北京大学

PEKING UNIVERSITY

上机报告

学　　院：数学科学学院

班　　级：硕士一班

姓　　名：谢地

学　　号：1801210098

2019~2020 第一学期

使用 LaTeX 撰写于 2019 年 11 月 18 日

目录

作业一 Algorithms for l_1 minimization

1.1 上机内容

考虑如下优化问题：

$$\min_x \quad \frac{1}{2}\|Ax-b\|_2^2+\mu\|x\|_1 \tag{1.1}$$

其中 $A\in\mathbb{R}^{m\times n}, b\in\mathbb{R}^m, \mu>0$ 是给定的.

1. 通过调用 CVX 和其中的 mosek,gurobi 求解器求解问题(1.1)
2. 通过直接调用 mosek 和 gurobi 求解问题(1.1)
3. 用投影梯度法求解问题(1.1)
4. 用次梯度方法求解问题(1.1)

1.2 问题一

CVX是一个强大的优化工具箱, 在 CVX 环境中目标函数可以直接用范数的形式表示, 但是 CVX 不能对范数进行幂运算, 因此需要将问题(1.1)中的 $\|Ax-b\|_2^2$ 项转化成 $(Ax-b)'(Ax-b)$ 的内积形式, 此时问题(1.1)可以直接用原问题的无约束形式即可. 若要使用不同的求解器, 则在 CVX 环境中特殊指明.

1.3 问题二

mosek和gurobi不能像 CVX 那样直接求解原问题, 只能对特殊类型的优化问题求解. 因此需要将原问题(1.1)进行等价的变形. 我们考虑将其变成二次规划问题 (Quadratic program,QP). 二次规划具体形式为

$$\begin{aligned} \min \quad & \tfrac{1}{2}x^TPx+q^Tx+r \\ \text{s.t.} \quad & Gx\le h \\ & Ax=b \end{aligned} \tag{1.2}$$

即目标函数为二次函数, 等式约束和不等式约束为线性函数. 下面将(1.1)转化成(1.2)的形式.

首先令 $y = Ax - b \in \mathbb{R}^m$, 则(1.1)等价于

$$\begin{aligned} \min_{x,y} \quad & \tfrac{1}{2} y^T y + \mu \|x\|_1 \\ \text{s.t.} \quad & y = Ax - b \end{aligned} \tag{1.3}$$

因为 $\|x\|_1 = \sum_{i=1}^{n} |x_i|$, 所以令 $z_i = |x_i|$, 则问题等价于

$$\begin{aligned} \min_{x,y,z} \quad & \tfrac{1}{2} y^T y + \mu 1^T z \\ \text{s.t.} \quad & y = Ax - b \\ & -x_i \le z_i \le x_i, \quad i = 1, \dots, n \end{aligned} \tag{1.4}$$

可以看出来这是一个二次规划问题, 具体对应到(1.2)中的话, 未知变量为 $(x^T, y^T, z^T)^T \in \mathbb{R}^{2n+m}$, 目标函数中的矩阵系数和向量系数分别为

$$P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad q = \begin{pmatrix} 0 \\ 0 \\ \mu 1 \end{pmatrix}$$

等式约束的矩阵系数和向量系数分别为

$$\bar{A} = \begin{pmatrix} A & -I & 0 \end{pmatrix}, \bar{b} = b$$

不等式约束的矩阵系数和向量系数分别为

$$G = \begin{pmatrix} I & 0 & -I \\ -I & 0 & -I \end{pmatrix}, h = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

其中 I 是单位矩阵,0 表示零矩阵,1 表示所有元素全为 1 的向量, 至此问题完全转化为二次规划问题.

以上的推导过程说明了(1.1)可以通过增加 $n + m$ 个辅助变量 $y \in \mathbb{R}^m, z \in \mathbb{R}^n$ 变成二次规划问题(1.4). 经过上面的分析, 我们可以对(1.4)使用 gurobi 进行求解. 因为二次规划问题都可以转化成二阶锥规划问题 (Second-order cone programming,SOCP),mosek 将所有二次规划问题都看成二阶锥规划问题求解, 因此我们需要进一步将(1.4)转化成 SOCP.

