SI151A
Convex Optimization and its Applications in Information Science, Fall 2024
Homework 3

Name: **Zhou Shouchen**
Student ID: **2021533042**

Due on Dec. 16, 2024, 11:59 AM

Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

- You are required to write down all the major steps towards making your conclusions; otherwise you may obtain limited points ($\leq 20\%$) of the problem.
- Write your homework in English; otherwise you will get no points of this homework.
- Do your homework by yourself. Any form of plagiarism will lead to 0 point of this homework. If more than one plagiarisms during the semester are identified, we will prosecute all violations to the fullest extent of the university regulations, including but not limited to failing this course, academic probation, or expulsion from the university.
- If you have any doubts regarding the grading, you need to contact the instructor or the TAs within two days since the grade is announced.

I. CVX Programming

1. Consider the following Tschebyshev approximation problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p} \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_\infty,$$

where $\mathbf{A}\in\mathbb{R}^{n\times p}$ and $\mathbf{b}\in\mathbb{R}^n$, and $\|\mathbf{u}\|_\infty := \max\{|\mathbf{u}_i| \mid 1\le i\le p\}$.

(a) Reformulate it in LP. (10 points)

(b) Use CVX to solve the original problem and the LP form and report your results respectively. The initialisation part is given below.(10 points)

```
n = 10; p = 20;
A = randn(n, p);
x_org = randn(p, 1);
b = A * x_org + (1e-2) *randn(n, 1);
```

Solution

(a) Let $t = \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_\infty \ge 0$, then we have $-t\mathbf{1} \le \mathbf{A}\mathbf{x}-\mathbf{b} \le t\mathbf{1}$.
So the origin problem can be reformulated as:

$$\begin{aligned} \min_{t\in\mathbb{R},\mathbf{x}\in\mathbb{R}^p} \quad & t \\ \text{s.t.} \quad & -t\mathbf{1} \le \mathbf{A}\mathbf{x}-\mathbf{b} \le t\mathbf{1} \\ & t \ge 0 \end{aligned}$$

Which is a LP problem.

(b) (1) The code to solve for the original problem and the reformulated LP problem is as follows:

```
% clear
clear; clc; close all;

% initialization
n = 10; p = 20;
A = randn(n, p);
x_org = randn(p, 1);
b = A * x_org + (1e-2) * randn(n, 1);

% origin problem
disp('==================== Origin Problem ====================');
cvx_begin
    variable x(p)
    minimize norm(A * x - b, inf)
cvx_end

disp('CVX Status:');
disp(cvx_status);
disp('Optimal value:');
disp(cvx_optval);

% reform problem
disp('==================== Reform Problem ====================');
cvx_begin
    variable x(p)
    variable t
    minimize (t)
    subject to
        -t * ones(n, 1) <= A * x - b;
        A * x - b <= t * ones(n, 1);
        t >= 0;
cvx_end

disp('CVX Status:');
disp(cvx_status);
disp('Optimal value:');
disp(cvx_optval);
```

