methods solve the numerical results of the compatible system in the specific example matrix
表 7. GRCD(ω)和 GRCD 两种方法在具体算例矩阵时求解相容系统的数值结果

名称		Worldcities	divorce	cage5	Cities	abtaha1
$m\times n$		315 × 100	50 × 99	37 × 37	55 × 46	14596 × 209
density		23.87%	50.00%	17.02%	50.34%	1.68%
Cond($\boldsymbol{A}$)		66.00	19.39	15.42	207.15	12.23
ω		1.8	1.8	1.6	1.8	1.7
GRCD	IT CPU	3834.5 0.067084	594 0.005716	2235 0.022035	60,142 0.669865	10,342 0.554940
GRCD(ω)	IT CPU	1185.5 0.020597	162.5 0.001580	760 0.007437	11,260 0.123660	4063.5 0.554940
Speed-up		3.26	3.66	2.94	5.34	2.55

Table 8. The GRCD(ω) and GRCD methods solve the numerical results of the incompatible system in the specific example matrix
表 8. GRCD(ω)和 GRCD 两种方法在具体算例矩阵时求解不相容系统的数值结果

名称	Worldcities	divorce	Cities	abtaha1
$m\times n$	315 × 100	50 × 99	55 × 46	14,596 × 209
density	23.87%	50.00%	50.34%	1.68%
Cond($\boldsymbol{A}$)	66.00	19.39	207.15	12.23
ω	1.8	1.8	1.8	1.7

续表

GRCD	IT CPU	4568 0.077293	601 0.005446	46,131 0.495157	13816.5 0.710776
GRCD(ω)	IT CPU	1267 0.021641	173.5 0.002749	11,973 0.130001	4283 0.228498
Speed-up		3.57	1.98	3.91	3.11

5. 总结

本文在 GRCD 算法的基础上提出了 GRCD(ω)算法，分析了算法的收敛性。数值实验结果表明，在选择适当的参数 ω 后，GRCD(ω)算法在迭代步数和计算时间方面明显优于 GRCD 算法。

参考文献

[1] Duan, L.-X. and Zhang, G.-F. (2021) Variant of Greedy Randomized Gauss-Seidel Method for Ridge Regression. *Numerical Mathematics*: *Theory, Methods and Applications*, **14**, 714-737. https://doi.org/10.4208/nmtma.oa-2020-0095

[2] Hefny, A., Needell, D. and Ramdas, A. (2017) Rows versus Columns: Randomized Kaczmarz or Gauss-Seidel for Ridge Regression. *SIAM Journal on Scientific Computing*, **39**, S528-S542. https://doi.org/10.1137/16m1077891

[3] Liu, Y. and Gu, C. (2019) Variant of Greedy Randomized Kaczmarz for Ridge Regression. *Applied Numerical Mathematics*, **143**, 223-246. https://doi.org/10.1016/j.apnum.2019.04.008

[4] Byrne, C. (2003) A Unified Treatment of Some Iterative Algorithms in Signal Processing and Image Reconstruction. *Inverse Problems*, **20**, 103-120. https://doi.org/10.1088/0266-5611/20/1/006

[5] Bouman, C.A. and Sauer, K. (1996) A Unified Approach to Statistical Tomography Using Coordinate Descent Optimization. *IEEE Transactions on Image Processing*, **5**, 480-492. https://doi.org/10.1109/83.491321

[6] Ye, J.C., Webb, K.J., Bouman, C.A. and Millane, R.P. (1999) Optical Diffusion Tomography by Iterative-Coordinate-Descent Optimization in a Bayesian Framework. *Journal of the Optical Society of America A*, **16**, 2400-2412. https://doi.org/10.1364/josaa.16.002400

[7] Chang, K.W., Hsieh, C.J. and Lin, C.J. (2008) Coordinate Descent Method for Large-Scale L2-Loss Linear Support Vector Machines. *Journal of Machine Learning Research*, **9**, 1369-1398.

[8] Breheny, P. and Huang, J. (2011) Coordinate Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection. *The Annals of Applied Statistics*, **5**, 232-253. https://doi.org/10.1214/10-aoas388

[9] Canutescu, A.A. and Dunbrack, R.L. (2003) Cyclic Coordinate Descent: A Robotics Algorithm for Protein Loop Closure. *Protein Science*, **12**, 963-972. https://doi.org/10.1110/ps.0242703

[10] Elad, M., Matalon, B. and Zibulevsky, M. (2007) Coordinate and Subspace Optimization Methods for Linear Least Squares with Non-Quadratic Regularization. *Applied and Computational Harmonic Analysis*, **23**, 346-367. https://doi.org/10.1016/j.acha.2007.02.002

[11] Scott, J.A. and Tůma, M. (2019) Sparse Stretching for Solving Sparse-Dense Linear Least-Squares Problems. *SIAM Journal on Scientific Computing*, **41**, A1604-A1625. https://doi.org/10.1137/18m1181353

[12] Scott, J. and Tůma, M. (2022) Solving Large Linear Least Squares Problems with Linear Equality Constraints. *BIT Numerical Mathematics*, **62**, 1765-1787. https://doi.org/10.1007/s10543-022-00930-2

[13] Ruhe, A. (1983) Numerical Aspects of Gram-Schmidt Orthogonalization of Vectors. *Linear Algebra and Its Applications*, **52**, 591-601. https://doi.org/10.1016/0024-3795(83)80037-8

[14] Strohmer, T. and Vershynin, R. (2008) A Randomized Kaczmarz Algorithm with Exponential Convergence. *Journal of Fourier Analysis and Applications*, **15**, 262-278. https://doi.org/10.1007/s00041-008-9030-4

[15] Leventhal, D. and Lewis, A.S. (2010) Randomized Methods for Linear Constraints: Convergence Rates and Conditioning. *Mathematics of Operations Research*, **35**, 641-654. https://doi.org/10.1287/moor.1100.0456

[16] Gower, R.M. and Richtárik, P. (2015) Randomized Iterative Methods for Linear Systems. *SIAM Journal on Matrix Analysis and Applications*, **36**, 1660-1690. https://doi.org/10.1137/15m1025487

[17] Liu, Y., Jiang, X. and Gu, C. (2021) On Maximum Residual Block and Two-Step Gauss-Seidel Algorithms for Linear Least-Squares Problems. *Calcolo*, **58**, Article No. 13. https://doi.org/10.1007/s10092-021-00404-x

[18] Du, K. and Sun, X. (2021) A Doubly Stochastic Block Gauss-Seidel Algorithm for Solving Linear Equations. *Applied*

Mathematics and Computation, **408**, Article 126373. https://doi.org/10.1016/j.amc.2021.126373

[19] Bai, Z. and Wu, W. (2019) On Greedy Randomized Coordinate Descent Methods for Solving Large Linear Least-Squares Problems. *Numerical Linear Algebra with Applications*, **26**, e2237. https://doi.org/10.1002/nla.2237

[20] Davis, T.A. and Hu, Y. (2011) The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software*, **38**, Article No. 1. https://doi.org/10.1145/2049662.2049663