

凸优化第二次上机报告

戴悦浩 (1800010660@pku.edu.cn)

2020 年 12 月 1 日

摘要

本次报告的主要内容是使用次梯度算法和光滑化梯度法求解优化问题1.0.1。通过数值实验，我们实现的算法精度可以接近 cvx 求解器的求解水平，在求解速度方面平均比 cvxmosek 快 30% 到 50% 左右。

1 问题重述

考虑组 LASSO 问题

$$\min_{x \in \mathbb{R}^{n \times l}} \frac{1}{2} \|Ax - b\|_F^2 + \mu \|x\|_{1,2} \quad (1.0.1)$$

其中 $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^{m \times l}$ 且 μ 是给定的正数, 这里

$$\|x\|_{1,2} = \sum_{i=1}^n \|x(i, 1:l)\|_2,$$

其中 $\|x(i, 1:l)\|_2$ 表示 x 的第 i 行。

2 次梯度方法

由于原问题1.0.1是凸函数, 且定义域没有约束, 因此次梯度存在. 记原问题的目标函数为 $f(x)$, 则对其的次梯度下降算法描述如下:

算法 1 次梯度算法

输入: 初始点 $x_0, k = 0$

输出: 最优点 x^*

- 1: **while** 未达到终止条件 **do**
 - 2: 选定步长 t_k
 - 3: 更新 $x_{k+1} = x_k - t_k \nabla f(x_k)$
 - 4: **end while**
 - 5: **return** $x^* = \arg \min_{x_k} f(x_k)$
-

对于原问题1.0.1中的非光滑部分 $\|x\|_{1,2}$, 可以求得其次微分为

$$\partial \|x\|_{1,2} = \begin{pmatrix} h(x(1, 1:l), x_{11}) & \cdots & h(x(1, 1:l), x_{1l}) \\ \vdots & & \vdots \\ h(x(n, 1:l), x_{n1}) & \cdots & h(x(n, 1:l), x_{nl}) \end{pmatrix}$$

其中

$$h(x(i, 1:l), x_{ij}) = \begin{cases} \frac{x_{ij}}{\|(x(i, 1:l))\|_2}, & \|(x(i, 1:l))\|_2 > 0 \\ g, & \|g\|_2 \leq 1, \quad \|(x(i, 1:l))\|_2 = 0 \end{cases}$$

2.1 连续化次梯度算法

由于 LASSO 问题的解一般都是稀疏的, 本报告考虑连续化策略, 首先从较大的对非光滑部分的惩罚开始求解问题, 尽快收敛到较为稀疏的解, 然后逐渐缩小惩罚系数到原问题的系数, 从而加速收敛. 通过实验, 增大惩罚系数后的收敛速度非常快, 因此求解的时候针对不同的 μ 应该设置不同的最大迭代轮数. 本报告中实现的算法在算法 2 中描述.

算法 2 连续化策略次梯度算法

输入: 初始点 $x_0, k = 0, c = 0$, 惩罚因子 μ_0 , 默认每轮最大迭代步数 $\text{maxiter} = 300$, 最终轮迭代步数倍数为 $\text{most} = 3.5$

输出: 最优点 x^*

```

初始化惩罚因子  $\mu = 10000\mu_0$ , 固定步长  $\alpha = 1/\lambda_{\max}(A^T A)$ 
2: while  $\mu \geq \mu_0$  do
    if  $\mu > \mu_0$  then
        4:  $c = 0$ 
        while 迭代步数  $c < \text{maxiter}$  do
            6: 固定步长  $t_k = \alpha$  使用次梯度下降法更新  $x_{k+1} = x_k - t_k g_k^T$ 
                if  $|f(x_{k+1}) - f(x_{k-70})|/|f(x_{k-70})| < 1E - 12$  then
                    8: 退出循环
                    end if
            10:  $k \leftarrow k + 1, c = c + 1$ 
            end while
        12: else
             $c = 0$ 
        14: while 迭代步数  $c < \text{most} * \text{maxiter}$  do
            取步长  $t_k = 300\alpha / \max\{c, 300\}$ , 使用次梯度下降法更新  $x_{k+1} = x_k - t_k g_k^T$ 
            16: if  $|f(x_{k+1}) - f(x_k)|/|f(x_k)| < 1E - 12$  then
                18: 退出循环
                end if
            20:  $k \leftarrow k + 1, c \leftarrow c + 1$ 
            end while
        22: end if
        end while
    24: return  $x^* = \arg \min_{x_k} f(x_k)$ 

