$\ell_1 - \ell_1$ Minimization & Sparse Inverse

*Project 1 on the Course "Algorithms for Big Data Analysis".

1st Chen Yihang

Peking University

1700010780

Abstract

In this report, we discuss two topic, i.e. the $\ell_1 - \ell_1$ minimization and sparse covariance estimation. For the first problem, we transform it into a Basis Pursuit problem, and propose the Bregman method with proximal gradient and its accelerated version, linearized Bregman method, and ADMM method based on the dual problem. We compare this method to the commercial software, i.e. CVX, MOSEK, GUROBI in terms of speed and accuracy. In summary, we find the error forgetting phenomenon [Yin and Osher, 2013] of the Bregman iteration, and the dual ADMM is rather inefficient when higher accuracy is required.

For the second problem, CVX adopts second order method, SDPT3, to solve this problem. We find that first order method, is much more efficient. We adopt Graphic Lasso method to solve the penalized likelihood maximization. Besides, we find an ADMM method, originally designed for penalized D-trace problem [Zhang and Zou, 2014], is well suited to solve our penalized Frobenius norm minimization. In the end, we perform penalty parameter search by estimating the precison on a synthetic dataset, and discuss the relation between this two method.

CONTENTS

I. Algorithms for $\ell_1 - \ell_1$ minimization

Consider the problem

$$\min_x \quad \rho\|x\|_1 + \|Ax - b\|_1 \tag{I.1}$$

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;
rho = 1e-2
```

For the numerical result $\tilde{x}$, we define its error to be $\|A\tilde{x} - b\|_1$, its relative error with respect to cvx mosek x_{cvx} to be $\|\tilde{x} - x_{\text{cvx}}\|_1/(1 + \|x_{\text{cvx}}\|_1)$.

A. CVX/LP solver

We solve Equation I.1 using CVX by calling different solvers mosek or gurobi. Besides, we transform Equation (I.1) into standard form and call mosek and gurobi directly. Specifically, assuming $x = x^+ - x^-$, $y = Ax - b$ and $y = y^+ - y^-$, where $x^+ = \max(x, 0)$, $x^- = \max(-x, 0)$, we can formulate Equation (I.1) into the standard form:

$$\begin{aligned} \min_{x^+,x^-,y^+,y^-} \quad & \rho\mathbb{1}^T \begin{pmatrix} x^+ & x^- & y^+ & y^- \end{pmatrix}^\top \\ \text{s.t.} \quad & \begin{pmatrix} A & -A & -I & I \end{pmatrix} \begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} = b \\ & \begin{pmatrix} x^+ & x^- & y^+ & y^- \end{pmatrix}^\top \geq 0 \end{aligned} \tag{I.2}$$

To reproduce the results in Figure (I.2), please execute "test1.m", which calls "l1l1_cvx_mosek.m", "l1l1_cvx_gurobi.m", "l1l1_mosek.m" and "l1l1_gurobi.m".

B. Reformulation as Basis Pursuit

We give a reformulation of problem (I.1) in this part. Assume $u = b - Ax$, hence the problem (I.1) can be written as

$$\begin{aligned} \min \quad & \|u\|_1 + \rho\|x\|_1 \\ \text{s.t.} \quad & Ax + u = b \end{aligned} \tag{I.3}$$

Denote

$$\hat{A} = (A \ \rho I)/\sqrt{1+\rho^2}, \quad \hat{b} = \rho b/\sqrt{1+\rho^2}, \quad \hat{x} = \begin{pmatrix} \rho x \\ u \end{pmatrix} \tag{I.4}$$

then, (I.3) can be rewritten as

$$\begin{aligned} \min \quad & \|\hat{x}\|_1 \\ \text{s.t.} \quad & \hat{A}\hat{x} = \hat{b} \end{aligned} \tag{I.5}$$

For simplicity, we use A to substitute $\hat{A}$ and so on.

C. Bregman iteration

Bregman iterative regularization was introduced to solve

$$\min_u \quad J(u) + H(u) \tag{I.6}$$

The Bregman distance based on a convex functional $J(\cdot)$ between points u and v is defined as

$$D_J^p(u,v) = J(u) - J(v) - \langle p, u - v\rangle \tag{I.7}$$

Instead solving I.6 directly, we choose to solve

$$u^{k+1} \leftarrow \arg\min_u \quad D_J^{p^k}(u, u^k) + H(u) \tag{I.8a}$$

$$p^{k+1} \leftarrow p^k - \nabla H(u^{k+1}) \in \partial J(u^{k+1}) \tag{I.8b}$$

We can use Bregman iteration scheme to solve the problem

$$\min_u \mu\|u\|_1 + \|Au - b\|^2 \tag{I.9}$$

where we define $J(u) = \mu\|u\|_1, H(u) = \|Au - b\|^2$.

Thus, the iteration becomes

$$u^{k+1} \leftarrow \arg\min_u \quad D_J^{p^k}(u, u^k) + \frac{1}{2}\|Au - b\|^2 \tag{I.10a}$$

$$p^{k+1} \leftarrow p^k - A^\top(Au^{k+1} - b) \tag{I.10b}$$

which is equivalent to a more simplified form

$$b^{k+1} \leftarrow b + (b^k - Au^k) \tag{I.11a}$$

$$u^{k+1} \leftarrow \arg\min_u \quad J(u) + \frac{1}{2}\|Au - b^{k+1}\|^2 \tag{I.11b}$$

where $b^0 = 0$, $u^0 = 0$. The subproblem I.11b can be soolved by any Lasso solver. Here, we choose to use proximal gradient method with BB stepsizes and FISTA to cope with the problem

$$\min_u \quad \mu\|x\|_1 + \frac{1}{2}\|Ax - b\|^2 \tag{I.12}$$

In the following three subsubsections, we denote $f(x) = g(x) + \mu h(x) = \frac{1}{2}\|Ax - b\|^2 + \mu\|x\|_1$, where we replae u by x to avoid confusion.

This algorithm is implemented in "l1_breg.m". The input "opts.method" specify different method to solve the lasso problem, where "0" means proximal gradient method with BB stepsize, "1" means Nesterov acceleration. The default value is set to be zero. We set $\mu = 0.01$ by default.

1) Proximal gradient method

Since $h(x) = \|x\|_1$, the proximal operator can be explicitly obtained. We adopt the continuation stategy from [Wen et al., 2010] and Barzila-Borwein step size. We set $\mu_{\max} = \max(\mu, \gamma_1\|A^\top b\|_\infty)$, where γ_1 is diminishing ratio for μ's each update. Then, we take $\mu_0 = \mu_{\max}$, and update μ_i according to criterions that will be introduced later.

In the first iteration, step size is set to be α_0. otherwise, in the $(k+1)$-th iteration, BB step size is adopted

$$\alpha^{k+1} = \frac{(x^k - x^{k-1})^\top(x^k - x^{k-1})}{(x^k - x^{k-1})^\top(\nabla g(x^k) - \nabla g(x^{k-1}))} \tag{I.13}$$

Thus, x^{k+1} is updated by

$$x^{k+1} = \text{soft}(x^k - \alpha^k \nabla g(x^k), \alpha^k \mu_i) \tag{I.14}$$

Define $f_i(x) = g(x) + \mu_i h(x)$, if $|\frac{f_i(x^k) - f_i(x^{k-1})}{f_i(x^{k-1})}| < \varepsilon_1$ and $\mu_i > \mu$, we update μ_i by

$$\mu_{i+1} = \max(\mu, \gamma_1 \min(\|\nabla g(x^k)\|_\infty, \mu_i)) \tag{I.15}$$

Once we update μ_i, we set the index of x^k to be zero, and adopt the inital stepsize α^0.

When $\mu_i = \mu$, and $|\frac{f_i(x^k) - f_i(x^{k-1})}{f_i(x^{k-1})}| < \varepsilon_2$, we terminate the whole algorithm.

This algorithm is implemented in "l1_proxgrad.m".

2) Accelerated proximal gradient method-Nesterov

We apply the same continuation strategy. The update rule of Nesterov can be summerized as

$$y^k = x^k + \frac{k-1}{k+2}(x^k - x^{k-1}) \tag{I.16a}$$

$$x^{k+1} = \text{soft}(y^k - \alpha \nabla g, \alpha) \tag{I.16b}$$

This algorithm is implemented in "l1_fprox_nesterov.m".

D. Linearized Bregman iteration

The linearized Bregman algorithms [Yuan et al., 2013][1] return the solution to I.5 by solving

$$\min_x \|x\|_1 + \frac{1}{2\alpha}\|x\|_2^2, \quad \text{s.t.} Ax = b \tag{I.17}$$

