Project 1 report

AAIS in PKU Weijie Chen 1901111420

April 14, 2020

1 Algorithms for l_1 Minimization

1.1 Problem Setting

Considering the problem

$$\min_x \mu\|x\|_1 + \|Ax - b\|_1 \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ are given. Test data are as follows:

- n = 1024, m = 512
- A = randn(m,n), u = sprandn(n,1,0.1)
- b = A*u, μ = 1e-2

In total, we use 4 ways to solve this problem, and we will show the algorithms, numerical results and their interpretation below. The 4 methods are:

- Mosek in CVX
- Gurobi in CVX
- Augmented Lagrangian method, where each augmented Lagrangian func- tion is minimized using proximal gradient method
- Alternating direction method of multipliers(ADMM) for the primal problem

1.2 Augmented Lagrangian Method

Note that the original problem is an unconstrained problem, so we do a variable substitution and rewrite the problem with the following form:

$$\min_x \mu\|x\|_1 + \|y\|_1 \quad \text{s.t.} Ax - b = y \tag{2}$$

Therefore, we have the Lagrangian function

$$L(x, y, \lambda) = \mu\|x\|_1 + \|y\|_1 + \lambda^T(Ax - b - y) \tag{3}$$

then we add the augmented term and get

$$L_\beta(x, y, \lambda) = \mu\|x\|_1 + \|y\|_1 + \lambda^T(Ax - b - y) + \frac{\beta}{2}\|Ax - b - y\|_2^2 \tag{4}$$

In order to get a more compact form, we denote that $z = (x, y) \in \mathbb{R}^{n+m}$, then rewrite the augmented Lagrangian function

$$L_\beta(z, \lambda) = \|\hat{\mu} z\|_1 + \lambda^T(\hat{A}z - b) + \frac{\beta}{2}\|\hat{A}z - b\|_2^2 \tag{5}$$

where $\hat{\mu} = \text{diag}(\mu \mathbb{I}_n, \mathbb{I}_m)$ and $\hat{A} = [A, -\mathbb{I}_m]$.

Now the problem is to minimize $L_\beta(z, \lambda)$. The update rule is

$$\begin{aligned} z_{k+1} &:= \underset{z}{\text{argmin}} \|\hat{\mu} z\|_1 + (\lambda_k)^T (\hat{A}z - b) + \tfrac{\beta}{2}\|\hat{A}z - b\|^2 \\ \lambda_{k+1} &:= \lambda_k + \alpha \left(\hat{A} z_{k+1} - b\right) \end{aligned} \tag{6}$$

To solve the subproblem of z, we consider proximal gradient method. Proximity operator (prox-operator) of convex funtion h is defined as following.

$$\text{prox}_h(x) = \text{argmin}_u \left(\frac{1}{2}\|u - x\|_2^2 + h(u)\right) \tag{7}$$

For general unconstrained optimization with objective split in two components

$$\min_z h(z) + g(z) \tag{8}$$

where $h(z) = \|\hat{\mu} z\|_1$ and $g(z) = (\lambda_k)^T (\hat{A}z - b) + \frac{\beta}{2}\|\hat{A}z - b\|^2$. Then we introduce to a iterative method with proximity operator

$$z_k^{l+1} = \text{prox}_{t_k h}\left(z_k^l - t_k \nabla g\left(z_k^l\right)\right) \tag{9}$$

where t_k is the step size of each update and upper index l denotes the l-th iteration in the subproblem. In l_1-minimization problem, the gradient of g can be calculated exactly that

$$\nabla g(z) = \hat{A}^T \lambda + \beta \hat{A}^T (\hat{A}z - b) \tag{10}$$

1.3 ADMM for the Primal Problem

From the above section, we notice that the joint minimum value of the augmented lagrangian isn't always explicit and inexpensive. However, coordinate minimizing is usually inexpensive when another variables fixed. That is the intuition of **alternating direction method of multipliers**. In each interation,

$$\begin{aligned} x_{k+1} &:= \underset{x}{\text{argmin}} \mu\|x\|_1 + (\lambda_k)^T (Ax - y_k - b) + \tfrac{\beta}{2}\|Ax - y_k - b\|^2 \\ y_{k+1} &:= \underset{y}{\text{argmin}} \|y\|_1 + (\lambda_k)^T (Ax_{k+1} - y - b) + \tfrac{\beta}{2}\|Ax_{k+1} - y - b\|^2 \\ \lambda_{k+1} &:= \lambda_k + \alpha (Ax_{k+1} - y_{k+1} - b) \end{aligned} \tag{11}$$

Similar to augmented Lagrangian method, we solve the subproblems with proximal gradient method.