```

3 光滑化梯度法

光滑化的思想是用一个可微函数近似原问题1.0.1中的不可微部分, 得到一个近似问题, 通过求解近似问题的解去逼近原问题的解. 本报告中采用类似 Huber 光滑化的方法, 对于向量二范数以及 $\sigma > 0$, 定义

$$H'_\sigma(x) = \begin{cases} \frac{1}{2\sigma} \|x\|_2^2, & \|x\|_2 \leq \sigma \\ \|x\|_2 - \frac{\sigma}{2}, & \|x\|_2 > \sigma \end{cases}$$

显然 H'_σ 是向量二范数的可微近似, 且 σ 越小近似精度越高. 于是可以得到 $n \times l$ 矩阵 $l_{1,2}$ 范数的可微近似

$$H_\sigma(x) = \sum_{i=1}^n H'_\sigma(x(i, 1:l)),$$

并且可得

$$\frac{\partial H'_\sigma(x)}{\partial x_i} = \begin{cases} \frac{x_i}{\sigma}, & \|x\|_2 \leq \sigma \\ \frac{x_i}{\|x\|_2}, & \|x\|_2 > \sigma \end{cases}$$

于是

$$\frac{\partial H_\sigma(x)}{\partial x_{ij}} = \begin{cases} \frac{x_{ij}}{\sigma}, & \|x(i, 1 : l)\|_2 \leq \sigma \\ \frac{x_{ij}}{\|x(i, 1 : l)\|_2}, & \|x(i, 1 : l)\|_2 > \sigma \end{cases}$$

本报告采用的光滑化梯度法也采用连续化策略，以加速收敛。带连续化策略的光滑化梯度法在算法 3 中描述。

算法 3 连续化策略光滑化梯度算法

输入: 初始点 $x_0, k = 0, c = 0$, 惩罚因子 μ_0 , 默认每轮最大迭代步数 maxiter = 300, 最终轮迭代步数倍数为 most = 3.5, 近似系数 $\sigma = 1E^{-7}$

输出: 最优点 x^*

初始化惩罚因子 $\mu = 10000\mu_0$, 固定步长 $\alpha = 1/\lambda_{\max}(A^T A)$

while $\mu \geq \mu_0$ **do**

3: **if** $\mu > \mu_0$ **then**

$c = 0$

while 迭代步数 $c < \text{maxiter}$ **do**

6: 固定步长 $t_k = \alpha$, 对光滑化近似问题使用梯度下降法更新 $x_{k+1} = x_k - t_k g_k^T$

if $|f(x_{k+1}) - f(x_{k-70})|/|f(x_{k-70})| < 1E - 12$ **then**

 退出循环

9: **end if**

$k \leftarrow k + 1, c = c + 1$

end while

12: **else**

$c = 0$

while 迭代步数 $c < \text{most} * \text{maxiter}$ **do**

15: 取步长 $t_k = 300\alpha / \max\{c, 300\}$, 对光滑化近似问题使用梯度下降法更新 $x_{k+1} = x_k - t_k g_k^T$

if $|f(x_{k+1}) - f(x_k)|/|f(x_k)| < 1E - 12$ **then**

 退出循环

18: **end if**

$k \leftarrow k + 1, c \leftarrow c + 1$

end while

21: $\mu = \mu/10$

end if

end while

24: **return** $x^* = \arg \min_{x_k} f(x_k)$

3.1 BB 步长梯度下降

作为一个尝试，我们也实现 BB 步长的 Hongchao and Hagger's 方法，¹ 同样采用连续化策略。主要的步骤算法 4 中描述。

¹Hongchao and Hagger's 方法在课堂课件中有所描述，本报告不赘述

算法 4 Hongchao and Hagger' s method

输入: 初始点 $x_0, k = 0, c = 0$, 惩罚因子 μ_0 , 默认每轮最大迭代步数 $\text{maxiter} = 300$, 最终轮迭代步数倍数为 $\text{most} = 3.5$, 近似系数 $\sigma = 1E^{-7}$

