ADDITIONAL NOTES FOR WEEK 1

THOMAS YU

1. Descent Directions Methods

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a C^1 function that you want to minimize over either the whole $\mathbb{R}^n$ or some specific subset U, i.e. you want to solve

$$\min_{x \in U} f(x).$$

Let's say $U = \prod_{i=1}^n [a_i, b_i]$. (Being a compact set in $\mathbb{R}^n$, a minimizer must exist.) With a powerful computer, how about simply sampling f over a fine grid on U and see which sample point gives you the lowest value of f? If you want to determine a minimizer accurately, you will need a fine grid with a reasonably large number of samples, say m, in each of the n dimensions. If m and n are just moderately big, you will have to deal with m^n samples, which is astronomically big. You are cursed by the dimensionality in such a brute-force approach.

To break the curse of dimensionality, we exploit the smoothness of f and we consider *iterative methods*, i.e. we start at an initial guess $\mathbf{x}_0$, then find a way to move to the next point with a lower value of f, and we iterate such a process.

To understand how to move to the next point, the first thing we should try to picture is how the level set of f at $\mathbf{x}_0$,

$$f^{-1}(f(\mathbf{x_0})) = \{\mathbf{x} : f(\mathbf{x}) = f(\mathbf{x_0})\}, \tag{1.1}$$

looks like near $\mathbf{x_0}$. This is the set in which the function neither increases nor decreases. Having a good picture of how this set looks like will help us understand in which directions the function increases or decreases.

The bad news: $f(\mathbf{x})$ can be an awfully nonlinear function, and the level set above can at best look like a complicated curved object. How are you supposed to say something useful about this level set?

The good news: Since $f(x)$ is assumed to be differentiable, near $\mathbf{x}_0$, $f(\mathbf{x})$ is well-approximated by an affine (i.e. linear + a shift) function, i.e.

$$f(\mathbf{x}) \approx f(\mathbf{x_0}) + \left[\frac{\partial f}{\partial x_1}(\mathbf{x_0}), \dots \frac{\partial f}{\partial x_1}(\mathbf{x_0})\right](\mathbf{x} - \mathbf{x_0}) =: L(\mathbf{x}). \tag{1.2}$$

(Note: $L(\mathbf{x})$ is the one and only one affine function with the approximation property that $L(\mathbf{x})$ approaches $f(\mathbf{x})$ *faster* than $\mathbf{x}$ approaches $\mathbf{x}_0$, i.e. $\|f(\mathbf{x}) - L(\mathbf{x})\| = o(\|\mathbf{x} - \mathbf{x}_0\|)$, or $\lim_{\mathbf{x} \to \mathbf{x}_0} \|f(\mathbf{x}) - L(\mathbf{x})\|/\|\mathbf{x} - \mathbf{x}_0\| = 0$.)

Date: April 6, 2020.

Here is the key heuristics: the part of the level set of $f(\mathbf{x})$ near $\mathbf{x_0}$ should look like the level set of the affine function $L(\mathbf{x})$ on the right-hand side. (Globally, the level set of $f(x)$ can be a highly curved and complicated, and may have self-intersections, etc.. We conveniently avoid the difficulty by thinking *locally*.)

But the level set of an affine function is easy, linear algebra tells you all about it. Notice that

$$L^{-1}(f(\mathbf{x_0})) = \mathbf{x_0} + \text{null}(\left[\frac{\partial f}{\partial x_1}(\mathbf{x_0}), \dots \frac{\partial f}{\partial x_1}(\mathbf{x_0})\right]) \tag{1.3}$$

$$= \{\mathbf{x_0} + \mathbf{d} : \nabla f(\mathbf{x_0})^T \mathbf{d} = 0\}. \tag{1.4}$$

If the vector $\nabla f(\mathbf{x_0})^T = \left[\frac{\partial f}{\partial x_1}(\mathbf{x_0}), \dots \frac{\partial f}{\partial x_1}(\mathbf{x_0})\right]$ is non-zero (i.e. you are not at a critical point yet), then the null space above has dimension $n-1$. In this case, $L^{-1}(f(\mathbf{x_0}))$ is called a hyperplane in $\mathbb{R}^n$. (If $n = 3$, it is like a plane cutting through our 3-dimensional space. If $n = 2$, it is a line in $\mathbb{R}^2$.)

Rephrasing the heuristics more mathematically: Since L approximates f near $\mathbf{x_0}$, we expect that the level surface $f^{-1}(f(\mathbf{x_0}))$ should be approximated by the hyperplane $L^{-1}(f(\mathbf{x_0}))$ also. We are then led to the conjecture that the hyperplane must be the ***tangent plane*** of the level surface at $\mathbf{x_0}$.

Proving this conjecture is essentially what the implicit function theorem is about. The IFT usually pertains to a function mapping from $\mathbb{R}^{n+m}$ to $\mathbb{R}^m$: here the m is 1 in our optimization problem.

