Convex Optimization
Homework 7

Spring 1400
Due date: 7th of Khordad

---

1. Consider the problem of minimizing a quadratic function:

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

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

   (a) Show that if $P \not\succeq 0$, 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 $Px^* = -q$ does not have a solution. Show that the problem is unbounded below.

2. Let Δx_{nsd} and Δx_{sd} be the normalized and unnormalized steepest descent directions at x, for the norm $\|.\|$. Prove the following identities.

   (a) $\nabla f(x)^T \Delta x_{nsd} = -\|\nabla f(x)\|_*$.
   (b) $\nabla f(x)^T \Delta x_{sd} = -\|\nabla f(x)\|_*^2$.
   (c) $\Delta x_{sd} = \text{argmin}_v(\nabla f(x)^T v + (1/2)\|v\|^2)$.

3. Consider the problem

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

where $\lambda > 0$ is smoothing parameter, ψ is a convex penalty function, and $x \in \mathbb{R}^n$ 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

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

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

4. Consider a standard form LP and its dual

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \succeq 0 \end{array} \qquad\qquad \begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & A^T y \preceq c, \end{array}$$

with $A \in \mathbb{R}^{m\times n}$ and $\mathbf{rank}(A) = m$. In the barrier method the (feasible) Newton method is applied to the equality constrained problem

$$\begin{array}{ll} \text{minimize} & tc^T x + \phi(x) \\ \text{subject to} & Ax = b, \end{array}$$

where $t > 0$ and $\phi(x) = -\sum_{i=1}^n \log x_i$. The Newton equation at a strictly feasible $\hat{x}$ is given by

$$\begin{bmatrix} \nabla^2\phi(\hat{x}) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ w \end{bmatrix} = \begin{bmatrix} -tc - \nabla\phi(\hat{x}) \\ 0 \end{bmatrix}.$$

Suppose $\lambda(\hat{x}) \leq 1$ where $\lambda(\hat{x})$ is the Newton decrement at $\hat{x}$.

(a) Show that $\hat{x} + \Delta x$ is primal feasible.

(b) Show that $y = -(1/t)w$ is dual feasible.

(c) Let p^* be the optimal value of the LP. Show that

$$c^T\hat{x} - p^* \leq \frac{n + \lambda(\hat{x})\sqrt{n}}{t}.$$

5. In the following three parts of this exercise, you will implement a barrier method for solving 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 $x \in \mathbb{R}^n$, where $A \in \mathbb{R}^{m\times n}$, with $m < n$. Throughout these exercises we will assume that A is full rank,and the sublevel sets $\{x \mid Ax = b, x \succeq 0, c^T x \leq \gamma\}$ are all bounded. (If this is not the case, the centering problem is unbounded below.)

(a) *Centering step.* Implement Newton's method for solving the centering problem

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

with variable x, given a strictly feasible starting point x_0. Your code should accept A, b, c, and x_0, and return x^*, the primal optimal point, ν^*, a dual optimal point, and the number of Newton steps executed.

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 $\lambda^2/2$ versus iteration k, for various problem data and initial points, to verify that your implementation gives asymptotic quadratic convergence. As stopping criterion, you can use $\lambda^2/2 \leq 10^{-6}$. Experiment with varying the algorithm parameters α and β, observing the effect on the total number of Newton steps required, for a fixed problem instance. Check that your computed x^* and ν^* (nearly) satisfy the KKT conditions.

To generate some random problem data (*i.e.*, A, b, c, x_0), we recommend the following approach. First, generate A randomly. (You might want to check that it has full rank.) Then generate a random positive vector x_0, and take $b = Ax_0$. (This ensures that x_0 is strictly feasible.) The parameter c can be chosen randomly. To be sure the sublevel sets are bounded, you can add a row to A with all positive elements. If you want to be able to repeat a run with the same problem data, be sure to set the state for the uniform and normal random number generators.

Here are some hints that may be useful.

- We recommend computing λ^2 using the formula $\lambda^2 = -\Delta x_{nt}^T \nabla f(x)$. You don't really need λ for anything; you can work with λ^2 instead. (This is important for reasons described below.)
- There can be small numerical errors in the Newton step Δx_{nt} that you compute. When x is nearly optimal, the computed value of λ^2, *i.e.*, $\lambda^2 = -\Delta x_{nt}^T \nabla f(x)$, can actually be (slightly) negative. If you take the squareroot to get λ, you'll get a complex number, and you'll never recover. Moreover, your line search will never exit. However, this only happens when x is nearly optimal. So if you exit on the condition $\lambda^2 \leq 10^{-6}$, everything will be fine, even when the computed value of λ^2 is negative.
- For the line search, you must first multiply the step size t by β until $x + t\Delta x_{nt}$ is feasible (*i.e.*, strictly positive). If you don't, when you evaluate f you'll be taking the logarithm of negative numbers, and you'll never recover.

