Math 484: Nonlinear Programming[1] Mikhail Lavrov

Chapter 3, Lecture 2: Newton's Method in $\mathbb{R}^n$

April 17, 2019 *University of Illinois at Urbana-Champaign*

1 Solving systems of equations

Given a function $f : \mathbb{R} \to \mathbb{R}$, we find a point x satisfying $f(x) = 0$ by an iterative procedure that approximates f linearly: from a point x_k, we approximate

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

and find x_{k+1} by setting this approximation equal to 0.

When we have a function $g : \mathbb{R}^n \to \mathbb{R}$, we can try to find a point $\mathbf{x} \in \mathbb{R}^n$ satisfying $g(\mathbf{x}) = 0$ by the same procedure, but doing this is slightly awkward. We can approximate $g(\mathbf{x})$ at a point $\mathbf{x}^{(k)}$ by

$$g(\mathbf{x}) \approx g(\mathbf{x}^{(k)}) + \nabla g(\mathbf{x}^{(k)}) \cdot (\mathbf{x} - \mathbf{x}^{(k)})$$

but if we set this approximation equal to 0, we will find many solutions. It's not clear which one of them is the best one to pick for $\mathbf{x}^{(k+1)}$.

Instead, the natural generalization of Newton's method to n dimensions is for solving a system of equations:

$$\begin{cases} g_1(\mathbf{x}) = 0, \\ g_2(\mathbf{x}) = 0, \\ \vdots \\ g_n(\mathbf{x}) = 0 \end{cases} \iff \mathbf{g}(\mathbf{x}) = \mathbf{0}$$

where $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^n$ is a function with components $g_1, g_2, \ldots, g_n$.

In this case, a linear approximation of $\mathbf{g}(\mathbf{x})$ at $\mathbf{x}^{(k)}$ is given by combining the linear approximations of $g_1, g_2, \ldots, g_n$:

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_n(\mathbf{x}) \end{bmatrix} \approx \begin{bmatrix} g_1(\mathbf{x}^{(k)}) \\ g_2(\mathbf{x}^{(k)}) \\ \vdots \\ g_n(\mathbf{x}^{(k)}) \end{bmatrix} + \begin{bmatrix} \nabla g_1(\mathbf{x}^{(k)})^\mathsf{T} \\ \nabla g_2(\mathbf{x}^{(k)})^\mathsf{T} \\ \vdots \\ \nabla g_n(\mathbf{x}^{(k)})^\mathsf{T} \end{bmatrix} (\mathbf{x} - \mathbf{x}^{(k)}).$$

This has the form $\mathbf{g}(\mathbf{x}) \approx \mathbf{b} + A\mathbf{x}$, where $\mathbf{b}$ is a vector of the values of $\mathbf{g}$ at $\mathbf{x}^{(k)}$ and A is a matrix of the derivatives of $\mathbf{g}$ at $\mathbf{x}^{(k)}$. The matrix, which we wrote above in terms of its rows (its i^{th} row

[1] This document comes from the Math 484 course webpage: https://faculty.math.illinois.edu/~mlavrov/courses/484-spring-2019.html

is the gradient of g_i at $\mathbf{x}^{(k)}$) can also be written as

$$\begin{bmatrix} \nabla g_1(\mathbf{x}^{(k)})^\mathsf{T} \\ \nabla g_2(\mathbf{x}^{(k)})^\mathsf{T} \\ \vdots \\ \nabla g_n(\mathbf{x}^{(k)})^\mathsf{T} \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1}(\mathbf{x}^{(k)}) & \frac{\partial g_1}{\partial x_2}(\mathbf{x}^{(k)}) & \cdots & \frac{\partial g_1}{\partial x_n}(\mathbf{x}^{(k)}) \\ \frac{\partial g_2}{\partial x_1}(\mathbf{x}^{(k)}) & \frac{\partial g_2}{\partial x_2}(\mathbf{x}^{(k)}) & \cdots & \frac{\partial g_2}{\partial x_n}(\mathbf{x}^{(k)}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1}(\mathbf{x}^{(k)}) & \frac{\partial g_n}{\partial x_2}(\mathbf{x}^{(k)}) & \cdots & \frac{\partial g_n}{\partial x_n}(\mathbf{x}^{(k)}) \end{bmatrix}.$$

