Nonlinear Least Squares Optimization

MA 348
Kurt Bryan

Least-Squares

We've already considered objective functions of the special form

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} f_i^2(\mathbf{x}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and the f_i are linear. Now we'll let the f_i be general nonlinear (but differentiable) functions. Such optimization problems usually occur in data fitting problems, for example, fitting $\phi(x) = a+bx+e^{cx}$ to (x, y) data points $(1.0, 2.2), (1.6, 2.8), (2.3, 3.9), (3.4, 4.4)$, and $(4.1, 5.2)$, by adjusting a, b, and c to minimize the squared error

$$f(a,b,c) = \frac{1}{2}((\phi(1.0)-2.2)^2+(\phi(1.6)-2.8)^2+(\phi(2.3)-3.9)^2+(\phi(3.4-4.4)^2+(\phi(4.1)-5.2)^2).$$

As in the linear case, it wouldn't be crazy to attack this optimization problem with any general nonlinear algorithm, e.g., conjugate gradients or quasi-Newton methods. But the objective function (1) has a rather special structure and it turns out you can exploit this to get (usually) better results with less hassle.

Modified Newton's Method

Suppose we attack the problem of minimizing $f(\mathbf{x})$ in equation (1) with Newton's Method. At each iteration we'll need the gradient of f, which you can easily calculate is given by

$$\nabla f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x})\nabla f_i(\mathbf{x}). \tag{2}$$

We also need the Hessian matrix, which you can check is given by

$$\mathbf{H}(\mathbf{x}) = \sum_{i=1}^{m}(\nabla f_i(\mathbf{x})\nabla f_i^T(\mathbf{x}) + f_i(\mathbf{x})\mathbf{H}_i(\mathbf{x})) \tag{3}$$

where $\mathbf{H}_i$ is of course the Hessian matrix of f_i. BE CAREFUL: The term $\nabla f_i(\mathbf{x})\nabla f_i^T(\mathbf{x})$ in equation (3) above is NOT the dot product of ∇f_i with itself (a scalar), but rather is an n by n matrix (the so-called outer product of a vector with itself).

Equations (2) and (3) can be written in an alternate form. Let $\mathbf{J}$ denote the matrix whose (i, j) entry is $\frac{\partial f_i}{\partial x_j}$ and let $\mathbf{f} = [f_1, \ldots, f_m]^T$. Then we can write

$$\nabla f(\mathbf{x}) = \mathbf{J}^T\mathbf{f} \tag{4}$$

$$\mathbf{H}(\mathbf{x}) = \mathbf{J}^T\mathbf{J} + \sum_{i=1}^{m} f_i(\mathbf{x})\mathbf{H}_i(\mathbf{x})). \tag{5}$$

Exercise 1

- Verify equations (2)-(5).

Armed with equations (2) and (3) we can minimize f using Newton's method or some variation. This is what we'll do, but with a small modification: we're going to replace the Hessian with an approximation

$$\tilde{\mathbf{H}}(\mathbf{x}) = \mathbf{J}^T\mathbf{J} \tag{6}$$

by DROPPING the terms $f_i(\mathbf{x})\mathbf{H}_i(\mathbf{x})$. The reasons are

1. If the model we're trying to fit to the data is good then the f_i should be fairly small near the minimum, so dropping these terms shouldn't change $\mathbf{H}$ too much. In fact if the f_i are zero then we don't change $\mathbf{H}$ at all (at the minimum).

2. The resulting matrix $\tilde{\mathbf{H}}$ is symmetric positive semi-definite, and "probably" positive definite, so the resulting algorithm is will likely be a descent method.

3. We don't have to compute second derivatives!

Exercise 2

- Verify that $\tilde{\mathbf{H}}$ is positive semi-definite. Under what circumstances would $\mathbf{v}^T\tilde{\mathbf{H}}\mathbf{v} = 0$?