输出: 最优点 x^*

初始化惩罚因子 $\mu = 5E7\mu_0$, 固定步长 $\alpha = 1/\lambda_{\max}(A^T A)$

while $\mu > \mu_0$ **do**

 使用 Hongchao and Hagger' s method 求解对应的优化问题

4: $\mu = \max\{\mu/10, \mu_0\}$

end while

使用 Hongchao and Hagger' s method 求解原问题

return $x^* = \arg \min_{x_k} f(x_k)$

4 数值实验及结果分析

4.1 算法稳定性

我们使用默认的随机种子解原问题1.0.1, 同时使用不同的随机种子进行试验, 以验证算法有效性.

图 1: 不同算法收敛曲线

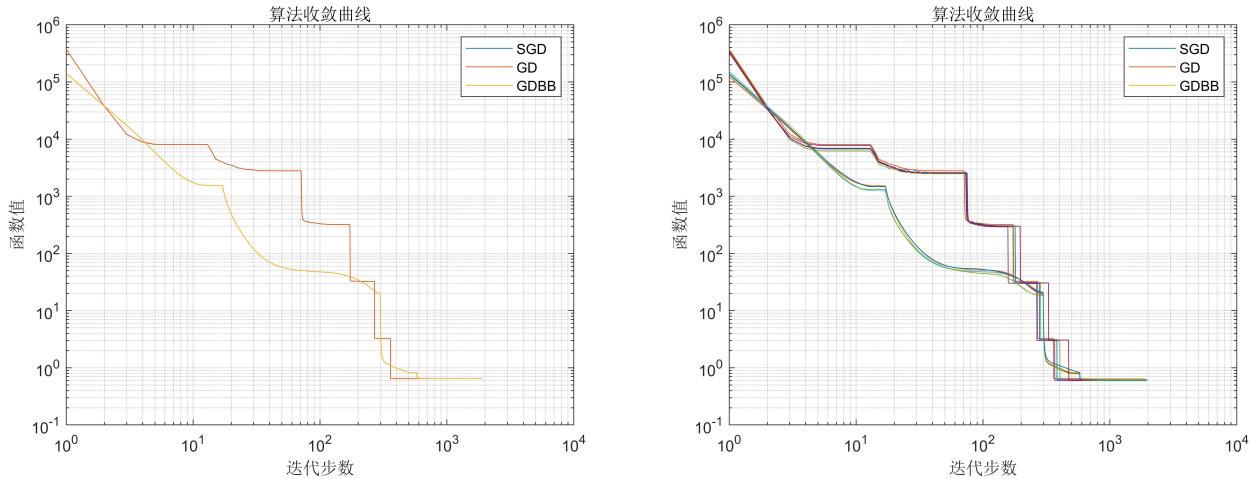
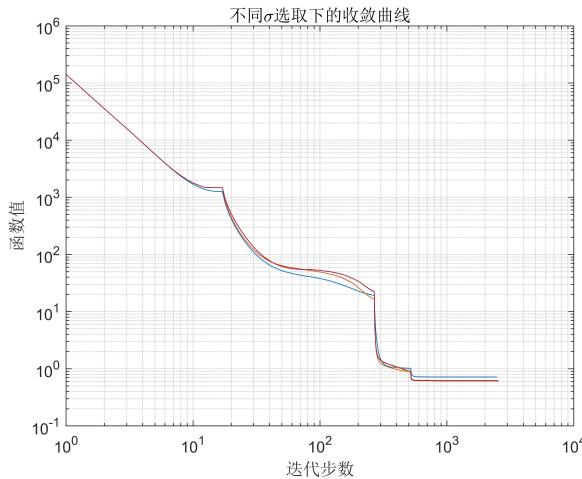
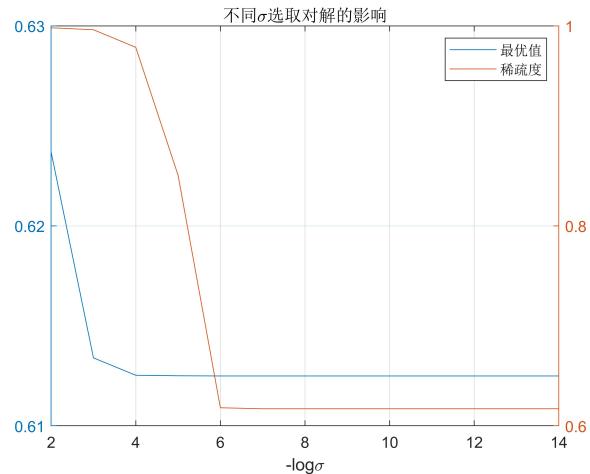