首先目标函数必须是线性的, 所以令 $w = y^T y$, 则有

$$\begin{aligned} \min_{x,y,z,w} \quad & \tfrac{1}{2}w + \mu 1^T z \\ \text{s.t.} \quad & y = Ax - b \\ & -x_i \le z_i \le x_i, \quad i = 1, \dots, n \\ & y^T y \le w \\ & w \ge 0 \end{aligned} \tag{1.5}$$

已知对于旋转锥 $w^T w \le xy, x, y > 0$, 等价于

$$\left\| \begin{pmatrix} 2w \\ x - y \end{pmatrix} \right\| \le x + y$$

所以问题又可以等价于

$$\begin{aligned} \min_{x,y,z,w} \quad & \tfrac{1}{2}w + \mu 1^T z \\ \text{s.t.} \quad & y = Ax - b \\ & -x_i \le z_i \le x_i, \quad i = 1, \dots, n \\ & \left(w + 1, \begin{pmatrix} 2y \\ w - 1 \end{pmatrix} \right) \ge_{\mathcal{Q}} 0 \\ & w \ge 0 \end{aligned} \tag{1.6}$$

到这里已经变成 SOCP 问题了. 但是通过我对 mosek 的学习, 好像 mosek 对二阶锥约束只能用原始变量表示, 不能用变量的其他形式, 也就是说(1.6)中的 $w + 1, w - 1, 2y$ 都是不能表示的, 因此我们还需要再引入变量来表示这三个变量, 最终问题等价于

$$\begin{aligned} \min \quad & \tfrac{1}{2}w + \mu 1^T z \\ \text{s.t.} \quad & y = Ax - b \\ & -x_i \le z_i \le x_i, \quad i = 1, \dots, n \\ & \left(p_3, \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \right) \ge_{\mathcal{Q}} 0 \\ & p_1 = 2y \\ & p_2 = w - 1 \\ & p_3 = w + 1 \\ & w \ge 0 \end{aligned} \tag{1.7}$$

至此可以使用 mosek 进行求解, 以上推导过程说明了(1.1)可以通过增加 $n+m+1$ 个辅助变量 $y \in \mathbb{R}^m, z \in \mathbb{R}^n, w \in \mathbb{R}$ 变成二阶锥规划问题(1.6), 再增加 $m+2$ 个辅助变量 $p_1 \in \mathbb{R}^m, p_2 \in \mathbb{R}, p_3 \in \mathbb{R}$ 变成 mosek 可求解的二阶锥规划问题(1.7).

1.4 问题三

使用投影梯度法 (Projection gradient method) 求解(1.1), 显然是需要对(1.1)做一些改变的. 首先目标函数要可导, 需要考虑的就是 $\|x\|_1$ 这项, 在问题二中我们通过令 $z_i = |x_i|$ 将目标函数变成了二次可导函数, 我们还可以用令一种方法, 将 x_i 分解有 $x_i = x_i^+ - x_i^-, x_i, x_i^- \geq 0$. 则绝对值可以表示成 $|x_i| = x_i^+ + x_i^-$. 因此(1.1)等价成另一种二次规划问题

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|A(x^+ - x^-) - b\|_2^2 + \mu 1^T(x^+ + x^-) \\ \text{s.t.} \quad & x^+, x^- \geq 0 \end{aligned} \tag{1.8}$$

转化成这种形式的好处是目标函数是可微的二次函数, 梯度为

$$\begin{pmatrix} A^T(Ax-b) \\ -A^T(Ax-b) \end{pmatrix} + \mu 1$$

而且约束条件是界约束的 (box constraints). 当我们对目标函数使用最速下降法的时候, 得到的下一个迭代点需要投影到(1.8)的约束集合内, 而界约束的投影是很好求得的, 为

$$\mathcal{P}(x)_i = \max(x_i, 0)$$

即把元素中小于 0 的数变为 0, 大于零的数不变. 总结, 我们通过使用最速下降法进行迭代并对每次的迭代点做一次约束集合上的投影, 就得到了我们的投影梯度法.

