2023-03-31

1 Optimization essentials

Last time, we mostly focused on solving systems of nonlinear equations. Today, we consider optimization problems. The two perspectives are very complimentary, but we will delve into that further after break.

Recall from your calculus classes the second-order Taylor expansion. If $\phi : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^2$ (i.e. if it is at least twice continuously differentiable) then we have the expansion

$$\phi(x+u) = \phi(x) + \phi'(x)u + \frac{1}{2}u^T H_\phi(x)u + o(\|u\|^2)$$

where $\phi'(x) \in \mathbb{R}^{1\times n}$ is the derivative of ϕ and H_ϕ is the *Hessian matrix* consisting of second derivatives:

$$(H_\phi)_{ij} = \frac{\partial \phi}{\partial x_i \partial x_j}.$$

The *gradient* $\nabla\phi(x) = \phi'(x)^T$ is a column vector (rather than a row vector).

Let's give an illustration comparing a function $\phi_{\mathrm{ex}} : \mathbb{R}^2 \to \mathbb{R}$ to the first and second-order Taylor expansions about some point x_0. To do this, we will need the gradient and the Hessian

$$\begin{aligned}
\phi(x) &= -\cos(x_1+x_2) + \sin(x_2)^2 \\
\nabla\phi(x) &= \begin{bmatrix} \sin(x_1+x_2) \\ \sin(x_1+x_2) + \sin(2x_2) \end{bmatrix} \\
H_\phi &= \begin{bmatrix} \cos(x_1+x_2) & \cos(x_1+x_2) \\ \cos(x_1+x_2) & \cos(x_1+x_2) + 2\cos(2x_2) \end{bmatrix}
\end{aligned}$$

We compare ϕ to first-order and second-order Taylor expansions in Figures 1-2.

If $\nabla\phi(x) \neq 0$ then $\nabla\phi(x)$ and $-\nabla\phi(x)$ are the directions of steepest ascent and descent, respectively (we will have more to say on this point presently). If $\nabla\phi(x) = 0$, then we say x is a *stationary point* or *critical point*. The first derivative test says that if x minimizes ϕ (and ϕ is differentiable) then the gradient of x must be zero; otherwise, there is a "downhill" direction, and a point near x achieves a smaller function value.

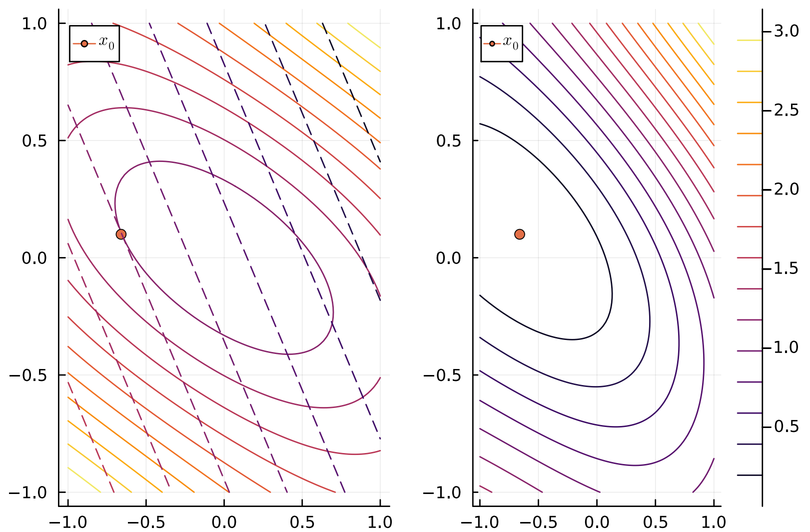

Figure 1: Contour plot of ϕ vs first-order Taylor approximation $\hat{\phi}$ (left) and plot of $\phi - \hat{\phi}$ (right).