(2) And the results ran by the code with CVX with **SDPT3** solver are as follows:

```
==================== Origin Problem ====================

Calling SDPT3 4.0: 51 variables, 20 equality constraints
```

```
------------------------------------------------------------

 num. of constraints = 20
 dim. of socp   var  = 20,   num. of socp blk  = 10
 dim. of linear var  = 10
 dim. of free   var  = 21 *** convert ublk to lblk
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
   NT      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|7.2e+00|1.5e+02|9.8e+04| 1.151555e-09  0.000000e+00| 0:0:00| chol  1  1
 1|1.000|0.747|1.1e-06|3.7e+01|6.3e+03| 4.408195e+01  4.641006e-01| 0:0:00| chol  1  1
 2|1.000|0.985|2.1e-06|5.7e-01|7.6e+01| 4.055019e+01  1.831781e-02| 0:0:00| chol  1  1
 3|1.000|0.856|1.7e-07|8.2e-02|5.4e+00| 4.248205e+00  2.388919e-03| 0:0:00| chol  1  1
 4|0.900|0.539|1.4e-08|3.8e-02|6.9e-01| 4.102975e-01  4.095772e-05| 0:0:00| chol  1  1
 5|0.982|0.974|2.8e-10|1.0e-03|1.2e-02| 7.698398e-03  1.757785e-05| 0:0:00| chol  1  1
 6|0.988|0.988|4.9e-12|1.3e-05|1.4e-04| 9.331806e-05  2.780875e-06| 0:0:00| chol  1  1
 7|0.989|0.988|2.0e-12|2.6e-06|1.1e-05| 1.037847e-06  3.121756e-07| 0:0:00| chol  1  1
 8|1.000|0.989|1.3e-12|2.0e-07|7.4e-07| 3.264173e-08  3.129853e-09| 0:0:00| chol  1  1
 9|1.000|0.989|1.2e-12|1.4e-08|5.2e-08| 2.271110e-09  8.884006e-12| 0:0:00| chol  1  2
10|0.596|0.944|6.2e-13|9.7e-10|4.7e-09| 1.198669e-09 -4.691367e-12| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   = 10
 primal objective value =  1.19866920e-09
 dual   objective value = -4.69136711e-12
 gap := trace(XZ)       = 4.72e-09
 relative gap           = 4.72e-09
 actual relative gap    = 1.20e-09
 rel. primal infeas (scaled problem)   = 6.19e-13
 rel. dual     "        "       "      = 9.71e-10
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 2.5e+00, 3.2e-01, 4.5e-01
 norm(A), norm(b), norm(C) = 2.2e+01, 1.1e+01, 2.4e+00
 Total CPU time (secs)  = 0.11
 CPU time per iteration = 0.01
 termination code       =  0
 DIMACS: 1.1e-12  0.0e+00  1.2e-09  0.0e+00  1.2e-09  4.7e-09
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +1.19867e-09

CVX Status:
Solved
Optimal value:
   1.1987e-09

==================== Reform Problem ====================

Calling SDPT3 4.0: 21 variables, 21 equality constraints
   For improved efficiency, SDPT3 is solving the dual problem.
------------------------------------------------------------

 num. of constraints = 21
 dim. of linear var  = 22
 number of nearly dependent constraints = 10
 To remove these constraints, re-run sqlp.m with OPTIONS.rmdepconstr = 1.
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
   NT      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|1.0e+02|4.8e+00|3.2e+03|-7.809891e-09  0.000000e+00| 0:0:00| chol  1  1
 1|0.994|0.996|6.0e-01|4.9e-02|3.3e+01|-2.470275e-05 -1.438311e+01| 0:0:00| chol  1  1
 2|1.000|1.000|8.6e-08|3.2e-03|4.6e+00| 2.686130e-09 -4.512945e+00| 0:0:00| chol  1  1
 3|1.000|0.989|1.9e-08|3.5e-04|5.7e-02| 6.729059e-10 -4.820207e-02| 0:0:00| chol  1  1
 4|1.000|0.989|6.2e-09|3.5e-05|1.2e-03| 2.201996e-10 -4.307715e-04| 0:0:00| chol  1  1
 5|1.000|0.989|1.7e-09|3.5e-06|6.0e-05| 6.093492e-11  5.114873e-06| 0:0:00| chol  1  1
```

```
 6|1.000|0.989|2.7e-10|3.9e-08|6.6e-07|-2.949282e-11  5.609093e-08| 0:0:00| chol  1  1
 7|1.000|0.989|1.8e-11|4.9e-10|1.5e-08|-6.288997e-13 -6.252401e-09| 0:0:00| chol  2  2
 8|1.000|0.989|3.3e-13|7.1e-12|3.6e-10| 3.948231e-14 -2.258790e-10| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   =  8
 primal objective value =  3.94823063e-14
 dual   objective value = -2.25878967e-10
 gap := trace(XZ)       = 3.56e-10
 relative gap           = 3.56e-10
 actual relative gap    = 2.26e-10
 rel. primal infeas (scaled problem)   = 3.27e-13
 rel. dual     "        "       "      = 7.10e-12
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 4.9e+00, 2.7e+00, 1.1e-09
 norm(A), norm(b), norm(C) = 2.1e+01, 2.0e+00, 1.5e+01
 Total CPU time (secs)  = 0.09
 CPU time per iteration = 0.01
 termination code       =  0
 DIMACS: 3.3e-13  0.0e+00  1.7e-11  0.0e+00  2.3e-10  3.6e-10
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +2.25879e-10

CVX Status:
Solved
Optimal value:
   2.2588e-10
```

And we can find that the origin problem and the reformulated problem solved by CVX have the similar results(The origin problem solved with optimal value: $1.1987e-09$, and the reformulated problem solved with optimal value: $2.25879e-10$), which has no significant difference.

And their running speed has no significant difference(The origin problem solved with CPU time: 0.11 in 10 iterations, and the reformulated problem solved with CPU time: $0.09s$ in 8 iterations).

2. Consider the SOCP problem in HW2:

$$\min_{x \in \mathbb{R}^p} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 + \lambda \|\mathbf{D}\mathbf{x}\|_1,$$

where $\mathbf{A} \in \mathbb{R}^{n \times p}$, $\mathbf{D} \in \mathbb{R}^{n \times p}$, and $\mathbf{b} \in \mathbb{R}^n$

(a) Use CVX to solve the original problem and SOCP form and report your results respectively. The initialisation part is given below.(10 points)

(b) Compare the results, and answer 1), how many iterations do both problems cost respectively? 2), What if we change the solver?(10 points)

```
n = 10; p = 20;
A = randn(n, p);
x_org = randn(p, 1);
b = A*x_org+1e-2*randn(n, 1);
D = randn(n, p);
```

Solution

(a) From homework 2, we know this problem could be reformulated as a SOCP:

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^p, t \in \mathbb{R}, \mathbf{s} \in \mathbb{R}^p} \quad & t + \lambda \sum_{i=1}^{p} s_i \\ \text{s.t.} \quad & \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \le t \\ & t \ge 0 \\ & -s_i \le (\mathbf{D}\mathbf{x})_i \le s_i, \quad i = 1, 2, \cdots, p \\ & s_i \ge 0, \quad i = 1, 2, \cdots, p \end{aligned}$$