The (i, j) entry of this matrix is the derivative of g_i with respect to x_j. We write $\nabla \mathbf{g}(\mathbf{x}^{(k)})$ for this matrix, and it is called the Jacobian matrix of $\mathbf{g}$ at $\mathbf{x}^{(k)}$.

Altogether, the n-dimensional Newton's method for solving $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ starting with a guess $\mathbf{x}^{(0)}$ is to compute a sequence $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ by letting $\mathbf{x}^{(k+1)}$ be the solution to

$$\mathbf{g}(\mathbf{x}^{(k)}) + \nabla \mathbf{g}(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \mathbf{0}.$$

In theory, we can solve for $\mathbf{x}^{(k+1)}$ to get the nice-looking formula

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \nabla \mathbf{g}(\mathbf{x}^{(k)})^{-1} \mathbf{g}(\mathbf{x}^{(k)})$$

which resembles the one-dimensional Newton's method. But when n is large, it's usually not efficient to solve the system of equations by computing the inverse.

It is not really required to have the number of variables be the same as the number of equations, it's just the nicest case. When we have fewer equations than variables, we'll end up making arbitrary choices at each step, because we're likely to have infinitely many solutions for $\mathbf{x}^{(k+1)}$.

It might make sense to use least-squares optimization to pick $\mathbf{x}^{(k+1)}$ as close to $\mathbf{x}^{(k)}$ as possible, but that's just me making things up. This would give us the system of equations

$$(\nabla \mathbf{g}(\mathbf{x}^{(k)}))^\mathsf{T} \nabla \mathbf{g}(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = (\nabla \mathbf{g}(\mathbf{x}^{(k)}))^\mathsf{T} \mathbf{g}(\mathbf{x}^{(k)}).$$

Once nice thing about doing this would be that we always get an answer, even in cases where the original system has no solution. (This corresponds to a case where the linear approximation never passes through $\mathbf{0}$, and it might make sense to go to a point where it's as close to $\mathbf{0}$ as possible.)

However, note that if there is a point $\mathbf{x}$ with $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ and $\nabla \mathbf{g}(\mathbf{x})$ is invertible, then $\nabla \mathbf{g}(\mathbf{x}^{(k)})$ will also be invertible for $\mathbf{x}^{(k)}$ close enough to $\mathbf{x}$ (assuming some continuity).

1.1 Example

For example, if we have $\mathbf{g}(x, y) = (x^2 + y^2 - 4, xy - 1)$ then

$$\nabla \mathbf{g}(x, y) = \begin{bmatrix} 2x & 2y \\ y & x \end{bmatrix}.$$

If we want to iterate from the point $(1, 0)$, then we have

$$\mathbf{g}(1, 0) = \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \qquad \nabla \mathbf{g}(1, 0) = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

and to find the next point (x_1, y_1) we solve the system of equations

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 - 1 \\ y_1 - 0 \end{bmatrix} = - \begin{bmatrix} -3 \\ -1 \end{bmatrix},$$

or $2(x_1 - 1) = 3$ and $y_1 = 1$, getting the point $(\frac{5}{2}, 1)$.

2 Minimizing functions of n variables

Another reason to consider systems of n equations in n variables is that this is the case we naturally end up in when we want to adapt the method to minimizing a function $f : \mathbb{R}^n \to \mathbb{R}$.

As we saw last time, really all we can do with Newton's method is find a critical point of f: solve the equation $\nabla f(\mathbf{x}) = \mathbf{0}$. But the good news is that this is exactly the situation in which we are happiest applying the n-dimensional Newton's method.