1.5 问题四

对原问题(1.1)使用次梯度方法 (Subgradient method), 令 $f(x) = \frac{1}{2}\|Ax-b\|_2^2 + \mu\|x\|_1$, 显然 $f(x)$ 在 $x=0$ 处不可导, 对 $f(x)$ 求次梯度有

$$\partial f = A^T(Ax-b) + \mu \partial \|x\|_1$$

其中 $\partial\|x\|_1$ 表示 $\|x\|_1$ 的次梯度, 即

$$(\partial\|x\|_1)_i = \begin{cases} 1, & x_i > 0 \\ [-1,1] & x_i = 0 \\ -1 & x_i < 0 \end{cases}$$

次梯度方法与最速下降方法思想是一样的, 给定一个初始点, 用当前点的负次梯度方向作为搜索方向, 再确定一个步长从而得到下一个迭代点, 如此反复迭代直至收敛. 即算法框架为

$$x^{(k)} = x^{(k-1)} - t_k g^{(k-1)}, \quad k = 1, 2, \ldots$$

其中 $g^{(k-1)}$ 是 $x^{(k-1)}$ 的次梯度,t_k 是迭代步长, 常用的三种步长取法为

- 固定步长, 即 $t_k = constant$
- 固定长度, 即 $t_k\|g^{(k-1)}\| = constant$, 也就是 $\|x^{(k)} - x^{(k-1)}\| = constant$
- 逐渐衰减, 即 $t_k \to 0, \sum_{k=1}^{\infty} t_k = \infty$

与最速下降法一样, 次梯度方法的收敛比较慢, 并且由于次梯度的性质, 不能保证每次迭代单调下降.

1.6 数值实验

在使用 CVX 求解器求解时, 不知道是不是我的电脑 CVX 的安装有些问题, 使用时第一次会出现错误, 运行第二次就可以正常运行, 所以如果您在运行时出现了错误, 请再运行一次即可正常使用.

在使用 mosek 和 gurobi 求解器求解时, 我把这两个所需要用到的函数的文件夹复制到了代码的目录下, 在使用 mosek 和 gurobi 时需要添加路径, 我在测试代码中也已加上了, 在这里仅说明一下.

在使用投影梯度法和次梯度方法时, 需要对一些参数进行设置, 这两个方法的参数只有每次迭代的步长 t, 问题四中介绍了三种选取方法, 经过不断地尝试, 在这里我们使用第一种取法, 即步长固定不变, 在这个问题中投影梯度法取 t =4.5e-4; 次梯度方法取 t =4.0e-4. 还有个问题是如果直接按照 $\mu = 0.001$ 去算的话, 这两种算法都只会收敛到 0.3 左右, 虽然函数值在下降, 但是基本收敛不动. 因此需要进行一些改进, 我们先将 μ 取一个比较大的值, 我们知道对于问题(1.1),μ 的值越大说明极小化越侧重于 $\|x\|_1$, 即 x 越稀疏, 当 x 有越多的元素变成 0

时, 再对 μ 逐渐取小至 0.001, 则函数可以很快地收敛到比较接近的值. 具体地在这个问题中经过不断调试, 我们始取 μ 从 100 到 0.001, 每次变化缩小十倍, 也就是说一共改变六次. 在代码中我们设置一个精度和每个 μ 值的最大迭代次数, 当迭代次数达到设置的最大值或者两次迭代的函数值的差小于设置的精度时, 即迭代已经不能使得函数值有明显下降时, 将 μ 取向下一个值, 直到 $\mu = 0.001$. 下面我们对问题一到问题四中的六种方法比较它们的函数值、运行时间、精度、恢复效果、稀疏性以及投影梯度法和次梯度方法的迭代次数. 其中精度为各种方法的解与直接调用 CVX 的 mosek 求解器得到的解的误差, 恢复效果用 $\|Ax - b\|_2$ 衡量, 稀疏性用 $\|x\|_1$ 衡量. 得到的结果如下

