MA5232/AY21-22 – Convex Programming : Assignment

Instructions

1. This is the homework for Part 3 of MA5232. There are five parts, and substantial programming is required.

2. While all languages are permitted, it is recommended that you code using Python or MATLAB.

3. You have **four** weeks to complete the assignment. It is due Friday, April 8, at 11:59pm.

4. You are required to write a short report for all parts. Please present your solutions neatly and clearly. A portion of the grading rubric depends on how clear your solutions are.

5. You are also required to submit an attachment with your code. You *must* write your own code.

6. While you are permitted to refer to textbooks or other online resources, you are *not* permitted to look at solutions that directly solve the homework problem. If you refer to resources, you *must* cite them.

7. You may verbally discuss your homeworks with your coursemates, but you must *not* refer to their solutions directly. If you have *any* discussions, you *must* acknowledge these discussions in your report. You also need to describe the extent of these discussions.

1 Convexity

(i) Suppose $\{C_i\}_{i=1}^{K}$ are convex sets. Show that $\cap_{i=1}^{K} C_i$ is also convex.

(ii) Show that this extends to intersections over arbitrary index sets; that is, show that $\cap_{i\in\mathcal{I}} C_i$ is convex if C_i, $i \in \mathcal{I}$, is a collection of convex sets.

(iii) A polyhedra is a set of the following form

$$\{x : Ax \leq b\}.$$

Show that polyhedra are convex sets.

(iv) Suppose $\{f_i\}_{i\in\mathcal{I}}$ is a collection of convex functions. Show that

$$f(x) = \sup\{f_i(x) : i \in \mathcal{I}\}$$

is also a convex function.

(v) Let X be a symmetric matrix. Briefly justify why

$$\lambda(X) = \sup_{\|u\|_2=1} u^T X u.$$

Here, $\lambda(X)$ is the largest eigenvalue of a matrix. Show that $\lambda(X)$ is a convex function in X. (Hint: $u^T X u$ is a linear function in X.)

[Note: In a previous version, the constraint was $\|u\|_2 \leq 1$. This is incorrect.]

2 Geometric Programming

A function $f : \mathbb{R}^n \to \mathbb{R}$ of the following form

$$f(x_1, \ldots, x_n) = c x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n},$$

where $x_i > 0$, $a_i \in \mathbb{R}$, and $c > 0$, is known as a *monomial*. A function that can be described as the sum of monomials is known as a *posynomial* $f(x_1, \ldots, x_n) = \sum_{j=1}^{N} c_j x_1^{a_{1j}} x_2^{a_{2j}} \ldots x_n^{a_{nj}}$.

An optimization problem of the form

$$\begin{aligned} \min_{x_1,\ldots,x_n>0} \quad & f_0(x_1,\ldots,x_n) \\ \text{s.t.} \quad & f_i(x_1,\ldots,x_n) \leq 1, \quad 1 \leq i \leq m \\ & h_j(x_1,\ldots,x_n) = 1, \quad 1 \leq j \leq p \end{aligned}$$

where $f_0, f_1, \ldots, f_m$ are posynomials and $h_1, \ldots, h_p$ are monomials, is known as a *geometric* program (GP). Geometric programs are *not* convex programs in general.

(i) By considering the transformation

$$y_i = \log x_i,$$

show that a GP can be transformed into a *convex program*.

(ii) We want to design a three dimensional box container. A box container can be specified in terms of its height, length, and width. We have an upper limit on the total surface area of this container, and an upper limit on the area of the ceiling. We also have an upper bound and a lower bound for the ratio between the height and the width of the container, as well as the ratio between the height and the length of the container. We wish to find the largest container (by volume) satisfying these constraints.

Express this problem in terms of a GP.

3 Compressed Sensing

In this example, we explore the compressed sensing problem. We explore the relationship between the sparsity of the signal, and the number of measurements required to recover the signal.

For this problem, you are required to solve a convex program multiple times.

If you are using MATLAB, you may consider using CVX from the following

http://cvxr.com/cvx/

If you are using Python, you may consider using CVXPY from the following

http://www.cvxpy.org/

You will need some amount of time to run the code, so do not leave this exercise to the last minute.

(i) Generate a random $m \times n$-dimensional matrix A where each entry is an i.i.d. random variable drawn from the standard normal distribution. Let $x^\star$ be a random sparse vector with sparsity s, and whose location of the non-zero entries are chosen uniformly at random. You may take the non-zero entries of the vector to be equal to one.

Let
$$y = Ax^\star.$$

Define $\hat{x}$ to be the optimal solution to the following
$$\min \quad \|x\|_1 \quad \text{s.t.} \quad y = Ax. \tag{1}$$

(ii) We say that a Linear Program is in standard form if it can be expressed as follows
$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Express (1) as a Linear Program in standard form.

(iii) We declare success if
$$\|\hat{x} - x^\star\| / \|x^\star\| \leq 10^{-4}.$$

(iv) We set $n = 50$. For every choice of $m \in \{1, \ldots, 50\}$ and $s \in \{1, \ldots, 50\}$ perform the above experiment 10 times. Record the average number of successes among these trials. Display the matrix recording the number of successes as a grayscale image. If you are using MATLAB, you may consider the `imagesc` function, and if you are using Python, you may consider the `imshow` function from the `matplotlib` package.

Do you observe a pattern? Specifically, do you observe a curve in which, if the number of measurements exceed the curve, then the probability of success is overwhelmingly high, and if the number of measurements is below the curve, then the probability of success is overwhelmingly low?