It has been shown that there exist α_0, such that for any $\alpha > \alpha_0$, tbe solution to I.17 is also the solution to I.5. In the experiments, α could be $1 \sim 10 \times \|u\|_0$, where u is the original signal. In our implementation, we take $\alpha = 5\|u\|_0$.

The Lagrangian of problem I.17 can be written as

$$L(x, y) = \|x\|_1 + \frac{1}{2\alpha}\|x\|^2 + y^\top (b - Ax) \tag{I.18}$$

[1]Since the performance of Bregman iteration is not satisfactory, I found an more efficient algorithm online and test its performace.

Minimizing L w.r.t x, we have $x = \alpha\ \mathrm{soft}(A^\top y, 1)$, and y can updated by gradient descent. Hence, the algorithm can be written as

$$y^{k+1} = y^k - s^k(Ax^k - b) \tag{I.19a}$$

$$x^k = \alpha\ \mathrm{soft}(A^\top y^k, 1) \tag{I.19b}$$

where s^k is the BB stepsize.

This algorithm is implemented in "l1l1_lbreg_bb.m". We find this algorithm is both efficient and easy for coding.

E. ADMM for the dual

We can derive the dual of I.5 can be written as

$$\begin{aligned} \min \quad & -b^\top y \\ \text{s.t.} \quad & z = A^\top y, \|z\|_\infty \le 1 \end{aligned} \tag{I.20}$$

which has an augmented Lagrangian problem of the form

$$\min_{y, \|z\|_\infty \le 1} \quad -b^\top y - x^\top(z - A^\top y) + \frac{\beta}{2}\|z - A^\top y\|^2 \tag{I.21}$$

Fixing x^k and y^k, minimizing I.21 w.r.t. z can be given explicitly by

$$z^{k+1} = \mathcal{P}_{\mathbf{B}_1^\infty}\left(A^\top y + x/\beta\right) \tag{I.22}$$

where $\mathbf{B}_1^\infty = \{x \in \mathbb{R}^n : \|x\|_\infty \le 1\}$, and $\mathcal{P}$ is the orthogonal projection. Specifically,

$$\mathcal{P}_{\mathbf{B}_1^\infty}(z)_i = \frac{z_i}{\max(1, |z_i|)} \tag{I.23}$$

Second, fixing x^k and z^{k+1}, minimizing w.r.t. y is a least sqquares problem, and we can get

$$\beta A A^\top y = \beta A z^{k+1} - (Ax^k - b) \tag{I.24}$$

The reference paper [Yang and Zhang, 2011] differentiates two cases $AA^\top = I$ and $AA^\top \ne I$. In the former case, I.24 can be readily solved. However, in the second case,

solving I.24 explicitly is computationally hard. Hence, we take a steepest descent in the y direction $y^{k+1} = y^k - \alpha^k g^k$, where α^k and g^k is given by

$$g^k = Ax^k - b + A(A^\top y^k - z^{k+1}), \quad \alpha^k = \frac{g^{k\top} g^k}{g^{k\top} \beta A A^\top g^k} \tag{I.25}$$

Third, we update x by

$$x^{k+1} = x^k - \gamma\beta(z^{k+1} - A^\top y^{k+1}) \tag{I.26}$$

where hyperparameter γ is adopted in [Yang and Zhang, 2011] and is suggested to be set within $(0, \frac{\sqrt{5}+1}{2})$. In our implementation, we simply set it to be $\frac{\sqrt{5}+1}{2}$.

Stopping critertion

We define the primal residue $r_p^k = Ax^k - b$ and the dual residue $r_d^k = A^\top y^k - z^k$. Besides, the duality gap is $\delta^k = b^\top y^k - \|x^k\|_1$.

If either $\|x^{k+1} - x^k\|_2 / \|x^k\| < tol/2$ or

$$\max\left(\frac{\|r_p^k\|_2}{\|b\|_2}, \frac{\|r_p^k\|_2}{\|z^k\|_2}, \frac{|\delta^k|}{\|x^k\|_1}\right) < tol \tag{I.27}$$

we stop the iteration. To reduce the computational effort, we do not check the criterion I.27 if $\|x^{k+1} - x^k\|_2 / \|x^k\| > tol$. Besides, in order to avoid division by zero error, we replace each denominator in the algorithm with the maximum of itself and the machine precision "eps=2^{-52}".

This algorithm is implemented in "l1l1_dual_ADMM.m".

F. Numerical results

The the following, we define the relative error to cvx mosek solver is by defining $\frac{\|x_k - x_{\text{mosek}}\|_1}{1+\|x_{\text{mosek}}\|+1}$. We find that Bregman iteration with Nesterov, with fixed stepsize, is competitive with BB step size. The Linearized Bregman method with BB stepsize is, on the contrary, significantly outperforms other methods. The Bregman method with BB stpesize is slightly inferior to cvx-mosek. As for the ADMM for dual problem, it converges relative fast in the first iterations, but slowes down when the residual approches 10^{-6}. Corresponding codes are "test1.m" and "test2.m". We set "rng(1234)" to ensure reproductivity.

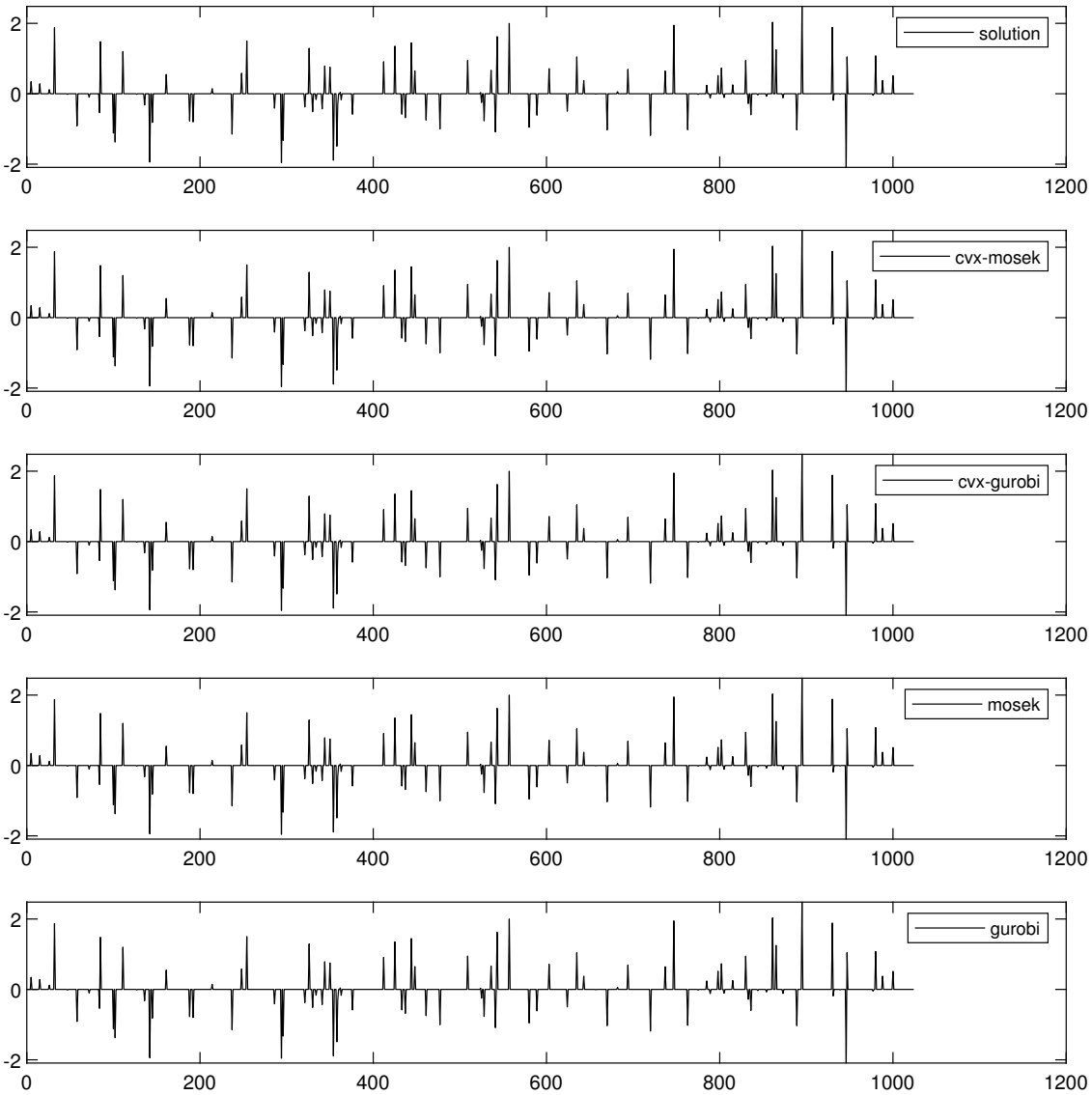


Fig. I.1: Numerical results by calling different solvers.

	$\|Ax_k - b\|_1$	$\|x_k\|_1$	CPU time	err-to-cvx-mosek
cvx-call-mosek	1.63e-09	7.40e+01	0.86	0
call-mosek	3.55e-09	7.40e+01	3.46	1.35e-10
cvx-call-gurobi	6.28e-05	7.40e+01	1.06	5.49e-09
call-gurobi	9.30e-10	7.40e+01	2.10	1.35e-10

TABLE I.1: Numerical results by different solvers.

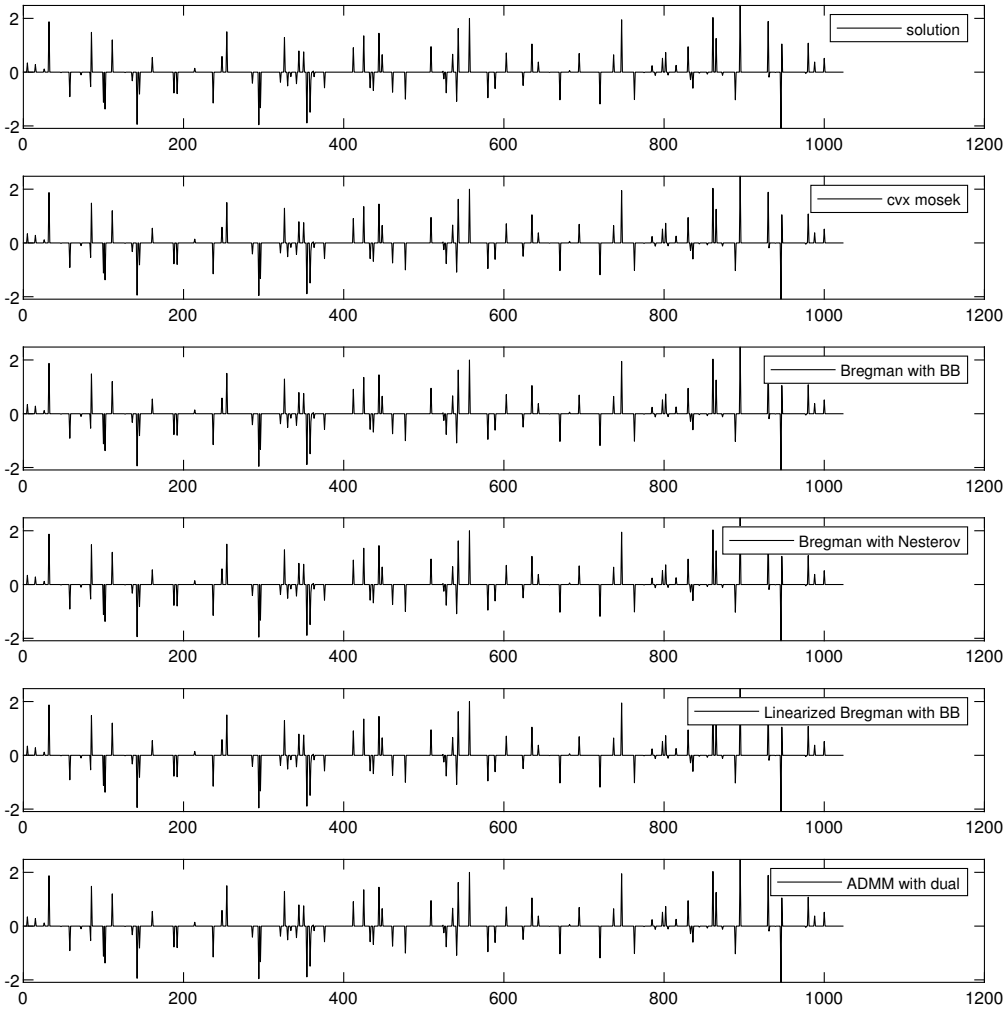


Fig. I.2: Numerical results by implementing different methods

	$\|Ax_k - b\|_1$	$\|x_k\|_1$	CPU time	err-to-cvx-mosek
cvx-call-mosek	1.63e-09	7.40e+01	0.86	0.00e+00
Bregman-BB	1.10e-10	7.40e+01	4.41	1.35e-10
Bregman-Nesterov	1.39e-12	7.40e+01	4.13	1.35e-10
LBregman-BB	1.92e-09	7.40e+01	0.642	5.47e-11
ADMM-dual	1.61e-06	7.40e+01	0.66	7.65e-10

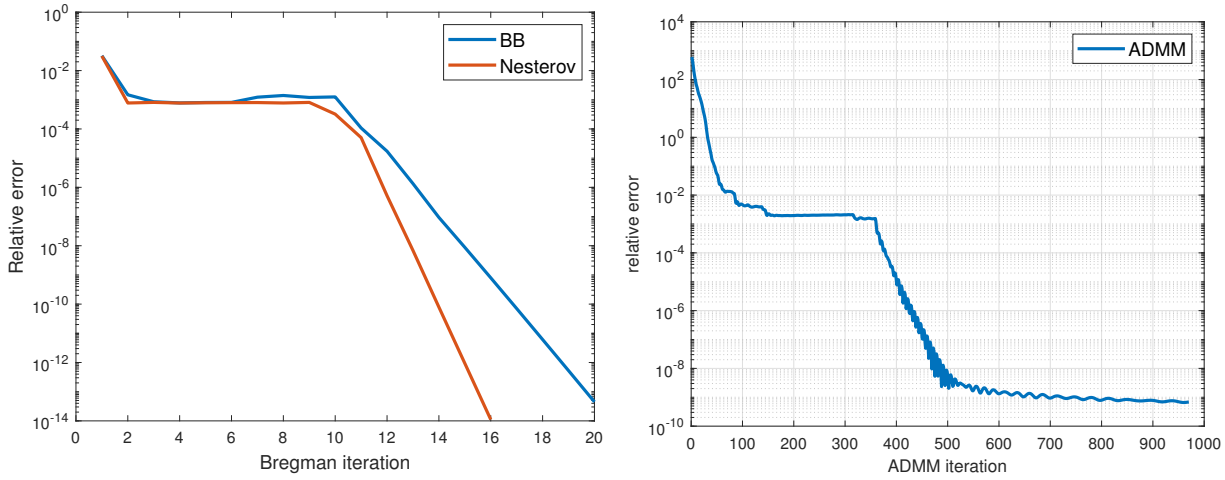
TABLE I.2: Numerical results by different methods.

G. Comparision of Bregman iteration and ADMM

An intriguing observation would be that we can set the convergence criterion of the inner cycle to be low, and the iteration will still converge. This phenomenon is analyzed in [Yin and Osher, 2013] which explains why solving Bregman subproblems at low accuracies (10^{-6}) gives a Bregman solution at near the machine precision (10^{-15}). In our implementation, we find that modifying tolerance in "l1l1_breg.m" from 10^{-8}, to 10^{-10}, and to 10^{-12} does not affect the number of iterations a lot, even we set the tolerance in solving the Lasso subproblem to be 10^{-5} or 10^{-6}.

However, we find that ADMM is stagnated in the region around 10^{-8} and is hard to achieve higher accuracy. Hence, semismooth Newton method was introduced to cope with this situation.

Setting "opts.display_err" in "l1l1_breg.m" and "l1l1_dual_ADMM.m" to be 1, we can verify this phenomenon of BB/Nesterov method. We plot numerical results in I.3. The relative error is defined by $\frac{\|x_k - x_{\text{real}}\|}{\|x_\|}$, where x is the variable in Basis Pursuit problem.



(a) Bregman iteration by proximal gradient with BB stepsize/ Nesterov acceleration. Inner cycle: tol 10^{-5}, outer cycle: tol 10^{-12}.

(b) Dual ADMM, tol 10^{-10}

Fig. I.3: Error forgetting and Stagnation

II. Sparse Inverse Covariance Estimation, Part I

Let $\mathcal{S}^n = \{X \in \mathbb{R}^{n\times n} | X^\top = X\}$. Let $S \in \mathcal{S}^n$ be a a given observation of covariance matrix. Consider the model

$$\max_{X\succeq 0} \log\det X - \mathrm{trace} SX - \rho\|X\|_1 \tag{II.1}$$

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

A. Dual problem

We introduce variable U, such that

$$\|X\|_1 = \max_{\|U\|_\infty \le 1} \langle U, X\rangle. \tag{II.2}$$

Hence, problem (II.1) can be transformed into

