CS 542G: Robustifying Newton, Constraints, Nonlinear Least Squares

Robert Bridson

October 29, 2008

1 Hessian Problems in Newton

Last time we fixed one of plain Newton's problems by introducing line search along the Newton direction. However, we are still left with the problem of bad Hessians, for example when H is singular and the linear equations $H\Delta x = -\nabla f$ have no solution. In fact, even if H is nonsingular, if it has any negative eigenvalues we could be in trouble.

Line search may fail unless the direction d is a **descent direction**: that is, the directional derivative of f along d is negative, so that we know we can make some progress towards a minimum for α small enough. The direction derivative along d is:

$$\frac{\partial f}{\partial d} = \nabla f^T d$$

Obviously, unless we're at a local min (so $\nabla f = 0$) the Steepest Descent direction $d = -\nabla f$ must make this negative. However, if we plug in the Newton direction $d = -H^{-1}\nabla f$ we get:

$$\frac{\partial f}{\partial d} = -\nabla f^T H^{-1} \nabla f$$

If H (and hence H^{-1}) has a negative eigenvalue, then if ∇f happens to be its eigenvector we find the Newton direction has a *positive* directional derivative, so line search may well fail.

Therefore we really prefer for H to be SPD, which also makes minimizing the model objective function well-posed. If we evaluate the Hessian and it's not SPD, we should modify the H to make it SPD—with the conviction that we must necessarily be far from Newton converging so the Newton step isn't optimal anyhow, and our primary goal is instead to make sure we get a descent direction for line search.

One way to force H to be SPD is to attempt to run Cholesky factorization for use in solving the Newton linear system. If Cholesky succeeds, H already was SPD and we're happy. Otherwise when we hit a zero or negative pivot we can simply replace it with a positive value of our choice; this gives us a matrix LL^T which is SPD but will no longer equal the Hessian—but in some sense is only minimally perturbed from it. This approach is often called **modified Cholesky**.

Another more common approach to making Newton robust, which doesn't rely on any particular method for solving the linear system (as modified Cholesky obviously does), is simply to add a multiple of the identity matrix to H:

$$H + \mu I$$

As long as $\mu > 0$ is larger in magnitude than the biggest non-positive eigenvalue of H, this must be SPD. Since we don't expect to know what that is we can simply guess a value (somewhere between 0 and $\|H\|$), adjusting it up if we hit a problem solving with H or don't get a descent direction. (And of course we should always give $\mu = 0$ a try to get back to plain Newton at convergence.)

Note that the Steepest Descent direction is the solution of $Id = -\nabla f$, compared to Newton using the solution of $Hd = -\nabla f$, and thus we can interpret this latest method $(H + \mu I)d = -\nabla f$ as a blend between the two.

We can also interpret this as a modification of the model objective function, a regularization of the Taylor series to make minimization well-posed. Instead of minimizing

$$\min_{\Delta x} f(x^{(k)}) + \nabla f(x^{(k)})\Delta x + \frac{1}{2}\Delta x^T H(x^{(k)})\Delta x$$

we are now minimizing

$$\min_{\Delta x} f(x^{(k)}) + \nabla f(x^{(k)})\Delta x + \frac{1}{2}\Delta x^T H(x^{(k)})\Delta x + \frac{\mu}{2}\|\Delta x\|_2^2$$

i.e. including a regularization term as discussed earlier to make the problem well-posed.

2 Trust-Region Methods

We've just seen an approach to making methods like Newton robust based around line search, with additional effort expended to force the search direction to be a descent direction. The **Trust Region** (TR) approach to robustifying is quite different.

The idea behind TR methods is that the Taylor polynomial, i.e. the model objective function, must be a good enough approximation to f in some bounded neighbourhood that minimizing it there will be a good idea. Instead of adding a line search and altering the search direction to make Newton more robust, the TR approach instead adds a constraint that Δx not be too large—it can only be taken from the region where we trust the model objective function is a good approximation to f. The sub-problem we solve every iteration is then:

$$\min_{x:\|x\|\leq D} f(x^{(k)}) + \nabla f(x^{(k)})\Delta x + \frac{1}{2}\Delta x^T H(x^{(k)})\Delta x$$

Here D is the trust region radius, which we also have to estimate as we go. It plays a role similar to the step size α in a line search method. If the solution Δx to the problem doesn't lead to a reduction in $f(x^{(k)} + \Delta x)$ relative to $f(x^{(k)})$ then D was obviously too large and we can try it again with D reduced; if the solution does give a redution, we can try increasing D in the next iteration. Near convergence, the limit D should stop playing a role (i.e. $\|\Delta x\| < D$ naturally) and plain Newton is recovered.

3 Constrained Optimization

3.1 Equality Constraints

The TR method involved solving a sub-problem with a constraint on the variable, a topic we haven't broached so far. Let's first take a look at a special case, which is somewhat easier, where constraints are expressed as equalities:

$$\min_{x:g(x)=0} f(x)$$

We've introduced a constraint function $g(x)$, where the only values we consider are those where $g(x) = 0$.

The approach you probably saw in calculus to dealing with constraints like this is to introduce **Lagrange multipliers**. At a local minimum subject to the constraint, the derivative of f in directions tangential to the constraint must be zero, but it might not be zero in directions orthogonal to the constraint. In other words, ∇f must be orthogonal to the constraint if it's not zero. The gradient ∇g is also orthogonal, and so the two gradients must be proportional to each other; the Lagrange multiplier λ will be the constant of proportionality. In summary, the following is satisfied at the minimum:

$$\begin{aligned} g(x) &= 0 \\ \nabla f(x) &= \nabla g(x)\lambda \end{aligned}$$

This generalizes to include multiple constraints (i.e. with g vector-valued), with the same number of components in λ.

