# Homework 5 Report
for “Convex Optimization”

Zehao Wang, 1700010718

December 17, 2019

Algorithms for ℓ_1 minimization

Consider the ℓ_1-regularized problem

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

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $\mu > 0$ are given.

Basic Setting

- seed = 97006855;
- Errfun$(x, y) = \frac{||x-y||}{1+||x||}$, all of the Errfun results are computing as y with result of l1_cvx_mosek as x
- default stop condition is $(||x_{now} - x_{prev}|| < err) \vee (num\_iter < max\_iter)$
- sparsity$(x) = \frac{\sum_i \mathbb{I}(abs(x_i)<threshold)}{length(x)}$, where $threshold = 1e-9$

Result Glance

Table 1: Results of all solvers.

Solver	Fval	Errfun	Time(s)	Iter	Sparsity
l1_cvx_mosek	0.080876507245464	N/A	N/A	N/A	N/A
l1_cvx_gurobi	0.080876507407497	2.01e-08	1.15	N/A	0.3525
l1_mosek	0.080876800468350	4.41e-06	1.48	N/A	0.0020
l1_gurobi	0.080876390758893	2.54e-06	1.95	N/A	0.7520
l1_PGD_primal	0.080876390321693	2.54e-06	1.15	630	0.8652
l1_SGD_primal	0.080876405664217	2.58e-06	0.66	8945	0.0322
l1_GD_primal	0.080876530783799	4.32e-06	0.56	7615	9.7656e-04
l1_FGD_primal	0.080876530455554	4.30e-06	0.37	4878	9.7656e-04
l1_ProxGD_primal	0.080876390303637	2.51e-06	0.24	2268	0.8662
l1_FProxGD_primal	0.080876390303716	2.50e-06	0.19	1814	0.8662
l1_ALM_dual	0.080876202050527	2.29e-06	1.21	525	0.8447
l1_ADMM_dual	0.080876390303716	5.70e-06	1.01	580	0
l1_ADMM_lprimal	0.080876390303716	3.06e-06	0.66	329	9.7656e-04

Remark

In the previous report, we use norm(*,2) in Matlab to compute our algorithms' final value, it will output 0.0811 as the Fval, but when we change the 2-norm's computation to vector's inner product, it turns to be 0.809.

Claim

I talk with Haotong Yang about some of my algorithms and his.

1 Problem 1

Solve 1 using CVX by calling different solvers mosek and gurobi.

2 Problem 2

First write down an equivalent model of 1 which can be solved by calling mosek and gurobi directly, then implement the codes.

2.1 Reformulate to QP

We can easily rewrite the ℓ_1 minimization problem to a Quadratic Programming as follows,

$$\begin{aligned} \min_y \quad & \tfrac{1}{2} y^T Q y + c^T y, \\ s.t. \quad & Ly \preceq u, \\ & -Ly \preceq -l, \end{aligned} \tag{2}$$

where

$$y \in \mathbb{R}^{(2n+m)}, \quad Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_m \end{bmatrix} \in \mathbb{R}^{(n+n+m)\times(n+n+m)},$$

$$c = \begin{bmatrix} 0 \\ \mu \mathbf{1}_n \\ 0 \end{bmatrix} \in \mathbb{R}^{(n+n+m)}, \quad L = \begin{bmatrix} I_n & -I_n & 0 \\ -I_n & -I_n & 0 \\ A & 0 & -I_m \end{bmatrix} \in \mathbb{R}^{(n+n+m)\times(n+n+m)},$$

$$l = \begin{bmatrix} -\infty \\ b \end{bmatrix} \in \mathbb{R}^{(2n+m)}, \quad u = \begin{bmatrix} 0 \\ b \end{bmatrix} \in \mathbb{R}^{(2n+m)},$$

3 Problem 3

First write down, then implement the following algorithms in Matlab (or Python):

3.1 (a) PGD

Projection gradient method by reformulating the primal problem as a quadratic program with box constraints.

3.1.1 Reformulate to QP

We can easily rewrite the ℓ_1 minimization problem to a Quadratic Programming for this method as follows,

$$\begin{aligned} \min_x \quad & \tfrac{1}{2} ||W\hat{x} - b||_2^2 + \mu \mathbf{1}_{2n}^T \hat{x}, \\ s.t. \quad & -\hat{x} \preceq 0, \end{aligned} \tag{3}$$

the feasible domain of this problem is $S = \mathbb{R}_+^{2n}$, then we can define the projection function as follows,

$$P_S(\hat{x}) = \arg\min_{y \succeq 0} ||\hat{x} - y||_2^2 = \max(\hat{x}, 0), \tag{4}$$

where max means element-wise maximization.