I would not bother you with the details of the IFT. What is more important for nonlinear optimization is to see that the hyperplane $L^{-1}(f(\mathbf{x_0}))$ divides the whole space $\mathbb{R}^n$ into two halves:

$$\{\mathbf{x_0} + \mathbf{d} : \nabla f(\mathbf{x_0})^T \mathbf{d} < 0\}$$

and

$$\{\mathbf{x_0} + \mathbf{d} : \nabla f(\mathbf{x_0})^T \mathbf{d} > 0\}.$$

If we take a vector $\mathbf{d} = [d_1, \dots, d_n]^T$ (think of it as emanating from $\mathbf{x_0}$) so that

$$\nabla f(\mathbf{x_0})^T \mathbf{d} < 0,$$

then, starting at $\mathbf{x_0}$, if you travel along this direction, you must see that your objective function $f(x)$ decrease for at least a while, it is because

$$\frac{d}{dt} f(\mathbf{x_0} + t\mathbf{d})\Big|_{t=0} = \nabla f(\mathbf{x_0})^T \mathbf{d},$$

which is negative by your choice of $\mathbf{d}$. Any such direction is called a **descent direction of f at $\mathbf{x}_0$**. See also Definition 4.1 of Beck.

To conclude: if you move along the level set, the function neither increases nor decreases. If you move in a direction that points to one side of the hyperplane, the function decreases for at least a while; in the opposite direction, the function increases.

This is what calculus can tell you. Be reminded that it is a local theory, it does not tell you whether this beautiful picture will sustain when you move too far away from $\mathbf{x_0}$. However, when you move to a different point, say $\mathbf{x_1}$, you can "redraw the picture" by centering yourself at $\mathbf{x_1}$ and "tilting"

the hyperplane by recomputing the gradient vector at $\mathbf{x_1}$, and so on. This way you get a new local picture around $\mathbf{x_1}$.

And then you can iterate and automate this process on computers. Any such method is called "descent direction method". The "gradient descent method" corresponds to the choice

$$\mathbf{d} = -\nabla f(\mathbf{x}).$$

Note that, unless $\mathbf{x}$ is already a critical point of $\mathbf{x}$ (i.e. $\nabla f(\mathbf{x}) = 0$), this is always a descent direction of f at $\mathbf{x}$, as

$$-\nabla f(x)^T \nabla f(x) = -\|\nabla f(x)\|^2 < 0.$$

2. A note on the multivariate Taylor expansion

The multivariate Taylor expansion comes from the 1-D Taylor expansion, the idea is simple: restrict $f : \mathbb{R}^n \to \mathbb{R}$ to a line going through x_0, parametrized as $t \mapsto \mathbf{x_0} + t\mathbf{h}$, and define

$$\phi(t) := f(\mathbf{x_0} + t\mathbf{h}).$$

Take the (1-D) Taylor expansion of $\phi(t)$ at $t = 0$, i.e.

$$\phi(t) \approx \phi(0) + \phi'(0)t + \frac{1}{2!}\phi''(0)t^2 + \cdots .$$

When you take this 1-D Taylor polynomial your mindset is that t is variable and $\mathbf{h}$ is set in stone. But *after* you do that, let $t = 1$ be set in stone and think of $\mathbf{h} = \mathbf{x} - \mathbf{x_0}$ as variable. As such, the above 1-D Taylor expansion becomes

$$f(\mathbf{x}) = f(\mathbf{x_0} + \mathbf{h}) \approx f(\mathbf{x_0}) + \frac{d}{dt} f(\mathbf{x_0} + t\mathbf{h})|_{t=0} + \frac{1}{2!}\frac{d^2}{dt^2} f(\mathbf{x_0} + t\mathbf{h})|_{t=0} + \cdots .$$

An application of the chain rule (with some details omitted) shows that

$$\frac{1}{k!}\frac{d^k}{dt^k} f(\mathbf{x_0} + t\mathbf{h})|_{t=0} = \frac{1}{k!} \sum_{i_1=1}^{n} \cdots \sum_{i_k=1}^{n} \frac{\partial^k f(\mathbf{x_0})}{\partial x_{i_1} \cdots \partial x_{i_k}} h_{i_1} \cdots h_{i_k}. \tag{2.1}$$

(The right-hand side may seem like a hideous summation, with n^k terms; but the left-hand side carries the clear geometric interpretation of the k-th order directional derivative of f at the point $\mathbf{x_0}$ in the direction $\mathbf{h}$.)

(Using the fact that the order of taking partial derivatives does not matter, and that multiplication is commutative, the right-hand of (2.1) can be 'compressed' into the following smaller sum:

$$\sum_{|\nu|=k} \frac{1}{\nu!} D^\nu f(\mathbf{x_0}) \mathbf{h}^\nu. \tag{2.2}$$

In above, I use the so-called "multi-index notations": $\nu = (\nu_1, \ldots, \nu_n)$, $|\nu| = \nu_1 + \cdots + \nu_n$, $\nu! = \nu_1! \cdots \nu_n!$, $\mathbf{h}^\nu = h_1^{\nu_1} \cdots h_n^{\nu_n}$, and I let you guess what $D^\nu f$ is supposed to mean. I won't need (2.2), but if you care then here is a good exercise: (i) prove that (2.2) and (2.1) are the same. (ii) How many terms are there in the summation of (2.2)?)

