Homework 7

Math 652
Spring 2020

Due Monday, April 20

The following problem verifies some important properties of the BFGS method. Here, the definitions of H_k, s_k, and y_k are taken from Chapter 6 in Nocedal and Wright. See also the lecture of 4/8.

Problem 1 (2+5+3 points)**.**

1. *Suppose that B_k is positive definite. Show that the Quasi-Newton search direction*
$$p_k = -B_k \nabla f(x_k)$$
is a descent direction.

2. *Assume that the curvature condition $y_k^{\mathrm{t}} s_k > 0$ holds and that H_k is symmmetric positive definite. Prove that H_{k+1} is symmetric positive definite.*

3. *Show that $H_{k+1} y_k = s_k$.*

Last semester, I demonstrated that the steepest descent method from linear algebra converges slowly for ill-conditioned problems. To be precise, recall that if A is SPD, then the unique minimizer u of the functional

$$I[v] = \frac{1}{2} v^{\mathrm{t}} A v - f^{\mathrm{t}} v$$

solves the linear equation $Au = f$. One can compute the minimizer by steepest descent with an exact linesearch, i.e. a linesearch that yields the exact minimum of ϕ_k. Last semester, we saw that for the steepest descent method,

$$\|x_k - u\|_A \leq \left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1} \right)^k \|x_0 - u\|_A.$$

(Here, $\|v\|_A := \langle v, Av \rangle$ is the A-norm.) This estimate correctly suggests that convergence is slow when the condition number $\kappa_2(A)$ is large. *Note*: The result above appears as Theorem 3.3 in Nocedal and Wright.

A similar difficulty arises in the minimization of general (not necessarily quadratic) functions f. If the Hessian $D^2 f(x_*)$ is ill-conditioned at the minimum x_*, then steepest descents will converge slowly. This makes sense, since

according to Taylor's theorem,

$$f(x) = f(x_*) + \nabla f(x_*)^{\mathrm{t}}(x - x_*) + \frac{1}{2}(x - x_*)^{\mathrm{t}} D^2 f(x_*)(x - x_*) + O(\|x - x_*\|^3).$$

Therefore, minimizing f over a neighborhood of the minimizer x_* is nearly the same thing as minimizing a quadratic functional such as I above with $A = D^2 f(x_*)$. (Theorem 3.4 in Nocedal and Wright is an elaboration of this idea.)

Of course, our statement about Hessians and condition numbers is rather abstract. What it really means for a function f to be ill-conditioned is that the value of f is very sensitive to some small perturbations of the inputs but not sensitive to others. One simple example is the Rosenbrock function

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2.$$

This very ill-conditioned function is often used to test optimization methods. Its unique minimizer is $(1, 1)$.

Problem 2 (2 points)**.** *Plot of the graph of the Rosenbrock function for $x \in [-2, 2]$ and $y \in [-1, 3]$. Observe that the level sets are highly elongated. This is the signature of an ill-conditioned function.* Hint: *I suggest that you adapt the code at* `https://matplotlib.org/3.1.1/gallery/images_contours_and_fields/contour_demo.html` *to produce the plot. Make sure that you print some contours for small values of the function, say less than one, and also some contours for slightly larger values.*

We observe that any function can be made ill-conditioned by changing the units of the inputs. For example, let $f : \mathbb{R}^2 \to \mathbb{R}$. Suppose that the first variable is measured in seconds and the second in meters. If we have measurements taken in microseconds and kilometers, we have to convert them to meters and seconds before applying f. In effect, we must define a new function

$$f^D(x) = f(Dx), \text{ where } D = \begin{pmatrix} 10^{-6} & 0 \\ 0 & 10^3 \end{pmatrix}.$$

Of course, D is ill-conditioned ($\kappa(D) = 10^9$), so f^D will generally be ill-conditioned even if f was not. As a consequence, steepest descents may converge quickly for f but slowly for f^D.