$$\max_{X\succeq 0} \min_{\|U\|_\infty \le \rho} \log\det X - \langle S, X\rangle - \langle U, X\rangle. \tag{II.3}$$

Hence, the Lagrange of this problem is

$$L(X, U, P) = \log\det X - \langle S + U - P, X\rangle \tag{II.4}$$

Since, the derivative of the function is $\nabla_X L = X^{-1} - (S + U - P)$, and we get the following dual to (II.3)

$$\min_{P\succeq 0, \|U\|_\infty \le \rho} -\log\det(S + U - P) - n \tag{II.5}$$

The duality gap is

$$\mathrm{dualgap} = n - \langle S, X\rangle - \rho\|X\|_1 \tag{II.6}$$

B. CVX solution

We use CVX to solve the primal problem (II.3), and use duality gap (II.6) to check the optimality condition.

C. First-order algorithm: GLasso

The problem (II.3) is a convex optimization problem: the ℓ_1 penalty is convex, the linear term is convex, the negative log determinant function is convex, and the set of positive semidefinite matrices is also convex. In particular, (II.3) is in the form of Semidefinite Programming (SDP) because the variable is constrained to be a PSD matrix. SDPs are known to be hard convex optimization problems. Although there exist algorithms for solving general SDPs, such as the interior point method, most of these algorithms/solvers usually become quite inefficient when the dimension of the variable matrix exceeds hundreds. Therefore, machine learning researchers have been focusing specifically on solving (II.3), and as a result developed the Glasso algorithm. [Friedman et al., 2008]

The main idea of the Glasso algorithm consists of the following three ingredients:

- Instead of the regularized maximum likelihood problem (II.3), solve its dual problem.
- Decompose the dual problem of (II.3) into a series of sub-problems, and iteratively solve the sub-problems until convergence.
- For each sub-problem, solve its dual via the Lasso algorithm.

Let W be the estimate of Σ. We show that one can solve the problem by optimizing over each row and corresponding column of W in a block coordinate descent fashion. Partitioning W and S

$$W = \begin{pmatrix} W_{11} & w_{12} \\ w_{12}^\top & w_{22} \end{pmatrix}, \quad S = \begin{pmatrix} S_{11} & s_{12} \\ s_{12}^\top & s_{22} \end{pmatrix}, \quad X = \begin{pmatrix} X_{11} & x_{12} \\ x_{12}^\top & x_{22} \end{pmatrix} \tag{II.7}$$

where $W_{11}, S_{11}, X_{11} \in \mathbb{R}^{(n-1)\times(n-1)}$, $w_{12}, s_{12}, x_{12} \in \mathbb{R}^n$, $w_{22}, s_{22}, x_{22} \in \mathbb{R}^n$.

If we only update the row and column j, according to the Schur complements, the solution w_{12} satisfies

$$w_{12} = \arg\min_y \left\{ y^\top W_{11}^{-1} y : \|y - s_{12}\|_\infty \le \rho \right\} \tag{II.8}$$

Using convex duality, we can show that solving (II.8) is equivalent to solving the dual

problem

$$\min_{\beta}\left\{\frac{1}{2}\|W_{11}^{1/2}\beta - b\|_2^2 + \rho\|\beta\|_1\right\} \tag{II.9}$$

where $b = W_{11}^{-1/2}s_{12}$. If β solves (II-C), then $w_{12} = W_{11}\beta$ solves (II.8). In addition, we can express (II-C) resembles a lasso regression.

Expanding relation $WX = I$ gives an expression that will be useful below

$$\begin{pmatrix} S_{11} & s_{12} \\ s_{12}^\top & s_{22} \end{pmatrix}\begin{pmatrix} X_{11} & x_{12} \\ x_{12}^\top & x_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0^\top & 1 \end{pmatrix} \tag{II.10}$$

Due to the sub-gradient equation for (II.3)

$$W - S - \rho\Gamma = 0 \tag{II.11}$$

where $\Gamma_{ij} = \text{sgn}(X_{ij})$ if $X_{ij} \neq 0$, else $\Gamma_{ij} \in [-1, 1]$ if $X_{ij} = 0$.

Since $X \succeq 0$, $x_{ii} > 0$, according the (II.11), the solution of W on its diagonal can be explicit written $w_{ii} = s_{ii} + \rho$. Hence, in our update, we keep the diagonal unchanged and update w_{12}.

We can derive that

$$\begin{aligned} W_{11}x_{12} + w_{12}x_{22} &= 0 \\ w_{12}^\top x_{12} + w_{22}x_{22} &= 1 \end{aligned} \tag{II.12}$$

Hence, $x_{12} = -x_{22}W_{11}^{-1}w_{12} = -x_{22}\beta$ and $1 = w_{12}^\top x_{12} + w_{22}x_{22} = (-w_{12}^\top\beta + w_{22})x_{22}$. This formula can used to update X according to W by each column.

Here is Glasso algorithm in detail

1) Start with $W = S + \rho I$. The diagonal of W remains unchanged in what follows.
2) For each $i, j = 1, 2, \cdots, n$, solve the lasso problem (II-C), which takes as input the inner products W_{11} and s_{12}. This gives a $p-1$ vector solution β. Fill in the corresponding row and column of W using $w_{12} = W_{11}\beta$
3) Continue until convergence, the criterion will be detailed later.
4) Construct X column by column according to $x_{22} = 1/(-w_{12}^\top\beta + w_{22})$ and $x_{12} = -x_{22}\beta$.

Stopping criterion

Take the subgradient of (II-C), we have

$$W_{11}\beta - s_{12} + \rho v = 0 \tag{II.13}$$

where $v \in \text{sgn}(\beta)$ element-wise. Hence, the convergence of lasso is equivalent to the fact that

$$\begin{aligned} |(W_{11}\beta)_i - (s_{12})_i| &\le \rho, \quad \beta_i = 0 \\ |(W_{11}\beta)_i - (s_{12})_i + \rho \text{sgn}(\beta_i)| &\le \epsilon_{\text{lasso}}, \quad \beta_i \neq 0 \end{aligned} \tag{II.14}$$

For the outer loop, we iterate the column from the leftest to the rightest, i.e. from column 1 to column n. After a full batch of iterations, when the absolute change in W is less than $\epsilon_{\text{out}}\|S_{-\text{diag}}\|$ where $S_{-\text{diag}}$ are the off-diagonal elements of the empirical covariance matrix S.

FISTA for solving Lasso problem

We are aimed to solve the Lasso subproblem

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \tag{II.15}$$

However, we do not have direct access to X, y, instead, we have $X^\top X, X^\top y$ from the definition of equation .

Still, one may wonder what whether W_{11} is positive semidefinite. Since W_{11} is the order principal minor determinant of a positive definite matrix, it should be positive definite. However, due to numerical instability, it might be indefinite during the process. Hence, we check if W_{11} is positive definite. Otherwise, we decompose $X^\top X = W_{11} = VDV^\top$, and makes the following replacement

$$\widehat{X^\top X}_{11} = V \max(D, 0)V^\top, \quad \widehat{X^\top y} = V\mathbb{1}(D > 0)V^\top(X^\top y) \tag{II.16}$$

We use the fast proximal gradient with Nesterov in Section I-C2 to solve the Lasso problem. Since we do not explicitly have X, y now. The stopping criterion can be simply set to be $\|x^k - x^{k-1}\|_1 < \varepsilon$. Still, we set the inital step size to be $\frac{1}{\|X^\top X\|_2}$. We find the Glasso outperforms cvx significantly. Detailed default parameters can be found in

“l1_fprox_nesterov.m”

III. Sparse Inverse Covariance Estimation, Part II

Let $\mathcal{S}^n = \{X \in \mathbb{R}^{n\times n} | X^\top = X\}$. Let $S \in \mathcal{S}^n$ be a a given observation of covariance matrix. Consider the model

$$\min_{X\succeq 0} \|X\|_1 + \frac{\sigma}{2}\|SX - I\|_F^2 \tag{III.1}$$