A very simple algorithm for minimizing f as defined by equation (1) would be

1. Make initial guess $\mathbf{x}_0$; set $k = 0$.

2. Solve
$$\tilde{\mathbf{H}}\mathbf{h}_k = -\nabla f(\mathbf{x}_k), \tag{7}$$
for $\mathbf{h}_k$, where $\tilde{\mathbf{H}}$ is defined by equation (6), and $-\nabla f(\mathbf{x}_k)$ can be computed from (2).

3. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{h}_k$ (like in straight Newton's method) or maybe do a line search from $\mathbf{x}_k$ in the direction $\mathbf{h}_k$ and let $\mathbf{x}_{k+1}$ be the minimum found by the line search.

4. Test for convergence. Increment k and return to step 2.

The general hope is that since $\mathbf{h}_k$ is likely to be a descent direction, the algorithm makes downhill progress. As we near a minimum (and if the function value here is close to zero) the $\tilde{\mathbf{H}} \approx \mathbf{H}$ and the algorithm converges with the full speed of Newton's Method.

In general one does not do a line search in the algorithm above, but rather takes $\mathbf{h}_k$ "as is." That's how we'll proceed in the rest of the handout. It may turn out that $\mathbf{h}_k$ is not a good step (if, say, $f(\mathbf{x}_k + \mathbf{h}_k) > f(\mathbf{x}_k)$). We'll deal with this in a moment.

Example 1: Let $n = 2$ and $m = 3$, with $f_1(\mathbf{x}) = 10(x_2 - x_1^2)$, $f_2(\mathbf{x}) = 1 - x_1$, $f_3(\mathbf{x}) =$

$x_1 + \sin(x_2)$. I used the modified Cholesky algorithm to solve $\tilde{\mathbf{H}}\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$ at each iteration, with $\delta = 0.01$ (if any pivot element in the routine is less than δ then it gets replaced by δ). I took $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{h}_k$ (no line search—just take $\mathbf{h}_k$ as the step). The starting point was $(-1, -1)$ (function value 203.7). The results were (stopping tolerance 10^{-3} on successive iterates)

iteration	point	function value
0	(-1,-1)	203.7
1	(0.926, -2.85)	686.9
2	(-0.428,-1.67)	173.4
3	(1.028, -1.063)	224.7
4	(0.549, 0.070)	2.97
5	(0.304, 0.029)	0.498
6	(0.322, 0.099)	0.319
7	(0.318, 0.097)	0.319
8	(0.319, 0.098)	0.319

Note that the method was NOT a descent method; the steps from 0 to 1, and from 2 to 3, both increased the objective function.

Exercise 3

- Use this method to solve the nonlinear least-squares problem at the start of this handout (so $m = 5$, $n = 3$).

Variations and Improvements

The matrix $\tilde{\mathbf{H}}$ may, under some circumstances, become singular or nearly singular. In fact, if $m < n$ this is guaranteed! In this case $\mathbf{h}_k$ is not a descent direction, and moreover, $\mathbf{h}_k$ may become very large.

Exercise 4

- If f_i is a function of n variables, show that the n by n matrix $\nabla f_i \nabla f_i^T$ is rank one (if $\nabla f_i \neq 0$), and so has a nullspace of dimension $n - 1$.
- Show that for $m < n$ the n by n matrix $\sum_{i=1}^{m} \nabla f_i \nabla f_i^T$ is at most of rank m, and hence NOT invertible. Hint: Use the fact that if V and W are subspaces of $\mathbb{R}^n$ of dimensions r and s, respectively, then $V \cap W$ is at least of dimension $\max(0, r + s - n)$.

It would be nice to safeguard against the possibility that $\tilde{\mathbf{H}}$ is singular, and to somehow exert more control over $\mathbf{h}_k$, to ensure that the algorithm makes downhill progress. One method is to replace equation (7) by

$$(\tilde{\mathbf{H}} + \mu\mathbf{I})\mathbf{h}_k = -\nabla f(\mathbf{x}_k) \tag{8}$$

where μ is some positive number. The matrix $\tilde{\mathbf{H}} + \mu\mathbf{I}$ is symmetric and positive definite (hence invertible) if $\mu > 0$, so the resulting $\mathbf{h}_k$ will be a descent direction. If μ is very large then $\mathbf{h}_k$ is essentially a multiple of $-\nabla f$, the direction of steepest descent.