And I chose $\lambda = 0.1$ as the hyperparameter.
The code to solve for the original problem and the reformulated LP problem is as follows:

```
% clear
clear; clc; close all;

% initialization
n = 10; p = 20;
A = randn(n, p);
x_org = randn(p, 1);
b = A * x_org + (1e-2) * randn(n, 1);
D = randn(n, p);
lambda = 0.1;

% origin problem
disp('==================== Origin Problem ====================');
cvx_begin
    variable x(p)
    minimize norm(A * x - b, 2) + lambda * norm(D * x, 1)
cvx_end

disp('CVX Status:');
disp(cvx_status);
disp('Optimal value:');
disp(cvx_optval);


% reform problem
disp('==================== Reform Problem ====================');
cvx_begin
    variable x(p)
    variable t
    variable s(n)
    minimize t + lambda * sum(s)
    subject to
        norm(A * x - b, 2) <= t;
        t >= 0;
        -s <= D * x;
        D * x <= s;
        s >= 0 * ones(n, 1);
cvx_end

```

```
disp('CVX Status:');
disp(cvx_status);
disp('Optimal value:');
disp(cvx_optval);
```

(b) (1) And the results ran by the code with CVX using the **SDPT3** solver are as follows:

```
==================== Origin Problem ====================

Calling SDPT3 4.0: 51 variables, 20 equality constraints
------------------------------------------------------------

 num. of constraints = 20
 dim. of socp   var  = 31,   num. of socp blk  = 11
 dim. of free   var  = 20 *** convert ublk to lblk
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
    NT      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|9.4e-01|2.1e+02|1.6e+05| 2.153345e+01  0.000000e+00| 0:0:00| chol  1  1
 1|1.000|0.980|2.2e-07|4.2e+00|6.9e+02| 1.823944e+01  2.272040e+00| 0:0:00| chol  1  1
 2|0.913|0.951|3.9e-07|2.2e-01|1.0e+01| 2.510379e+00  5.194200e-01| 0:0:00| chol  1  1
 3|0.932|0.899|6.2e-07|2.3e-02|7.4e-01| 6.017452e-01  1.946068e-01| 0:0:00| chol  1  1
 4|0.982|0.550|2.8e-07|1.0e-02|1.6e-01| 2.421165e-01  2.233709e-01| 0:0:00| chol  1  1
 5|0.223|0.131|2.1e-07|1.3e-02|2.4e-01| 1.598451e-01  2.236800e-01| 0:0:00| chol  1  1
 6|0.348|0.355|1.4e-07|8.3e-03|1.6e-01| 5.519695e-02  2.133340e-01| 0:0:00| chol  1  1
 7|1.000|0.151|1.2e-07|1.0e-02|2.4e-01| 1.401985e-01  1.605340e-01| 0:0:00| chol  1  1
 8|0.538|0.437|5.5e-08|5.7e-03|1.4e-01| 2.968233e-02  1.293434e-01| 0:0:00| chol  1  1
 9|1.000|0.549|6.2e-09|2.6e-03|7.5e-02| 1.585422e-02  4.907454e-02| 0:0:00| chol  1  1
10|0.989|0.920|1.3e-09|2.1e-04|6.2e-03| 6.610269e-04  3.247607e-03| 0:0:00| chol  1  1
11|0.986|0.983|3.2e-10|1.0e-04|8.5e-04| 9.614834e-06  5.406719e-05| 0:0:00| chol  1  1
12|0.989|0.989|1.2e-11|1.4e-05|1.1e-04| 1.069771e-07  5.995755e-07| 0:0:00| chol  1  1
13|1.000|0.944|7.4e-14|1.9e-06|1.5e-05| 2.023181e-07 -2.204021e-07| 0:0:00| chol  1  1
14|1.000|0.988|2.3e-14|2.5e-07|2.1e-06| 3.607555e-08 -5.014128e-08| 0:0:00| chol  1  1
15|0.574|0.943|3.1e-14|3.4e-08|3.0e-07| 1.903887e-08 -1.087372e-08| 0:0:00| chol  1  1
16|0.584|0.938|5.8e-15|5.0e-09|5.1e-08| 1.006371e-08 -5.251112e-09| 0:0:00| chol  1  1
17|0.530|0.932|2.6e-14|8.5e-10|1.2e-08| 5.319355e-09 -1.740390e-09| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   = 17
 primal objective value =  5.31935487e-09
 dual   objective value = -1.74038977e-09
 gap := trace(XZ)       = 1.22e-08
 relative gap           = 1.22e-08
 actual relative gap    = 7.06e-09
 rel. primal infeas (scaled problem)   = 2.63e-14
 rel. dual     "        "       "      = 8.53e-10
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 3.5e+01, 1.6e-09, 1.0e+00
 norm(A), norm(b), norm(C) = 2.9e+01, 1.6e+01, 2.0e+00
 Total CPU time (secs)  = 0.08
 CPU time per iteration = 0.00
 termination code       =  0
 DIMACS: 4.7e-14  0.0e+00  8.7e-10  0.0e+00  7.1e-09  1.2e-08
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +5.31935e-09

CVX Status:
Solved
Optimal value:
   5.3194e-09

==================== Reform Problem ====================

Calling SDPT3 4.0: 43 variables, 32 equality constraints
   For improved efficiency, SDPT3 is solving the dual problem.
------------------------------------------------------------

 num. of constraints = 32
```

```
 dim. of socp   var  = 11,    num. of socp blk  =  1
 dim. of linear var  = 32
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
    NT      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|4.7e+01|3.8e+00|3.3e+03| 0.000000e+00  0.000000e+00| 0:0:00| chol  1  1
 1|0.920|1.000|3.7e+00|3.6e-02|2.8e+02|-7.704659e-03 -2.253207e+01| 0:0:00| chol  1  1
 2|1.000|1.000|2.4e-07|3.6e-03|1.3e+01|-8.187216e-06 -1.317604e+01| 0:0:00| chol  1  1
 3|1.000|0.989|1.1e-07|3.9e-04|1.5e-01|-3.448245e-07 -1.440054e-01| 0:0:00| chol  1  1
 4|1.000|0.987|3.5e-08|4.0e-05|1.9e-03|-6.545340e-08 -1.620071e-03| 0:0:00| chol  1  1
 5|1.000|0.970|8.8e-09|4.7e-06|5.9e-05|-7.348773e-08 -2.117986e-05| 0:0:00| chol  1  1
 6|1.000|0.989|1.5e-09|5.4e-08|6.7e-07|-1.229281e-08 -2.361627e-07| 0:0:00| chol  1  1
 7|1.000|0.992|8.2e-12|5.8e-10|1.2e-08|-6.736756e-11 -7.761193e-09| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   =  7
 primal objective value = -6.73675598e-11
 dual   objective value = -7.76119257e-09
 gap := trace(XZ)       = 1.25e-08
 relative gap           = 1.25e-08
 actual relative gap    = 7.69e-09
 rel. primal infeas (scaled problem)   = 8.16e-12
 rel. dual     "        "       "      = 5.76e-10
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 1.3e+00, 3.5e+01, 5.1e-08
 norm(A), norm(b), norm(C) = 2.6e+01, 2.0e+00, 1.6e+01
 Total CPU time (secs)  = 0.11
 CPU time per iteration = 0.02
 termination code       =  0
 DIMACS: 8.4e-12  0.0e+00  1.0e-09  0.0e+00  7.7e-09  1.2e-08
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +7.76119e-09

CVX Status:
Solved
Optimal value:
   7.7612e-09
```