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

A. Duality Gap

We introduce variable U, such that

$$\|X\|_1 = \max_{\|U\|_\infty \le 1} \langle U, X\rangle. \tag{III.2}$$

Hence, problem (III.1) can be transformed into

$$\min_{X\succeq 0} \max_{\|U\|_\infty \le 1} \langle U, X\rangle + \frac{\sigma}{2}\|SX - I\|_F^2 \tag{III.3}$$

Introducing $Y \succeq 0$, the Lagrange of this problem is

$$L(X, U, Y) = \langle U, X\rangle + \frac{\sigma}{2}\|SX - I\|_F^2 - \langle Y, X\rangle \tag{III.4}$$

Since, the derivative of the function is $\nabla_X L = U - Y + \sigma\frac{S^2X + XS^2}{2} - S$, and we get the following dual to (III.1)

$$\begin{aligned} \max \quad & \langle U, X\rangle + \frac{\sigma}{2}\|SX - I\|_F^2 - \langle Y, X\rangle \\ \text{s.t.} \quad & \sigma\frac{S^2X + XS^2 - 2S}{2} = Y - U, \quad Y \succeq 0, \quad \|U\|_\infty <= 1 \end{aligned} \tag{III.5}$$

The duality gap is

$$\text{dualgap} = \|X\|_1 - \langle U - Y, X\rangle = \|X\|_1 + \sigma(\|SX\|_F^2 - \text{trace}SX). \tag{III.6}$$

B. CVX solution

We use CVX to solve the primal problem (III.1), and use duality gap (III.6) to check the optimality condition.

C. Alternating linearization method

Following the methods in [Yang and Zhang, 2011] to solve II.1, we can reformulate the problem III.1 into

$$\begin{aligned} \min_{X,Y\geq 0} \quad & f(X)+g(Y) \\ \text{s.t.} \quad & X=Y \end{aligned} \tag{III.7}$$

where $f(X)=\frac{\sigma}{2}\|SX-I\|_F^2, g(Y)=\|Y\|_1$.

The augmented Lagrange can be written as

$$L(x,y,\lambda)=f(X)+g(Y)-\langle\Lambda,X-Y\rangle+\frac{1}{2\mu}\|X-Y\|_F^2 \tag{III.8}$$

An anlternating direction versison of the augmented Lagranagian can be written as

$$\begin{cases} X^{k+1} & \leftarrow \arg\min_X L(X,Y^k,\Lambda_Y^k) \\ \Lambda_X^{k+1} & \leftarrow \Lambda_Y^k-(X^{k+1}-Y^k)/\mu \\ Y^{k+1} & \leftarrow \arg\min_Y L(X^{k+1},y,\Lambda_X^{k+1}) \\ \Lambda_Y^{k+1} & \leftarrow \Lambda_X^{k+1}-(X^{k+1}-Y^{k+1})/\mu \end{cases} \tag{III.9}$$

We have $-\Lambda_Y^{k+1}\in\partial g(Y^{k+1}), \Lambda_X^{k+1}=\nabla f(X^{k+1})$ due to first-order optimality condition. Note that g is not differentiable everywhere. Hence, we keep Λ_Y^k in updating X^k and Y^k. In proving the convergence [Yang and Zhang, 2011], we need to keep $F(X^{k+1})\leq Q(X^{k+1},Y)$, where $F(X)=f(X)+g(X)$ and $Q(X,Y)=f(X)+g(Y)-\langle\Lambda^k,X-Y^k\rangle+\frac{1}{2\mu}\|X-Y\|_F^2$. Otherwise, we set $X^{k+1}=Y^k$.

The algorithm can be summerized in 1. We adopt a decreasing μ_k. Specifically, $\mu_k=\mu_0$ initially, and for every μ_N iteration, we decrease μ_k to $\max(\mu_r\mu_k,\mu_{\min})$. In our implementation, we tune the hyperparameters to be $\mu_0=0.1,\mu_N=40,\mu_r=3/4,\mu_{\min}=10^{-6}$.

Note that in solving X^{k+1}, we are required to solve the linear system with the form $AX+XB=C$, where A,B are positive definite. According to the property of such Sylvester equation, the solution uniquely exists. For simplicity, we use MATLAB built-

Algorithm 1 Alternating linearization method (ALM)

Input: $X^0 = Y^0$, μ_0.
for $k = 0, 1, \cdots$ **do**
0. Pick $\mu_{k+1} \le \mu_k$.
1. $X^{k+1} \le \arg\min_{X \succeq 0} f(X) + g(Y^k) - \langle \Lambda^k, X - Y^k \rangle + \frac{1}{2\mu_{k+1}} \|X - Y^k\|_F^2$;
2. If $g(X^{k+1}) > g(Y^k) - \langle \Lambda^k, X^{k+1} - Y^k \rangle + \frac{1}{2\mu_{k+1}} \|X^{k+1} - Y^k\|_F^2$, then $X^{k+1} := Y^k$.
3. $Y^{k+1} \leftarrow \arg\min_Y f(X^{k+1}) + \langle \nabla f(X^{k+1}), Y - X^{k+1} \rangle + \frac{1}{2\mu_{k+1}} \|Y - X^{k+1}\|_F^2 + g(Y)$;
4. $\Lambda^{k+1} \leftarrow \nabla f(X^{k+1}) - (X^{k+1} - Y^{k+1})/\mu_{k+1}$.
end for

in function "sylvester" to deal with the subproblem.

The alternating linearization method is able to solve the problem III.1, yet requires finetuning hyperparameters and is hard to reduce gap below 10^{-6}. However, we still implement such algorithm in the file "spinv_alm.m". The stopping criterion will be detailed in the next subsection. Since the gap can only be reduced to around $\sim 10^{-2}$ by this algorithm, which inspires us to find a better way.

D. ADMM method

Since Algorithm 1 is not satisfactory, we find another variant of ADMM method to deal with the problem. The method is modified from [Wang and Jiang, 2020], which solves the following problem:

$$\arg\min_{\Omega} \frac{1}{4}(\mathrm{trace}(\Omega^\top S \Omega + \Omega S \Omega^\top)) - \mathrm{trace}(\Omega) + \lambda \|\Omega\|_1$$

which is closely related to our problem III.1. Hence, we anticipate adopting their method should have better performace.

We consider the augmented Lagrangian in the following form

$$L(X, Y, \Lambda) = \frac{\sigma}{2} \|SX - I\|_F^2 + \|Y\|_1 + \frac{1}{2\mu} \|X - Y + \Lambda\|_F^2 \tag{III.10}$$

Hence, the ADMM algorithm should be

$$\begin{cases} X^{k+1} & \leftarrow \arg\min_X L(X, Y^k, \Lambda^k) \\ Y^{k+1} & \leftarrow \arg\min_Y L(X^{k+1}, Y, \Lambda^k) \\ \Lambda^{k+1} & \leftarrow \Lambda^k + X^{k+1} - Y^{k+1} \end{cases} \tag{III.11}$$

The second minimization can be obtained explicitly by

$$Y^{k+1} = \text{soft}(X^{k+1} + \Lambda^k, \mu)$$

whereas obtaining X^{k+1} is slightly more complicated. By differentiate L w.r.t. X, we have

$$X^{k+1}\frac{\mu\sigma S^2}{2} + (\frac{\mu\sigma S^2}{2} + I)X^{k+1} = \mu\sigma S + Y^k - \Lambda^k \tag{III.12}$$

which is also a Sylvester equation, and the solution uniquely exists, since $S^2 = S^\top S$ is positive definite. Since if X is a solution to III.12, $X^\top$ is also a solution to III.12, we have X is symmetric. Initially, we set $X^0 = 0, Y^0 = \mathbb{1}_{n\times n}$, which is respectively the sparsest and densest matrix, and then converge to the same matrix in the end.

Here, we do not need to tune μ to get better performace. We set $\mu = \mu_0 = 0.1$. We find that such algorithm is significantly more efficient. Hence, in presenting the results, we use this algorithm instead of Algorithm 1. The MATLAB file "spinv_admm.m" implements the algorithm.

The major disadvantage of our algorithm is the loss of positive definiteness. From the perspective of optimization, it is ideal to find the solution over the convex cone of positive definite matrices. However, this could be costly, as we would need to guarantee the solution in each iteration to be positive definite. As suggested by [Cai et al., 2010], a practical proposal to avoid this would be to project the estimator to the space of positive-semidefinite matrices under the operator norm. More specifically, one may first diagonalize X and then replace negative eigenvalues by 0. The resulting estimator is then positive-semidefinite. To reduce computational complexity, we use the following codes to implement this idea, since function "chol" is the most efficient way to check

if positive definite, and "eps" is equal to 2^{-52}.