One drawback to this procedure is that if we fix $\mu > 0$ then $\tilde{\mathbf{H}} + \mu\mathbf{I}$ will never equal the true Hessian, even if all f_i equal zero, and so the quadratic convergence rate of Newton's method will be lost. One strategy is to update the value of μ at each iteration, increasing μ if the algorithm isn't make good progress (forcing steepest descent steps) and decreasing μ toward zero if things are going well (so the method is more like straight Newton).

We can quantify how well the algorithm is making progress by computing the ratio

$$\rho = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})}{\nabla f(\mathbf{x}_k) \cdot \mathbf{h}_k + \frac{1}{2}\mathbf{h}_k^T \tilde{\mathbf{H}} \mathbf{h}_k} \tag{9}$$

where $\mathbf{h}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$. The numerator is the decrease in function value from one iteration to the next; the denominator is the decrease predicted by the local quadratic model $f(\mathbf{x}_{k+1}) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{h}_k + \frac{1}{2}\mathbf{h}_k^T \tilde{\mathbf{H}} \mathbf{h}_k$. If this ratio is close to one then this indicates that the local quadratic model is good, and we should decrease μ (so Newton's method can run unfettered). If $\rho << 1$ we should increase μ, thus forcing the method to take steepest descent-like steps. There's lots of room for experimentation, but I'll use the rule that if $\rho > 0.9$ then μ should be cut by a factor of 10. If $\rho < 0.1$ then μ should be increased by a factor of 10. Otherwise, leave μ alone. Also, if $f(\mathbf{x}_k + \mathbf{h}_k) < f(\mathbf{x}_k)$ (so progress was made) then accept the step $\mathbf{h}_k$ and increment k; otherwise, reject the step, increase μ, and try again. Note that for a sufficiently small value of μ we must obtain $f(\mathbf{x}_k + \mathbf{h}_k) < f(\mathbf{x}_k)$.

Example 2: Same setting as the previous example (but with $\delta = 0$ in the Cholesky decomposition routine) and initial value $\mu = 1.0$. The algorithm converges in 7 iterations. The first few look like

iteration	point	function value
0	(-1,-1)	203.7
1	(-0.01, -0.976)	48.5
2	(0.434,-0.02)	2.40
3	(0.304, 0.072)	0.334
4	(0.322,0.10)	0.319
5	(0.318,0.097)	0.319
6	(0.319,0.098)	0.319
7	(0.319,0.098)	0.319

Note that the objective function now decreases monotonically.

More Improvements

There is (at least) one other significant improvement that can be made. At each iteration we have to solve $(\tilde{\mathbf{H}} + \mu\mathbf{I})\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$, that is,

$$(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\mathbf{h}_k = -\nabla f(\mathbf{x}_k). \tag{10}$$

When $\mu = 0$ this is $\mathbf{J}^T\mathbf{J}\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$, which, given that $\nabla f = \mathbf{J}^T\mathbf{f}$, you might recognize as the normal equations for the linear least-squares problem of minimizing $\|\mathbf{J}\mathbf{h}_k + \mathbf{f}\|^2$. In this case we've seen that QR decomposition is the numerically preferred method, so if $\mu = 0$ that's what we should do. In fact, even if $\mu > 0$ equation (10) is still the normal equation for a least squares problem, namely that of finding the least squares solution to the linear system

$$\begin{bmatrix} \mathbf{J} \\ \sqrt{\mu}\mathbf{I} \end{bmatrix} \mathbf{h}_k = \begin{bmatrix} -\mathbf{f}(x_k) \\ \mathbf{0} \end{bmatrix} \tag{11}$$

where the $\mathbf{0}$ on the right is an $m - n$ by n matrix of zeros. Thus QR decomposition remains the method of choice.

