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 12 月 14 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. 通过增广拉格朗日方法 (ALM) 求解问题(1.1)的对偶问题
2. 通过交替乘子方向法 (ADMM) 求解问题(1.1)的对偶问题
3. 通过使用线搜索的交替乘子方向法求解问题(1.1)

1.2 问题一

首先写出(1.1)的对偶问题. 令 $y=Ax-b$ 将问题变成带约束的优化问题

$$\begin{aligned}\min_{x,y} \quad & \tfrac{1}{2}\|y\|_2^2+\mu\|x\|_1\\ \text{s.t.} \quad & y=Ax-b\end{aligned} \tag{1.2}$$

(1.2)的拉格朗日函数为

$$\begin{aligned}\mathcal{L}(x,y,\lambda)&=\mu\|x\|_1+\frac{1}{2}\|y\|_2^2-\lambda^T(Ax-b-y)\\ &=\mu\|x\|_1-(A^T\lambda)^Tx+\frac{1}{2}\|y\|_2^2+\lambda^Ty+b^T\lambda\end{aligned} \tag{1.3}$$

分别对拉格朗日函数中的 x,y 求极小值. 对于 x, 因为 $\mu\|x\|_1-(A^T\lambda)^Tx=\sum_i\mu|x_i|-(A^T\lambda)_ix_i$, 而当 $|(A^T\lambda)_i|\le\mu$ 时, 在 $x=0$ 处 $\mu|x_i|-(A^T\lambda)_ix_i$ 取极小值 0. 对于 y, 这是个二次函数, 令

导数等于 0 可以得到在 $y=-\lambda$ 处 $\frac{1}{2}\|y\|_2^2+\lambda^T y$ 取极小值 $-\frac{1}{2}\|\lambda\|_2^2$. 所以有(1.1)的对偶问题为

$$\begin{aligned} \max \quad & b^T\lambda-\tfrac{1}{2}\|\lambda\|_2^2 \\ \text{s.t.} \quad & \|A^T\lambda\|_\infty \le \mu \end{aligned} \tag{1.4}$$

为了使用 ALM 方法求解对偶问题(1.4), 需要有等式约束, 因此将问题(1.4)等价于

$$\begin{aligned} \min \quad & -b^T\lambda+\tfrac{1}{2}\|\lambda\|_2^2 \\ \text{s.t.} \quad & A^T\lambda=s, \quad \|s\|_\infty \le \mu \end{aligned} \tag{1.5}$$

则有增广拉格朗日函数

$$\mathcal{L}(\lambda,s,x)=-b^T\lambda+\frac{1}{2}\|\lambda\|^2+x^T(A^T\lambda-s)+\frac{1}{2\tau}\|A^T\lambda-s\|^2 \tag{1.6}$$

若假设当前迭代点为 x^k,λ^k,s^k, 则 ALM 方法的框架为

- 固定 x^k, 求解如下 DL 子问题

$$\min_{\lambda,s}\mathcal{L}(\lambda,s,x^k), \quad \text{s.t. } \|s\|_\infty \le \mu \tag{1.7}$$

- 更新拉格朗日乘子

$$x^{k+1}=x^k+\frac{A^T\lambda^{k+1}-s^{k+1}}{\tau} \tag{1.8}$$

因此我们需要求解 DL 子问题, 这是一个约束优化问题, 由于目标函数是可微的并且约束集合很特殊, 因此我们可以考虑使用投影梯度法, 迭代公式为

$$\begin{aligned} \lambda_{j+1} &= \lambda_j - t\nabla_\lambda\mathcal{L}(\lambda_j,s_j,x^k) \\ s_{j+1} &= \mathcal{P}(s_j - t\nabla_s\mathcal{L}(\lambda_{j+1},s_j,x^k)) \end{aligned}$$

其中

$$\begin{aligned} \nabla_\lambda\mathcal{L} &= \lambda-b+Ax+\frac{1}{\tau}A(A^T\lambda-s) \\ \nabla_s\mathcal{L} &= -x-\frac{1}{\tau}(A^T\lambda-s) \end{aligned}$$

$\mathcal{P}(x)$ 为投影算子, 将 x 的每个分量投影到区间 $[-\mu,\mu]$, 初始迭代点为 $\lambda_0=\lambda^k, s_0=s^k$. 一直迭代直到增广拉格朗日函数(1.6)没有明显下降趋势, 即 λ,s 达到收敛, 用收敛点表式 λ^{k+1},s^{k+1}.

