Convex Optimization

Homework 7

Spring 1401
Due date: 13th of Khordad

---

1. *Minimizing a quadratic function.* Consider the problem of minimizing a quadratic function:

$$\text{minimize } f(x) = (1/2)x^T P x + q^T x + r,$$

where $P \in \mathbf{S}^n$ (but we do not assume $P \succeq 0$).

   (a) Show that if $P \succeq 0$, i.e., the objective function f is not convex, then the problem is unbounded below.

   (b) Now suppose that $P \succeq 0$ (so the objective function is convex), but the optimality condition $Px^\star = -q$ does not have a solution. Show that the problem is unbounded below.

2. *Smoothed fit to given data.* Consider the problem

$$\text{minimize } f(x) = \sum_{i=1}^{n} \psi\left(x_i - y_i\right) + \lambda \sum_{i=1}^{n-1} \left(x_{i+1} - x_i\right)^2$$

where $\lambda > 0$ is smoothing parameter, ψ is a convex penalty function, and $x \in \mathbf{R}^n$ is the variable. We can interpret x as a smoothed fit to the vector y.

   (a) What is the structure in the Hessian of f ?

   (b) Extend to the problem of making a smooth fit to two-dimensional data, i.e., minimizing the function

$$\sum_{i,j=1}^{n} \psi\left(x_{ij} - y_{ij}\right) + \lambda \left(\sum_{i=1}^{n-1} \sum_{j=1}^{n} \left(x_{i+1,j} - x_{ij}\right)^2 + \sum_{i=1}^{n} \sum_{j=1}^{n-1} \left(x_{i,j+1} - x_{ij}\right)^2 \right),$$

   with variable $X \in \mathbf{R}^{n \times n}$, where $Y \in \mathbf{R}^{n \times n}$ and $\lambda > 0$ are given.

3. Derive the Newton equation for the unconstrained minimization problem

$$\text{minimize } (1/2)x^T x + \log \sum_{i=1}^{m} \exp\left(a_x^T x + b_i\right).$$

Give an efficient method for solving the Newton system, assuming the matrix $A \in \mathbf{R}^{m \times n}$ (with rows a_i^T) is dense with $m \ll n$. Give an approximate flop count of your method.

4. *Schur complements.* Consider a matrix $X = X^T \in \mathbf{R}^{n \times n}$ partitioned as

$$X = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

where $A \in \mathbf{R}^{k \times k}$. If $\det A \neq 0$, the matrix $S = C - B^T A^{-1} B$ is called the Schur complement of A in X. Schur complements arise in many situations and appear in many important formulas and theorems. For example, we have $\det X = \det A \det S$. (You don't have to prove this.)

   (a) The Schur complement arises when you minimize a quadratic form over some of the variables. Let $f(u, v) = (u, v)^T X (u, v)$, where $u \in \mathbf{R}^k$. Let $g(v)$ be the minimum value of f over u, i.e., $g(v) = \inf_u f(u, v)$. Of course $g(v)$ can be $-\infty$. Show that if $A \succ 0$, we have $g(v) = v^T S v$.

(b) The Schur complement arises in several characterizations of positive definiteness or semidefiniteness of a block matrix. As examples we have the following three theorems:

- $X \succ 0$ if and only if $A \succ 0$ and $S \succ 0$.
- If $A \succ 0$, then $X \succeq 0$ if and only if $S \succeq 0$.
- $X \succeq 0$ if and only if $A \succeq 0, B^T\left(I - AA^{\dagger}\right) = 0$ and $C - B^T A^{\dagger} B \succeq 0$, where $A^{\dagger}$ is the pseudo-inverse of A. ($C - B^T A^{\dagger} B$ serves as a generalization of the Schur complement in the case where A is positive semidefinite but singular.)

Prove one of these theorems. (You can choose which one.)

