Nonlinear Optimization

Spring 2013, Rutgers University

Prof. Eckstein

Homework 3: Unconstrained Optimization

1. *Linear convergence of gradient methods near "flat" local minima:* Consider the same function $f : \mathbb{R} \to \mathbb{R}$ given by $f(x) = x^4$ from the second problem of the previous homework. Consider attempting to minimize f by a Newton method with Armijo line search with $s = 1$.

   (a) Show that the stepsize α_k does not depend on x^k, so long as $x_k \neq 0$ (that is, the stepsize is always the same unless the algorithm lands right on the global minimum). You do not have to calculate the exact value of α_k.

   (b) Show that from any starting point $x_0 \neq 0$, the algorithm converges linearly to 0 with rate no better than 2/3. Why does this result not contradict the superlinear convergence theorem in the class notes?

2. *Modifying Newton methods to tolerate negative curvature.* In this problem, we will investigate ways of modifying Newton methods so they have more desirable global convergence properties in cases where they may encounter Hessians that are not positive definite. *Instructor's note:* these modifications are simplified and not as computationally efficient or robust as "professional grade" approaches, but are similar in spirit.

   (a) Describe what happens when we apply the `newtonArmijo` algorithm on the class website to the function $f_6(x_1, x_2) = \log(1+x_1^2+5x_2^2)$ (defined as `f6` in `funcdefs.m`), from the starting point $(2, 2)$ = `[2;2]`? Explain why.

   (b) Consider the following modification of Newton's method: at each iteration, factor the Hessian $H_k = \nabla^2 f(x^k)$ into $H_k = QDQ^\top$, where D is a diagonal matrix containing the eigenvalues of H_k, and Q is matrix whose columns are corresponding unit-length eigenvectors. Using some given parameter $\nu > 0$, construct a modifed diagonal matrix $\widetilde{D}$ as follows: if $d_{jj} \geq \nu$, then $\widetilde{d}_{jj} = d_{jj}$, but if $d_{jj} < \nu$, then $\widetilde{d}_{jj} = 1$. Then let $\widetilde{H}^k = Q\widetilde{D}Q^\top$, and compute the search direction via $d^k = -[\widetilde{H}^k]^{-1}\nabla f(x^k)$. Show that this procedure produces gradient-related search directions even if the H_k matrices encountered might not be positive definite.

   (c) Another alternative is as follows: if $|\lambda| < \nu$ for any eigenvalue λ of H_k, use $-\nabla f(x^k)$ as the search direction. Otherwise calculate the Newton search direction $d_{\mathrm{n}} = -[H_k]^{-1}\nabla f(x^k)$, and check whether $\langle \nabla f(x^k), d_{\mathrm{n}} \rangle \leq -\nu \|\nabla f(x^k)\|^2$. If so, set $d^k = d_{\mathrm{n}}$, and otherwise set $d^k = -\nabla f(x^k)$. Show that this method will also produce a gradient-related sequence, even when the H_k matrices encountered might not be positive definite.

(d) Implement each of the above methods in MATLAB in a similar style to the code distributed on the class website. Turn in a listing of the `.m`-file for each method. Also hand in printouts of the sequence of points generated by each method for the example of part (a) with the same starting point `[2;2]`. Use $\nu = 0.001$ in both cases, and the Armijo stepsize rule with parameters $s = 1$, $\sigma = 0.25$, and $\beta = 0.5$. Notes: the MATLAB statement `[Q,D] = eig(H)` will compute the factors of the matrix `H` in the manner described in part (b). Similarly, the statement `lambda = eig(H)` will place a one-dimensional array of `H`'s eigenvalues into `lambda`.