Exercise 5

- Verify equation (11).

The Trust Region Approach

If $\mu = 0$ in equation (10) or (11) then $\mathbf{h}_k$ is the full unconstrained Newton step, while if μ is very large then $\mathbf{h}_k \approx -\nabla f(x_k)/\mu$ and the algorithm becomes something like steepest descent. We increase or decrease μ according to how well the algorithm is proceeding.

There's a slightly different point of view on this process for controlling the step $\mathbf{h}_k$, called the "trust region" approach. Consider the problem of solving for the stepsize $\mathbf{h}_k$ by minimizing $\|\mathbf{J}\mathbf{h}_k + \mathbf{f}\|^2$ (this is the $\mu = 0$ case of (10) or (11)) but with the additional constraint that the step $\mathbf{h}_k$ must not exceed a certain magnitude Δ, that is,

$$\|\mathbf{h}_k\| \leq \Delta. \tag{12}$$

I claim that for any fixed $\Delta > 0$ there is some unique μ such that the problem of minimizing $\|\mathbf{J}\mathbf{h}_k + \mathbf{f}\|^2$ subject to the constraint (12) is equivalent to solving equation (10).

To prove this, let $\Delta > 0$ be given and consider minimizing $\|\mathbf{J}\mathbf{h}_k + \mathbf{f}\|^2$ subject to the constraint (12). At the solution (which clearly exists—why?) we either have $\|\mathbf{h}_k\| < \Delta$ or $\|\mathbf{h}_k\| = \Delta$. If $\|\mathbf{h}_k\| < \Delta$ (so the constraint is not "active") the gradient of the function $\|\mathbf{J}\mathbf{h}_k + \mathbf{f}\|^2$ with respect to the components of $\mathbf{h}_k$ must be zero. This gradient is easy to compute and is given by $2(\mathbf{J}^T\mathbf{J}\mathbf{h}_k + \mathbf{J}^T\mathbf{f})$. Setting this to zero yields equation (10) in the case $\mu = 0$, so for those Δ we associate $\mu = 0$.

Suppose, on the other hand, we have $\|\mathbf{h}_k\| = \Delta$; I'll write this constraint as $\|\mathbf{h}_k\|^2 - \Delta^2 = 0$. A Calc III Lagrange multiplier argument shows that for some λ we have the gradient of $\|\mathbf{J}\mathbf{h}_k + \mathbf{f}\|^2$ is proportional to the gradient of $\|\mathbf{h}_k\|^2 - \Delta^2$, where all gradients are taken with respect to the components of $\mathbf{h}_k$. The gradient of $\|\mathbf{h}_k\|^2 - \Delta^2$ is just $2\mathbf{h}_k$, so we obtain

$$2(\mathbf{J}^T\mathbf{J}\mathbf{h}_k + \mathbf{J}^T\mathbf{f}) = 2\lambda\mathbf{h}_k.$$

With $\mu = -\lambda$ this is exactly equation (10) (recall that $\nabla f = \mathbf{J}^T\mathbf{f}$).

Thus to each $\Delta > 0$ we can associate a unique $\mu \geq 0$. Something like the converse is true too: For each $\mu > 0$ there is some unique $\Delta > 0$ (but for $\mu = 0$ any sufficiently large Δ will work). We can show this by showing that the norm of the solution $\|\mathbf{h}_k\|$ to equation (10) is strictly increasing with respect to $\mu > 0$. To prove this, note that the matrix $\mathbf{J}^T\mathbf{J}$ is symmetric and (as we've assumed all along) positive definite, hence can be diagonalized as $\mathbf{J}^T\mathbf{J} = \mathbf{P}\mathbf{D}\mathbf{P}^T$, where $\mathbf{P}$ is the matrix of orthonormal eigenvectors (hence $\mathbf{P}$ is an orthogonal matrix) and $\mathbf{D}$ is diagonal with positive entries. Equation (10) can be written as

$$\mathbf{P}(\mathbf{D} + \mu\mathbf{I})\mathbf{P}^T\mathbf{h}_k = -\nabla f(\mathbf{x}_k).$$