```
try
    chol(X+eps*eye(size(X)));
catch
    [V,D] = eig(X);
    d = diag(D); d = max(d,0);
    X = (V*diag(d))*V';
end
```

Since the projection is quite computationally expensive, we will first updating X^k, Y^k until convergence, and then check if X^k is positive semidefinite. Otherwise, we resume the iteration with projection. Numerically, we find that strategy is quite efficient.

Stopping criterion

Define $F(X) = \frac{\sigma}{2}\|SX - I\|_F^2 + \|X\|_1$, we have that if

$$\max\left\{\frac{|F(X^k) - F(X^{k-1})|}{\max\{F(X^k), F(X^{k-1}), 1\}}, \frac{\|X^k - X^{k-1}\|_F}{\max\{\|X^k\|_F, \|X^{k-1}\|_F, 1\}}, \frac{\|Y^k - Y^{k-1}\|_F}{\max\{\|Y^k\|_F, \|Y^{k-1}\|_F, 1\}}\right\} < tol_{\text{rel}} \tag{III.13}$$

where $tol_{\text{rel}} = 10^{-12}$, or the duality gap w.r.t. X^k and Y^k is smaller than $tol_{\text{gap}} = 10^{-6}$.

Finally, we calculate

$$\frac{\|X^k - Y^k\|_F}{\max\{\|X^k\|_F, \|Y^k\|_F, 1\}}$$

in order to check convergence. Numerically, we find out that in our algorithm, this value is about $\sim 10^{-8}$. Hence, the convergence of the algorithm is guaranteed.

IV. Summary of Sparse Inverse Covariance Estimation

A. Simulation Models on Sparse Inverse Covariance Estimation

Suppose $S \in \mathbb{R}^{n\times n}$, we set $n = 30$. The two models are excerpted from [Cai et al., 2011].

- **Model 1**: $S_{ij} = 0.6^{|i-j|}$.
- **Model 2**: We let $S = B + \delta I$, 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 n. Finally, the matrix is standardized to have unit diagonals.

Model 1 has a banded structure, and the values of the entries decay as they move away from the diagonal. Model 2 is an example of a sparse matrix without any special sparsity patterns.

In implementing model 2, there are one pitfall. We cannot directly use the code