We do not want the performance of an optimization method to depend on the choice of units. One way to ensure that it does not is to use only scale invariant methods. Let $f : \mathbb{R}^d \to \mathbb{R}$ be an objective function, and let $F \in \mathrm{GL}(d)$ be a linear change of variable. Define

$$f^F(x) = f(F(x)).$$

Let x_k be the sequence of iterates produced by a certain optimization method when applied to f starting from x_0. Let x_k^F be the sequence produced by the same method when applied to f^Fstarting from the point $F^{-1}x_0$. An optimization method is *scale invariant* if and only if

$$x_k^F = F^{-1}x_k$$

for all $F \in \mathrm{GL}(d)$ and $k \geq 0$. For a scale invariant method, the rate of convergence is indeed independent of the choice of units.

Problem 3 (1+1+3 points)**.**

1. *Show that*
$$\nabla f^F(z) = F^{\mathrm{t}} \nabla f(Fz).$$
Note: *The formula is backwards from the usual chain rule (the factor of F is on the left), since the gradient is always understood to be a column vector, whereas the derivative f' is always a row vector. It has to be this way, because you take ∇f to be a column vector when you do Newton's method.*

2. *Show that*
$$D^2 f^F(z) = F^{\mathrm{t}} D^2 f(Fz) F.$$
Hint: *Use that $D^2 f^F(z)w = \frac{d}{d\varepsilon}\big|_{\varepsilon=0} \nabla f^F(z + \varepsilon w)$ for all $w \in \mathbb{R}^d$.*

3. *Show that Newton's method is scale invariant.*

The BFGS method is also scale invariant, which is a major motivation for its particular rule for updating the approximations H_k to the inverse Hessians.

Problem 4 (5 points)**.** *Implement the steepest descent method using the backtracking line search, which is Algorithm 3.1 in Nocedal and Wright. Apply your steepest descent algorithm to the Rosenbrock function. Use the parameters $\bar{\alpha} = 1$, $c = 10^{-4}$, and $\rho = \frac{1}{2}$. Start with $x_0 = (-2, -1)$. Plot a sequence of iterates. Do you see roughly why convergence is so slow?*

You may choose any reasonable termination condition for your steepest descent algorithm. One standard choice is to terminate whenever the size of gradient shrinks below some tolerance, e.g. when $\|\nabla f(x_k)\|_2 \leq 10^{-10}$. Another is to terminate when too little progress is made in lowering the value of the objective, e.g. when $f(x_k) - f(x_{k+1}) \leq 10^{-10}$. Sophisticated implementations will use some combination of the two. If you choose the gradient based condition with a small tolerance, you will probably find that the steepest descent method simply cannot find any points meeting the condition. Thus, you may want to break your loop if no progress is made, no matter what.

Important Note: *Your code will have to be robust to overflow! The gradient of the Rosenbrock function is large at x_0. Therefore, the first value tried in your line search will be far from x_0. In fact, it will be so far from the origin that the value of the Rosebrock function will be too large for floating point. You might look into the function numpy.isfinite for detecting infinite values. When such a value arises, there is no cause for concern, you can simply multiply the step by ρ like you would do for any other value that does not satisfy the sufficient decrease condition.*

You may recall from last semester that Newton's method converges superlinearly. In fact, let x_* be a local minimizer of the function f, and assume that

the Hessian of f is positive definite at x_*. One can show that there exists $\rho > 0$ so that if $\|x_0 - x_*\| \leq \rho$, then

$$\|x_{k+1} - x_*\| \leq c\|x_k - x_*\|^2$$

for some constant $c > 0$. That is, Newton's method converges quadratically. (Compare the estimate for above for steepest descent, which only shows linear convergence.) BFGS also converges quadratically under conditions outlined in Chapter 6 of Nocedal and Wright.

Problem 5 (5 points)**.** *Use BFGS to find the minimum of the Rosenbrock function. Start with $x_0 = (-2, -1)$. You may use SciPy's built-in optimization function to do this. Try the following:*

```
from scipy.optimize import minimize
BFGS_output=minimize(rosen,
    np.array([-2,-1]),jac=rosen_der,tol=10**(-16),method='BFGS',
    options={'return_all': True}).
```

Does it look like BFGS converges quadratically or at least superlinearly for this problem? How does the approximate inverse Hessian generated by BFGS compare with the actual inverse Hessian at the minimizer?