还可以使用一种比较简单的思想, 即将两个变量分开独立求解, 具体迭代公式为

$$\begin{aligned} \lambda_{j+1} &= (AA^T + \tau I)^{-1}(\tau(b - Ax^k) + As_j) \\ s_{j+1} &= \mathcal{P}_{[-\mu,\mu]}(A^T\lambda_{j+1} + \mu x^k) \end{aligned} \tag{1.9}$$

然后迭代直到增广拉格朗日函数(1.6)没有明显下降趋势, 即 λ, s 达到收敛, 用收敛点表式 λ^{k+1}, s^{k+1}, 这样的好处是变量有显式解且简单易求.

通过以上两种方法可以求解 DL 子问题, 然后更新拉格朗日乘子, 一直迭代直到达到收敛准则.

1.3 问题二

ADMM 方法是 ALM 方法的一个简化版, 在 ALM 方法中, 求解 DL 子问题一般不是很容易求解的, 会带来比较大的运算量,ADMM 方法的思想是近似求解 DL 子问题, 然后再更新拉格朗日乘子. 具体的近似求解方法为将 λ, s 看成独立的变量分别求解极值问题, 这时 λ, s 都是有显式解的, 其实也就是问题一中求解 DL 子问题时的第二种方法, 只是更新一次 λ, s 就更新一次拉格朗日乘子, 没有对 λ, s 迭代. 因此 ADMM 方法的更新步骤为

$$\begin{aligned} \lambda^{k+1} &= (AA^T + \tau I)^{-1}(\tau(b - Ax^k) + As^k) \\ s^{k+1} &= \mathcal{P}_{[-\mu,\mu]}(A^T\lambda^{k+1} + \tau x^k) \\ x^{k+1} &= x^k + \frac{A^T\lambda^{k+1} - s^{k+1}}{\tau} \end{aligned} \tag{1.10}$$

ADMM 方法就是一直对上面三步进行迭代, 直到达到收敛准则.

1.4 问题三

为了使用 ADMM 方法求解原问题(1.1), 需要有等式约束, 因此将问题(1.1)等价于

$$\begin{aligned} \min_{x,y} \quad & \tfrac{1}{2}\|Ax - b\|_2^2 + \mu\|y\|_1 \\ \text{s.t.} \quad & x - y = 0 \end{aligned} \tag{1.11}$$

则有增广拉格朗日函数

$$\mathcal{L}(x,y,z)=\frac{1}{2}\|Ax-b\|_2^2+\mu\|y\|_1+z^T(x-y)+\frac{\tau}{2}\|x-y\|_2^2 \tag{1.12}$$

若 y,z 已知, 增广拉格朗日函数为关于 x 的二次函数, 求导并令其等于零, 因此有极小值

$$x=(A^TA+\tau I)^{-1}(A^Tb+\tau y-z)$$

若 x,z 已知, 求增广拉格朗日函数关于 y 的极小值为

$$\begin{aligned}&\min \quad \mu\|y\|_1+z^T(x-y)+\frac{\tau}{2}\|x-y\|_2^2\\ \Longleftrightarrow &\min \quad \frac{\mu}{\tau}\|y\|_1+\frac{1}{\tau}z^T(x-y)+\frac{1}{2}\|x-y\|_2^2\\ \Longleftrightarrow &\min \quad \frac{\mu}{\tau}\|y\|_1+\frac{1}{2}\|x-y+z/\tau\|_2^2\end{aligned}$$