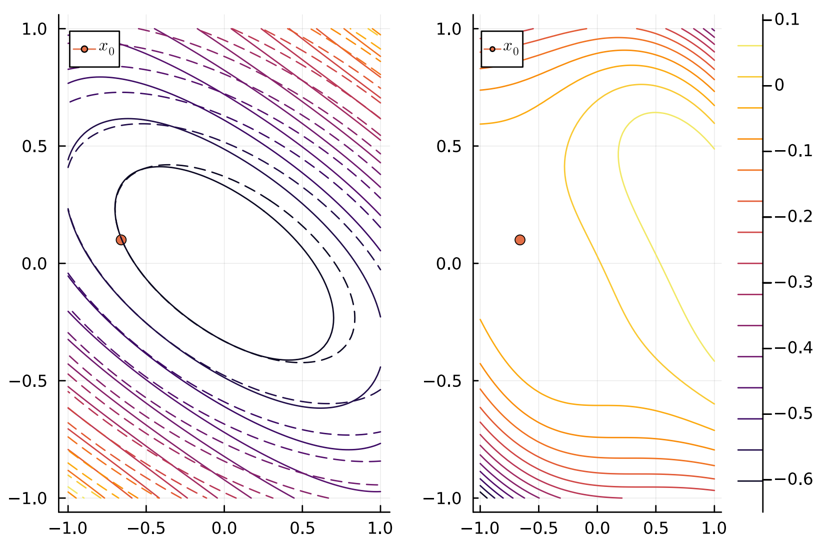

Figure 2: Contour plot of ϕ vs second-order Taylor approximation $\hat{\phi}$ (left) and plot of $\phi - \hat{\phi}$ (right).

A stationary point does not need to be a local minimizer; it might also be a maximizer, or a saddle point. The *second* derivative test says that for a critical point x to be a (local) minimizer, the Hessian $H_\phi(x)$ must be at least positive semi-definite. If x is a stationary point and H_ϕ is strictly positive definite, then x must be a local minimizer; in this case, we call x a *strong* local minimizer.

One approach to the problem of minimizing ϕ is to run Newton iteration on the critical point equation $\nabla\phi(x) = 0$. The Jacobian of the function $\nabla\phi(x)$ is simply the Hessian matrix, so Newton's iteration for finding the critical point is just

$$x_{k+1} = x_k - H_\phi(x_k)^{-1}\nabla\phi(x_k).$$

We can derive this in the same way that we derived Newton's iteration for other nonlinear equations; or we can derive it from finding the critical point of a quadratic approximation to ϕ:

$$\hat{\phi}(x_k + p_k) = \phi(x_k) + \phi'(x_k)p_k + \frac{1}{2}p_k^T H_\phi(x_k)p_k.$$

The critical point occurs for $p_k = -H_\phi(x_k)^{-2}\nabla\phi(x_k)$; but this critical point is a strong local minimum iff $H_\phi(x_k)$ is positive definite.

```
function plot_convergence(ϕ, xs, resids)
    xx = range(-1.0, 1.0, length=100)
    p1 = plot(xx, xx, ϕ, st=:contour)
    plot!([x[1] for x in xs], [x[2] for x in xs], marker=true, label=false)
    p2 = plot(resids[resids .> 0], yscale=:log10, legend=false)
    plot(p1, p2, layout=(1,2))
end

let
    x = [0.0; 0.5]

    # Newton loop -- plot where we're going
    resids = []
    xs = [copy(x)]

    for k = 1:5
```