This result is using the **SDPT3** solver.

And we can find that the origin problem and the reformulated problem solved by CVX have the similar results(The origin problem solved with optimal value: $5.3194e-09$, and the reformulated problem solved with optimal value: $7.7612e-09$), which has no significant difference.

And for their running speed: The origin problem solved with CPU time: $0.08s$ in 17 iterations, and the reformulated problem solved with CPU time: $0.11s$ in 7 iterations. The reformulated problem has less iterations but more CPU time.

(2) Then we change the solver to **SeDuMi**, and the results are as follows:

```
==================== Origin Problem ====================

Calling SeDuMi 1.3.4: 51 variables, 20 equality constraints
------------------------------------------------------------
SeDuMi 1.3.4 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
eqs m = 20, order n = 25, dim = 53, blocks = 3
nnz(A) = 430 + 0, nnz(ADA) = 400, nnz(L) = 210
 it :     b*y       gap     delta   rate    t/tP*   t/tD*    feas cg cg   prec
  0 :            4.71E+00 0.000
  1 :   2.52E+00 1.68E+00 0.000 0.3573 0.9000 0.9000   2.09  1  1  1.3E+00
  2 :   6.64E-01 9.30E-01 0.000 0.5525 0.9000 0.9000   2.84  1  1  5.1E-01
  3 :   2.54E-01 2.56E-01 0.000 0.2750 0.9000 0.9000   2.08  1  1  1.6E-01
  4 :   2.09E-01 1.19E-01 0.000 0.4641 0.9000 0.9000   1.15  1  1  1.1E-01
  5 :   1.65E-01 8.27E-02 0.000 0.6974 0.9000 0.9000   0.49  1  1  1.4E-01
  6 :   5.28E-02 2.69E-02 0.000 0.3255 0.9000 0.9000   0.61  1  1  1.5E-01
  7 :   9.76E-04 6.04E-04 0.000 0.0224 0.9900 0.9900   0.99  1  1  1.6E-01
  8 :   5.75E-09 8.86E-10 0.132 0.0000 1.0000 1.0000   1.00  1  1  8.1E-07
```

```
  9 :    2.47E-11 2.90E-12 0.208 0.0033 0.9979 0.9979   1.00  1  1  2.9E-09

iter seconds digits       c*x               b*y
  9      0.3   6.5  3.4358369747e-11  2.4675681591e-11
|Ax-b| =   6.9e-12, [Ay-c]_+ =   2.5E-12, |x|=  2.7e+01, |y|=  5.0e-12

Detailed timing (sec)
   Pre          IPM          Post
2.658E-01     2.996E-01     1.089E-02
Max-norms: ||b||=5.255533e+00, ||c|| = 1,
Cholesky |add|=0, |skip| = 0, ||L.L|| = 1.81398.
------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +3.43584e-11

CVX Status:
Solved
Optimal value:
   3.4358e-11

==================== Reform Problem ====================

Calling SeDuMi 1.3.4: 43 variables, 32 equality constraints
   For improved efficiency, SeDuMi is solving the dual problem.
------------------------------------------------------------
SeDuMi 1.3.4 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
eqs m = 32, order n = 35, dim = 44, blocks = 2
nnz(A) = 634 + 0, nnz(ADA) = 894, nnz(L) = 463
 it :     b*y       gap    delta  rate   t/tP*  t/tD*   feas cg cg  prec
  0 :            1.03E+00 0.000
  1 :  -1.68E+00 4.23E-01 0.000 0.4105 0.9000 0.9000   2.81  1  1  2.0E+00
  2 :  -3.34E-01 2.42E-01 0.000 0.5717 0.9000 0.9000   4.18  1  1  5.3E-01
  3 :  -1.12E-02 4.64E-02 0.000 0.1919 0.9000 0.9000   3.34  1  1  9.2E-02
  4 :  -9.63E-05 2.07E-04 0.000 0.0045 0.9990 0.9990   1.13  1  1  6.0E-03
  5 :  -4.66E-12 3.46E-11 0.000 0.0000 1.0000 1.0000   1.00  1  1  3.0E-10

iter seconds digits       c*x               b*y
  5      0.1   7.5  2.9291503328e-14 -4.6586468057e-12
|Ax-b| =   3.9e-11, [Ay-c]_+ =   1.7E-11, |x|=  8.6e-01, |y|=  1.7e+01

Detailed timing (sec)
   Pre          IPM          Post
4.458E-02     3.980E-02     2.444E-03
Max-norms: ||b||=1, ||c|| = 5.255533e+00,
Cholesky |add|=0, |skip| = 0, ||L.L|| = 2.96617.
------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +4.65865e-12

CVX Status:
Solved
Optimal value:
   4.6586e-12
```

