MAT 280: Assignment #2

Due on Monday, May 2, 2016

Prof. Thomas Strohmer MF 13:30 - 15:00

Wenhao Wu

Contents

Problem 1

Let $i = \sqrt{-1}$ and set

$$\mathbf{A} = \begin{bmatrix} i & 0 & -i \\ 0 & i & -i \end{bmatrix}.$$

Using the null space property, show that l_1-minimization will recover any 1-sparse vector $\mathbf{x}$, given $\mathbf{Ax} = \mathbf{y}$.

Answer: The null space of $\mathbf{A}$ is $\{[1,1,1]^H\}$, therefore for any $\|\mathcal{S}\| = 1$ and any $\mathbf{h} \in \text{null}(\mathbf{A})\backslash\{0\}$ we have

$$\|\mathbf{h}_{\mathcal{S}^C}\|_1 = 2\|\mathbf{h}_{\mathcal{S}}\|_1 > 0 \tag{1}$$

which suggests $\|\mathbf{h}_{\mathcal{S}^C}\|_1 > \|\mathbf{h}_{\mathcal{S}}\|_1$, therefore $\mathbf{A}$ satisfies the nullspace property w.r.t all $\mathcal{S}$ satisfying $|\mathcal{S}| \leq 1$. Consequently, l_1-minimization will recover any 1-sparse vector $\mathbf{x}$.

Problem 2

On the connection between (in)coherence parameter μ and restricted isometry constant δ_s: Show that $\delta_1 = 0$, $\delta_2 = \mu$, and $\delta_s \leq (s-1)\mu$.

Answer: Denote $\mathbf{A} \in \mathbb{C}^{k \times d}$ as a matrix with unit 2-norm columns $\mathbf{a}_1, \ldots, \mathbf{a}_d$. For 1-sparse vector $\mathbf{x}$, assuming $x_i \neq 0$, we have

$$\|\mathbf{Ax}\|_2^2 = \|x_i \mathbf{a}_i\|_2^2 = |x_i|^2 = \|\mathbf{x}\|_2^2, \tag{2}$$

therefore $\delta_1 = 0$. For 2-sparse vector $\mathbf{x}$, assuming $x_i \neq 0$, $x_j \neq 0$, $i \neq j$, we have

$$\|\mathbf{Ax}\|_2^2 = \|x_i \mathbf{a}_i + x_j \mathbf{a}_j\|_2^2 = |x_i|^2 + |x_j|^2 + 2\,\text{Re}\big\{x_i x_j^* \langle \mathbf{a}_i, \mathbf{a}_j \rangle\big\}, \tag{3}$$

Since

$$2\,\text{Re}\big\{x_i x_j^* \langle \mathbf{a}_i, \mathbf{a}_j \rangle\big\} \leq (|x_i|^2 + |x_j|^2)|\langle \mathbf{a}_i, \mathbf{a}_j \rangle|, \tag{4a}$$

$$2\,\text{Re}\big\{x_i x_j^* \langle \mathbf{a}_i, \mathbf{a}_j \rangle\big\} \geq -(|x_i|^2 + |x_j|^2)|\langle \mathbf{a}_i, \mathbf{a}_j \rangle|, \tag{4b}$$

where equalities hold if and only if $x_j = \pm |x_i| \exp(i \arg\langle \mathbf{a}_i, \mathbf{a}_j \rangle)$, respectively. Consequently, we have

$$\|\mathbf{Ax}\|_2^2 \leq (1 + |\langle \mathbf{a}_i, \mathbf{a}_j \rangle|)\|\mathbf{x}\|_2^2 \leq (1+\mu)\|\mathbf{x}\|_2^2 \tag{5a}$$

$$\|\mathbf{Ax}\|_2^2 \geq (1 - |\langle \mathbf{a}_i, \mathbf{a}_j \rangle|)\|\mathbf{x}\|_2^2 \geq (1-\mu)\|\mathbf{x}\|_2^2 \tag{5b}$$

therefore $\delta_2 = \mu$. For s-sparse vector $\mathbf{x}$, assuming its support is $\mathcal{S} = \{i_1, \ldots, i_s\}$. We have

$$\begin{aligned}
\|\mathbf{Ax}\|_2^2 &= \left\|\sum_{p=1}^{s} x_p \mathbf{a}_p\right\|_2^2 \\
&= \sum_{p=1}^{s} |x|_{i_p}^2 + \sum_{p=1}^{s-1} \sum_{q=p+1}^{s} 2\,\text{Re}\Big\{x_{i_p} x_{i_q}^* \langle \mathbf{a}_{i_p}, \mathbf{a}_{i_q} \rangle\Big\} \\
&\leq \sum_{p=1}^{s} |x|_{i_p}^2 + \sum_{p=1}^{s-1} \sum_{q=p+1}^{s} (|x_{i_p}|^2 + |x_{i_q}|^2)|\langle \mathbf{a}_{i_p}, \mathbf{a}_{i_q} \rangle| \\
&\leq \sum_{p=1}^{s} |x|_{i_p}^2 + \mu \sum_{p=1}^{s-1} \sum_{q=p+1}^{s} (|x_{i_p}|^2 + |x_{i_q}|^2) \\
&= (1 + (s-1)\mu)\|\mathbf{x}\|_2^2
\end{aligned} \tag{6}$$

Similarly, we can prove that $\|\mathbf{Ax}\|_2^2 \geq (1 - (s-1)\mu)\|\mathbf{x}\|_2^2$. Consequently, we have $\delta_s \leq (s-1)\mu$.