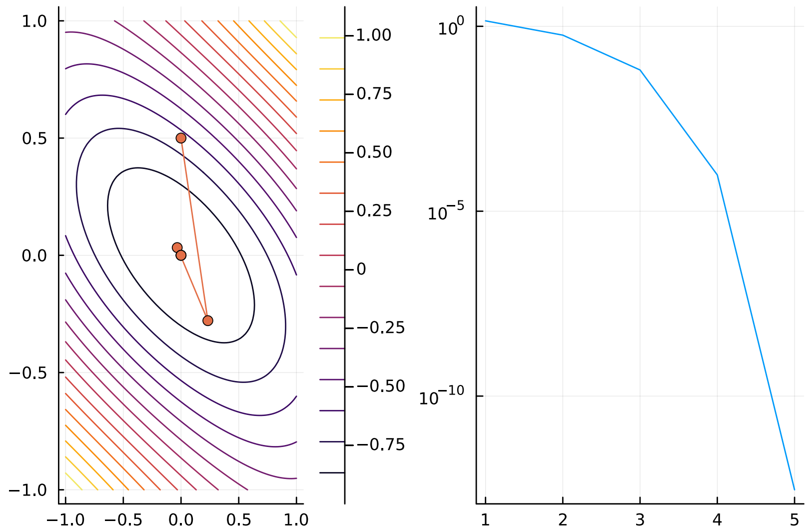

Figure 3: Convergence of Newton iteration on example ϕ.

```
        # Compute the gradient and record the norm
        g = ∇ϕex(x)
        push!(resids, norm(g))

        # Take a Newton step and record the point
        x -= Hϕex(x)\g
        push!(xs, copy(x))

    end

    # Plot the function and the iterates
    plot_convergence((x,y)->ϕex([x; y]), xs, resids)
end
```

There are a couple reasons we might want to not just say "run Newton to find a critical point" and call it a day. Three key ones are:

- Computing Hessians can be a bit of a pain.

- We can take advantage of the fact that this is *not* a general system of nonlinear equations in devising and analyzing methods.
- If we only seek to solve the critical point equation, we might end up finding a maximizer or saddle point as easily as a minimizer.

For this reason, we will discuss a different class of iterations, the (scaled) gradient descent methods and their relatives.

1.1 Questions

For all these questions, assume ϕ is twice continuously differentiable and that the Hessian H_ϕ is Lipschitz with constant M.

1. Change the starting point in the iteration above (e.g. to the point $(0, 0.6)$). What changes?
2. Explain why $\phi(x+u) = \phi(x) + \nabla\phi(x)^T u + \frac{1}{2} u^T H_\phi(x + \xi u) u$ for some $\xi \in [0, 1]$.
3. Argue that $|\phi(x) + \nabla\phi(x)^T u + \frac{1}{2} u^T H_\phi(x) u - \phi(x+u)| \leq \frac{M}{2} \|u\|^3$.
4. Let p be a Newton update starting from x, and let $w = \phi(x) - \phi(x+p)$ be the improvement in the function value during that Newton step. Show that $\left| w - \frac{1}{2} p^T H_\phi(x) p \right| \leq \frac{M}{2} \|p\|^3$.

2 Gradient descent

One of the simplest optimization methods is the *steepest descent* or *gradient descent* method

$$x_{k+1} = x_k + \alpha_k p_k$$

where α_k is a *step size* and $p_k = -\nabla\phi(x_k)$.

Let's try an example using the test function above. Play with α. What happens to the convergence?

```
let
    x = [0.0; 0.5]
    α=0.1

    # Gradient descent loop -- plot where we're going
    resids = []
```

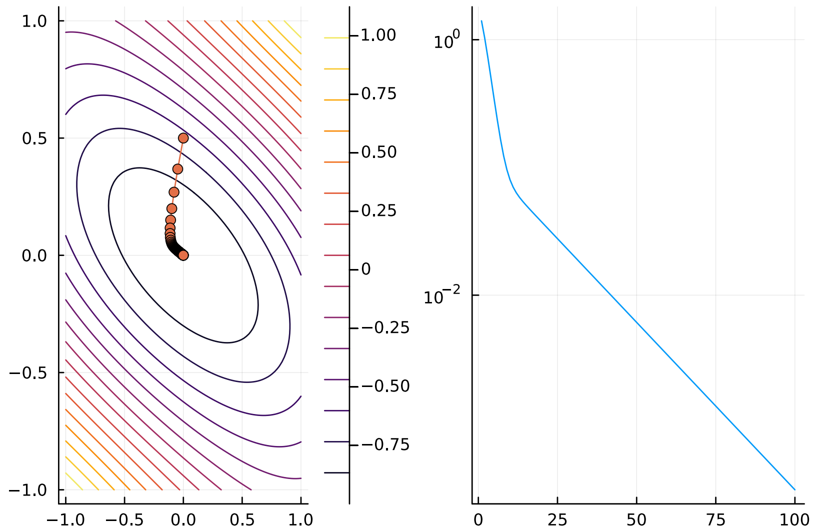

Figure 4: Convergence of gradient descent on example ϕ.

```
    xs = [copy(x)]

    for k = 1:100

        # Compute the gradient and record the norm
        g = ∇ϕex(x)
        push!(resids, norm(g))

        # Take a gradient descent step and record the point
        x -= α*g
        push!(xs, copy(x))

    end

    # Plot the function and the iterates
    plot_convergence((x,y)->ϕex([x; y]), xs, resids)
end
```