3.1.2 Algorithm & Numerical Experiment

The basic Progection Gradient Method (PGD) is shown in Algorithm 1,

Algorithm 1 Basic PGD Method

```
Input: $x_0$
  $x = x_0$
  while Stop Conditions do
    compute gradient $g$
    choose a proper step size $t$
    $x = P_S(x - tg)$
  end while
Output: $x$
```

We can improve the basic PGD method (2) to adapt our specific problem,

Algorithm 2 Improved PGD Method for this problem

```
Input: $x_0,\ A,\ b,\ \mu$, Multi_$\mu$_Schedule
  $W = [A - A]$
  $x = x_0$
  for $c$ in Multi_$\mu$_Schedule do
    $\mu_{now} = c\mu$
    while Stop Conditions do
      $x^+ = P_S(x)$
      $x^- = x^+ - x$
      $\hat{x} = [x^+; x^-]$
      $g = W^T(W\hat{x} - b) + \mu_{now}\mathbf{1}_{2n}$ # compute gradient
      $t = \frac{g^T g}{g^T W^T W g}$ # compute exact step size
      $\hat{x} = P_S(\hat{x} - tg)$
      $x = \hat{x}(1:n) - \hat{x}(n+1:2n)$
    end while
  end for
Output: $x$
```

Let me explain the idea in above algorithm, first we consider that this problem has several local minima so that we should choose a not-bad initial value x_0 for it, but it can only get x_0 randomly, so I make a Multi_μ_Schedule in order to choose some easier μ first, after the easier problem iterates for some rounds, use the output of this iteration for next μ in the schedule, the final μ in this schedule is the original μ. This idea is come from Sparse Reconstruction.

Because of the existence of the explicit optima of update of $\hat{x}$, we choose this exact step size for update per step.

Where the hyperparameters mentioned above are as follows,

- err: 1e-8
- Multi_μ_Schedule: [1e+4,1e+3,1e+2,1e+1,1]
- Max_Iter_Schedule: [500,500,500,500,1e+6]

3.2 (b) SGD

Subgradient method for the primal problem.

3.2.1 Algorithm & Numerical Experiment

We can use Subgradient method (SGD) directly for this problem. We choose $g = A^T(Ax - b) + \mu\text{sgn}(x) \in \partial f$ as the subgradient to compute.

First we show the basic SGD method in Algorithm 3,

Algorithm 3 Basic SGD Method

```
Input: $x_0$
  $x = x_0$
  while Stop Conditions do
    compute subgradient $g$
    choose a proper step size $t$
    $x = x - tg$
  end while
Output: $x$
```

We can improve the basic SGD method (4) to adapt our specific problem,

The reason why we import two schedules to this algorithm is to accelerate this algorithm and help it to find the global optimum. Multi_μ_Schedule is as same as before. Because of the implicit existence of the optimal step size per step, we choose $t = \frac{\alpha}{\sqrt{k}}$ as our step size, Step_Size_Schedule is designed for α followed the following principle,

1. diminishing step size
2. alternating size to find global optimum quickly

Where the hyperparameters mentioned above are as follows,

- err: 1e-7
- Multi_μ_Schedule: [1e+4,1e+3,1e+2,1e+1,1]

Algorithm 4 Improved SGD Method for this problem

```
Input: $x_0,\ A,\ b,\ \mu$, Multi_$\mu$_Schedule, Step_Size_Schedule
  $x = x_0$
  for $c$ in Multi_$\mu$_Schedule, $\alpha$ in Step_Size_Schedule do
    $\mu_{now} = c\mu$
    $k = 1$
    while Stop Conditions do
      $g = A^T(Ax - b) + \mu\mathrm{sgn}(x)$ # compute subgradient
      $t = \frac{\alpha}{\sqrt{k}}$ # compute step size
      $x = x - tg$
      $k = k + 1$ # update number of iteration in this epoch
    end while
  end for
Output: $x$
```

- Max_Iter_Schedule: [1e+3,1e+3,1e+3,1e+3,1e+6]
- Step_Size_Schedule: [3*1e-3,3*1e-3,3*1e-3,1e-3,3*1e-3]

3.3 (c) GD

Gradient method for the smoothed primal problem.