This result is using the **SeDuMi** solver.
And we can find that the origin problem and the reformulated problem solved by CVX have the similar results(The origin problem solved with optimal value: $3.4358 - 011$, and the reformulated problem solved with optimal value: $4.6586 - 02$), which has no significant difference.

And for their running speed: The origin problem solved with CPU time: $0.57629s$ in 9 iterations, and the reformulated problem solved with CPU time: $0.086824s$ in 5 iterations. The reformulated problem has less iterations and much less CPU time.

II. Sparse Optimization

Consider a *linear system of equations* $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}$, where $\mathbf{D}$ is an *underdetermined* $m \times p$ matrix ($m < p$) and $\mathbf{x} \in \mathbb{R}^m$, $\boldsymbol{\alpha} \in \mathbb{R}^p$. The matrix $\mathbf{D}$ (typically assumed to be full-rank) is referred to as the *dictionary*, and $\mathbf{x}$ is a signal of interest. The core sparse representation problem is defined as the quest for the sparsest possible representation $\boldsymbol{\alpha}$ satisfying $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}$. Due to the underdetermined nature of $\mathbf{D}$, this linear system admits in general infinitely many possible solutions, and among these, we seek the one with the fewest non-zeros. This is the most popular problem in *compress sensing*.

1. Given $\mathbf{x}$ and $\mathbf{D}$, we want to find the $\boldsymbol{\alpha}$ with the fewest non-zeros. Derive the problem. (5 points) Relax the problem so that the **objective function** is convex. (5 points) (*hint*: You can use the L_0 norm to form the problem and use some other norm to approximate the L_0 norm.)
2. Derive the **Lagrangian** $L(\mathbf{x}, \lambda, \mathbf{v})$ of the relaxed problem. (5 points) Consider the problem $\min\limits_{\mathbf{x}} L(\mathbf{x}, \lambda, \mathbf{v})$, reformulate it in the ADMM form. (5 points)

Solution

(1) <1>. To have fewest non-zeros, which means that we want to minimize $\operatorname{card}(\boldsymbol{\alpha})$, where for a scalar a, $\operatorname{card}(a) = \begin{cases} 1, & \text{if } a \neq 0, \\ 0, & \text{if } a = 0. \end{cases}$. Then $\operatorname{card}(\boldsymbol{\alpha}) = \sum\limits_{i=1}^{p} \operatorname{card}(\alpha_i)$.

Thus, the problem can be formulated as

$$\begin{aligned} \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \quad & \operatorname{card}(\boldsymbol{\alpha}) \\ \text{s.t.} \quad & \mathbf{x} = \mathbf{D}\boldsymbol{\alpha}. \end{aligned}$$

<2>. Since $\operatorname{card}(\boldsymbol{\alpha})$ or we can also say L_0-norm is non-convex, we can relax the problem by using L_1-norm. The problem can be formulated as

$$\begin{aligned} \min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \quad & \|\boldsymbol{\alpha}\|_1 \\ \text{s.t.} \quad & \mathbf{x} = \mathbf{D}\boldsymbol{\alpha}. \end{aligned}$$

Now the objective function is convex, since L_1-norm is convex.