5. *Infeasible start Newton method for LP centering problem.* Implement the infeasible start Newton method for solving the centering problem arising in the standard form L.P,

$$\begin{array}{ll} \text{minimize} & c^T x - \sum_{i-1}^{n} \log x_i \\ \text{subject to} & Ax = b, \end{array}$$

with variable x. The data are $A \in \mathbf{R}^{m \times n}$, with $m < n, c \in \mathbf{R}^n$, and $b \in \mathbf{R}^m$. You can assume that A is full rank. This problem cannot be solved when it is infeasible or unbounded below. Your code should accept A, b, c, and x_0, and return $x^\star$, the primal optimal point, $\nu^\star$, a dual optimal point, and the number of Newton steps executed. The initial point $x^{(0)}$ must satisfy $x^{(0)} \succ 0$, but it need not satisfy the equality constraints.

Use the block elimination method to compute the Newton step. (You can also compute the Newton step via the KKT system, and compare the result to the Newton step computed via block elimination. The two steps should be close, but if any x_i is very small, you might get a warning about the condition number of the KKT matrix.)

Plot $\|r(x, \nu)\|_2$, the norm of the concatenated primal and dual residuals, versus iteration k for various problem data and initial points, to verify that your implementation achieves quadratic convergence. As stopping criterion, you can use $\|r(x, \nu)\|_2 \leq 10^{-6}$ (which means the problem was solved) or some maximum number of iterations (say, 50) was reached, which means it was not solved (likely because the problem is either infeasible or unbounded below). For a fixed problem instance, experiment with varying the algorithm parameters α and β, observing the effect on the total number of Newton steps required.

To generate problem data (i.e., A, b, c, x_0) that are feasible, you can first generate A, then random positive vector p, and set $b = Ap$. You can be sure that the problem is not unbounded by making one row of A have positive entries. You may also want to check that A is full rank.

Test the behavior of your implementation on data instances that are not feasible, and also ones that are unbounded below.

6. *Standard form LP barrier method with infeasible start Newton method.* Implement the barrier method for the standard form LP,

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \quad x \succeq 0 \end{array}$$

with variable $x \in \mathbf{R}^n$, where $A \in \mathbf{R}^{m \times n}$, with $m < n$, with A full rank. (Your method will of course fail if the problem is not strictly feasible, or if it is unbounded.)

Use the centering code that you developed in question 5. Your LP solver should take as argument A, b, c, and return primal and dual optimal points $x^\star, \nu^\star$, and $\lambda^\star$.

You can terminate your barrier method when the duality gap, as measured by n/t, is smaller than 10^{-3}. (If you make the tolerance much smaller, you might run into some numerical trouble.)

Check your LP solver against the solution found by CVX* for several problem instances. The comments in question 5 on how to generate random data hold here too.

Experiment with the parameter μ to see the effect on the number of Newton steps per centering step, and the total number of Newton steps required to solve the problem.

Plot the progress of the algorithm, for a problem instance with $n = 500$ and $m = 100$, showing duality gap (on a log scale) on the vertical axis, versus the cumulative total number of Newton steps (on a linear scale) on the horizontal axis.

Your algorithm should return a $2 \times k$ matrix history, (where k is the total number of centering steps), whose first row contains the number of Newton steps required for each centering step, and whose second row shows the duality gap at the end of each centering step. In order to get a plot that looks like the ones in the book (e.g., figure 11.4, page 572), in Julia, with PyPlot you can use the following:

```
using PyPlot
step(cumsum(history[1,:]'), history[2,:]')
```

In Python, also with PyPlot you should use:

```
import matplotlib.pyplot as plt
plt.step(np.cumsum(history [0,:]), history [1,:], where="post")
plt. yscale("log")
plt. show()
```

7. **Optional**. *Estimation of a vector from one-bit measurements.* A system of m sensors is used to estimate an unknown parameter $x \in \mathbf{R}^n$. Each sensor makes a noisy measurement of some linear combination of the unknown parameters, and quantizes the measured value to one bit: it returns $+1$ if the measured value exceeds a certain threshold, and -1 otherwise. In other words, the output of sensor i is given by

$$y_i = \operatorname{sign}\left(a_i^T x + v_i - b_i\right) = \begin{cases} 1 & a_i^T x + v_i \geq b_i \\ -1 & a_i^T x + v_i < b_i, \end{cases}$$