3.3.1 Algorithm & Numerical Experiment

First we should choose a proper smoothed function for $||*||_1$. Due to the lecture, we can use Huber panalty as smoothed absolute value, then it will be easy to construct a smoothed function ψ as follows,

$$\phi_\alpha(z) = \begin{cases} z^2/2(\alpha) & |z| \le \alpha \\ |z| - \alpha/2 & |z| \ge \alpha \end{cases}, \tag{5}$$

$$\psi_\alpha(x_1, ..., x_n) = \sum_{i=1}^{n} \phi_\alpha(x_i). \tag{6}$$

α controls accuracy and smoothness. The detailed analysis of the accuracy and complexity can be found in the lecture.

First we show the basic Gradient Descent Method in Algorithm 5

Then we treat $\frac{1}{2}||Ax - b||_2^2 + \mu\psi_\alpha(x)$ as our new objective function. The gradient of this function is

$$\begin{aligned} & A^T(Ax - b) + \mu\nabla\psi_\alpha(x) \\ = \quad & A^T(Ax - b) + \mu(\phi'_\alpha(x_i))_{i\in[n]} \end{aligned} \tag{7}$$

where $\phi'_\alpha(x_i) = \mathrm{sgn}(x_i)\min\left\{\frac{|x_i|}{\alpha}, 1\right\} = \begin{cases} 1, x_i \ge \alpha \\ \frac{x_i}{\alpha}, -\alpha < x_i < \alpha \\ -1, x_i \le -\alpha \end{cases}$.

Algorithm 5 Basic GD Method

```
Input: $x_0$
  $x = x_0$
  while Stop Conditions do
    compute gradient $g$
    choose a proper step size $t$
    $x = x - tg$
  end while
Output: $x$
```

Then we can implement GD Method to this function. Basic idea is as same as above, we show the improved GD Method as follows,

Algorithm 6 Improved GD Method for this problem

```
Input: $x_0,\ A,\ b,\ \mu$, Multi_$\mu$_Schedule, Step_Size_Schedule
  $x = x_0$
  for $c$ in Multi_$\mu$_Schedule, $\alpha$ in Step_Size_Schedule do
    $\mu_{now} = c\mu$
    $k = 1$
    while Stop Conditions do
      $g = A^T(Ax - b) + \mu\nabla\psi_\alpha(x)$ # compute gradient
      $t = \frac{\alpha}{\sqrt{k}}$ # compute step size
      $x = x - tg$
      $k = k + 1$ # update number of iteration in this epoch
    end while
  end for
Output: $x$
```

Where the hyperparameters mentioned above are as follows,

- err: 1e-10
- Multi_μ_Schedule: [1e+4,1e+3,1e+2,1e+1,1]
- Max_Iter_Schedule: [1e+3,1e+3,1e+3,1e+3,1e+6]
- Step_Size_Schedule: 3e-3
- multi_α: 1e-3

3.4 (d) FGD

Fast gradient method for the smoothed primal problem.

3.4.1 Algorithm & Numerical Experiment

Smoothed function is the same as it described above.

First we show the basic Fast Gradient Descent Method in Algorithm 7.

Algorithm 7 Basic FGD Method

```
Input: $x_0$
  $x^{(0)} = x_0$
  $v^{(0)} = x_0$
  $k = 1$
  while Stop Conditions do
    $y = (1 - \theta_k)x^{(k-1)} + \theta_k v^{(k-1)}$
    compute gradient $g$ at $y$
    choose a proper step size $t$
    $x^{(k)} = y - tg$
    $v^{(k)} = x^{(k-1)} + \frac{1}{\theta_k}(x^{(k)} - x^{(k-1)})$
    $k = k + 1$
  end while
Output: $x^{(k-1)}$
```

Here we choose $\theta_k = \frac{2}{k+1}$.