(2) The Lagrangian of the relaxed problem is

$$L(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \|\boldsymbol{\alpha}\|_1 + \boldsymbol{\lambda}^\top (\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})$$

To find the $\min\limits_{\boldsymbol{\alpha} \in \mathbb{R}^p} L(\boldsymbol{\alpha}, \boldsymbol{\lambda})$, we can get the optimization problem as:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} \|\boldsymbol{\alpha}\|_1 + \boldsymbol{\lambda}^\top (\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}) = \|\boldsymbol{\alpha}\|_1 - (\mathbf{D}^\top \boldsymbol{\lambda})^\top \boldsymbol{\alpha} + \boldsymbol{\lambda}^\top \mathbf{x}$$

And since $\boldsymbol{\lambda}$ is the multiplier, $\mathbf{D}$ and $\mathbf{x}$ could be considered as constants, so we can let $\mathbf{c} = -\mathbf{D}^\top \boldsymbol{\lambda}$, and ignore $\boldsymbol{\lambda}^\top \mathbf{x}$. Then the problem can be formulated as:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^p} = \|\boldsymbol{\alpha}\|_1 + \mathbf{c}^\top \boldsymbol{\alpha}$$

To formulate the problem in the ADMM form, we reformulate the problem as

$$\begin{aligned} \min_{\boldsymbol{\alpha}, \mathbf{z} \in \mathbb{R}^p} \quad & \mathbf{c}^\top \boldsymbol{\alpha} + \|\mathbf{z}\|_1 \\ \text{s.t.} \quad & \boldsymbol{\alpha} = \mathbf{z} \end{aligned}$$

Which means that through ADMM, we can get the update rules as

$$\begin{cases} \boldsymbol{\alpha}^{k+1} = \arg\min\limits_{\boldsymbol{\alpha}} \left(\mathbf{c}^\top \boldsymbol{\alpha} + \dfrac{\rho}{2} \|\boldsymbol{\alpha} - \mathbf{z}^k + \mathbf{u}^k\|_2^2 \right) \\ \mathbf{z}^{k+1} = \arg\min\limits_{\mathbf{z}} \left(\|\mathbf{z}\|_1 + \dfrac{\rho}{2} \|\boldsymbol{\alpha}^{k+1} - \mathbf{z} + \mathbf{u}^k\|_2^2 \right) \\ \mathbf{u}^{k+1} = \mathbf{u}^k + \boldsymbol{\alpha}^{k+1} - \mathbf{z}^{k+1} \end{cases}$$

1. For $\boldsymbol{\alpha}$: Let $\phi(\boldsymbol{\alpha}) = \mathbf{c}^\top \boldsymbol{\alpha} + \dfrac{\rho}{2} \|\boldsymbol{\alpha} - \mathbf{z}^k + \mathbf{u}^k\|_2^2$, then

$$\begin{aligned} \frac{\partial \phi}{\partial \boldsymbol{\alpha}} &= \mathbf{c} + \rho(\boldsymbol{\alpha} - \mathbf{z}^k + \mathbf{u}^k) \\ \frac{\partial^2 \phi}{\partial \boldsymbol{\alpha}^2} &= \rho \mathbf{I} \succ 0 \end{aligned}$$

So $\phi(\boldsymbol{\alpha})$ is a convex function, so we just need to let $\dfrac{\partial \phi}{\partial \boldsymbol{\alpha}} = 0 \Rightarrow \boldsymbol{\alpha}^{k+1} = \mathbf{z}^k - \mathbf{u}^k - \dfrac{1}{\rho}\mathbf{c}$.

2. For $\mathbf{z}$: $\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \left(\|\mathbf{z}\|_1 + \frac{\rho}{2}\|\boldsymbol{\alpha}^{k+1} - \mathbf{z} + \mathbf{u}^k\|_2^2 \right) = \arg\min_{\mathbf{z}} \left(\sum_{i=1}^{p} |z_i| + \frac{\rho}{2} \sum_{i=1}^{p} (\alpha_i^{k+1} - z_i + u_i^k)^2 \right)$.
We can see that this is seperable. So let $\psi(z_i) = |z_i| + \frac{\rho}{2}\|\boldsymbol{\alpha}^{k+1} - z_i + \mathbf{u}^k\|_2^2$, since $\psi(z_i)$ is not differentiable, so to find the minimize, we use the subgradient: $0 \in \partial\psi(z_i^*)$.
Where $\partial\psi(z_i) = \partial(|z_i|) + \rho\left(\alpha_i^{k+1} - z_i + u_i^k\right)$.
<1>. If $z_i^* > 0$, then

$$
\begin{aligned}
&\Rightarrow |z_i^*| = z_i \Rightarrow \partial(|z_i^*|) = 1 \\
&\Rightarrow 0 = 1 + \rho\left(\boldsymbol{\alpha}^{k+1} - z_i^* + u_i^k\right) \\
&\Rightarrow z_i^* = \alpha_i^{k+1} + u_i^k - \frac{1}{\rho} > 0 \\
&\Rightarrow \alpha_i^{k+1} + u_i^k > \frac{1}{\rho}
\end{aligned}
$$

<2>. If $z_i^* < 0$, then

$$
\begin{aligned}
&\Rightarrow |z_i^*| = -z_i \Rightarrow \partial(|z_i^*|) = -1 \\
&\Rightarrow 0 = -1 + \rho\left(\boldsymbol{\alpha}^{k+1} - z_i^* + u_i^k\right) \\
&\Rightarrow z_i^* = \alpha_i^{k+1} + u_i^k + \frac{1}{\rho} < 0 \\
&\Rightarrow \alpha_i^{k+1} + u_i^k < -\frac{1}{\rho}
\end{aligned}
$$