where a_i and b_i are known, and v_i is measurement error. We assume that the measurement errors v_i are independent random variables with a zero-mean unit-variance Gaussian distribution (i.e., with a probability density $\phi(v) = (1/\sqrt{2\pi})e^{-v^2/2}\big)$. As a consequence, the sensor outputs y_i are random variables with possible values ± 1. We will denote $\operatorname{prob}(y_i = 1)$ as $P_i(x)$ to emphasize that it is a function of the unknown parameter x :

$$P_i(x) = \operatorname{prob}(y_i = 1) = \operatorname{prob}\left(a_i^T x + v_i \geq b_i\right) = \frac{1}{\sqrt{2\pi}} \int_{b_i - a_i^T x}^{\infty} e^{-t^2/2} dt$$

$$1 - P_i(x) = \operatorname{prob}(y_i = -1) = \operatorname{prob}\left(a_i^T x + v_i < b_i\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{b_i - a_i^T x} e^{-t^2/2} dt.$$

The problem is to estimate x, based on observed values $\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_m$ of the m sensor outputs. We will apply the maximum likelihood (ML) principle to determine an estimate $\hat{x}$. In maximum likelihood estimation, we calculate $\hat{x}$ by maximizing the log-likelihood function

$$l(x) = \log\left(\prod_{\bar{y}_i = 1} P_i(x) \prod_{\bar{y}_i = -1} (1 - P_i(x))\right) = \sum_{\bar{y}_i = 1} \log P_i(x) + \sum_{\bar{y}_i = -1} \log(1 - P_i(x)).$$

(a) Show that the maximum likelihood estimation problem maximize $l(x)$ is a convex optimization problem. The variable is x. The measured vector $\bar{y}$, and the parameters a_i and b_i are given.

(b) Solve the ML estimation problem with data defined in one_bit_meas_data.m, using Newton's method with backtracking line search. This file will define a matrix A (with rows a_i^T), a vector b, and a vector $\bar{y}$ with elements ± 1.

Remark. The Matlab functions erfc and erf cx are useful to evaluate the following functions:

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} e^{-t^2/2} dt = \frac{1}{2} \operatorname{erfc}\left(-\frac{u}{\sqrt{2}}\right), \quad \frac{1}{\sqrt{2\pi}} \int_{u}^{\infty} e^{-t^2/2} dt = \frac{1}{2} \operatorname{erfc}\left(\frac{u}{\sqrt{2}}\right)$$

$$\frac{1}{\sqrt{2\pi}} e^{u^2/2} \int_{-\infty}^{u} e^{-t^2/2} dt = \frac{1}{2} \operatorname{erfcx}\left(-\frac{u}{\sqrt{2}}\right), \quad \frac{1}{\sqrt{2\pi}} e^{u^2/2} \int_{u}^{\infty} e^{-t^2/2} dt = \frac{1}{2} \operatorname{erfcx}\left(\frac{u}{\sqrt{2}}\right).$$

8. **Optional**. *Bounding portfolio risk with incomplete covariance information.* Consider the following instance of the problem described in §4.6, on p 171 − 173 of Convex Optimization. We suppose that Σ_{ii}, which are the squares of the price volatilities of the assets, are known. For the off-diagonal entries of Σ, all we know is the sign (or, in some cases, nothing at all). For example, we might be given that $\Sigma_{12} \geq 0, \Sigma_{23} \leq 0$, etc. This means that we do not know the correlation between p_1 and p_2, but we do know that they are nonnegatively correlated (i.e., the prices of assets 1 and 2 tend to rise or fall together). Compute $\sigma^2_{\rm wc}$, the worst-case variance of the portfolio return, for the specific case

$$x = \begin{bmatrix} 0.1 \\ 0.2 \\ -0.05 \\ 0.1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 0.2 & + & + & \pm \\ + & 0.1 & - & - \\ + & - & 0.3 & + \\ \pm & - & + & 0.1 \end{bmatrix}$$

where a "t" entry means that the element is nonnegative, a "„ $means the entry is nonpositive, and$"±" means we don't know anything about the entry. (The negative value in x represents a short position: you sold stocks that you didn't have, but must produce at the end of the investment period.) In addition to σ^2_{wc}, give the covariance matrix Σ_{wc} associated with the maximum risk. Compare the worst-case risk with the risk obtained when Σ is diagonal.

Good Luck!