(b) *LP solver with strictly feasible starting point.* Using the centering code from part **(a)**, implement a barrier method to solve 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 $x \in \mathbb{R}^n$, given a strictly feasible starting point x_0. Your LP solver should take as argument A, b, c, and x_0, and return x^*.

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 part **(a)** 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), you should use the following code:

```
[xx, yy] = stairs(cumsum(history(1,:)),history(2,:));
semilogy(xx,yy);
```

(c) *LP solver.* Using the code from part **(b)**, implement a general standard form LP solver, that takes arguments A, b, c, determines (strict) feasibility, and returns an optimal point if the problem is (strictly) feasible.

You will need to implement a phase I method, that determines whether the problem is strictly feasible, and if so, finds a strictly feasible point, which can then be fed to the code from part **(b)**. In fact, you can use the code from part **(b)** to implement the phase I method. To find a strictly feasible initial point x_0, we solve the phase I problem

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & Ax = b \\ & x \succeq (1-t)\mathbf{1}, \quad t \geq 0 \end{array}$$

with variables x and t. If we can find a feasible (x, t), with $t < 1$, then x is strictly feasible for the original problem. The converse is also true, so the original LP is strictly feasible if and only if $t^* < 1$, where t^* is the optimal value of the phase I problem.

We can initialize x and t for the phase I problem with any x^0 satisfying $Ax^0 = b$, and $t^0 = 2 - \min_i x_i^0$. (Here we can assume that $\min_i x_i^0 \leq 0$; otherwise x^0 is already a strictly feasible point, and we are done.) You can use a change of variable $z = x + (t-1)\mathbf{1}$ to transform the phase I problem into the form in part **(b)**.

Check your LP solver against CVX* on several numerical examples, including both feasible and infeasible instances.

Optional

1. Consider the unconstrained problem

$$\text{minimize} \quad f(x) = -\sum_{i=1}^m \log(1 - a_i^T x) - \sum_{i=1}^n \log(1 - x_i^2),$$

with variable $x \in \mathbb{R}^n$, and $\mathbf{dom} f = \{x \mid a_i^T x < 1, i = 1, ..., m, |x_i| < 1, i = 1, ..., n\}$. This is the problem of computing the analytic center of the set of linear inequalities

$$a_i^T x \leq 1, \quad i = 1, \ldots, m, \quad |x_i| \leq 1, \quad i = 1, \ldots, n.$$

Note that we can choose $x^{(0)} = 0$ as our initial point. You can generate instances of this n problem by choosing a_i from some distribution on $\mathbb{R}^n$.

(a) Use the gradient method to solve the problem, using reasonable choices for the backtracking parameters, and a stopping criterion of the form $\|f(x)\|_2 \leq \eta$. Plot the objective function and step length versus iteration number. (Once you have determined p^* to high accuracy, you can also plot $f - p^*$ versus iteration.) Experiment with the backtracking parameters α and β to see their effect on the total number of iterations required. Carry these experiments out for several instances of the problem, of different sizes.

(b) Repeat using Newton's method, with stopping criterion based on the Newton decrement λ. Look for quadratic convergence. You do not have to use an efficient method to compute the Newton step; you can use a general purpose dense solver, although it is better to use one that is based on a Cholesky factorization.

Hint. Use the chain rule to find expressions for $\nabla f(x)$ and $\nabla^2 f(x)$.

2. The cost of Newton's method is dominated by the cost of evaluating the Hessian $\nabla^2 f(x)$ and the cost of solving the Newton system. For large problems, it is sometimes useful to replace the Hessian by a positive definite approximation that makes it easier to form and solve for the search step. In this problem we explore some common examples of this idea.

   For each of the approximate Newton methods described below, test the method on some instances of the analytic centering problem described in the previous problem, and compare the results to those obtained using the Newton method and gradient method.

   (a) *Re-using the Hessian.* We evaluate and factor the Hessian only every N iterations, where $N > 1$, and use the search step $\Delta x = -H^{-1}\nabla f(x)$, where H is the last Hessian evaluated. (We need to evaluate and factor the Hessian once every N steps; for the other steps, we compute the search direction using back and forward substitution.)

   (b) *Diagonal approximation.* We replace the Hessian by its diagonal, so we only have to evaluate the n second derivatives $\partial^2 f(x)/\partial x_i^2$, and computing the search step is very easy.

Good luck!