9. Minimization problems

We consider two classes of problems: the first is the unconstrained minimization problem (UCMP) and the second is nonlinear least squares (NLLS).

UCMP: Given $f : \mathbb{R}^N \to \mathbb{R}$, find $\boldsymbol{x} \in \mathbb{R}^N$ which minimizes $f(\boldsymbol{x})$.

NLLS: Given $\boldsymbol{F} = (f_1, \ldots, f_m)^T : \mathbb{R}^N \to \mathbb{R}^m$, with $m \geq N$, find $\boldsymbol{x} \in \mathbb{R}^N$ which minimizes $\phi(\boldsymbol{x}) = (1/2) \sum_{k=1}^{m} [f_k(\boldsymbol{x})]^2$.

Note that the second problem is a special case of the first, but with more structure.

9.1. Newton's method and steepest descent.

To solve UCMP, we can look for $\boldsymbol{x}^*$ at which

$$\nabla f = (\partial f/\partial x_1, \ldots, \partial f/\partial x_N) = \boldsymbol{0}$$

and the Hessian matrix $H_f = (\partial^2 f/\partial x_i \partial x_j)$ is symmetric and positive definite. Thus the problem becomes one of solving a nonlinear systems of equations $\boldsymbol{F}(\boldsymbol{x}) = 0$, where $\boldsymbol{F} = \nabla f$, i.e., $\boldsymbol{F}_i = \partial f/\partial x_i$.

If we apply Newton's method, we get the iteration

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - J_{\nabla f}(\boldsymbol{x}^n)^{-1} \nabla f(\boldsymbol{x}^n) = \boldsymbol{x}^n - H_f(\boldsymbol{x}^n)^{-1} \nabla f(\boldsymbol{x}^n),$$

since $(J_F)_{ij} = \partial F_i/\partial x_j$ and hence

$$(J_{\nabla f})_{ij} = \frac{\partial}{\partial x_j} \frac{\partial f}{\partial x_i} = \frac{\partial^2 f}{\partial x_i \partial x_j} = H_f.$$

Note that H_f is symmetric, so for a minimum at x^*, it will be sufficient to have $H_f(\boldsymbol{x}^*)$ to be positive definite.

In the special case of UCMP where $f = (1/2)\boldsymbol{x}^T A \boldsymbol{x} - \boldsymbol{x}^T b$, we considered methods of the form $\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \alpha_n \boldsymbol{p}^n$, where $\boldsymbol{p}^n$ is a search direction and α_n is a scalar. In the method of steepest descent, we choose $\boldsymbol{p}^n = -\nabla f(\boldsymbol{x}^n)$. With this choice and the special form of f, we can solve explicitly for the best choice of α_n. In the more general case, we would consider methods of the form

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - \alpha_n \nabla f(\boldsymbol{x}^n),$$

where α_n is chosen to guarantee that $f(\boldsymbol{x}^{n+1}) < f(\boldsymbol{x}^n)$. The advantage of this approach is that we do not need to compute the Hessian. However, the method converges slowly. To compromise, we could consider methods of the form

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - \alpha_n B_n^{-1} \nabla f(\boldsymbol{x}^n),$$

where B_n is symmetric and positive definite and α_n is chosen to insure that $f(\boldsymbol{x}^{n+1}) < f(\boldsymbol{x}^n)$. For example, let $B_n = (H_f(\boldsymbol{x}^n) + \mu_n I)$, with $\mu_n > 0$ chosen so that B_n is positive definite. For large values of μ_n this method behaves like steepest descent and for small values of μ_n, it behaves like Newton's method. The difficult part is deciding how to choose the parameter μ_n.

9.2. **Quasi-Newton methods.** Analogous to the case of quasi-Newton methods for nonlinear equations, we now wish to avoid computation of the Hessian at each iteration. In this case, we want to generate a sequence of symmetric, positive definite matrices B_n such that B_n approximates $H_f(\boldsymbol{x}^n)$, but can be computed easily from B_{n-1}. Since we are now solving the system $\nabla f(\boldsymbol{x}) = \mathbf{0}$, the appropriate Taylor series expansion is:

$$\nabla f(\boldsymbol{x}^n) = \nabla f(\boldsymbol{x}^{n+1}) + H_f(\boldsymbol{x}^{n+1})(\boldsymbol{x}^n - \boldsymbol{x}^{n+1}) + O(\boldsymbol{x}^n - \boldsymbol{x}^{n+1})^2.$$