1.4 Numerical Result

The cpu time, error with cvx-mosek and optimal value of different methods were shown below. The implemention of these methods had been shown in each *.m file.

The main technique is continuation method, that using larger μ to make the l_1 penalty of x have the same weight as the other term at the beginning. Then, we decrease μ till the target value while optimizing the problem.

Method	cpu time/s	error with cvx-mosek	optimal value
cvx-call-mosek	1.57	0.00e+00	0.72844399488363
cvx-call-gurobi	3.97	1.49e-07	0.72844462365042
Augmented Lagrangian	1.39	7.35e-07	0.73040764808376
ADMM primal	1.04	9.12e-08	0.72868194363288

Table 1: Efficiency (cpu time) and accuracy

2 Algorithms for Sparse Inverse Covariance Estimation

2.1 Problem Setting

Let $\mathbb{S}^n = \{X \in \mathbb{R}^{n\times n} | X^T = X\}$. Let $S \in \mathbb{S}^n$ be a given observation of covariance matrix. Considering the sparse inverse problem

$$\max_{X\succeq 0} \log\det X - \mathrm{Tr}(SX) - \rho\|X\|_1 \tag{12}$$

where $\|X\|_1 = \sum_{ij} |X_{ij}|$.

2.2 Dataset

As in the reference [1], we let $n = 30$ and generate model1 and model2. Here we make a slight adjustment to make matrix generated from model2 to be symmetric.

- model1: $S_{i,j} = 0.6^{|i-j|}$
- model2: $S = B + \delta\mathbb{I}_n$, where each off-diagonal entry in B is generated independently and equals 0.5 with probability 0.1 or 0 with probability 0.9. δ is chosen such that the conditional number (the ratio of maximal and minimal singular values of a matrix) is equal to p. Finally, the matrix is standardized to have unit diagonals.

2.3 Dual Problem

To derive the dual form, we introduce variable U, that

$$\|X\|_1 = \max_{\|U\|_\infty \le 1} < U, X > \tag{13}$$

where $\|U\|_\infty := \max |U_{ij}|$, thus rewrite the original problem to

$$\max_{X\succeq 0} \min_{\|U\|_\infty \le \rho} \mathrm{logdet}(X) - < S, X > - < U, X > \tag{14}$$

$$L(X, U, Y) = \log\det(X) - < S + U, X > + < Y, X > \tag{15}$$

The Lagrangian function is

$$L(X, U, Y) = \log\det(X) - < S + U, X > + < Y, X > \tag{16}$$

where $X \succeq 0, Y \succeq 0$ and $\|U\|_\infty \le \rho$.

The derivative of the Lagrangian function is

$$\nabla_X L = X^{-1} - (S + U - Y) \tag{17}$$

Therefore, the dual problem is

$$\min_{Y\succeq 0, \|U\|_\infty \le \rho} -\log\det(S + U - Y) - n \tag{18}$$

The duality gap is $n - < S, X > -\rho\|X\|_1$.

2.4 Solving Primal Using First-Order Type Algorithm

In this section, we directly generalize gradient descent method to subgradient method. For sparse inverse problem, the subgradient as following

$$g = X^{-1} - S - \rho \operatorname{sign}(X) \tag{19}$$

Then, we implement the subgradient algorithm with line search technique.

Algorithm 1: Solving Sparse Inverse Using Subgradient Method

Input : covariance matrix S, hyperparameter ρ, stop criteria ϵ
Output: estimation X, optimal value, duality gap

Initialize $X = \mathbb{I}_n$;
repeat
  $X_0 := X$;
  $g = X_0^{-1} - S - \rho \operatorname{sign}(X_0)$;
  Line Search to choose step size α;
  $X := X_0 + \alpha g$;
until $\|X_0 - X\|_2 < \epsilon$;