```
f = @(x)cond(B+x*eye(n)) - n;
[delta,~,flag] = fsolve(f, 1);
```

to find desired δ. However, since B is not positive definite. Starting search x from 0 will leads to a solution that is positive indefinite. Since "cond" involves singular values instead of eigenvalues. Hence, we adopt the following strategy.

```
f = @(x)cond(B+(x+5)*eye(n)) - n;
[delta,~,flag] = fsolve(f, 1);
```

and uses

```
try
    chol(W+eps*eye(n)); % check if positive definite
catch MSE
    flag = 0;
end
```

wher "flag = 0" means an invalid solution. Since in most cases, "$B + 5I$" is positive definite, such stategy works well empirically.

Besides, it is still possible that "fsolve" cannot find a solution. According to the documentation of "fsolve", we use "flag" to encode whether a solution δ is found. In all, the output is valid if and only if "flag=1". Otherwise, we raise an error and suggest the user to try again. In our implementation, that has never happened.

B. Experiments Settings

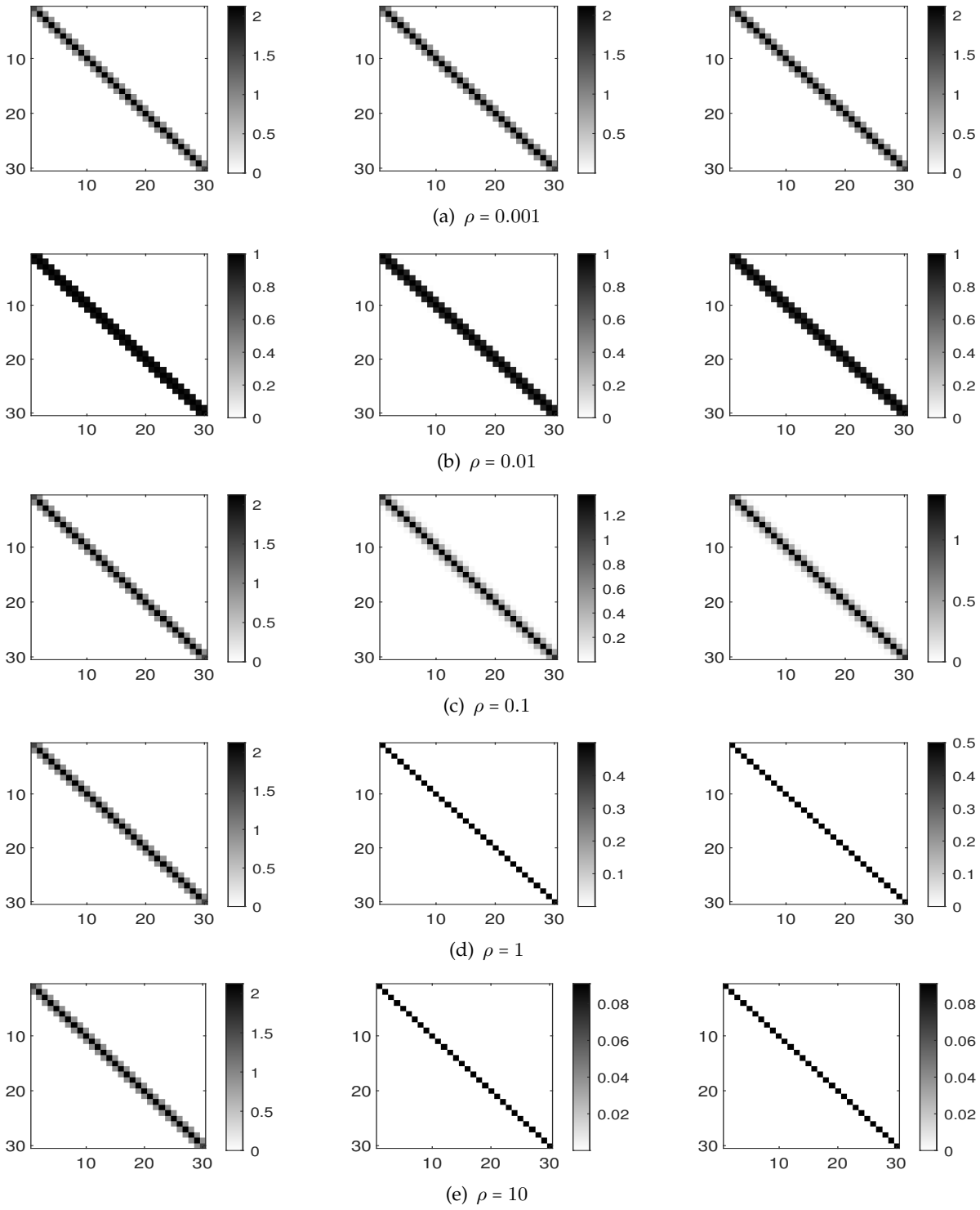
In Part I of the covariance inverse estimation, we set $\rho = 0.001, 0.1, 10$ for Model 1 and 2. To reproduce the results, first specify the model in "test1.m", then execute the scripts.

Algorithm	ρ	Objective Value	Duality Gap	Time	Iteration	Error to CVX SDPT3
CVX SDPT3	0.001	-17.17430565	9.23e-05	6.9355317	NaN	0
	0.01	-18.19217144	0.000102716	6.631535	NaN	0
	0.1	-26.10807442	0.000208115	6.4952199	NaN	0
	1	-50.79441552	0.00138182	5.4659974	NaN	0
	10	-101.9368582	1.07e-05	8.2433067	NaN	0
Glasso	0.001	-17.17430564	1.33e-06	0.7930596	36	3.29e-06
	0.01	-18.19217143	-2.54e-06	0.7233722	38	3.17e-06
	0.1	-26.10807441	-1.33e-06	0.2454903	25	6.91e-06
	1	-50.79441542	0	0.0109431	2	4.61e-05
	10	-101.9368582	0	0.012671	2	1.78e-07

TABLE IV.1: Part 1, Model 1

Algorithm	ρ	Objective Value	Duality Gap	Time	Iteration	Error to CVX SDPT3
CVX SDPT3	0.001	-28.82568208	2.60e-04	8.3034009	NaN	0
	0.01	-29.31224024	0.000269697	7.1003089	NaN	0
	0.1	-32.74842041	0.000355329	6.0068386	NaN	0
	1	-50.79441546	0.001381937	5.3271661	NaN	0
	10	-101.9368582	1.07e-05	7.8253934	NaN	0
Glasso	0.001	-28.82568207	1.18e-07	0.1719492	6	9.69e-06
	0.01	-29.31224022	-2.85e-08	0.0957352	7	8.98e-06
	0.1	-32.74842041	4.40e-09	0.0430902	5	1.18e-05
	1	-50.79441542	0	0.0142118	2	4.61e-05
	10	-101.9368582	0	0.0114573	2	1.78e-07

TABLE IV.2: Part 1, Model 2



(a) $\rho = 0.001$

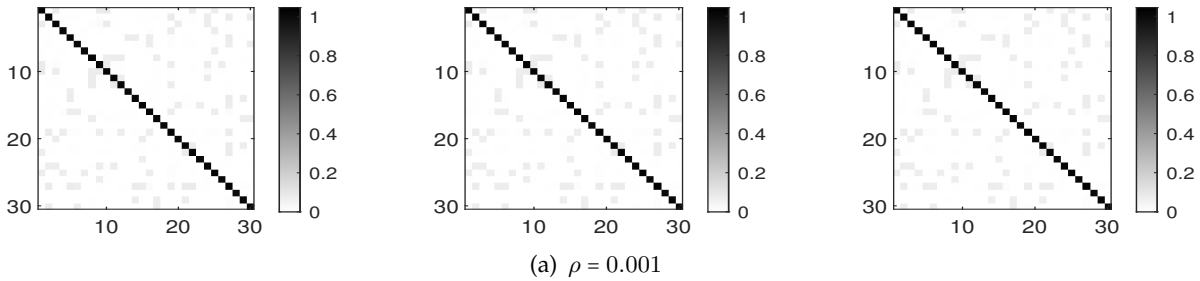
(b) $\rho = 0.01$

(c) $\rho = 0.1$

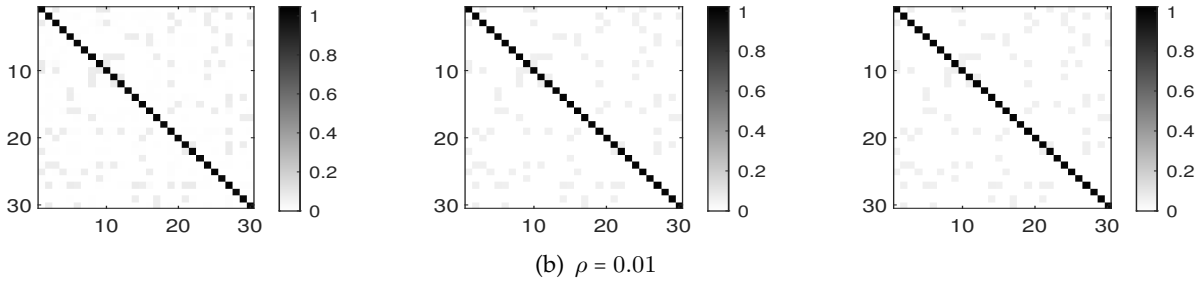
(d) $\rho = 1$

(e) $\rho = 10$

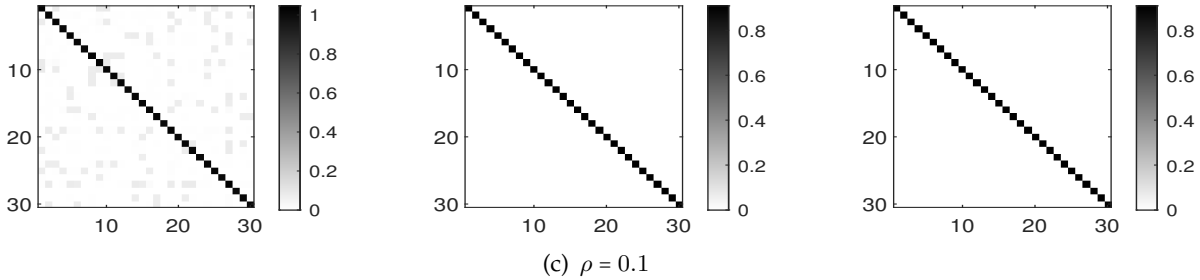
Fig. IV.1: Part 1, Model 1. **Left**: accurate inverse. **Middle:** solution by cvx. **Right**: solution by the first order algorithm.



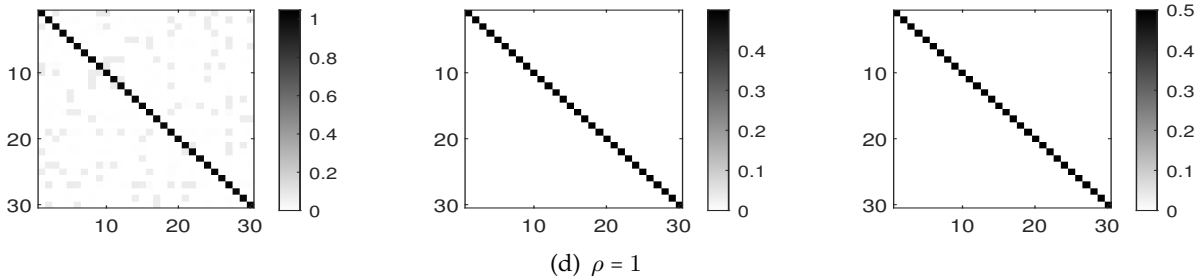
(a) $\rho = 0.001$



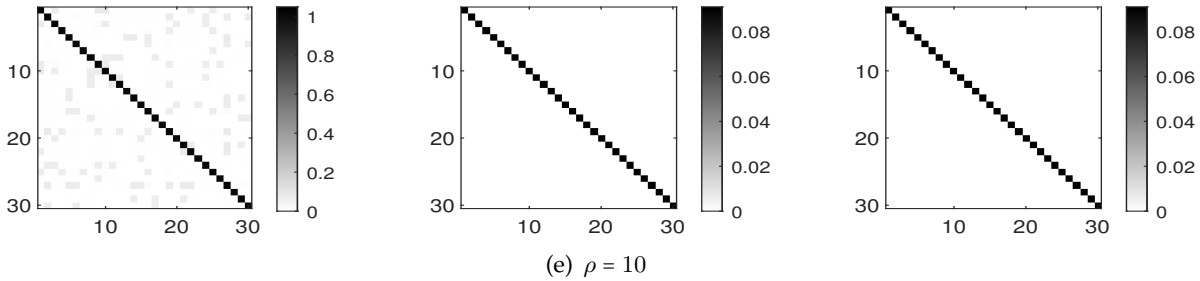
(b) $\rho = 0.01$



(c) $\rho = 0.1$



(d) $\rho = 1$



(e) $\rho = 10$

Fig. IV.2: Part 1, Model 2. **Left**: accurate inverse. **Middle:** solution by cvx. **Right**: solution by the first order algorithm.

In Part II, the minima is zero matrix when σ is small. Thus, we take $\sigma = 0.1, 1, 10, 100, 1000$, and find out that when setting $\sigma = 1$, the minimum attains around $X = 0$. Hence, lower σ is almost meaningless. To reproduce the results, first specify the model in "test2.m", then execute the scripts.

Algorithm	σ	Objective Value	Duality Gap	Time	Iteration	Error to CVX SDPT3
CVX SDPT3	1000	116.6301638	-7.38e-06	2.2758444	20	0
	100	113.3016374	0.00010081	1.6106223	16	0