表 1.1: 六种方法的比较

Solver	Fval	Time	Errfun	Iter	Sparisity	recovery
cvx_mosek	8.0877e-2	1.34	–	–	80.8764	5.2e-04
cvx_gurobi	8.0876e-2	1.61	2.71e-06	–	80.8763	4.8e-04
mosek	8.0892e-2	1.17	4.46e-05	–	80.8794	7.9e-04
gurobi	8.0876e-2	2.81	2.54e-06	–	80.8763	4.8e-04
PGD	8.0876e-2	0.32	1.50e-06	1577	80.8762	5.6e-04
SGD	8.0877e-2	0.36	1.27e-06	1662	80.8764	6.5e-04

可以看到, 每种方法都可以收敛到极小值, 因此这六种方法是有效的. 下面从各个指标上对六种方法进行比较.

在最优值上, 显然函数值越小对问题(1.1)的求解效果越好. 总体来说,mosek 求解器的效果要差一些,gurobi、PCG 和 SGD 的效果要好一些.

在运行时间上,PCG 和 SGD 都要比直接使用求解器求解的要快很多, 再结合迭代次数中 SGD 的迭代次数要比 PGD 的迭代次数少一些, 也大致说明了 SGD 要比 PGD 方法的运行时间少一些, 这说明了 SGD 要比 PCG 能更快地收敛到最优解. 而在四种求解器中, 直接使用 mosek 求解器最快, 直接使用 gurobi 求解器最慢, 整体上 mosek 要比 gurobi 要快.

在最优解精度上, 可以看到 gurobi 和 PCG 效果要相对好一些,mosek 和 SGD 效果要相对差一些.

在稀疏度上,$\|x\|_1$ 越小说明越稀疏表1.1表明了六种方法的稀疏度基本一致, 如果将很接近 0 的 x_i, 例如 $|x_i| < 10^{-5}$ 认为是 0 的话, 则 x 的稀疏度, 即 $|x|_0$ 应该是相等的.

在恢复效果上,PCG 和 SDG 的效果整体要比求解器的效果差一些, 在四种求解器中,gurobi 整体上要比 mosek 效果要好.

总的来说,PCG 和 SGD 在达到其他指标相同精度的前提下大大缩短了运行时间, 在性能上比求解器有了明显提升. 因为 PCG 和 SGD 中有可以设置的参数, 因此我们也可以更多的调试, 比如设置步长的类型找出更加合适的参数, 达到更好的效果。

实验总结

这次的上机作业对我来说是一个很好的锻炼, 在这次的作业中, 我学习了 CVX、mosek、gurobi 求解器的使用, 通过将问题转化成求解器可以求解的类型, 即二次规划问题、二阶锥问题, 并用 CVX、moesk、gurobi 的语言编写目标函数和约束条件求解问题(1.1). 然后就是用投影梯度法和次梯度方法解决问题(1.1), 虽然两种方法的思想很简单, 都是最速下降法的简单推广, 但是由于问题的问题本身的一些性质和使用梯度方法所带来的收敛慢的问题, 在具体问题上还需要作出一些改变, 比如问题(1.1)中先让 μ 增大再逐渐减小, 步长形式的选取和步长参数的设置以及算法终止条件, 这些都是我在编写程序中一点点探索测试得到的, 无论是 mosek 和 gurobi 语言的学习, 还是投影梯度法和次梯度方法在求解问题是遇到的问题, 都在我不断学习中克服了. 对此我对完成这次的上机作业有很大的成就感.

通过实验的结果得出了投影梯度法和次梯度方法在性能上优于求解器, 因为这两种方法是针对问题(1.1)所设计的方法, 所以两种方法对这种问题有更多的处理形式, 我们可以根据自身的需要设置合适的参数, 比如如果实际上不需要很高的精度, 则可以通过减小最大迭代次数和迭代终止精度损失一定的精度但是可以提高算法速度等等, 但是正因为这是针对这一个问题写出的算法, 所以其适用性自然要比求解器要差, 因此对于不同的情况我们可以采用合适的方法解决问题.