Then we can implement FGD Method to this function. Basic idea is as same as above, we show the improved FGD Method as follows,

Algorithm 8 Improved FGD Method for this problem

```
Input: $x_0,\ A,\ b,\ \mu$, Multi_$\mu$_Schedule, Step_Size_Schedule
  for $c$ in Multi_$\mu$_Schedule, $\alpha$ in Step_Size_Schedule do
    $x^{(0)} = x_0$
    $v^{(0)} = x_0$
    $\mu_{now} = c\mu$
    $k = 1$
    while Stop Conditions do
      $\theta_k = \frac{2}{k+1}$
      $y = (1 - \theta_k)x^{(k-1)} + \theta_k v^{(k-1)}$
      $g = A^T(Ay - b) + \mu\nabla\psi_\alpha(y)$ # compute gradient at $y$
      $t = \frac{\alpha}{\sqrt{k}}$ # compute step size
      $x^{(k)} = y - tg$
      $v^{(k)} = x^{(k-1)} + \frac{1}{\theta_k}(x^{(k)} - x^{(k-1)})$
      $k = k + 1$ # update number of iteration in this epoch
    end while
  end for
Output: $x^{(k-1)}$
```

Where the hyperparameters mentioned above are as follows,

- err: 1e-10

- Multi_μ_Schedule: [1e+4,1e+3,1e+2,1e+1,1]
- Max_Iter_Schedule: [1e+3,1e+3,1e+3,1e+3,1e+6]
- Step_Size_Schedule: 3e-3
- multi_α: 1e-3

3.4.2 Result

Shown in Table 1.

3.5 (e) ProxGD

Proximal gradient method for the primal problem.

3.5.1 Algorithm & Numerical Experiment

First let objective function $f(x) = g(x) + h(x)$, where

$$g(x) = \frac{1}{2}||Ax - b||_2^2,$$

$$h(x) = \mu||x||_1.$$

Then we have

$$\text{Prox}_{th}(x)_i = \text{sign}(x_i)\text{max}(|x_i| - t\mu, 0) = \begin{cases} x_i - t\mu, & x_i \geq t\mu \\ 0, & -t\mu \leq x_i \leq t\mu \\ x_i + t\mu, & x_i \leq t\mu \end{cases}, \quad (8)$$

where t is the step-size.

First we show the basic Proximal Gradient Descent Method in Algorithm 9

Algorithm 9 Basic ProxGD Method

```
Input: $x_0$
  $x = x_0$
  while Stop Conditions do
    compute gradient $g$ of the first function $g(x)$
    choose a proper step size $t$
    $x = \text{Prox}_{th}(x - tg)$
  end while
Output: $x$
```

Then we can implement ProxGD Method to this problem. Basic idea is as same as above, we show the improved ProxGD Method to adapt our problem in order to converge quickly as follows,

Where the hyperparameters mentioned above are as follows,

- err: 1e-10

Algorithm 10 Improved ProxGD Method for this problem

```
Input: $x_0,\ A,\ b,\ \mu$, Multi_$\mu$_Schedule, Step_Size_Schedule
  $x = x_0$
  $t = \frac{1}{||A||_2^2}$ # set the fixed step size
  for $c$ in Multi_$\mu$_Schedule, $\alpha$ in Step_Size_Schedule do
    $\mu_{now} = c\mu$
    $k = 1$
    while Stop Conditions do
      $g = A^T(Ax - b)$ # compute gradient of $g(x)$
      $x = \text{Prox}_{th}(x - tg)$
      $k = k + 1$ # update number of iteration in this epoch
    end while
  end for
Output: $x$
```

- Multi_μ_Schedule: [1e+4,1e+3,1e+2,1e+1,1]
- Max_Iter_Schedule: [5e+2,5e+2,5e+2,5e+2,1e+6]

3.6 (f) FProxGD

Fast proximal gradient method for the primal problem.

3.6.1 Algorithm & Numerical Experiment

Proximal function and other detailed information are the same as they described above.

First we show the basic Fast Proximal Gradient Descent Method in Algorithm 11

Algorithm 11 Basic FProxGD Method

```
Input: $x_0$
  $x^{(0)} = x_0$
  $v^{(0)} = x_0$
  $k = 1$
  while Stop Conditions do
    $y = (1 - \theta_k)x^{(k-1)} + \theta_k v^{(k-1)}$
    compute gradient $g$ at $y$
    choose a proper step size $t$
    $x^{(k)} = \text{Prox}_{th}(y - tg)$
    $v^{(k)} = x^{(k-1)} + \frac{1}{\theta_k}(x^{(k)} - x^{(k-1)})$
    $k = k + 1$
  end while
Output: $x^{(k-1)}$
```

Then we can implement FProxGD Method to this function. Basic idea is as same as above, we show the improved FProxGD Method as follows,