(Both the univariate and multivariate Taylor's theorem tell you also a way to express the remainder when you truncate the Taylor expansion, I will need that later in this course, but not now.)

For the rest of this note, I only need (2.1) for $k = 1$ and $k = 2$. You have seen how we write the $k = 1$ expression in matrix notation:

$$\sum_{i=1}^{n} \frac{\partial f(\mathbf{x_0})}{\partial x_i} h_i = \left[\frac{\partial f}{\partial x_1}(\mathbf{x_0}), \dots \frac{\partial f}{\partial x_1}(\mathbf{x_0}) \right] \mathbf{h}.$$

And now for $k = 2$ notice that

$$\frac{1}{2} \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \frac{\partial^2 f(\mathbf{x_0})}{\partial x_{i_1} \partial x_{i_2}} h_{i_1} h_{i_2} = \frac{1}{2} \mathbf{h}^T H_f(\mathbf{x_0}) \mathbf{h},$$

where $H_f(x_0)$ is the so-called Hessian matrix of f at $\mathbf{x_0}$, defined by

$$(H_f(\mathbf{x_0}))_{i,j} = \frac{\partial^2 f(\mathbf{x_0})}{\partial x_i \partial x_j}, \;\; 1 \le i, j \le n.$$

Note: in linear algebra an $n \times n$ matrix is usually thought of as a linear map from $\mathbb{R}^n$ to $\mathbb{R}^n$, but here the Hessian matrix represents the quadratic part of a function from $\mathbb{R}^n$ to $\mathbb{R}^1$, which is nonlinear! [1]

3. Newton's Method: the basic idea

After you decide on a decent direction $\mathbf{d}$, how far should you go down the line? Ideally, we would like to find a value α that solves

$$\min_{\alpha \ge 0} f(\mathbf{x_0} + \alpha \mathbf{d}). \tag{3.1}$$

Albeit one-dimensional, this line search problem is still a nonlinear minimization problem and may have multiple local minimizers. Even if it has a unique local (and also global) minimizer, it may be too costly to locate the minimizer accurately. Typically, one is content with a choice of α that is not too small and decreases f by a fair amount. Section 7.2.3 discusses what 'fair amount' may mean in practice. The coverage there, however, is rather coarse. For a more in-depth study on line search methods, consider reading Chapter 3 of Nocedal and Wright's *Numerical Optimization*.

Regardless, solving the line search problem (3.1) typically requires some ad hoc choices of parameters.

Here is a different idea, called the Newton's method, that will free you from the sort of ad hoc things typically done in a line search method: if your $\mathbf{x_0}$ is close enough to a local minimizer $\mathbf{x}^*$, assumed to be such that $H_f(\mathbf{x}^*)$ is positive definite, then by C^2 continuity, the nearby Hessian must also be positive definite. Therefore, the second order Taylor expansion

$$f(\mathbf{x_0}) + \nabla f(\mathbf{x_0})^T (\mathbf{x} - \mathbf{x_0}) + \frac{1}{2} (\mathbf{x} - \mathbf{x_0})^T H_f(\mathbf{x_0}) (\mathbf{x} - \mathbf{x_0}),$$

[1] The best way to think of the Hessian matrix is to view it as a symmetric bilinear map:

$$(u, v) \in \mathbb{R}^n \times \mathbb{R}^n \mapsto \sum_i \sum_j (H_f(\mathbf{x}_0))_{i,j} u_i u_j = u^T H_f(\mathbf{x}_0) v.$$

Similarly, the higher derivative 'tensor', namely, the k-dimensional array $[\frac{\partial^k f(\mathbf{x_0})}{\partial x_{i_1} \cdots \partial x_{i_k}}]_{i_1, \dots, i_k}$, is best viewed as a symmetric k-linear map. (But there are a couple more observations one need to make before you may feel comfortable about this view.)

being a quadratic polynomial with a positive definite Hessian, must have a unique global minimizer given by

$$-H_f(\mathbf{x_0})^{-1}\nabla f(\mathbf{x_0}).$$

(You will be asked to explore it in details in the HW assignment.) Once you understand this, call this *exact* minimizer of the *approximating* quadratic polynomial $\mathbf{x_1}$, and iterate. This is the Newton's method for nonlinear optimization.

The price you have to pay is big, as you need to solve an $n \times n$ linear system in every iteration, which requires $O(n^2)$ storage and $O(n^3)$ time if solved using Gaussian elimination. For large problem, it usually does not worth it, or is simply infeasible. This is why **quasi-Newton methods** were invented. They are widely used in practice; such methods are not discussed in Beck but are discussed in details in Norcedal and Wright.

Thomas Yu, Department of Mathematics, Drexel University, 3141 Chestnut Street, 206 Korman Center, Philadelphia, PA 19104, U.S.A.

E-mail address: yut@drexel.edu