	10	80.01637194	-0.000127646	1.245693	15	0
	1	15.0	3.69e-08	1.5746467	25	0
	0.1	1.5	3.93e-09	1.3832566	17	0
ADMM	1000	116.6302438	1.77e-03	1.3586509	max	4.02e-07
	100	113.301638	9.95e-07	0.4821643	1489	3.76e-07
	10	80.01637378	9.90e-07	0.0400372	198	6.25e-06
	1	15.0	7.03e-07	0.0271157	79	1.02e-04
	0.1	1.5	1.08e-11	0.0125717	18	7.81e-10

TABLE IV.3: Part 2, Model 1

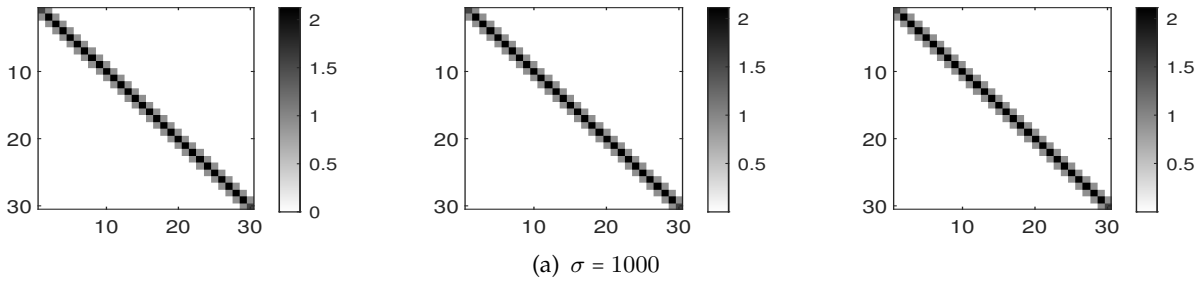
Algorithm	σ	Objective Value	Duality Gap	Time	Iteration	Error to CVX SDPT3
CVX SDPT3	1000	48.75919911	-3.72e-04	2.1926627	18	0
	100	44.87945079	2.16e-05	1.4598811	18	0
	10	33.70038151	-4.41e-06	1.2664943	18	0
	1	15	2.20e-08	1.6382531	25	0
	0.1	1.5	4.61e-09	1.3882479	17	0
ADMM	1000	48.75920008	9.99e-07	0.3943388	1156	1.42e-07
	100	44.87945173	9.54e-07	0.0839815	187	2.04e-08
	10	33.70038211	7.67e-07	0.0135485	37	3.26e-08
	1	15	9.73e-07	0.0468608	125	1.07e-04
	0.1	1.5	1.31e-10	0.0152287	17	9.17e-10

TABLE IV.4: Part 2, Model 2.

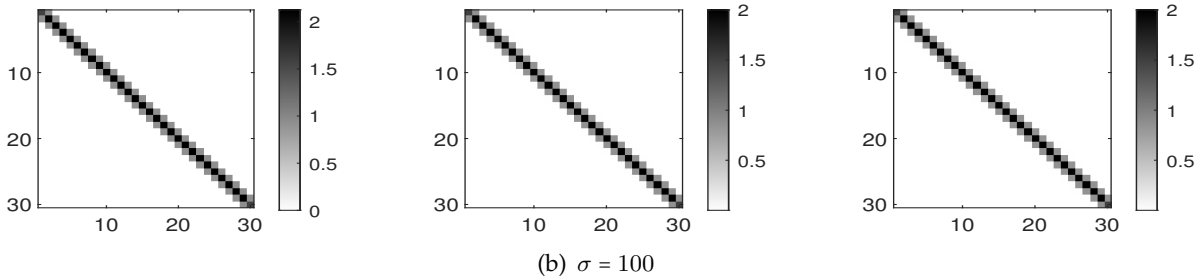
We find that, generally, first order algorithm performs significantly better than CVX solver, which utilizes SDPT3 [Toh et al., 1999]. Glasso achieves 10x improvement on performace, and ADMM based method achieves 50x improvement in reasonable size σ.

C. Penalty Parameter Estimation

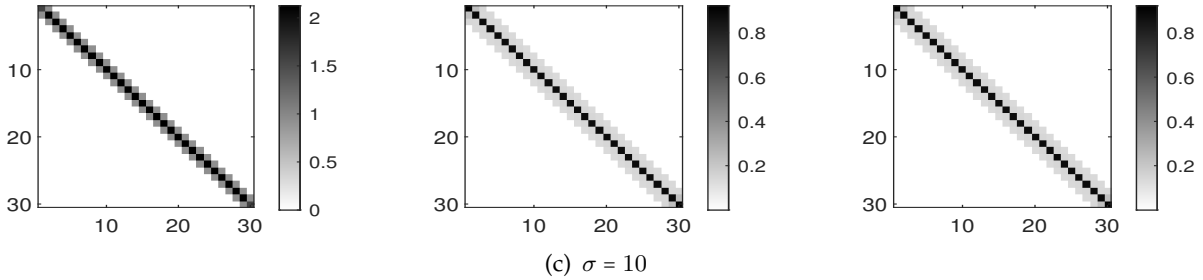
Since first order mehod is significantly better, we validate two approches II.1 and III.1 to deal with the sparse inverse problem. Given a covariance matrix S, we generate m



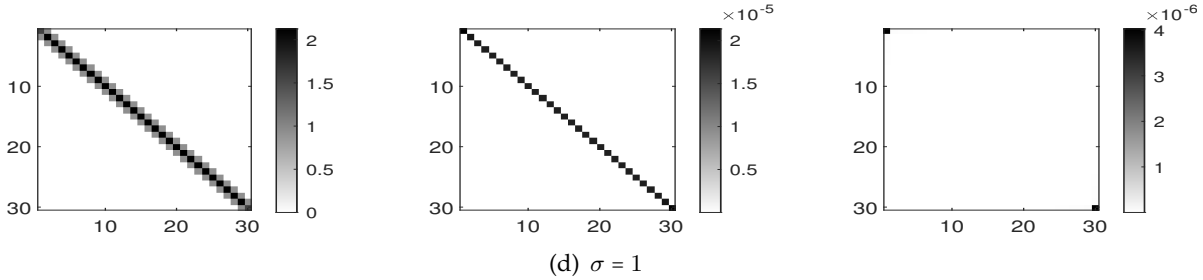
(a) $\sigma = 1000$



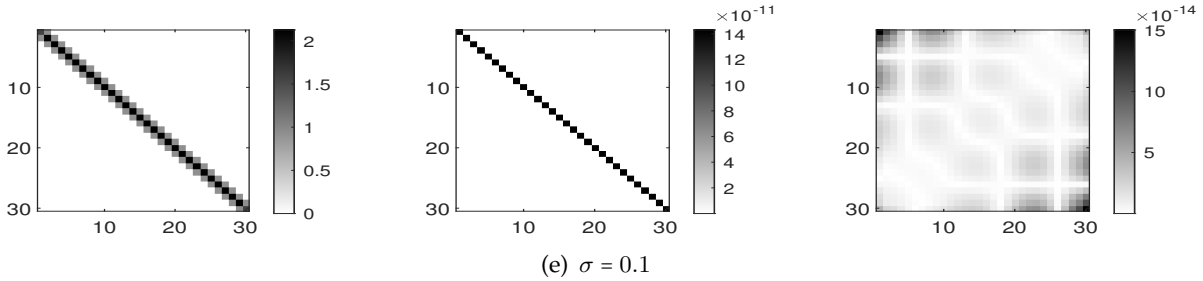
(b) $\sigma = 100$



(c) $\sigma = 10$



(d) $\sigma = 1$



(e) $\sigma = 0.1$

Fig. IV.3: Part 2, Model 1. **Left**: accurate inverse. **Middle:** solution by cvx. **Right**: solution by the first order algorithm.

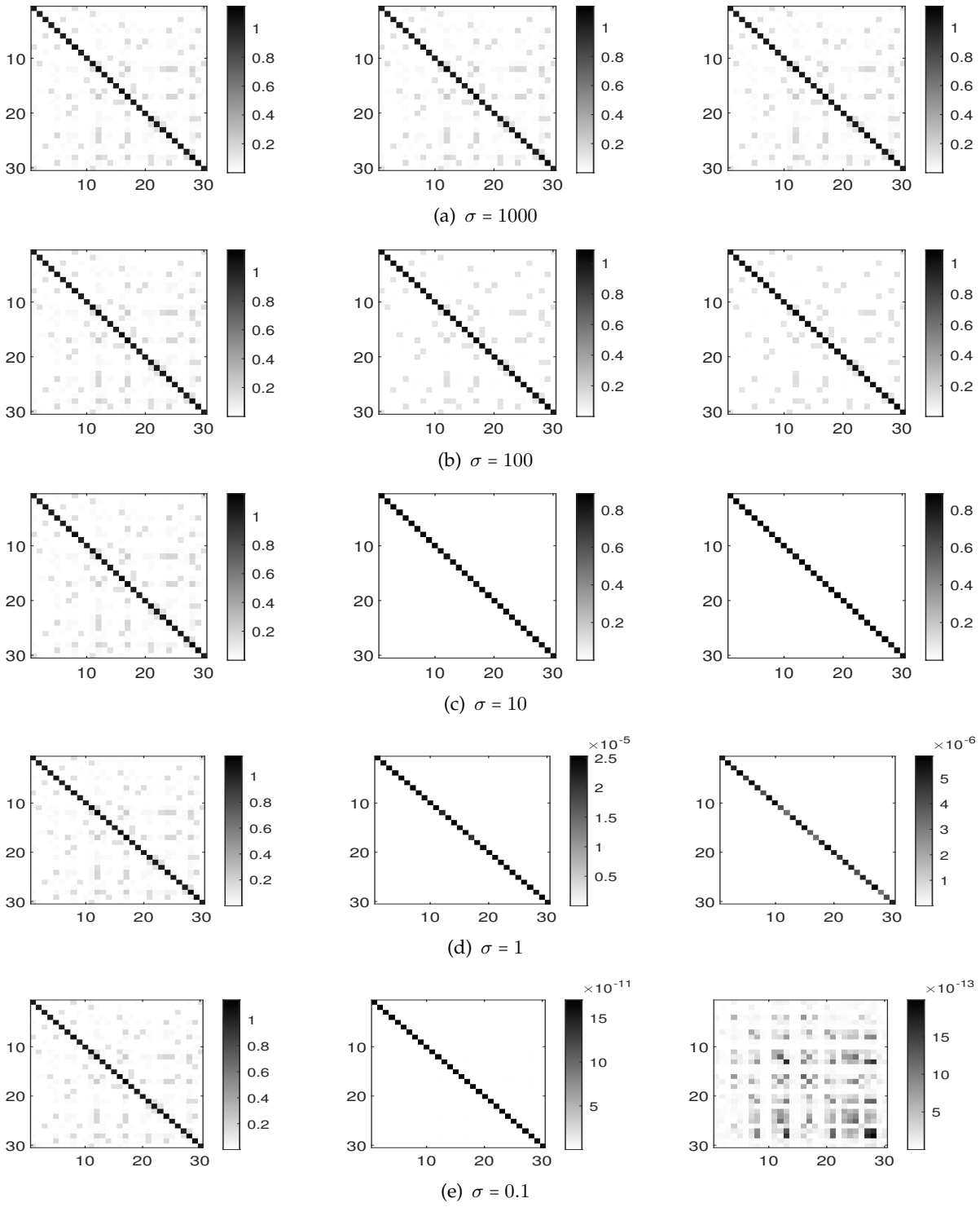


Fig. IV.4: Part 2, Model 1. **Left**: accurate inverse. **Middle:** solution by cvx. **Right**: solution by the first order algorithm.

samples according to a multivariate distribution with mean 0 and covariance matrix S. Based on the m samples x_i, we calculate the empirical covariance matrix as

$$\widehat{S}_m = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x})(x_i - \bar{x})^{\top} \tag{IV.1}$$

Then we calculate the estimated precision matrix $\widehat{\Sigma}_m$ according to $\widehat{S}_m$. The performace of the estimation is defined by the likelihood loss

$$L(\Sigma, S) = \langle \Sigma, S \rangle - \log\det(\Sigma) \tag{IV.2}$$

which we try to minimize. The test set is generated independently from the same distribution that generates training set. We set the size of the test set to be 100.

For criterion II.1, we perform grid search on ρ to determine the optimal value. By setting $m = 20, 40, 80, 160$, and (1) $\rho = 0.01i, 1 \le i \le 30$ for Model 1, or (2) $\rho = 0.01 + 0.05i, 0 \le i \le 20$ for Model 2. We are able to discover that the loss is in the form of "V", as Figure IV.5, IV.6 shows. In general, as m increases, the optimal ρ seems to increase as well. Compare $m = 20$ and $m = 40$, we find that the fact that $m < n$ drastically impair the performace. Under Model 1, setting $\rho = 0.05 \sim 0.1$ seems to be the best choice. While for Model 2, we had better set $\rho = 0.1 \sim 0.2$ To reproduce the results, first specify the model in "test3.m", then execute the scripts.

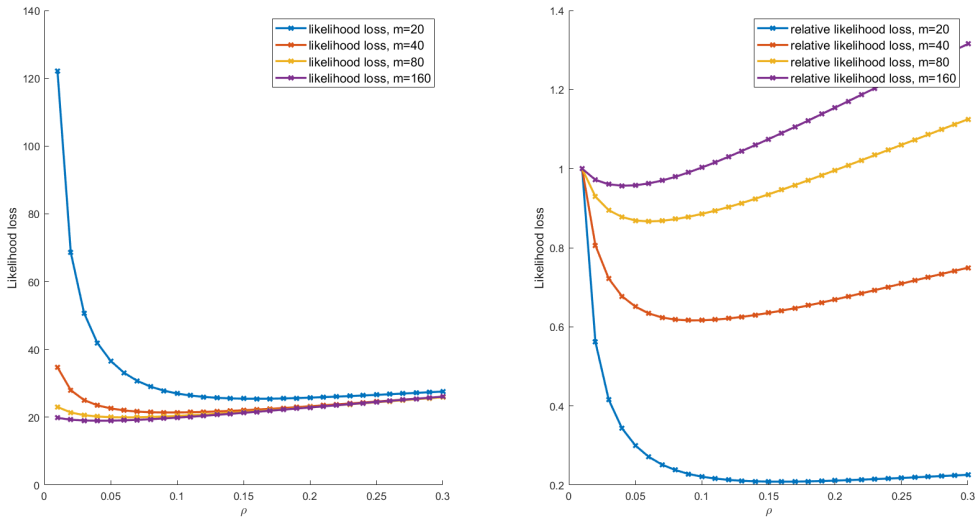


Fig. IV.5: Search for ρ in range $[0.01, 1]$ based on Model 1

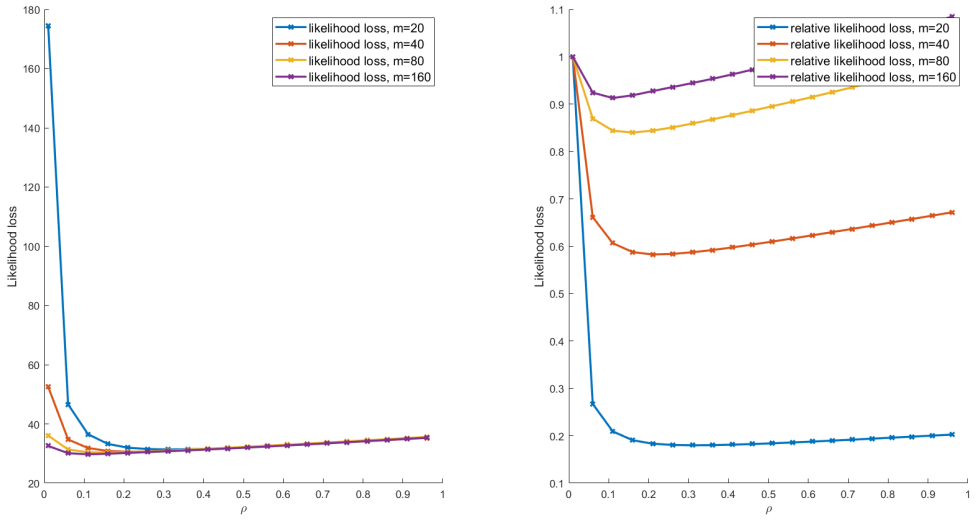