图 2: (左) 不同 σ 选取对收敛的影响



(右) 不同 σ 选取对最优值的影响



根据图 1 可知对于不同的数据, 不同的算法的收敛曲线变化不大, 说明本报告实现的算法具有较好的稳定性.

光滑化梯度法中的近似参数 σ 的选取对求解效率和精度有较大影响。通过我们采取不同的 σ 值对算法 3 进行数值实验得到的结果(图 2)可见, 近似参数 σ 的选取需要足够小才能保证解的稀疏性, 在本问题中 σ 至少需要小于 $1E-6$ 才能得到良好的稀疏性。另一方面显然 σ 并不是越小越好, 因为当 σ 趋于 0 的时候, 光滑化问题就趋向于原问题, 于是光滑化算法在接近不可微点时的光滑性质就不再能保持。从另一个角度说, 过小的 σ 也将使得光滑问题的梯度李普希兹常数变大, 于是相对次梯度下降的提升就会减少。由这些考虑, 本报告实现的算法中默认的 σ 为 $1E-7$, 后面的结果分析表明这个值已经能得到较好的效果。

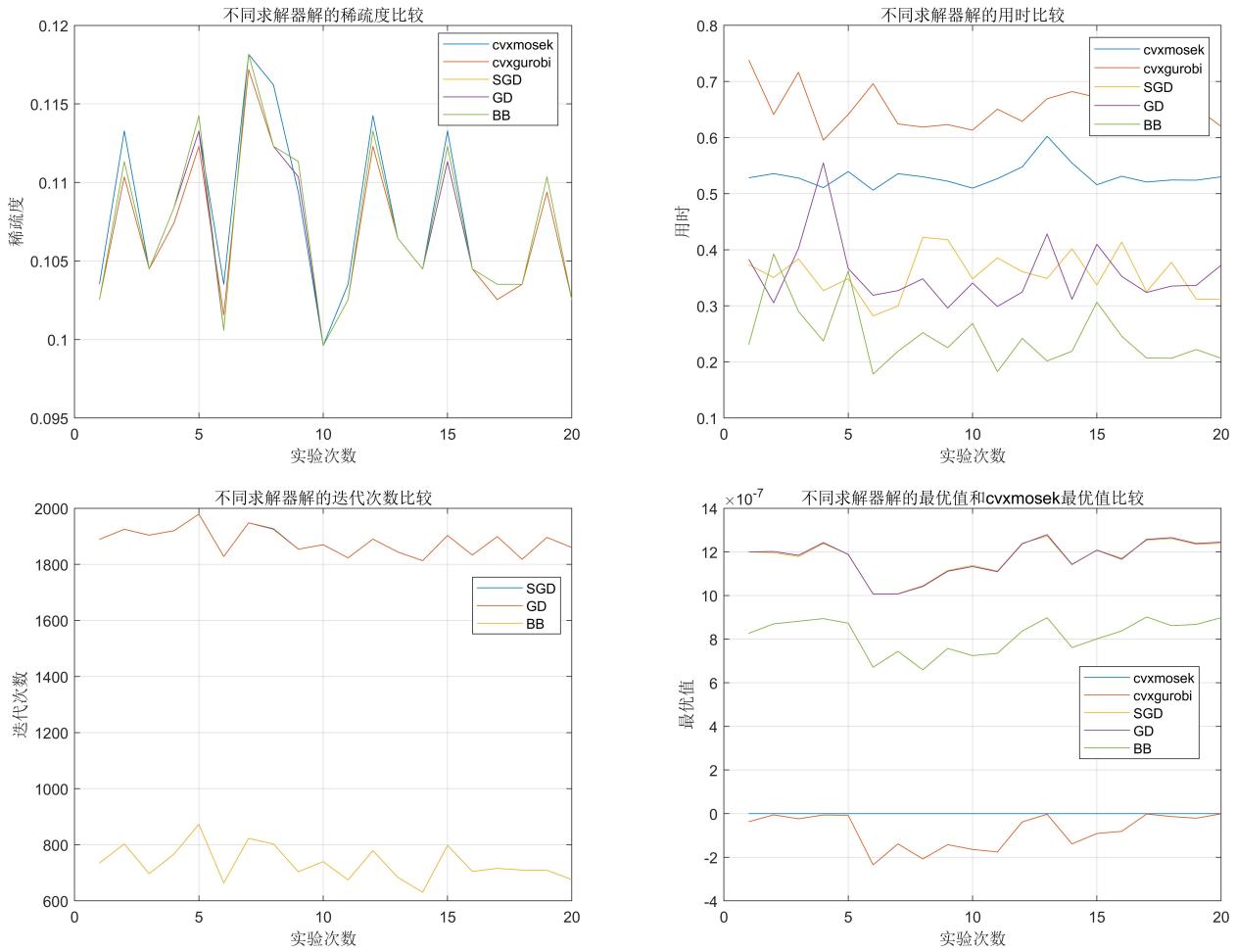
4.2 与主流优化软件比较

将各算法中表现最好的参数作为算法参数设置, 和 cvx 使用 mosek 和 gurobi 软件进行求解的结果进行对比, 展现在表 1 中²。另外也指定不同的随机种子进行随机试验, 将实验结果呈现在图 3 中。

表 1: 不同求解方式效率比较

Solver	Time	OptVal	Err-To-Exact	Iter	Sparsity	Err-To-Cvx-Mosek	Err-To-Cvx-Gurobi
cvx_mosek	0.54	5.80556E-01	3.78E-05	11	0.105	0.00E+00	4.57E-07
cvx_gurobi	0.64	5.80556E-01	3.77E-05	14	0.103	4.57E-07	0.00E+00
SGD_primal	0.38	5.80557E-01	3.81E-05	1904	0.104	8.64E-07	8.84E-07
GD_primal	0.41	5.80557E-01	3.81E-05	1904	0.104	8.64E-07	8.84E-07