Algorithm 12 Improved FProxGD Method for this problem

```
Input: $x_0,\ A,\ b,\ \mu$, Multi_$\mu$_Schedule, Step_Size_Schedule
  for $c$ in Multi_$\mu$_Schedule, $\alpha$ in Step_Size_Schedule do
    $t = \frac{1}{||A||_2^2}$ # set the fixed step size
    $x^{(0)} = x_0$
    $v^{(0)} = x_0$
    $\mu_{now} = c\mu$
    $k = 1$
    while Stop Conditions do
      $\theta_k = \frac{2}{k+1}$
      $y = (1-\theta_k)x^{(k-1)} + \theta_k v^{(k-1)}$
      $g = A^T(Ay - b)$ # compute gradient of $g(x)$ at $y$
      $x^{(k)} = \text{Prox}_{th}(y - tg)$
      $v^{(k)} = x^{(k-1)} + \frac{1}{\theta_k}(x^{(k)} - x^{(k-1)})$
      $k = k + 1$ # update number of iteration in this epoch
    end while
  end for
Output: $x^{(k-1)}$
```

Where the hyperparameters mentioned above are as follows,

- err: 1e-10
- Multi_μ_Schedule: [1e+4,1e+3,1e+2,1e+1,1]
- Max_Iter_Schedule: [5e+2,5e+2,5e+2,5e+2,1e+6]

3.7 (g) ALM_dual

Augmented Lagrangian method for the dual problem.

3.7.1 Algorithm & Numerical Experiment

First we can reformulate the primal problem as follows,

$$\begin{aligned} \min_{x,y} \quad & \tfrac{1}{2}||y-b||_2^2 + \mu||x||_1 \\ s.t. \quad & Ax - y = 0. \end{aligned} \tag{9}$$

Then we can write the Lagrange function and the dual problem of the above problem,

$$L(x,y,\lambda) = \frac{1}{2}||y-b||_2^2 + \mu||x||_1 + \lambda^T(Ax-y), \tag{10}$$

$$g(\lambda) \ = \ \inf_{x,y} L(x,y,\lambda) = -(\frac{1}{2}||\lambda||_2^2 + b^T\lambda) - I_{||\cdot||_\infty \le \mu}(A^T\lambda), \tag{11}$$

where I is the indicator function.

After that, we rewrite this dual problem as follows,

$$\begin{array}{ll} \min_{z,w} & f_1(z) + f_2(w) \\ s.t. & A^T z - w = 0, \end{array} \tag{12}$$

where

$$f_1(z) \ = \ \frac{1}{2}||z||_2^2 + b^T z, \tag{13}$$

$$f_2(w) \ = \ I_{||\cdot||_\infty \le \mu}(w). \tag{14}$$

Our goal is to implement Augmented Lagrange Method (ALM) to the above problem, the point is to compute

$$(\hat{z}, \hat{w}) = \arg\min_{z,w}(\frac{1}{2}||z||_2^2 + b^T z + I_{||\cdot||_\infty \le \mu}(w) + \frac{t}{2}||A^T z - w + \frac{1}{t}\lambda||_2^2)$$

$$= \ \arg \min_{z,||w||_\infty \le \mu} (\frac{1}{2}||z||_2^2 + b^T z + \frac{t}{2}||A^T z - w + \frac{1}{t}\lambda||_2^2). \tag{15}$$

First we compute the gradient w.r.t. z:

$$z + b + A(tA^T z - tw + \lambda) = 0,$$

then we have

$$z = (I + tAA^T)^{-1}(tAw - A\lambda - b). \tag{16}$$

Consider the inequality constraint of w, we can get the equation which w should have,

$$w = P_{||\cdot||_\infty \le \mu}(A^T z + \frac{1}{t}\lambda), \tag{17}$$

where $P_C(u)$ is the projection of u on set C. But it's difficult to solve it, so we only solve it approximately.

We can regard this sub-optimization problem in this iteration as a two level optimization, that is, we can first fix w to compute an optimal $z = z(w)$, then put this $z(w)$ into the object function and compute the optimal w of current object function which is only w.r.t. w.

