最优化方法程序作业 Report

王晨宇 2100010654

Author

王晨宇

Date

2023 年 12 月

Contents

1 综述

本篇文章主要是对文再文老师所开授的最优化方法课程中，期末程序作业所对应的报告.

本次作业主要围绕下述 Group Lasso 优化问题进行展开.

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

其中 $A\in\mathbb{R}^{m\times n}$, $b\in\mathbb{R}^{m\times l}$, 以及 $\mu>0$ 为给定的常数. 在本次作业中，我们指定 $n=512$, $m=256$, $l=2$.

接下来，本次作业借助 Matlab 编译器，分别尝试使用了如下方法该问题进行优化.

首先，为设置目标与对比，尝试调用专业的优化求解器 mosek 与 gurobi 进行求解。

1. 调用 Matlab 中 CVX 工具包的 mosek 求解器进行求解.
2. 调用 Matlab 中 CVX 工具包的 gurobi 求解器进行求解.
3. 直接调用 mosek 求解器进行求解.
4. 直接调用 gurobi 求解器进行求解.

之后，尝试根据现有算法自行设置求解器与求解过程，使用了如下算法.

1. 次梯度下降算法求解原问题.
2. 邻近算子梯度法求解原问题.
3. 快速邻近算子梯度法求解原问题.
4. 增广拉格朗日函数法求解对偶问题.
5. 交替方向乘子法求解对偶问题.
6. 增广拉格朗日函数法求解线性化原问题.

接下来，我将一一介绍每个算法以及其在 Group Lasso 问题中的实现细节.

2 算法细节与代码实现

2.1 CVX 工具包中的 mosek 与 gurobi 求解器

Matlab 中的 CVX 工具包较为智能，在安装好 mosek 与 gurobi 求解器之后，不需要过多改变原问题的形式，即可完成优化求解.

求解过后经过对比，可以发现，在这个问题中收敛速度以及精确度方面，mosek 求解器要优于 gurobi 求解器.

2.2 直接调用 mosek 与 gurobi 求解器

在直接调用 mosek 与 gurobi 求解器之前，我们需要将原问题转化为一个标准的锥优化问题. 引入新变量 $y_k, k = 1, 2, \cdots, n$, $z_{p,q}, p = 1, 2, \cdots, m, q = 1, 2$ 及约束:

$$y_k \geq \|x(k,:)\|_2, k = 1, 2, \cdots, n$$

$$z_{p,q} \geq ((Ax - b)(p,q))^2, p = 1, 2, \cdots, m, q = 1, 2$$

由此，目标函数变成了关于变量 y, z 的线性函数，而约束即可转化为锥约束.

将约束按照 mosek 与 gurobi 求解器的格式进行代入，即可完成优化求解. 与通过 CVX 调用求解器的结果类似，可以发现 mosek 求解器的性能在这个问题上要优于 gurobi 求解器.

2.3 次梯度下降算法求解原问题

次梯度下降算法较为基础. 原问题可看作无约束优化问题，且 $\|Ax-b\|_F^2$ 可微，$\mu\|x\|_{1,2}$ 不可微. 因此我们需要计算出 $\|x\|_{1,2}$ 的次梯度.

由于

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

而对于一个向量 α, $\|\alpha\|_{1,2}$ 的次梯度可以表示为

$$\partial\|\alpha\|_2 = \begin{cases} \{\frac{\alpha}{\|\alpha\|_2}\}, & \text{if } \|\alpha\|_2 > 0 \\ \{\beta | \|\beta\|_2 \ \leq 1\}, & \text{if } \|\alpha\|_2 = 0 \end{cases}$$

在实际代码实践中，很难做到判断某一个数是否精确等于 0，因此我们设定 $opts.thres = 1e-5$, 当某一个实数的绝对值取值小于 1e-5 的时候，可

以将其认为为 0. 在本作业中，若某个向量的 2 范数小于 1e-5, 那么我们取它的次梯度为 0 向量.

在步长选取时，由于实验发现取固定步长会导致收敛速度很慢，因此在此作业中我采用了如下步长:

$$\alpha_i = \frac{f(x^i) - f^*}{\|g^i\|_2^2}$$

其中 $f(x^i)$ 为当前函数值, f^* 为先验的函数最优值, g^i 为当前点处的次梯度向量. 先验的函数最优值即位上一问中所解得的结果，设为 $f^* = 0.580556$.