Problem 3

Let $\mathbf{A} \in \mathbb{R}^{k \times d}$ be a Gaussian random matrix. Give an estimate for the coherence μ of $\mathbf{A}$.

Answer: Assume that $A_{ij} \sim \mathcal{N}(0,1)$ identically and independently, $i = 1, \ldots, k$, $j = 1, \ldots, d$. After normalizing $\mathbf{A}$ so that each column has unit 2-norm, from Problem 1 of Homework 1 we have

$$\langle \mathbf{a}_i, \mathbf{a}_j \rangle \leq \frac{\epsilon}{1-\epsilon} \tag{7}$$

with probability at least $1 - 6\exp\left(-k\epsilon^2/\alpha \log k\right) - 2k^{-\alpha+1}$ for $\alpha > 1$. Consequently, with union bound, we have

$$\begin{aligned} \mathbb{P}\left(\mu \leq \frac{\epsilon}{1-\epsilon}\right) &= 1 - \mathbb{P}\left(\max_{i \neq j} \langle \mathbf{a}_i, \mathbf{a}_j \rangle > \frac{\epsilon}{1-\epsilon}\right) \\ &\geq 1 - \sum_{i \neq j} \mathbb{P}\left(\langle \mathbf{a}_i, \mathbf{a}_j \rangle > \frac{\epsilon}{1-\epsilon}\right) \\ &\geq 1 - d(d-1)\left[3\exp\left(-k\epsilon^2/\alpha \log k\right) + k^{-\alpha+1}\right]. \end{aligned} \tag{8}$$

Problem 4

Consider $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a 100×400 Gaussian random matrix and $\mathbf{x}$ is an s-sparse vector of length 400. The locations of the non-zero entries of $\mathbf{x}$ are chosen uniformly at random and the non-zero coefficients of $\mathbf{x}$ are normal-distributed. For $s = 1, 2, \ldots$, solve

$$\min_{\mathbf{z}} \|\mathbf{z}\|_1, \text{ subject to } \mathbf{A}\mathbf{z} = \mathbf{y}, \tag{9}$$

(e.g. using the toolbox CVX). For each fixed s repeat the experiment 10 times. Create a graph plotting s versus the relative reconstruction error (averaged over the ten experiments for each s). Starting with which value of s (approximately) does l_1-minimization fail to recover $\mathbf{x}$?

Answer: The relative error

$$\epsilon = \frac{\|\hat{\mathbf{z}} - \mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2} \tag{10}$$

are evaluated with 100 randomly generated $\mathbf{A}$ and $\mathbf{x}$ for each s, where $\mathbf{z}$ is the solution of the l_1-minimimization problem (9) or the l_1-non-negative-minimization problem (11) computed using CVXPY. The mean and stand-deviation of ϵ are plotted in Fig. 1. The two problems start to fail to recover $\mathbf{x}$ from $s = 20$ and $s = 27$, respectively. It appears that l_1-non-negative-minimization can recover $\mathbf{x}$ over a larger range than l_1-minimization.
The python code for this simulation is as follows:

```
import numpy as np
import scipy as sp
import cvxpy as cvx

import timeit
import sys
from IPython.display import clear_output