2.5 Numerical Result

Parameter	cpu time/s	optimal value	duality gap
CVX,$\rho = 10$	6.75	-101.9369	1e-5
First-order,$\rho = 10$	8.66	-101.9446	-5e-2
CVX,$\rho = 0.1$	5.73	-26.1081	2e-5
First-order,$\rho = 0.1$	3.49	-26.1083	5e-2
CVX,$\rho = 0.001$	5.56	-17.1743	9e-5
First-order,$\rho = 0.001$	0.42	-17.1743	8e-3

Table 2: Efficiency (cpu time) and accuracy of model1

Parameter	cpu time/s	optimal value	duality gap
CVX,$\rho = 10, p = 30$	9.12	-101.9369	1e-5
First-order,$\rho = 10, p = 30$	9.00	-101.9457	-4e-2
CVX,$\rho = 0.1, p = 30$	7.35	-28.0226	2e-4
First-order,$\rho = 0.1, p = 30$	3.77	-28.0231	4e-2
CVX,$\rho = 0.001, p = 30$	4.50	-418.1216	3e+1
First-order,$\rho = 0.001, p = 30$	0.10	-30.0132	-1e-2
CVX,$\rho = 10, p = 60$	9.41	-101.9369	1e-5
First-order,$\rho = 10, p = 60$	9.22	-101.9457	-4e-2
CVX,$\rho = 0.1, p = 60$	6.96	-28.0226	2e-4
First-order,$\rho = 0.1, p = 60$	3.81	-28.0231	4e-2
CVX,$\rho = 0.001, p = 60$	4.68	-418.1216	3e+1
First-order,$\rho = 0.001, p = 60$	0.10	-30.0132	-1e-2

Table 3: Efficiency (cpu time) and accuracy of model2 ($p = 30, 60$)

Parameter	cpu time/s	optimal value	duality gap
CVX,$\rho = 10, p = 90$	10.09	-101.9369	1e-5
First-order,$\rho = 10, p = 90$	9.93	-101.9457	-4e-2
CVX,$\rho = 0.1, p = 90$	7.28	-28.0226	2e-4
First-order,$\rho = 0.1, p = 90$	3.95	-28.0231	4e-2
CVX,$\rho = 0.001, p = 90$	4.46	-418.1216	3e+1
First-order,$\rho = 0.001, p = 90$	0.10	-30.0132	-1e-2
CVX,$\rho = 10, p = 120$	9.18	-101.9369	1e-5
First-order,$\rho = 10, p = 120$	8.99	-101.9456	-4e-2
CVX,$\rho = 0.1, p = 120$	6.77	-27.9252	2e-4
First-order,$\rho = 0.1, p = 120$	4.21	-27.9257	4e-2
CVX,$\rho = 0.001, p = 120$	4.47	-416.7642	3e+1
First-order,$\rho = 0.001, p = 120$	0.10	-30.0130	-1e-2
CVX,$\rho = 10, p = 200$	9.37	-101.9369	1e-5
First-order,$\rho = 10, p = 200$	9.18	-101.9459	-4e-2
CVX,$\rho = 0.1, p = 200$	7.01	-27.7810	2e-4
First-order,$\rho = 0.1, p = 200$	4.30	-27.7816	5e-2
CVX,$\rho = 0.001, p = 200$	4.72	-414.9480	3e+1
First-order,$\rho = 0.001, p = 200$	0.10	-30.0127	-1e-2

Table 4: Efficiency (cpu time) and accuracy of model2 ($p = 90, 120, 200$)

In this section we show the numerical results of both cvx and first-order methods for solving the problem when ρ takes different numbers. We show the results with the optimal value ,the duality gap and cpu time to show both the accuracy and efficiency performance of the algorithms.

From the result, we notice that the condition number P has no significant influence on the problem in model2. First-order methods suffer from lower accuracy than CVX. An explanation is that the subgradient is not accurate enough to solve the problem. And in order to achieve the theoretical convergence bound, it needs more iteration times.

3 README

The environment of numerical experiment as following:

- MacBook Pro(15-inch, 2019)
- 2.6 GHz 6-Core Intel Core i7
- 16 GB 2400 MHz DDR4
- macOS Catalina version 10.15.1
- Matlab R2019a

Reference

[1] Tony Cai, Weidong Liu, and Xi Luo. A constrained l1 minimization approach to sparse precision matrix estimation. Journal of the American Statistical Association, 106(494):594–607, 2011.