因此, 迭代公式为 $x^{i+1} = x^i - \alpha_i g^i$. 停机条件设置为 $|f(x^i) - f^*| < 1e - 5$. 此时只需要 7000 步迭代.

2.4 邻近算子梯度法求解原问题

在邻近算子梯度法的每一步迭代中，若步长为 α_i, 则迭代法则即位

$$x^{i+1} = prox_{\mu\alpha_i\|\cdot\|_{1,2}}(x^i - \alpha_i A^T(Ax^i - b))$$

因此最重要的即为计算函数 $prox_{\mu\alpha_i\|\cdot\|_{1,2}}(x)$. 我们有

$$prox_{\mu\alpha_i\|\cdot\|_{1,2}}(x)(i,:) = \begin{cases} 0, & \text{if } \|x(i,:)\|_2 \le \mu\alpha_i \\ x(1,:)(1 - \frac{\mu\alpha_i}{\|x(i,:)\|_2}), & \text{if } \|x(i,:)\|_2 > \mu\alpha_i \end{cases}$$

再选取步长时，我们采用 BB 步长进行迭代. 令 $s^k = x^{k+1} - x^k$, $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$. 则我们定义如下两种 BB 步长的拓展版本.

$$\alpha_k^1 = \frac{\|(s^k)^T s^k\|_2}{\|(s^k)^T y^k\|_2}$$

$$\alpha_k^2 = \frac{\|(s^k)^T y^k\|_2}{\|(y^k)^T y^k\|_2}$$

当 k 为奇数时，选取步长 α_k^1; 当 k 为偶数时，选取步长 α_k^2. 如何在计算步长时发现分母非常接近零，则将此时的步长选为固定步长，以防报错.

2.5 快速邻近算子梯度法求解原问题

在快速邻近算子梯度法中，我的步长选取规则与邻近算子计算规则与上一个算法相同. 与邻近算子梯度法的区别在于，快速邻近算子梯度法采用了如下迭代法则.

$$y^{k+1} = x^k + \frac{k-1}{k+2}(x^k - x^{k-1})$$
$$x^{k+1} = prox_{\mu\alpha_k\|\cdot\|_{1,2}}(y^k - \alpha_k A^T(Ay^k - b))$$

实验显示，FISTA 算法确实也有着更加快速的收敛速度与迭代步数.

2.6 增广拉格朗日函数法求解对偶问题

首先分析出原问题的对偶问题的具体形式.
令 $Z = Ax - b$, 则原问题转化为:

$$\min_{x\in\mathbb{R}^{n\times l}} \frac{1}{2}\|Z\|_F^2 + \mu\|x\|_{1,2} \text{ s.t. } Ax - b = Z$$

引入拉格朗日乘子 $L \in \mathbb{R}^{m\times l}$, 则有

$$L(x, Z, L) = \frac{1}{2}\|Z\|_F^2 + \mu\|x\|_{1,2} + \langle L, Ax - b - Z\rangle$$

对 x 与 Z 求极小值，则我们可得到如下形式的对偶问题:

$$\min_{L\in\mathbb{R}^{m\times l}} \frac{1}{2}\|L\|_F^2 + \langle L, b\rangle$$
$$\text{s.t. } \|A^T L\|_{\infty,2} \leq \mu$$

再进行变形，得到

$$\min_{L\in\mathbb{R}^{m\times l}} \frac{1}{2}\|L\|_F^2 + \langle L, b\rangle$$
$$\text{s.t. } S = A^T L$$
$$\|S\|_{\infty,2} \leq \mu$$

保留最后一个约束，引入拉格朗日乘子，构建罚函数:

$$L_\sigma(L, S, x) = \frac{1}{2}\|L\|_F^2 + \langle L, b\rangle + \langle x, A^T L - S\rangle + \frac{\sigma}{2}\|A^T L - S\|_F^2$$
$$\text{s.t. } \|S\|_{\infty,2} \leq \mu$$

上述问题的每次迭代分为两步, 第一步同时选择 L 与 S 使得原函数取得最小值，第二步更新拉格朗日乘子 x.

$$(L^{k+1}, \bar{S}^{k+1}) = argmin_{L\in\mathbb{R}^{m\times l}, S\in\mathbb{R}^{n\times l}} L_\sigma(L^k, S^k, x^k)$$

$$S^{k+1} = project(S^k)$$

$$x^{k+1} = x^k + \sigma(A^T L - S)$$

最终选取 $\sigma = 1$, 停机条件为 $|f(x^i) - f^*| < 1e-5$.