```

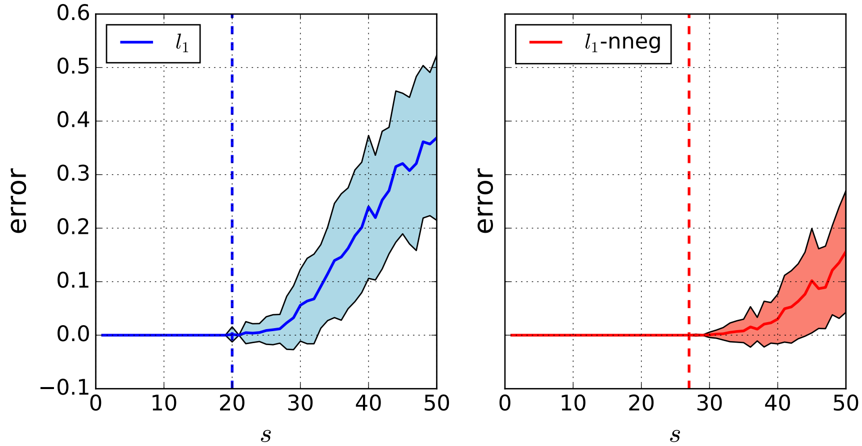

Figure 1: The mean and standard deviation of the error for the $l1$-minimizaion problem (left) and the $l1$-non-negative-minimizaion problem (right).

```
N = 100 # Number of repetition
k = 100
d = 400

s_range = np.arange(1, 51, dtype="int32") # range of s

A = np.random.randn(k, d, N)

err = np.zeros([len(s_range), N], dtype="float64") # The relative error
    ↪ for l1-minimization
err_nn = np.zeros([len(s_range), N], dtype="float64") # The relative error
    ↪  for non-neg l1-minimization

for idx, s in enumerate(s_range):
    X = np.random.randn(s, N)
    err_tmp = np.zeros(N, dtype="float64")
    for n in range(N):
        S = np.random.choice(d, s, replace = False) # Uniformly choose the
            ↪  locations of non-zero elements

        x = np.zeros(d, dtype="float64")
        x[S] = X[:, n]
        y = np.dot(A[:, :, n], x)
        z = cvx.Variable(d)
        prob = cvx.Problem(cvx.Minimize(cvx.norm(z, 1)), [A[:, :, n] * z
            ↪ == y,])
        prob.solve()
        err[idx, n] = (np.linalg.norm(z.value.flatten() - x) ** 2) / (np.
            ↪ linalg.norm(x) ** 2)

        x = np.zeros(d, dtype="float64")
```

```
        x[S] = np.absolute(X[:, n])
        y = np.dot(A[:, :, n], x)
        z = cvx.Variable(d)
        prob = cvx.Problem(cvx.Minimize(cvx.norm(z, 1)), [A[:, :, n] * z
            ↪ == y, z >= 0])
        prob.solve()
        err_nn[idx, n] = (np.linalg.norm(z.value.flatten() - x) ** 2) / (
            ↪ np.linalg.norm(x) ** 2)

    #process.stdout
    clear_output()
    print("s={0}, err={1}, err_nn={2}".format(s, err[idx, :].mean(),
        ↪ err_nn[idx, :].mean()))
    sys.stdout.flush()

err_mean = err.mean(axis = 1)
err_std = err.std(axis = 1)

err_nn_mean = err_nn.mean(axis = 1)
err_nn_std = err_nn.std(axis = 1)

threshold = 1e-10
idx_nz = np.where(err_mean > threshold)[0].min() # The first non-zero
    ↪ position
idx_nn_nz = np.where(err_nn_mean > threshold)[0].min()

import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline

axis_font = {'size':'20'}
mpl.rcParams['xtick.labelsize'] = 16
mpl.rcParams['ytick.labelsize'] = 16

fig, axs = plt.subplots(1, 2, sharey=True, sharex=True, figsize=(10, 5))

axs[0].plot(s_range, err_mean, 'blue', linewidth=2, label="$l_1$")
axs[0].fill_between(s_range, err_mean-err_std, err_mean+err_std, facecolor
    ↪ ='lightblue')
axs[0].axvline(s_range[idx_nz], color='blue', linestyle='--', linewidth=2)
axs[0].legend(prop={'size':16}, loc=2)
axs[0].grid()
axs[0].set_xlabel('$s$', **axis_font)
axs[0].set_ylabel('error', **axis_font)

axs[1].plot(s_range, err_nn_mean, 'red', linewidth=2, label="$l_1$-nneg")
axs[1].fill_between(s_range, err_nn_mean-err_nn_std, err_nn_mean+
    ↪ err_nn_std, facecolor='salmon')
axs[1].axvline(s_range[idx_nn_nz], color='red', linestyle='--', linewidth
```

```
    ↪ =2)
axs[1].legend(prop={'size':16}, loc=2)
axs[1].grid()
axs[1].set_xlabel('$s$', **axis_font)
axs[1].set_ylabel('error', **axis_font)

fig.savefig('error.pdf', dpi=10, bbox_inches='tight')
```

Problem 5

Same setup as in Problem 4, but now the non-zero entries of $\mathbf{x}$ are non-negative. Taking this information into account, we now solve

$$\min_{\mathbf{z}} \|\mathbf{z}\|_1, \text{ subject to } \mathbf{A}\mathbf{z} = \mathbf{y} \text{ and } \mathbf{z} \geq \mathbf{0} \tag{11}$$

(here, $\mathbf{z} \geq \mathbf{0}$ is meant entrywise, i.e., for each k: $z_k \geq 0$). (The positivity constraint is easy to include in CVX). Repeat the simulations as described in Problem 4. Compare your findings to the results from your experiments of Problem 4 and try to quantify the difference regarding the range for $\mathbf{s}$ for which recovery is still possible in this case.

Answer: See Problem 4.