这是一个关于一范数的 proximal 算子, 因此可以得到 y 的值为

$$y=prox_{\mu/\tau\|\cdot\|_1}(x+z/\tau)$$

综合以上我们可以得到 ADMM 方法的迭代公式

$$\begin{aligned}x^{k+1}&=(A^TA+\tau I)^{-1}(A^Tb+\tau y^k-z^k)\\ y^{k+1}&=prox_{\mu/\tau\|\cdot\|_1}(x^{k+1}+z^k/\tau)\\ z^{k+1}&=z^k+\tau(x^{k+1}-y^{k+1})\end{aligned} \tag{1.13}$$

1.5 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.

在使用以上三种方法求解问题时, 与前两次上机实验一样, 我们对 μ 从比较高的值逐渐递减到 $\mu=0.001$, 这样能让算法更快地收敛到极小值. 比如在第一次上机实验中, 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. 在这次实验的三个方法的实际编程中, 有些值是在迭代中反复使用的, 比如(1.9)和(1.10)中的 $(AA^T+\tau I)^{-1}$,(1.13)中的 $(AA^T+\tau I)^{-1}$ 和 A^Tb. 为了减少运算量, 我们可以提前计算出这些值. 因为这三种方法都引入了新的变量, 因此需要对这些变量赋初值, 在这里使用 rand 函数产生随机的初始变量, 同时为了保证代码的可复现, 选取种子 rng(0).

为了能更好地比较这几种方法的优劣性, 将第一次上机实验中的投影梯度法和次梯度方法和第二次实验中的近端梯度法也放入到这次的实验中与 CVX 的 mosek 求解器进行比较, 因为在第一次上机实验中我们有 gurobi 求解器与 mosek 的解误差有 10^{-6} 的量级, 因此在这里我们对六种方法进行调参, 在达到相同量级精度下与 mosek 的运行时间的比较, 以及六种方法中迭代次数等一些其他值的比较.

需要说明的是, 在这次实验中的三种方法与上两次实验的三种方法相比, 没有迭代步长参数 t, 同时增加了惩罚项参数 τ. 特别地, 对于问题一中的 ALM 方法的 DL 子问题, 经过实验我们使用第二种迭代方式, 即将 λ, s 看作是独立变量分别求解迭代, 不采用投影梯度法. 这时则又增加两个参数, 即 DL 子问题迭代的最大迭代次数和终止精度, 实际上 DL 子问题不需要求解的很精确, 并且 DL 子问题的迭代收敛很快, 因此在这里取 DL 子问题的最大迭代次数为 Max = 10, 终止精度为 eps = 1e-3.

需要调节的参数有初始 μ 值、固定迭代步长 t、固定惩罚系数、算法终止的精度 eps 和每个 μ 值的最大迭代次数 Max. 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 eps 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 $\mu = 0.001$. 具体的参数值见下表

表 1.1: 六种方法的参数设置

Parameter	PGD	SGD	ProxGD	ALM_dual	ADMM_dual	ADMM_lprimal
μ	100	100	10	1	1	1
t	4.5e-4	4e-4	7e-4	-	-	-
τ	-	-	-	100	100	100
eps	1e-9	1e-10	1e-9	1e-9	1e-9	1e-10
Max	200	300	250	150	150	200

调整好参数后, 我们对问题一到问题三, 以及第一、二次上机实验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1.2

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. 在与 CVX 的 mosek 求解器的解的误差精度达到 10^{-6} 的条件下, 六种方法得到的最优值都很接近. 其中 SGD 方法的解相对差一些, 其他五种方法效果相对更好.