Thus, we want the approximation B_{n+1} to $H_f(\boldsymbol{x}^{n+1})$ to satisfy the quasi-Newton equation

$$\nabla f(\boldsymbol{x}^n) = \nabla f(\boldsymbol{x}^{n+1}) + B_{n+1}(\boldsymbol{x}^n - \boldsymbol{x}^{n+1}).$$

To simplify notation, let

$$\boldsymbol{y}^n = \nabla f(\boldsymbol{x}^{n+1}) - \nabla f(\boldsymbol{x}^n), \qquad \boldsymbol{s}^n = \boldsymbol{x}^{n+1} - \boldsymbol{x}^n,$$

so the quasi-Newton equation is $B_{n+1}\boldsymbol{s}^n = \boldsymbol{y}^n$. If we look for a symmetric, single rank (the maximum number of linearly independent rows is one) update satisfying the quasi-Newton equation, then provided $(\boldsymbol{y}^n - B_n\boldsymbol{s}^n)^T\boldsymbol{s}^n \neq 0$, the only one is given by

$$B_{n+1} = B_n + \frac{(\boldsymbol{y}^n - B_n\boldsymbol{s}^n)(\boldsymbol{y}^n - B_n\boldsymbol{s}^n)^T}{(\boldsymbol{y}^n - B_n\boldsymbol{s}^n)^T\boldsymbol{s}^n}.$$

It turns out that this method does not work well, so we look for a double rank update. It can be shown that the general symmetric rank 2 update is given by:

$$B_{n+1} = B_n + \frac{(\boldsymbol{y}^n - B_n\boldsymbol{s}^n)(\boldsymbol{c}^n)^T + \boldsymbol{c}^n(\boldsymbol{y}^n - B_n\boldsymbol{s}^n)^T}{(\boldsymbol{c}^n)^T\boldsymbol{s}^n} - \frac{(\boldsymbol{y}^n - B_n\boldsymbol{s}^n)^T\boldsymbol{s}^n\boldsymbol{c}^n(\boldsymbol{c}^n)^T}{[(\boldsymbol{c}^n)^T\boldsymbol{s}^n]^2},$$

where $\boldsymbol{c}$ is an arbitrary vector such that $(\boldsymbol{c}^n)^T\boldsymbol{s}^n \neq 0$.

If we choose $\boldsymbol{c}^n = \boldsymbol{s}^n$, we get the Powell symmetric Broyden update. Since we would also like to have the property that B_n positive definite implies that B_{n+1} is positive definite, a better choice is $\boldsymbol{c}^n = \boldsymbol{y}^n$ (called the Davidon-Fletcher-Powell method).

Finally, as in the case of nonlinear equations, instead of updating B_n and then having to solve a linear system of equations at each step, we can update the inverse directly. If we let $H_n = B_n^{-1}$, we then get the iteration $\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - \alpha_n H_n \nabla f(\boldsymbol{x}^n)$, where

$$H_{n+1} = H_n + \frac{(\boldsymbol{s}^n - H_n\boldsymbol{y}^n)(\boldsymbol{s}^n)^T + \boldsymbol{s}^n(\boldsymbol{s}^n - H_n\boldsymbol{y}^n)^T}{(\boldsymbol{s}^n)^T\boldsymbol{y}^n} - \frac{(\boldsymbol{s}^n - H_n\boldsymbol{y}^n)^T\boldsymbol{y}^n\boldsymbol{s}^n(\boldsymbol{s}^n)^T}{[(\boldsymbol{s}^n)^T\boldsymbol{y}^n]^2}.$$

This method in which the inverse is updated directly is known as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method.

9.3. **Nonlinear least squares.** Finally, we consider the nonlinear least squares problem of minimizing $\phi(\boldsymbol{x}) = (1/2)\sum_{k=1}^m [f_k(\boldsymbol{x})]^2$. Let $\boldsymbol{F} = [f_1, \cdots, f_m]^T$. Then J_F, the Jacobian matrix of partial derivatives of $\boldsymbol{F}$ is given by $[J_f(\boldsymbol{x})]_{kj} = (\partial f_k/\partial x_j)$.