As we said above, first we put (16) into (15), then it will be only w.r.t. w as follows,

$$\hat{w} \ = \ \arg \min_{||w||_\infty \le \mu} -\frac{1}{2}(tAw - A\lambda - b)^T \left(I + tAA^T\right)^{-1} (tAw - A\lambda - b) - \lambda^T w + \frac{t}{2} w^T w, \tag{18}$$

then compute the gradient of (18) w.r.t. w without the inequality constraint,

$$\left(t^2 A^T \left(I + tAA^T\right)^{-1} A - tI\right) w = -\lambda + tA^T \left(I + tAA^T\right)^{-1} (A\lambda + b),$$

that is,

$$w = (t^2 A^T \left(I + tAA^T\right)^{-1} A - tI\right)^{-1} \left(-\lambda + tA^T \left(I + tAA^T\right)^{-1} (A\lambda + b)\right), \quad (19)$$

But if we now project this w to its constrained set, in experiments we found it can not converge to the correct optimal value. We conjecture that it may not be the optimal w of the 'primal' sub-optimization problem, because the current w's projection may be on the boundary of its constrained set like a 'cusp', so it could be far away from the correct optimal w of this problem. So we'd like to add a penalty function of w's norm to let it be closed to the correct optimal, i.e., far away from the cusps of the set's boundary.

According to the above discuss, we improve the object function (18) as follows,

$$\hat{w} = \arg \min_{||w||_\infty \leq \mu} ... + \boldsymbol{\alpha} * \frac{\boldsymbol{t}}{\boldsymbol{2}} \boldsymbol{w^T w}$$
$$= (t^2 A^T \left(I + tAA^T\right)^{-1} A - (1 + \boldsymbol{\alpha}) tI\right)^{-1} \left(-\lambda + tA^T \left(I + tAA^T\right)^{-1} (A\lambda + b)\right) \quad (20)$$

Finally,

$$z = (I + tAA^T)^{-1}(tAw - A\lambda - b). \quad (21)$$

According to KKT condition, we can easily compute x from z as follows,

$$Ax = z + b,$$

and for every element in $A^T z$, if it doesn't equal to μ or $-\mu$, we should delete this column and corresponding element in x, then after the following calculation (22), add 0 to the deleted positions in x,

$$x = (A^T A)^{-1} A^T (z + b). \quad (22)$$

First we show the basic Augmented Lagrangian Method in Algorithm 13,

Algorithm 13 Basic Augmented Lagrangian Method

```
Input: $x_0$
  $x^{(0)} = x_0$
  $w = 0, z = 0$
  step_size $t = 1$
  while Stop Conditions do
    update $w$ by (20)
    project $w$ to its constrained set
    update $z$ by (21)
  end while
  compute $x$ from $z$ by (22)
Output: $x$
```

Then we can implement Augmented Lagrangian Method to this problem. Basic idea is as same as above, we show the improved ALM as follows,

Where the hyperparameters mentioned above are as follows,

Algorithm 14 Improved ALM for this problem

```
Input: $x_0,\ A,\ b,\ \mu$, Multi_$\mu$_Schedule, Step_Size_Schedule
  for $c$ in Multi_$\mu$_Schedule, $\alpha$ in Step_Size_Schedule do
    $t = 1$ # set the fixed step size
    $k = 1$
    while Stop Conditions do
      update $w$ by (20)
      project $w$ to its constrained set
      update $z$ by (21)
      $k = k + 1$ # update number of iteration in this epoch
    end while
  end for
  compute $x$ from $z$ by (22)
Output: $x$
```

- err: 1e-10
- Multi_μ_Schedule: [1e+4,1e+3,1e+2,1e+1,1]
- Max_Iter_Schedule: [50,50,50,50,1e+6]
- slack for w: 1

3.8 (h) ADMM_dual

Alternating direction method of multipliers for the dual problem.

3.8.1 Algorithm & Numerical Experiment

The dual form of the primal problem is as same as (12), then we have,

$$\begin{aligned} w_k &= \arg\min_w (I_{||\cdot||_\infty \le \mu}(w) - \lambda_{k-1}^T w + \frac{t}{2}||A^T z - w||_2^2) \\ &= P_{||\cdot||_\infty \le \mu}(A^T z_{k-1} + \frac{1}{t}\lambda_{k-1}), \quad (23) \\ z_k &= \arg\min_z (b^T z + \frac{1}{2}||z||_2^2 + \lambda_{k-1}^T A^T z + \frac{t}{2}||A^T z - w||_2^2) \\ &= (I + tAA^T)^{-1}(tAw_k - A\lambda_{k-1} - b), \quad (24) \\ \lambda_k &= \lambda_{k-1} + t(A^T z - w). \quad (25) \end{aligned}$$

According to KKT condition, we can easily compute x from λ as follows,

$$x = -\lambda, \quad (26)$$

so during the iteration, we can use x to compute directly instead of λ.
First we show the basic ADMM in Algorithm 15