<3>. If $z_i^* = 0$, then

$$
\begin{aligned}
&\Rightarrow |z_i^*| = 0 \Rightarrow \partial(|z_i^*|) \in [-1, 1] \\
&\Rightarrow 0 \in [-1, 1] + \rho\left(\boldsymbol{\alpha}^{k+1} - z_i^* + u_i^k\right) \\
&\Rightarrow -1 \le \rho\left(\boldsymbol{\alpha}^{k+1} - z_i^* + u_i^k\right) \le 1 \\
&\Rightarrow -\frac{1}{\rho} \le \alpha_i^{k+1} + u_i^k \le \frac{1}{\rho}
\end{aligned}
$$

Thus, we can get the update rule for $\mathbf{z}$ as

$$
z_i^{k+1} = \begin{cases}
\alpha_i^{k+1} + u_i^k - \frac{1}{\rho}, & \text{if } \alpha_i^{k+1} + u_i^k > \dfrac{1}{\rho}, \\
\alpha_i^{k+1} + u_i^k + \frac{1}{\rho}, & \text{if } \alpha_i^{k+1} + u_i^k < -\dfrac{1}{\rho}, \\
0, & \text{otherwise}
\end{cases}
$$

i.e. $z_i^{k+1} = S_{\frac{1}{\rho}}(\alpha_i^{k+1} + u_i^k) \Rightarrow z^{k+1} = S_{\frac{1}{\rho}}(\boldsymbol{\alpha}^{k+1} + \mathbf{u}^k)$, where $S_{\frac{1}{\rho}}(\cdot)$ is the soft-thresholding operator.

So above all, the update rules for ADMM are

$$
\begin{cases}
\boldsymbol{\alpha}^{k+1} = \mathbf{z}^k - \mathbf{u}^k - \dfrac{1}{\rho}\mathbf{c} \\
\mathbf{z}^{k+1} = S_{\frac{1}{\rho}}(\boldsymbol{\alpha}^{k+1} + \mathbf{u}^k) \\
\mathbf{u}^{k+1} = \mathbf{u}^k + \boldsymbol{\alpha}^{k+1} - \mathbf{z}^{k+1}
\end{cases}
$$

III. **Low-Rank Optimization**

1. Consider a rating matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$ with R_{ij} representing the rating user i gives to movie j. But some R_{ij} are unknown since no one watches all the movies. Thus, the $\mathbf{R}$ may look like blow

$$\mathbf{R} = \begin{bmatrix} 2 & 3 & ? & ? & 5 & ? \\ 1 & ? & 4 & ? & 3 & ? \\ ? & ? & 3 & 2 & ? & 5 \\ 4 & ? & 3 & ? & 2 & 4 \end{bmatrix}$$

According to the above background, we would like to predict how users will like unwatched movies. Unfortunately, the rating matrix is very big, $480,189$ (number of users) times $17,770$ (number of movies) in the Netflix case. But there are much fewer types of people and movies than there are people and movies. So it is reasonable to assume that for each user i, there is a k-dimensional vector $\mathbf{p}_i$ explaining the user's movie taste, and for each movie j, there is also a k-dimensional vector $\mathbf{q}_j$ explaining the movie's appeal. The inner product between these two vectors, $\mathbf{p}_i^\top \mathbf{q}_j$, is the rating user i gives to movie j, i.e.,

$$R_{ij} = \mathbf{p}_i^\top \mathbf{q}_j.$$

Or equivalently in matrix form, $\mathbf{R}$ is factorized as

$$\mathbf{R} = \mathbf{P}^\top \mathbf{Q},$$

where $\mathbf{P} \in \mathbb{R}^{k\times m}$, $\mathbf{Q} \in \mathbb{R}^{k\times n}$, $k \ll \min(m, n)$. It is the same as assuming the matrix $\mathbf{R}$ is of low rank. However, the true rank k is unknown. A natural approach is to find the minimum rank solution $\mathbf{X}$.

(a) Given the rating matrix $\mathbf{R}$ with known entries R_{ij}, where $(i, j) \in \Omega$ and Ω is the set of observed entries, derive the optimization problem. (Not that $\mathbf{P}$ and $\mathbf{Q}$ are also unknown, so they should not be used in the formulation.)(10 points)

(b) In practice, instead of requiring strict equality for the observed entries, we may allow some error ϵ between the entry of solution X_{ij} and the corresponding entry of the observation R_{ij}. Modify the problem in 1 to satisfy the above requirements.(10 points)

Solution

(a) The optimization problem is

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad & \operatorname{rank}(\mathbf{X}) \\ \text{s.t.} \quad & X_{ij} = R_{ij}, \quad (i, j) \in \Omega. \end{aligned}$$

(b) Since we allow some error ϵ between the entry of solution X_{ij} and the corresponding entry of the observation R_{ij}, the optimization problem is

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad & \operatorname{rank}(\mathbf{X}) \\ \text{s.t.} \quad & |X_{ij} - R_{ij}| \le \epsilon, \quad (i, j) \in \Omega. \end{aligned}$$