4 Matrix Completion

In this example, we consider the matrix completion problem. Let $Z \in \mathbb{R}^{m\times n}$ be a matrix. We observe some entries of the matrix Z indexed by the set Ω. Our goal is to fill in the missing entries in Ω^c.

To do so, we consider the optimization instance

$$\begin{aligned} \min_X \quad & \|X\|_* \\ \text{s.t.} \quad & \boxed{X_{ij} = Z_{ij} \quad \text{for all} \quad (i,j) \in \Omega} \end{aligned} \tag{2}$$

Here, recall that $\|X\|_*$ is the matrix nuclear norm, and is defined as the sum of the singular values of X.

$$\min_X \underline{\|X\|_*} = \min_{W_1 W_2} \tfrac{1}{2}\mathrm{tr}(W_1+W_2)$$

s.t $X \in \Theta$

$$\begin{pmatrix} W_1 & X \\ X^T & W_2 \end{pmatrix} \in PSD$$

(i) A semidefinite program is one of the form

$$\begin{aligned} \min \quad & \mathrm{tr}(CX) \\ \text{s.t.} \quad & \mathrm{tr}(A_iX) = b_i \\ & \underline{\boxed{X \text{ is PSD}}} \sim \tilde{X}, \end{aligned}$$

$$\min_{\boxed{\tilde{X}, W_1, W_2}} \cancel{\tfrac{1}{2}\mathrm{tr}(W_1+W_2)} \; \boxed{\tfrac{1}{2}\mathrm{tr}(\tilde{X})}$$

s.t [illegible] $\in PSD$

$\tilde{X} =$ [illegible] $\in \underline{PSD}$

By noting that

$$\|X\|_* \quad = \quad \frac{1}{2}\inf\{\mathrm{tr}(W_1) + \mathrm{tr}(W_2) : \begin{pmatrix} W_1 & X \\ X^T & W_2 \end{pmatrix} \text{ is PSD } \},$$

show that (2) can be expressed as a SDP.

(ii) We test the problem on a random instance. We pick $(m,n) = (100,100)$. Generate a rank $r = 2$ random matrix Z of size $m \times n$. One way to generate a rank r matrix is to generate two matrices of dimensions $m \times r$ and $n \times r$, denoted by U and V, where each entry is drawn from the standard normal distribution. Then take

$$Z = UV^T.$$

$\frac{1}{2} \geq$ $\quad X$ [illegible]

~~$X =$~~ [illegible] $\in \mathbb{R}$

(iii) Next, we select the entries Ω – these are the location of the entries we will pick. Choose these locations uniformly at random so that $|\Omega| = k$, where k is chosen from the set $\{100, 200, \ldots, 3000\}$.

(iv) Solve the instance (2). Compute the Mean Squared Error for the unobserved entries

$$\frac{1}{|\Omega^c|}\sum_{ij\in\Omega^c}(X_{ij} - Z_{ij})^2$$

Plot a graph of the MSE over the different choices of $k \in \{100, 200, \ldots, 3000\}$.

(v) *(Removed – ignore this problem)* Repeat the experiment with 10 different instances of Z, and plot the graphs of the MSE (of all 10 instances) on the same graph.

5 Movie Lens

In this example, we consider a practical implementation of the matrix completion problem.

(i) First, download the dataset `data.csv`. The dataset is a collection of actual movie ratings by users, provided by MovieLens.

`https://grouplens.org/datasets/movielens/`

The dataset has 100,000 movie ratings. The first column records the user id, the second column records the movie id, while the third column records the movie rating. All ratings are integer values between 1 to 5, inclusive.

Note. Every user only rates a subset of movies.

(ii) We divide the dataset into two sets. Randomly select 90% to form the training dataset and the remainder to form the testing dataset.

(iii) We specify a baseline estimator. Given a movie id, the baseline computes the average rating of all users who have rated the movie *in the training dataset*, and outputs the value as the estimated rating for all other users. Compute the squared error loss of this estimator on the *testing dataset*.

(iv) Next, we apply our matrix completion algorithm. In principle, we would want to solve

$$\begin{aligned} \min_{X} \quad & \operatorname{rank}(X) \\ \text{s.t.} \quad & X_{ij} = Z_{ij} \quad \text{for all} \quad (i,j) \in \Omega \end{aligned}$$

Unfortunately, this problem is *non convex*. There is a natural convex relaxation based on the matrix nuclear norm; however, you will find that, on the scale of the problem size we are currently working with, such methods quickly run out of memory and can be very expensive to execute. Instead, we consider a heuristic that is much simpler to compute, and gives a reasonably good solution.

This is the procedure we apply: Fix some rank parameter r. Initialize the matrix X to be all zeros.

(a) Set the entries X_{ij} to be equal to the MovieLens rating if the pair (i,j) appears in the training dataset.

(b) Compute the SVD of the matrix $USV^T = X$.

(c) Let S_{new} be the matrix obtained by setting all but the r largest entries of S to be equal to zero.

(d) Update $X_{new} = US_{new}V^T$. What we are effectively computing is the closest matrix of rank r to X in the Frobenius norm.

(e) Record the error $\|X - X_{new}\|_F$.

(f) Set $X = X_{new}$.

(g) Repeat the procedure for 200 iterations. Plot the errors over these iterations to see if the matrix X converges.

(v) Now we know that all ratings are on a scale of 1 to 5. The completed matrix might not necessarily obey this property. Hence, clip all values below 1 to be equal to 1, and all values exceeding 5 to be equal to 5.

What is the squared error loss on the testing dataset? How does it compare with the performance of the baseline estimator?

(vi) Repeat the process for $r \in \{1, 2, \ldots, 20\}$. Compare the squared error loss of the prediction with the baseline over these values of r. What do you observe?