Algorithm 15 Basic ADMM

```
Input: $x_0$
  $x^{(0)} = x_0$
  $k = 1$
  while Stop Conditions do
    compute gradient $g$ at $y$
    choose a proper step size $t$
    update $w$ by (23)
    update $z$ by (24)
    update $\lambda$ by (25)
    $k = k + 1$
  end while
  $x = -\lambda$
Output: $x$
```

Algorithm 16 Improved ADMM for this problem

```
Input: $x_0,\ A,\ b,\ \mu$, Multi_$\mu$_Schedule, Step_Size_Schedule
  for $c$ in Multi_$\mu$_Schedule, $\alpha$ in Step_Size_Schedule do
    $t = 1$ # set the fixed step size
    $x^{(0)} = x_0$
    $\mu_{now} = c\mu$
    $k = 1$
    while Stop Conditions do
      update $w$ by (23)
      update $z$ by (24)
      update $\lambda$ by (25)
      $k = k + 1$ # update number of iteration in this epoch
    end while
  end for
  $x = -\lambda$
Output: $x$
```

Then we can implement ADMM to this problem. Basic idea is as same as above, we show the improved ADMM as follows,

Where the hyperparameters mentioned above are as follows,

- err: 1e-8
- Multi_μ_Schedule: [1e+4,1e+3,1e+2,1e+1,1]
- Max_Iter_Schedule: [50,50,50,50,1e+6]

3.9 (i) ADMM_lprimal

Alternating direction method of multipliers with linearization for the primal problem.

3.9.1 Algorithm & Numerical Experiment

First we can rewrite the primal problem as follows,

$$\begin{aligned} \min_{x,w} \quad & \tfrac{1}{2}||Ax-b||_2^2 + \mu||w||_1 \\ s.t. \quad & x - w = 0, \end{aligned} \tag{27}$$

the Lagrange function of this problem is

$$\frac{1}{2}||Ax-b||_2^2 + \mu||z||_1| + \lambda^T(x-z) + \frac{t}{2}||x-w||_2^2 \tag{28}$$

then we implement ADMM to (27),

$$\begin{aligned} x_k &= \arg\min_x(\frac{1}{2}||Ax-b||_2^2 + \frac{t}{2}||x - w_{k-1} + \frac{\lambda_{k-1}}{t}||_2^2) \\ &= (tI + A^TA)^{-1}(A^Tb + tw_{k-1} - \lambda_{k-1}), \end{aligned} \tag{29}$$

$$\begin{aligned} w_k &= \arg\min_w(\mu||w||_1| + \frac{t}{2}||x_k - w + \frac{\lambda_{k-1}}{t}||_2^2)) \\ &\approx P_{\frac{\mu}{t}||\cdot||_1}(x_k + \frac{\lambda_{k-1}}{t}), \end{aligned} \tag{30}$$

$$\lambda_k = \lambda_{k-1} + t(x_k - w_k). \tag{31}$$

First we show the basic ADMM in Algorithm 17

Algorithm 17 Basic ADMM

```
Input: $x_0$
  $x^{(0)} = x_0$
  $k = 1$
  while Stop Conditions do
    compute gradient $g$ at $y$
    choose a proper step size $t$
    update $x$ by (29)
    update $w$ by (30)
    update $\lambda$ by (31)
    $k = k + 1$
  end while
Output: $x$
```

Then we can implement ADMM to this problem. Basic idea is as same as above, we show the improved ADMM as follows,

Where the hyperparameters mentioned above are as follows,

- err: 1e-8
- Multi_μ_Schedule: [1e+4,1e+3,1e+2,1e+1,1]
- Max_Iter_Schedule: [50,50,50,50,1e+6]

Algorithm 18 Improved ADMM for this problem

```
Input: $x_0,\ A,\ b,\ \mu$, Multi_$\mu$_Schedule, Step_Size_Schedule
  for $c$ in Multi_$\mu$_Schedule, $\alpha$ in Step_Size_Schedule do
    $t = 1$ # set the fixed step size
    $x^{(0)} = x_0$
    $\mu_{now} = c\mu$
    $k = 1$
    while Stop Conditions do
      update $x$ by (29)
      update $w$ by (30)
      update $\lambda$ by (31)
      $k = k + 1$ # update number of iteration in this epoch
    end while
  end for
Output: $x$
```