2.7 交替方向乘子法求解对偶问题

ADMM 与 ALM 算法的理论基础相似，不同的地方在于 ADMM 遵循如下的交替迭代法则:

$$L^{k+1} = argmin_{L\in\mathbb{R}^{m\times l}} L_\sigma(L^k, S^k, x^k)$$

$$\bar{S}^{k+1} = argmin_{S\in\mathbb{R}^{n\times l}} L_\sigma(L^{k+1}, S^k, x^k)$$

$$S^{k+1} = project(S^k)$$

$$x^{k+1} = x^k + \sigma(A^T L - S)$$

最终依旧选取 $\sigma = 1$, 停机条件为 $|f(x^i) - f^*| < 1e-5$.

2.8 增广拉格朗日函数法求解线性化原问题

$$\min_{x\in\mathbb{R}^{n\times l}} \frac{1}{2}\|Z\|_F^2 + \mu\|x\|_{1,2} \text{ s.t. } Ax - b = Z$$

在对原问题直接应用 ALM 算法时，由于存在不可微项，难以准确更新变量 x 与 Z, 因此对原问题采用线性化的方法，近似求得变量 x 与 Z 的最优取值.

设线性化之后的增广拉格朗日函数为 $L_\sigma(x, Z, L)$, 则有迭代法则:

$$(x^{k+1}, Z^{k+1}) = argmin_{x\in\mathbb{R}^{n\times l}, Z\in\mathbb{R}^{m\times l}} L_\sigma(x^k, Z^k, L^k)$$

$$L^{k+1} = L^k + \sigma(Ax - b - Z)$$

在本问题中，设置 $\sigma = 0.01$, 停机条件为 $|f(x^i) - f^*| < 1e-5$.

3 优化结果

参照文老师提供的'Test_group_lasso.m' 代码，我也添加了一些其他功能，本次作业所实现的代码性能如下表所示:

Table 1: 不同算法下优化结果的表现

Method	CPU (s)	Iter	Optval	Sparsity	Err-to-exact
CVX-Mosek	2.93	NaN	5.80569×10^{-1}	0.100	1.22×10^{-9}
CVX-Gurobi	11.20	NaN	5.80569×10^{-1}	0.100	8.54×10^{-8}
Mosek	0.72	10	5.80556×10^{-1}	0.103	3.75×10^{-5}
Gurobi	2.98	13	5.80558×10^{-1}	0.117	3.91×10^{-5}
SGD Primal	3.91	7667	5.80566×10^{-1}	0.183	8.30×10^{-5}
ProxGD Primal	3.52	15548	5.80564×10^{-1}	0.164	6.90×10^{-5}
FProxGD Primal	2.51	11228	5.80566×10^{-1}	0.119	4.75×10^{-5}
ALM Dual	0.83	270	5.80565×10^{-1}	0.100	6.03×10^{-5}
ADMM Dual	1.63	532	5.80557×10^{-1}	0.100	4.40×10^{-5}
ADMM Primal	5.34	18002	5.80583×10^{-1}	0.119	3.88×10^{-5}

Table 2: 算法精确度差异

Method	Err-to-CVX-Mosek	Err-to-CVX-Gurobi
CVX-Mosek	0.00E+00	8.44×10^{-8}
CVX-Gurobi	8.44×10^{-8}	0.00E+00
Mosek	3.75×10^{-5}	3.75×10^{-5}
Gurobi	3.91×10^{-5}	3.90×10^{-5}
SGD Primal	8.30×10^{-5}	8.29×10^{-5}
ProxGD Primal	6.90×10^{-5}	6.89×10^{-5}
FProxGD Primal	4.75×10^{-5}	4.74×10^{-5}
ALM Dual	6.03×10^{-5}	6.02×10^{-5}
ADMM Dual	4.40×10^{-5}	4.39×10^{-5}
ADMM Primal	3.88×10^{-5}	3.88×10^{-5}

4 参考资料

1. 文再文老师课程讲义

2. 文再文老师个人主页算法实例
http://faculty.bicmr.pku.edu.cn/ wenzw/optbook/pages/contents/contents.html

谢谢阅读!