GD_primal(BB)	0.19	5.80557E-01	3.91E-05	704	0.104	1.66E-06	1.78E-06

图 3: 不同求解器比较



²所有程序在 cpu 为 intel i7-9750H 的机器上运行

表 2: 不同求解方式平均效率比较

Solver	Ave Time	Err-To-Exact	Ave Sparsity
cvx_mosek	0.5254	3.7300E-05	0.1077
cvx_gurobi	0.6312	3.7129E-05	0.1069
SGD_primal	0.3563	3.7790E-05	0.1072
GD_primal	0.3569	3.7781E-05	0.1072
GD_primal(BB)	0.2448	3.8703E-05	0.1073

根据表 1 可见带有连续化策略的次梯度下降法和光滑化梯度法的求解速度高于 cvxmosek, 并且解的稀疏度也低于 cvxmosek, 也就是对解的恢复更好. 最优值方面和 cvxmosek 相当.

在求解速度方面, 本报告实现的次梯度下降算法和光滑化算法的求解速度相比 cvxmosek 平均要快 30% 左右, 采用 BB 步长的光滑化算法比 cvxmosek 快 50% 左右.

在解的恢复方面, 本报告实现的所算法的平均稀疏性比 cvxmosek 更低, 因此对解的恢复效果更好.

在最优值方面, 本报告实现的次梯度下降算法和光滑化算法求得的最优值相比 mosek 平均大 1E-7 左右, 在迭代末期, 为了达到更高的精度需要迭代的步数非常多. 数值实验表明, 为了达到和 cvxmosek 相近的稀疏性, 只需要迭代 1500 步左右. 使用了 BB 步长的 Hongchao and Hagger's 方法的迭代次数显著少于固定步长与消失步长的次梯度下降以及光滑化梯度下降, 但是线搜索需要的时间会部分抵消迭代步数的减少. 另一方面使用 BB 步长得到的解的性质相比稍差, 体现在平均的稀疏性以及和精确解的距离稍差, 但是函数最优值比固定步长稍好.