To understand the convergence of this method, consider gradient descent with a fixed step size α for the quadratic model problem

$$\phi(x) = \frac{1}{2}x^T Ax + b^T x + c$$

where A is symmetric positive definite. We have computed the gradient for a quadratic before:

$$\nabla\phi(x) = Ax + b,$$

which gives us the iteration equation

$$x_{k+1} = x_k - \alpha(Ax_k + b).$$

Subtracting the fixed point equation

$$x_* = x_* - \alpha(Ax_* + b)$$

yields the error iteration

$$e_{k+1} = (I - \alpha A)e_k.$$

If $\{\lambda_j\}$ are the eigenvalues of A, then the eigenvalues of $I-\alpha A$ are $\{1-\alpha\lambda_j\}$. The spectral radius of the iteration matrix is thus

$$\min\{|1 - \alpha\lambda_j|\}_j = \min\left(|1 - \alpha\lambda_{\min}|, |1 - \alpha\lambda_{\max}|\right).$$

The iteration converges provided $\alpha < 1/\lambda_{\max}$, and the optimal α is

$$\alpha_* = \frac{2}{\lambda_{\min} + \lambda_{\max}},$$

which leads to the spectral radius

$$1 - \frac{2\lambda_{\min}}{\lambda_{\min} + \lambda_{\max}} = 1 - \frac{2}{1 + \kappa(A)}$$

where $\kappa(A) = \lambda_{\max}/\lambda_{\min}$ is the condition number for the (symmetric positive definite) matrix A. If A is ill-conditioned, then, we are forced to take very small steps to guarantee convergence, and convergence may be heart breakingly slow. We will get to the minimum in the long run — but, then again, in the long run we all die.

Our example problem is not quadratic, of course. But close to the minimum at $(0, 0)$, it is close enough to quadratic that we get a good picture of the convergence from the quadratic approximation.

If we take a somewhat less well-conditioned problem, we get a significantly slower optimal rate of convergence (Figure 5).

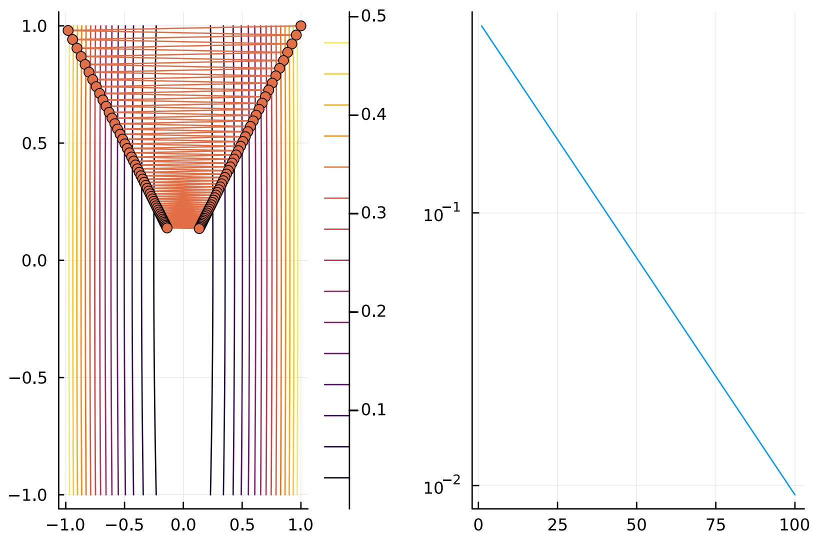

Figure 5: Convergence of gradient descent on an ill-conditioned problem.

```
let
    A = [1.0 0.0; 0.0 1e-2]
    x = [1.0; 1.0]
    λs = eigvals(A)
    α = 2/(λs[1]+λs[2])

    # Steepest descent loop
    xs = [copy(x)]
    resids = []
    for k = 1:100
        x -= α*A*x
        push!(xs, copy(x))
        push!(resids, 0.5*x'*A*x)
    end

    # Plot the function and the iterates
    plot_convergence((x,y)->0.5*[x; y]'*A*[x; y], xs, resids)
end
```

