CO 602: Fundamentals of Optimization
Fall 2010
Problem Set 8
S. Vavasis

Handed out: 2010-Nov-26.

Due: 2010-Dec-3.

1. Recall the vertex cover problem, which is NP-hard: given a graph $G = (N, E)$, find a subset $S \subset N$ of minimum size such that for every $\{i, j\} \in E$, either $i \in S$ or $j \in S$ (or both). Assume that G has no nodes of degree 0, else the problem is unsolvable. The following algorithm, due to Yannakakis and Gavril independently, is guaranteed to compute a vertex cover that is within a factor of 2 of optimal: Repeat the following iteration until no nodes remain. Select an arbitrary edge in the graph, say $\{u, v\}$, insert both u and v into S, and then delete both u and v from G as well as any edges incident upon u or v.

   (a) Show that the set S computed by this algorithm is indeed a vertex cover.

   (b) Show that its size is at most twice as large as the minimum cover. [Hint: Let the sequence of edges $\{u, v\}$ selected on each iteration be called the 'distinguished' edges. There are no shared endpoints among any of the distinguished edges. Why not? And how does this answer the question?]

2. Consider the problem of minimizing $f(\mathbf{x}) = \delta_1 x_1^2 + \cdots + \delta_n x_n^2$ where $\delta_1, \ldots, \delta_m < 0$ and $\delta_{m+1}, \ldots, \delta_n > 0$ for some $m \in \{1, \ldots, n-1\}$.

   (a) This problem is nonconvex and has a unique stationary point at the origin. Explain.

   (b) Suppose steepest descent is applied to minimize f. Suppose that at least one coordinate entry of the initial point $\mathbf{x}^0$ from among $x_1^0, \ldots, x_m^0$ is nonzero. Assume that exact line search is used. Show that convergence to $\mathbf{0}$ is impossible. More specifically, show that either the line search will be unbounded, or else $\|\mathbf{x}^{k+1}(1:m)\| > \|\mathbf{x}^k(1:m)\|$.

3. Consider minimizing the univariate quadratic function $f(x) = x^2$ using the descent algorithm discussed in lecture. Suppose an Armijo line search is used. Assume $x^k \neq 0$. Show that there exists a scalar $c < 1$ such that $|x^{k+1}| \leq c|x^k|$. Explicitly determine an upper bound on the value of c, which should depend on the two parameters β and σ of the Armijo line search. Note: your solution should assume that d^k is a descent direction but should not make any other assumptions about it—in particular, it is not necessarily the negative derivative.

   [Hint: This question requires a sequence of elementary but slightly tricky inequalities. Let w denote $\alpha^k d^k$ and let x denote x^k. What is the sign of w/x? First, manipulate the desired inequality $|x^{k+1}| \leq c|x^k|$ into the form $m_1 \leq w/x \leq m_2$, where m_1, m_2 are constants you determine. Next, write down both inequalities implied by Armijo and manipulate them until they are quadratic inequalities involving w/x. Factor these

inequalities to find n_1, n_2 such that $n_1 \leq w/x \leq n_2$. Finally, by comparing m_1 to n_1 and m_2 to n_2 complete the problem by writing c in terms of σ and β.]

4. The analysis presented in lecture of steepest descent with Armijo line search shows that the condition number of the Hessian at the root governs the convergence rate near the root. It also happens some times that this condition number governs the overall rate of convergence.

   Write a Matlab code for steepest descent with Armijo line search. The code should apply these algorithms to the problem of minimizing $f(\mathbf{x}) = \sum_{i=1}^{m} \exp(\mathbf{a}_i^T \mathbf{x} - b_i)$, where $\mathbf{a}_1^T, \ldots, \mathbf{a}_m^T$ are the rows of a given $m \times n$ matrix A and $\mathbf{b}$ is a given m-vector. (Note: you will have to work out Matlab expressions for both ∇f and $\nabla^2 f$.)

   Construct random matrices A and vectors $\mathbf{b}$ (use `randn`) and try your code out on problems of size $n = 10, 20, 30, 40$. Take $m = 3n$ in all cases and run multiple trials at each size. Use $\beta = 1/2$, $\sigma = 1/10$, $\mathbf{x}^0 = \mathbf{0}$. Use as a termination test a derivative norm less that 10^{-6}. Note that occasionally because of unlucky choice of random numbers you may hit an instance that runs for a very long time. If a run continues for more than a period of a minute or two, please stop it and try again.

   Plot the number of iterations against n, the problem size. Also plot the number of iterations against $\mathrm{cond}(H)$, the condition number of the Hessian at the computed minimizer. Both plots should be carried out with the `loglog` function for logarithmic axis spacing. Which one better predicts the running time?

   Hand in listings of your code, plots of how many iterations were required as a function of n or of $\mathrm{cond}(H)$, and your answer to the question in the previous paragraph.