Fig. IV.6: Search for ρ in range $[0.01, 0.1]$ based on Model 2

For criterion III.1, we also conduct similar experiments in "test4.m". For Model 1, we set $m = 20, 30, 40, 80, 160$, and $\sigma = 0.01i, 1 \leq i \leq 30$. From Figure IV.7, the optimal choice seems to be $1/\sigma = 0.1$. For Model 2, we set $m = 20, 30, 40, 80, 160$, and $\sigma = 0.01 + 0.05i, 0 \leq i \leq 19$. From Figure IV.8, the optimal choice still seems to be $1/\sigma = 0.1$.

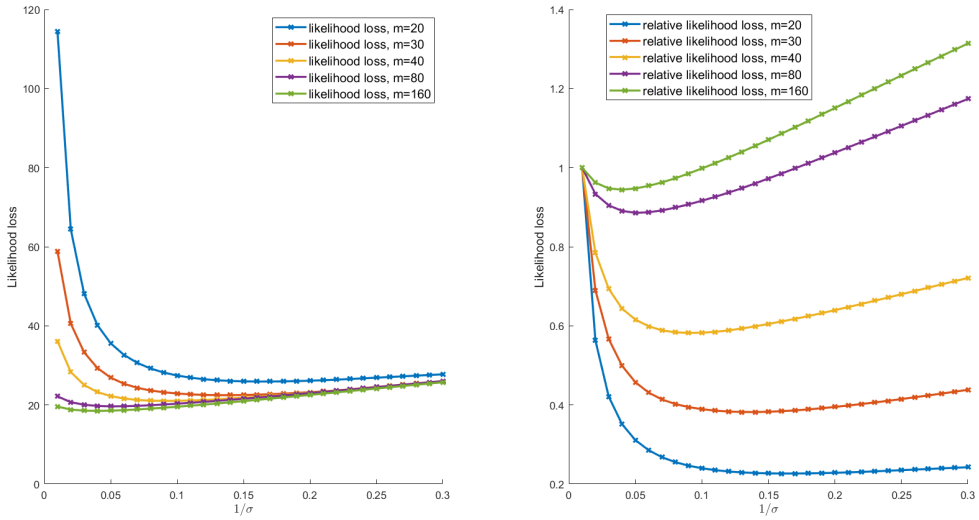


Fig. IV.7: Search for $1/\sigma$ based on Model 1

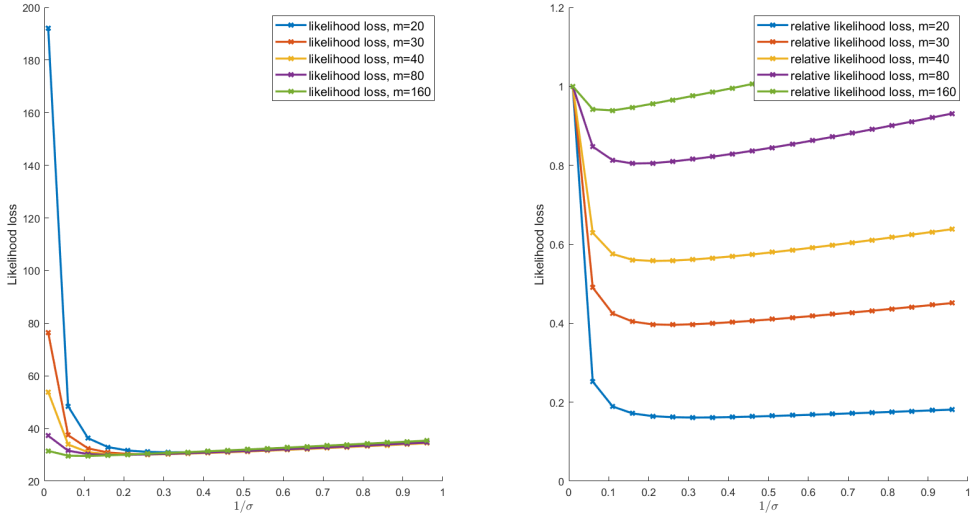


Fig. IV.8: Search for $1/\sigma$ in range based on Model 2

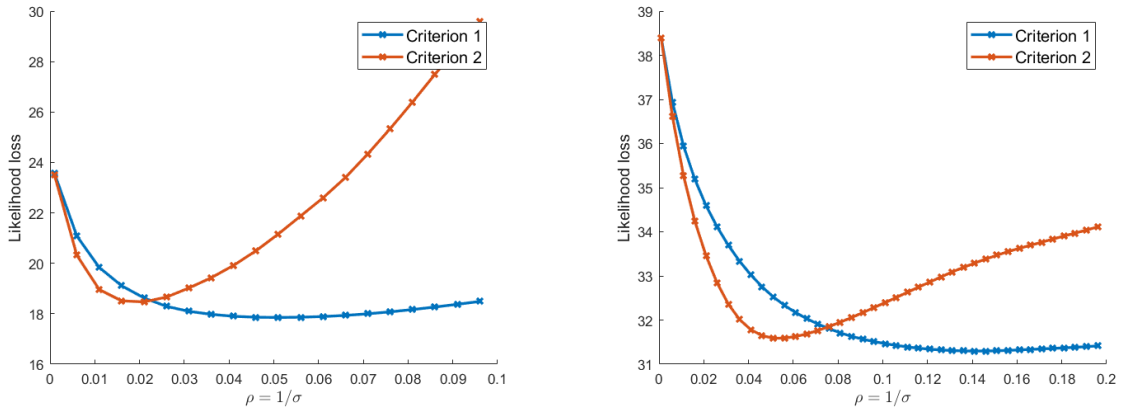
In Summary, $\rho = 0.1$ and $1/\sigma = 0.05$ will be advantageous. Besides, we observe that by setting the penalty parameters properly, it is possible to estimte the precision matrix given $m < n$ with low likelihood error. For small $m < n$, usually a slightly larger ρ or $1/\sigma$ is required, which could be twice of what in the normal setting.

D. Comparison between two criteria

From Section IV-C, the optimal choice of ρ or $1/\sigma$ is around 0.1. Hence, in this subsection, we set $m = 100$, and compare the solution of two different method in "test5.m". From IV.9, we find that only under small $1/sigma$, the criterion III.1 will be advantageous. It is not suprising, since criterion II.1 is the penalty form of likelihood. However, if we change the loss into

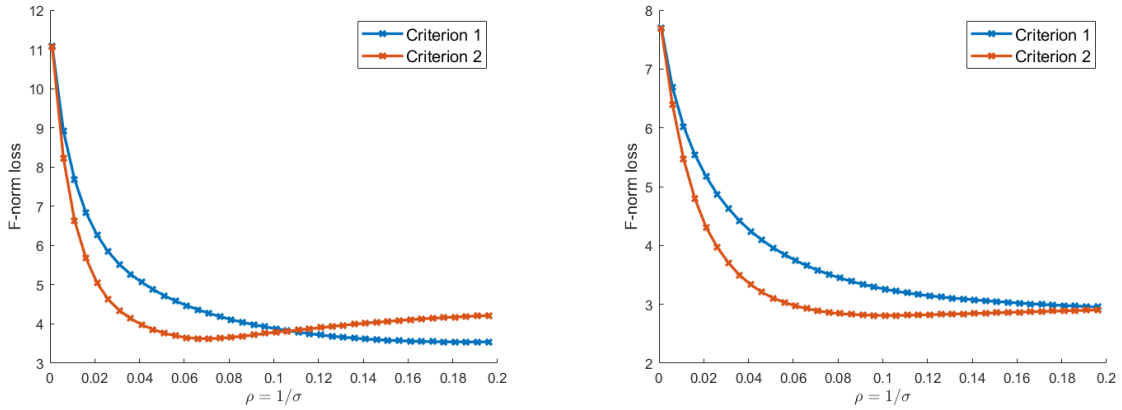
$$L(\Sigma, S) = \|\Sigma S - I\|_F \tag{IV.3}$$

and we get the Figure IV.10. We find that under most situations, III.1 outperforms II.1. However, we still observe that III.1 favors lower $1/\sigma$.



(a) Model 1 (b) Model 2

Fig. IV.9: Comparision by likelihood loss



(a) Model 1 (b) Model 2

Fig. IV.10: Comparision by F-norm loss

REFERENCES

[Cai et al., 2011] Cai, T., Liu, W., and Luo, X. (2011). A constrained ell_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607.

[Cai et al., 2010] Cai, T. T., Zhang, C.-H., Zhou, H. H., et al. (2010). Optimal rates of convergence for covariance matrix estimation. *The Annals of Statistics*, 38(4):2118–2144.

[Friedman et al., 2008] Friedman, J. H., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

[Toh et al., 1999] Toh, K.-C., Todd, M. J., and Tütüncü, R. H. (1999). Sdpt3—a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581.

[Wang and Jiang, 2020] Wang, C. and Jiang, B. (2020). An efficient admm algorithm for high dimensional precision matrix estimation via penalized quadratic loss. *Computational Statistics & Data Analysis*, 142:106812.

[Wen et al., 2010] Wen, Z., Yin, W., Goldfarb, D., and Zhang, Y. (2010). A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation. *SIAM Journal on Scientific Computing*, 32(4):1832–1857.

[Yang and Zhang, 2011] Yang, J. and Zhang, Y. (2011). Alternating direction algorithms for \ell_1-problems in compressive sensing. *SIAM journal on scientific computing*, 33(1):250–278.

[Yin and Osher, 2013] Yin, W. and Osher, S. (2013). Error forgetting of bregman iteration. *Journal of Scientific Computing*, 54(2-3):684–695.

[Yuan et al., 2013] Yuan, K., Ling, Q., Yin, W., and Ribeiro, A. (2013). A linearized bregman algorithm for decentralized basis pursuit. In *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5. IEEE.

[Zhang and Zou, 2014] Zhang, T. and Zou, H. (2014). Sparse precision matrix estimation via lasso penalized d-trace loss. *Biometrika*, 101(1):103–120.