Now

$$\frac{\partial \phi}{\partial x_j} = \sum_{k=1}^m f_k(\boldsymbol{x})\frac{\partial f_k}{\partial x_j} = \{[J_f(\boldsymbol{x})]^T F(\boldsymbol{x})\}_j, \quad (H_\phi)_{ij} = \frac{\partial^2 \phi}{\partial x_i \partial x_j} = \sum_{k=1}^m \left[f_k(\boldsymbol{x})\frac{\partial^2 f_k}{\partial x_i \partial x_j} + \frac{\partial f_k}{\partial x_i}\frac{\partial f_k}{\partial x_j}\right].$$

Hence, $\nabla\phi(\boldsymbol{x}) = J_F(\boldsymbol{x})^T\boldsymbol{F}(\boldsymbol{x})$ and

$$H_\phi(\boldsymbol{x}) = J_F(\boldsymbol{x})^T J_F + \sum_{k=1}^m f_k(\boldsymbol{x}) H_{f_k}(\boldsymbol{x}).$$

Newton's method for the system $\nabla\phi(\boldsymbol{x}) = J_F(\boldsymbol{x})^T\boldsymbol{F}(\boldsymbol{x}) = 0$, is then given by

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - [H_\phi(\boldsymbol{x}^n)]^{-1}\nabla\phi(\boldsymbol{x}^n).$$

$$= \boldsymbol{x}^n - [J_F(\boldsymbol{x})^T J_F + \sum_{k=1}^m f_k(\boldsymbol{x}) H_{f_k}(\boldsymbol{x})]^{-1}[J_F(\boldsymbol{x}^n)]^T F(\boldsymbol{x}^n).$$

Since this is fairly complicated to compute, we seek a simpler method. One approach is to linearize $f_k(\boldsymbol{x})$ about $\boldsymbol{x}^n$ and minimize the resulting quadratic functional instead. Using Taylor series, we approximate

$$f_k(\boldsymbol{x}) \approx f_k(\boldsymbol{x}^n) + [\nabla f_k(\boldsymbol{x}^n)]^T(\boldsymbol{x} - \boldsymbol{x}^n),$$

since we expect the remainder to be small if $\boldsymbol{x} - \boldsymbol{x}^n$ is small. Inserting this approximation,

$$\phi(\boldsymbol{x}) \approx (1/2)\sum_{k=1}^m [f_k(\boldsymbol{x}^n)]^2 + \sum_{k=1}^m f_k(\boldsymbol{x}^n)[\nabla f_k(\boldsymbol{x}^n)]^T(\boldsymbol{x}-\boldsymbol{x}^n) + (1/2)\sum_{k=1}^m \{[\nabla f_k(\boldsymbol{x}^n)]^T(\boldsymbol{x}-\boldsymbol{x}^n)\}^2.$$

Then

$$\nabla\phi(\boldsymbol{x}) \approx \sum_{k=1}^m f_k(\boldsymbol{x}^n)\nabla f_k(\boldsymbol{x}^n) + \sum_{k=1}^m \nabla f_k(\boldsymbol{x}^n)[\nabla f_k(\boldsymbol{x}^n)]^T(\boldsymbol{x} - \boldsymbol{x}^n).$$

Since we want to find $\boldsymbol{x}$ such that $\nabla\phi(\boldsymbol{x}) = \boldsymbol{0}$, we choose $\boldsymbol{x}^{n+1}$ to satisfy

$$\sum_{k=1}^m \nabla f_k(\boldsymbol{x}^n)[\nabla f_k(\boldsymbol{x}^n)]^T(\boldsymbol{x}^{n+1} - \boldsymbol{x}^n) = -\sum_{k=1}^m f_k(\boldsymbol{x}^n)\nabla f_k(\boldsymbol{x}^n).$$

This may be written in the form

$$J_F^T(\boldsymbol{x}^n) J_F(\boldsymbol{x}^n)(\boldsymbol{x}^{n+1} - \boldsymbol{x}^n) = -J_F^T(\boldsymbol{x}^n) F(\boldsymbol{x}^n).$$

Hence, this approximation amounts to dropping the term $\sum_{k=1}^m f_k(\boldsymbol{x}) H_{f_k}(\boldsymbol{x})$ in Newton's method. We would expect this to be small if the minimum of ϕ is near zero. This method is known as the Gauss-Newton method.

In practice, a modified version of Gauss-Newton, known as the Levenberg-Marquardt method, is used. This method is given by the iteration

$$[\alpha_n I + J_F^T(\boldsymbol{x}^n) J_F(\boldsymbol{x}^n)](\boldsymbol{x}^{n+1} - \boldsymbol{x}^n) = -J_F^T(\boldsymbol{x}^n) F(\boldsymbol{x}^n),$$

where $\alpha_n \geq 0$ is an appropriately chosen scalar. The idea is that for α_n large, the method behaves like steepest descent (to ensure that the function $\phi(\boldsymbol{x})$ is being decreased), while for α_n small, it behaves more like Newton's method, which will converge faster. In general, we would increase α_n if we take a step that increases $\phi(\boldsymbol{x})$, and decrease α_n if $\phi(\boldsymbol{x})$ is decreasing.