The behavior of steepest descent iteration on a quadratic model problem is indicative of the behavior more generally: if x_* is a strong local minimizer of some general nonlinear ϕ, then gradient descent with sufficiently small step size will converge locally to x_*. But if $H_\phi(x_*)$ is ill-conditioned, then one has to take small steps, and the rate of convergence can be quite slow.

Not all problems are terrible ill-conditioned, and so in many cases simple gradient descent algorithms can work quite well. For ill-conditioned problems, though, we would like to change something about the algorithm. One approach is to keep the gradient descent direction and adapt the step size in a clever way; the Barzelei-Borwein (BB) method and related approaches follow this approach. These remarkable methods deserve to be better known, but in the interest of fitting the course into the semester, we will turn instead to the problem of choosing better directions.

3 "Steepest" descent

At a point x, a direction p is a *descent* direction for ϕ if $\phi'(x)p < 0$. We measure how "steep" a descent direction is by the rate at which the function value decreases if we move at unit speed in that direction: $-\phi'(x)p/\|p\|$. If $\|\cdot\|$ is the ordinary Euclidean norm on $\mathbb{R}^n$, we have that the direction of steepest descent is $-\phi'(x)^T/\|\phi'(x)\|$. But what happens if we use a different norm for measuring distance?

Put differently, consider the problem of maximizing $-g^T u$ over the ball $\|u\| \leq 1$. Then

- Over $\|u\|_2 \leq 1$, we have $u = -g/\|g\|$
- Over $\|u\|_\infty \leq 1$, we have $u = -\operatorname{sign}(g)$
- Over $\|u\|_1 \leq 1$, we have $u = -\operatorname{sign}(g_k)e_k$ where g_k is the largest magnitude entry of g.

Of course, these are not the only norms out there. In particular, if M is a positive definite matrix, then there is an associated inner product

$$\langle x, y\rangle_M = y^* M x$$

and an associated Euclidean norm

$$\|x\|_M = \sqrt{x^T M x}.$$

Maximizing $-g^T u/\|u\|_M$ gives us $u \propto -M^{-1}g$.

4 Scaled gradient descent

The *scaled* gradient descent iteration takes the form

$$x_{k+1} = x_k + \alpha_k p_k, \quad M_k p_k = -\nabla\phi(x_k).$$

where α_k and p_k are the step size and direction, as before, and M_k is a symmetric positive definite *scaling matrix*. We can also see this as ordinary steepest descent, but steepest descent with respect to the adaptively-chosen M_k Euclidean norms.

Positive definiteness of M_k guarantees that p_k is a *descent direction*, i.e.

$$\phi'(x_k)p_k = \nabla\phi(x_k)^T p_k = -\nabla\phi(x_k)^T M_k^{-1} \nabla\phi(x_k) < 0;$$

this in turn guarantees that if α_k is sufficiently small, $\phi(x_{k+1})$ will be less than $\phi(x_k)$ — unless $\phi(x_k)$ is a stationary point (i.e. $\nabla\phi(x_k) = 0$).

How does scaling improve on simple gradient descent? Consider again the quadratic model problem

$$\nabla\phi(x) = Ax + b,$$

and let M and α be fixed. With a little work, we derive the error iteration

$$e_{k+1} = (I - \alpha M A)e_k$$

If $\alpha M = A^{-1}$, the iteration converges in a single step! Going beyond the quadratic model problem, if x_* was a known strong local minimizer, we could use $M_k = H_\phi(x_*)$ – which will give us superlinear convergence.