Multiply by $\mathbf{P}^T$ and then $(\mathbf{D} + \mu\mathbf{I})^{-1}$ on both sides to obtain

$$\mathbf{P}^T\mathbf{h}_k = -(\mathbf{D} + \mu\mathbf{I})^{-1}\mathbf{P}^T\nabla f(\mathbf{x}_k).$$

Take the norm of both sides above, noting that since $\mathbf{P}^T$ is orthogonal we have $\|\mathbf{P}^T\mathbf{h}_k\| = \|\mathbf{h}_k\|$ and hence

$$\|\mathbf{h}_k\| = \|(\mathbf{D} + \mu\mathbf{I})^{-1}\mathbf{b}\| \tag{13}$$

where $\mathbf{b} = \mathbf{P}^T\nabla f(\mathbf{x}_k)$. It's easy to see that the right side above is strictly decreasing in μ (for $\mu > 0$).

Exercise: Prove the right side of (13) is strictly decreasing in μ.

From the above exercise we can conclude that for each $\mu > 0$ there is only one Δ.

The trust region approach to the algorithm updates the parameter Δ instead of μ. The idea is that the quadratic model ((4) and (6)) which underlies our whole procedure may be good only for sufficiently small $\mathbf{h}$. As such, minimizing the full least-squares functional by trying to minimize $\|\mathbf{J}\mathbf{h}_k + \mathbf{f}\|^2$ may yield poor results if the resulting $\mathbf{h}_k$ is allowed to be too large. We thus minimize $\|\mathbf{J}\mathbf{h}_k + \mathbf{f}\|^2$ subject to the restriction that $\mathbf{h}_k$ lie in a region where the approximation is valid (so that $\mathbf{h}_k$ probably goes a good job of minimizing the full non-linear least-squares functional in this range too). The size of the region is governed by Δ, and in the trust region approach it is Δ that is increased or decreased as the algorithm proceeds. The inequality $\|\mathbf{h}_k\| \leq \Delta$ defines what is called the "trust region" about the current operating point $\mathbf{x}_k$, where we think our quadratic approximation is valid.

Typically, if the algorithm is proceeding well (say ρ in (9) is greater than 0.9) the trust region is expanded, by increasing Δ by some factor, typically 10. If the progress is poor (say $\rho < 0.1$) we decrease Δ by a factor of 10. Note that at each step we have to solve a minimization problem, that of minimizing $\|\mathbf{J}\mathbf{h}_k + \mathbf{f}\|^2$ subject to (12). Ironically, this is done with a root-finding approach, by varying μ in equation (11) until we obtain a solution with $\|\mathbf{h}_k\| = \Delta$.

The Levenberg-Marquardt Algorithm

The ideas above form the basis for the *Levenberg-Marquardt Algorithm*, the standard for non-linear least squares problems. One other refinement that can be made is to replace the condition $\|\mathbf{h}_k\| \leq \Delta$ with $\|\Lambda\mathbf{h}_k\| \leq \Delta$ where Λ is some diagonal "scaling matrix" that can be updated from iteration to iteration. This can be helpful if the least-squares functional has greater or lesser sensitivity to the various independent variables $x_1, \ldots, x_n$; in this case equation (10) is replaced by $(\mathbf{J}^T\mathbf{J} + \mu\Lambda)\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$, but otherwise the analysis is similar. One common choice is to take the (i,i) diagonal entry of Λ as $\Lambda_{ii} = \frac{\partial f}{\partial x_i}(\mathbf{x}_0)$ (so Λ is fixed at the outset). The paper "The Levenberg-Marquardt Algorithm: Implementation and Theory" by Jorge J. Moré has a thorough description for a robust and efficient implementation.