If we are finding places where $\mathbf{g} = \nabla f$ is zero, then $\nabla \mathbf{g} = Hf$ is the Hessian matrix of f. Therefore finding a critical point gives us the following iterative step: we get $\mathbf{x}^{(k+1)}$ from $\mathbf{x}^{(k)}$ by solving

$$\nabla f(\mathbf{x}^{(k)}) + Hf(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \mathbf{0}.$$

This can also be phrased as a quadratic approximation: we approximate

$$f(\mathbf{x}) \approx f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)}) \cdot (x - \mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(k)})^{\mathsf{T}} Hf(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)})$$

and find the critical point of this approximation.

In the next lecture, we will discuss the method of steepest descent: our first attempt to do something which actually *tries* to minimize (rather than maximize) f.

3 Guarantees on Newton's method

There are many things that can go wrong with Newton's method. We may end up at a point where we cannot continue the iteration, because the equation

$$\mathbf{g}(\mathbf{x}^{(k)}) + \nabla \mathbf{g}(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \mathbf{0}$$

has no solution. (Even for $g : \mathbb{R} \to \mathbb{R}$, we might end up with this problem if $g'(x_k) = 0$.) It is rare, but possible for a multi-point cycle to appear: e.g. from $\mathbf{x}^{(1)}$ we go to $\mathbf{x}^{(2)}$, from $\mathbf{x}^{(2)}$ we go to $\mathbf{x}^{(3)}$, but from $\mathbf{x}^{(3)}$ we happen to end up back at $\mathbf{x}^{(1)}$. We have seen examples where convergence is drastically slowed, and we have seen examples where Newton's method actually runs away from the root. Even when we do get an answer, it might not be the answer wanted.

We can say some things about problems for which we're guaranteed to converge, and to converge quickly, from most initial points. But one thing that fundamentally does not go wrong is correctness.

Theorem 3.1. *Suppose that we are using Newton's method to solve* $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ *for some* $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^n$ *(with continuous first partial derivatives). If the sequence of points* $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ *that we get converges to some limit* $\mathbf{x}^*$*, then* $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$.

Proof. The idea behind this proof is a simple but important one that shows up in a lot of places, and variations on this proof can guarantee correctness for a lot of our iterative methods.

At all times, the equation

$$\mathbf{g}(\mathbf{x}^{(k)}) + \nabla\mathbf{g}(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \mathbf{0}$$

holds, because that's how we compute $\mathbf{x}^{(k+1)}$ from $\mathbf{x}^{(k)}$ to begin with. The left-hand side is a continuous function of $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$...

...so we can take the limit as $k \to \infty$ on both sides and still get a true statement. That limit says that

$$\mathbf{g}(\mathbf{x}^*) + \nabla\mathbf{g}(\mathbf{x}^*)(\mathbf{x}^* - \mathbf{x}^*) = \mathbf{0}.$$

The difference $\mathbf{x}^* - \mathbf{x}^*$ simplifies to $\mathbf{0}$; we don't know very much about $\nabla\mathbf{g}(\mathbf{x}^*)$, but whatever it is, multiplying it by $\mathbf{0}$ will give us $\mathbf{0}$ again. So this equation simplifies to $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$, which is exactly what we wanted. □

Another instance of this argument is the rule that if we start at a value x and keep applying a function h, the sequence

$$x, h(x), h(h(x)), h(h(h(x))), \ldots$$

will either not converge at all, or converge to a limit x^* which is a fixed point of h: it satisfies $h(x^*) = x^*$. (What we're doing is not too different.)

Maybe some of you have tried this experiment when you were bored in school: take a scientific calculator, and keep pressing the COS button over and over again. Eventually you end up with a value close to 0.739085 which doesn't change when you press that button. This value is just the (not terribly interesting) solution to the equation $\cos x = x$.