在运行时间上, 六种方法都要比直接使用求解器要快很多, 其中 ADMM_dual 和 ProxGD

表 1.2: 六种方法的比较

Solver	Fval	Time	Errfun	Iter	Sparisity	recovery
cvx_mosek	8.0877e-2	1.27	–	–	80.8764	5.2e-04
PGD	8.0876e-2	0.21	1.53e-06	1136	80.8762	5.6e-04
SGD	8.0877e-2	0.29	1.27e-06	1662	80.8764	6.5e-04
ProxGD	8.0876e-2	0.15	1.83e-06	980	80.8763	5.4e-04
ALM_dual	8.0876e-2	0.40	2.86e-06	343	80.8763	5.0e-04
ADMM_dual	8.0876e-2	0.18	2.84e-06	344	80.8763	5.0e-04
ADMM_lprimal	8.0876e-2	0.35	2.75e-06	479	80.8763	4.8e-04

最快, 可以控制在 0.2 秒,PGD 和 SGD 可以控制在 0.3 秒, 而 ALM_dual 和 ADMM_dual 相对较慢, 可以控制在 0.4 秒. 可以发现, 虽然三种增广拉格朗日类方法迭代次数与其他三种方法比起来要小很多, 但是运行时间并没有成比例减少, 这点从迭代公式可以看出来. 首先对于 PGD、SGD、ProxGD, 它们每次迭代的最大运算量主要是在算梯度、次梯度、Prox 算子时的两个矩阵乘向量上; 而对于 ADMM_dual, 从迭代公式(1.10)可以看到, 每次迭代计算 λ 需要两次矩阵乘向量, 计算 s 需要一次矩阵乘向量, 计算 x 需要一次矩阵乘向量, 因此一次迭代需要四次矩阵乘向量, 是前三种方法的两倍, 再加上需要提前计算 512 维的矩阵乘向量的逆, 所以虽然迭代次数比前三种方法少很多, 但运行时间并没有相差很多. 再看 ALM_dual 方法, 因为 ALM_dual 方法是在计算 λ, s 时进行多次迭代, 所以每次迭代要计算更多次的矩阵乘向量运算, 而在迭代次数上, 因为 ALM_dual 每次计算 DL 子问题都要比 ADMM_dual 更精确, 所以迭代次数自然也会少一些, 但是可以看到实际上只比 ADMM_dual 少迭代一次, 而运行的时间却是 ADMM_dual 的两倍多, 因此 ADMM 要比 ALM 方法更好. 最后再看 ADMM_lprimal 方法, 从迭代公式(1.13)可以看到, 每次迭代只有在计算 x 的时候需要用到两次矩阵乘向量, 但是 ADMM_lprimal 引入两个 1024 维的向量, 需要提前计算一个 1024 维的矩阵的逆, 而矩阵的求逆运算量是很大的. 因此总的来说 ADMM_lprimal 的运算时间也比较大.

在稀疏度上,$\|x\|_1$ 越小说明越稀疏. 表1.2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,$\|Ax - b\|_2$ 越小说明恢复效果越好. 可以看到这次实验的增广拉格朗日类方法比前两次实验的三种方法要相对好一些.

总的来说, 近端梯度法和对偶问题的 ADMM 方法是综合比较好的方法. 其次如果希望有少的迭代次数和恢复效果则可以使用对偶问题的 ALM、对偶问题的 ADMM 以及原问题的 ADMM. 如果希望有更好的稀疏度和运行时间则可以使用投影梯度法和近端梯度法. 随机梯度法的效果总体来说不如其他几种方法要好.

实验总结

这次上机实验主要是学习编程一类使用增广拉格朗日方法求解有等式约束的极值问题的方法. 与前两次的上机实验的方法思想上有些不同, 是一类通过更新拉格朗日乘子实现收敛的方法. 通过实验对这类的三种方法与前两次实验的三种方法进行比较, 发现 ADMM 方法确实有很好的收敛效果.

经过这三次的上机实验, 我学习了许多方法去求解简单的回归问题(1.1), 虽然这个问题的形式很简单, 但是我想这个上机的真正目的是掌握这些方法的思想, 比如如何将问题转化成 QP、SOCP、对偶等问题, 线性化目标函数, 将目标函数拆分等等. 只要能熟练掌握这些技巧, 相信以后遇到更加复杂的问题依然可以去求解.