2. Consider the low-rank matrix recovery problem

$$\begin{aligned} \min_{\mathbf{L}} \quad & \sum_{i=1}^{m} \left(y_i - \mathbf{x}_i^\top \mathbf{L} \mathbf{x}_i\right)^2, \\ s.t. \quad & \operatorname{rank}(\mathbf{L}) \leq r, \end{aligned}$$

where $\mathbf{L} \in \mathbb{R}^{p \times p}$ is a low-rank matrix with $\operatorname{rank}(\mathbf{L}) \leq r \ll p$, $y_i \in \mathbb{R}$, and $\mathbf{x} \in \mathbb{R}^p$. Since the low-rank property is hard to address in practice, we would like to introduce some assumptions on $\mathbf{L}$. Assume that $\mathbf{L} \in \mathbb{S}_+^p$, the factorization method can be used. (Fracterization Method: in the previous problem, the low–rank matrix $\mathbf{R}$ can be factorized as $\mathbf{R} = \mathbf{P}^\top \mathbf{Q}$, where $\mathbf{P} \in \mathbb{R}^{k \times m}$, $\mathbf{Q} \in \mathbb{R}^{k \times n}$, $k \ll \min(m, n)$.)

   (a) Derive the optimization problem with factorization. (10 points)

   (b) Derive the gradient of the objective function in (a). (5 points) Is the objective function convex, concave, or neither? (5 points)

Solution

(a) Since $\operatorname{rank}(\mathbf{L}) \leq r$, and $\mathbf{L} \in \mathbb{S}_+^p$, we can factorize $\mathbf{L}$ as $\mathbf{L} = \mathbf{P}^\top \mathbf{P}$, where $\mathbf{P} \in \mathbb{R}^{r \times p}$. Since $r \ll p$, so $\operatorname{rank}(\mathbf{L}) \leq \min(r, p) = r \Rightarrow \operatorname{rank}(\mathbf{L}) = \operatorname{rank}\left(\mathbf{P}^\top \mathbf{P}\right) = \operatorname{rank}(\mathbf{P}) \leq r$, and $\mathbf{P}^\top \mathbf{P} \in \mathbb{S}_+^p$.

Then the optimization problem can be rewritten as

$$\min_{\mathbf{P} \in \mathbb{R}^{r \times p}} \sum_{i=1}^{m} \left(y_i - \mathbf{x}_i^\top \mathbf{P}^\top \mathbf{P} \mathbf{x}_i\right)^2$$

(b) <1>. Let $f(\mathbf{P}) = \sum_{i=1}^{m} \left(y_i - \mathbf{x}_i^\top \mathbf{P}^\top \mathbf{P} \mathbf{x}_i\right)^2$.

Since $\dfrac{\partial \mathbf{b}^\top \mathbf{X}^\top \mathbf{X} \mathbf{c}}{\partial \mathbf{X}} = \mathbf{X}\left(\mathbf{b}\mathbf{c}^\top + \mathbf{c}\mathbf{b}^\top\right) \Rightarrow \dfrac{\partial \left(\mathbf{x}_i^\top \mathbf{P}^\top \mathbf{P} \mathbf{x}_i\right)}{\partial \mathbf{P}} = 2\mathbf{P}\mathbf{x}_i\mathbf{x}_i^\top$.

So we can get that

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{P}} &= 2\sum_{i=1}^{m} \left(\mathbf{x}_i^\top \mathbf{P}^\top \mathbf{P} \mathbf{x}_i - y_i\right) \cdot \frac{\partial \left(\mathbf{x}_i^\top \mathbf{P}^\top \mathbf{P} \mathbf{x}_i\right)}{\partial \mathbf{P}} \\ &= 4\sum_{i=1}^{m} \left(\mathbf{x}_i^\top \mathbf{P}^\top \mathbf{P} \mathbf{x}_i - y_i\right) \mathbf{P}\mathbf{x}_i\mathbf{x}_i^\top \end{aligned}$$

<2>. Since the gradient of the objective function is a matrix. In formal, we should take the gradient of the gradient, which is the Hessian matrix, to judge it is positive semidefinite or not to determine the convexity of the objective function.

However, the gradient of a matrix is not a matrix, but a fourth-order tensor. Which is hard to express.

So take $m = r = p = 1$ as an example:

y, x, P are all scalars.

Then we have:

$$\begin{aligned} f(P) &= \left(y - x^2 P^2\right)^2 \\ f'(P) &= 4\left(x^2 P^2 - y\right) \cdot P x^2 = 4P^3 x^4 - 4Pyx^2 \\ f''(P) &= 4x^2(3x^2 P^2 - y) \end{aligned}$$

Since $f''(P)$ is positive or not depends on the value of y, so the objective function is neither convex nor concave in this case.

And we may generalize this conclusion to the general cases. i.e. the objective function is neither convex nor concave.

So above all, the gradient of the objective function is $\dfrac{\partial f}{\partial \mathbf{P}} = 4\sum_{i=1}^{m} \left(\mathbf{x}_i^\top \mathbf{P}^\top \mathbf{P} \mathbf{x}_i - y_i\right) \mathbf{P}\mathbf{x}_i\mathbf{x}_i^\top$.

And the objective function is neither convex nor concave.

References

[1] D. Amelunxen, M. Lotz, M. B. McCoy, and J. A. Tropp, "Living on the edge: Phase transitions in convex programs with random data," *Inf. Inference*, vol. 3, pp. 224-294, Jun. 2014.