The convergence of the scaled gradient descent iteration below is shown in Figure 6.

```
let
    x = [0.0; 0.5]
    α=1.0

    # Scaled steepest descent loop -- plot where we're going
    resids = []
    xs = [copy(x)]
    M = Hϕex([0; 0])
```

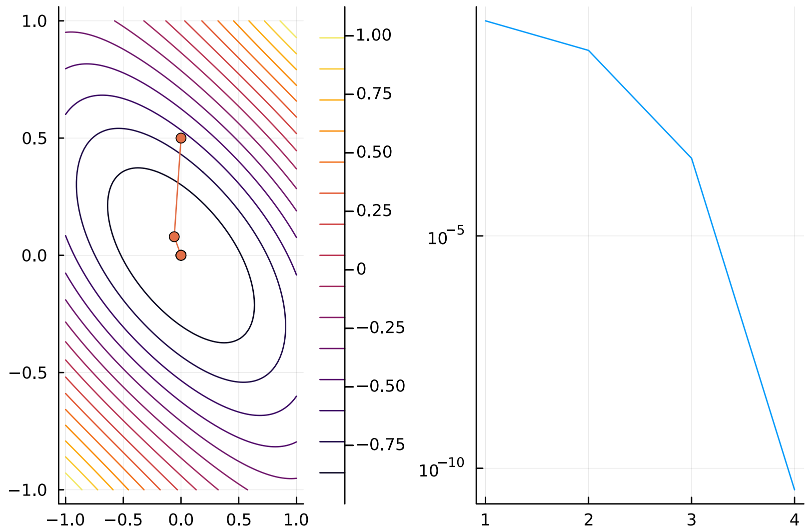

Figure 6: Convergence of scaled gradient descent.

```
    for k = 1:5

        # Compute the gradient and record the norm
        g = M\∇ϕex(x)
        push!(resids, norm(g))

        # Take a gradient descent step and record the point
        x -= α*g
        push!(xs, copy(x))

    end

    # Plot the function and the iterates
    plot_convergence((x,y)->ϕex([x; y]), xs, resids)
end
```

In general, we cannot scale with $H_\phi(x_*)$, since we don't know the location of x_*! But if $H_\phi(x_k)$ is positive definite, we might choose $M_k = H_\phi(x_k)$ — which would correspond to a Newton step. Of course, $H_\phi(x_k)$ does not have to be positive definite everywhere! Thus, most minimization codes based on Newton scaling use $M_k = H_\phi(x_k)$ when it is positive definite, and otherwise use some modification. One possible modification is to choose a diagonal shift $M_k = H_\phi(x_k) + \beta I$ where β is sufficiently large to guarantee positive definiteness. Another common approach is to compute a *modified Cholesky* factorization of $H_\phi(x_k)$. The modified Cholesky algorithm looks like ordinary Cholesky, and is identical to ordinary Cholesky when $H_\phi(x_k)$ is positive definite. But rather than stopping when it encounters a negative diagonal in a Schur complement, the modified Cholesky approach replaces that element with something else and proceeds.

4.1 Questions

Suppose ϕ is a C^2 function and H_ϕ is Lipschitz with constant M. Let x_* be a strong local minimizer for ϕ and consider the scaled steepest descent iteration

$$x_{k+1} = x_k - H_\phi(x_*)^{-1} \nabla \phi(x_k)$$

Subtract off the fixed point equation to get

$$e_{k+1} = e_k - H_\phi(x_*)^{-1} \left(\nabla \phi(x_* + e_k) - \nabla \phi(x_*) \right)$$

Conclude that for some $\xi \in [0,1]$,

$$\|e_{k+1}\| \leq \|H_\phi(x_*)^{-1} \left(H_\phi(x_*) - H_\phi(x_* + \xi e_k) \right) e_k\|,$$

which implies under the Lipschitz assumption that

$$\|e_{k+1}\| \leq \|H_\phi(x_*)^{-1}\| M \|e_k\|^2.$$

Therefore, the iteration converges quadratically for good enough initial guesses. Can you also give a condition on $\|e_0\|$ that guarantees an initial guess is "good enough"?