It's fairly natural to extend the Newton approach of replacing a general function with a simpler polyonmial model here: for example, in an iteration replacing the condition $g(x) = 0$ with the linearization $g(x^{(k)}) + \nabla g(x^{(k)})\Delta x = 0$. This results in a linear system for Δx and λ.

A somewhat different approach to constraints is to use **penalties**. If we introduce a penalty function

$$G(x) = \begin{cases} 0 & : \quad g(x) = 0 \\ \infty & : \quad g(x) \neq 0 \end{cases}$$

then clearly the unconstrained problem $\min_x f(x) + G(x)$ is equivalent to the original constrained problem. However, this objective is very badly behaved and not amenable to solution directly (since it's not even continuous). Instead, we introduce a parameterized family of softer penalty functions, $G_\epsilon(x)$, each smooth but converging to the hard penalty $G(x)$ as $\epsilon \to 0$. For example, we could take

$$G_\epsilon(x) = \frac{1}{\epsilon} g(x)^2$$

or something similar. Then, during an iterative method to solve the unconstrained problem $\min_x f(x) + G_\epsilon(x)$ we steadily slide ϵ down towards zero, forcing the unconstrained method to respect the constraint in the limit. Despite the attractiveness of this approach's simplicity, there should obviously be some concern for ill-conditioned matrices appearing as $\epsilon \to 0$.

3.2 Inequality Constraints

Inequality constrainted problems, of the form

$$\min_{x:g(x)\leq 0} f(x),$$

raise a number of additional complications. If we know the answer, the minimum, actually satisfies $g(x) < 0$, then the constraint is "inactive" there: that x is also a local minimum for the unconstrained problem $\min_x f(x)$. On the other hand, if the answer satisfies $g(x) = 0$, then the constraint is "active", and x is also a local minimum for the equality constrained problem $\min_{x:g(x)=0} f(x)$. In some sense, the inequality adds an extra combinatorial aspect to the problem, determining if a constraint is active or not. If there are k inequality constraints, all 2^k active/inactive combinations could be tried, but that's obviously inefficient; the challenge of inequality constrained optimization is to avoid that exponential cost.

The two same approaches from above can be applied here too. With Lagrange multipliers, the question of active/inactive boils down to deciding if a λ should be nonzero (active) or zero (inactive); a bit more analysis or geometric intuition[1] can show there is an extra condition $\lambda \leq 0$ that goes alongside

[1]If the solution to the inequality constrained problem lies on the boundary $g(x) = 0$, then f must be increasing as you go inside the region $g(x) < 0$, which means ∇f points inwards. On the other hand, ∇g points outwards from the region, so the two gradients must be opposite each other.

$g(x) \le 0$, with at least one of λ or $g(x)$ needing to be zero. For penalty methods, the penalty functions have to be designed a little differently so they are zero (at least in the limit) when $g(x) \le 0$.

3.3 The Trust Region Newton Problem

Let's take a look now at the trust region sub-problem we encountered above:

$$\min_{x:\|x\|^2-D^2\le 0} f(x^{(k)}) + \nabla f(x^{(k)})\Delta x + \frac{1}{2}\Delta x^T H(x^{(k)})\Delta x$$

We've manipulated the inequality constraint to get rid of the square root in the 2-norm and put it in ≤ 0 form (so our constraint function is $g(x) = \|x\|^2 - D^2$). At a local minimum for this problem, there is a Lagrange multiplier $\lambda \le 0$ where

$$\nabla f(x^{(k)}) + H(x^{(k)})\Delta x = \lambda 2\Delta x$$

In other words, there is a $\mu \ge 0$ (with $\mu = -\frac{1}{2}\lambda$) so that

$$\begin{array}{rrcl} & H\Delta x + \mu\Delta x & = & -\nabla f \\ \Leftrightarrow & (H + \mu I)\Delta x & = & -\nabla f \end{array}$$

We're back at the same regularized linear system we saw before! (This can also be derived with a penalty method.) The only difference this time is the strategy for picking μ: it's chosen essentially so that no line search is needed, i.e. the update $x^{(k+1)} = x^{(k)} + \Delta x$ is an improvement.

4 Nonlinear Least Squares

An important class of optimization problems is **nonlinear least squares**, a generalization of the linear least squares we saw earlier. As before, we have a set of n data points (labeled $\{b_i\}_{i=1}^n$ for this section) which we want to fit a parameterized function to: $f_i(x) \approx b_i$. Here we've changed notation significantly, so that x is a vector of the k parameters: in the linear case earlier we called it α, the coefficients of the linear combination of basis functions, but now we're not assuming f depends linearly on its parameters. We still define the residual vector $r_i = b_i - f_i(x)$ and seek to minimize its 2-norm $\|r(x)\|$, but this is no longer a simple quadratic in x so the linear algebra solution we derived doesn't apply.

We can of course treat $\min_x \|r(x)\|^2$ as a generic optimization problem and try out any of the strategies discussed above. However, there is more "structure" known about this problem that we can gainfully exploit. Furthermore, if we did happen to have a linear least squares problem (so $r(x) = b - Ax$ for some constant rectangular matrix A) then a method like Newton would end up being equivalent to solving the normal equations—which we know can be problematic when A is ill-conditioned.

4.1 Gauss-Newton

This leads us to the **Gauss-Newton** algorithm. Just like Newton's method replaces the general objective function with a simpler model using Taylor series, one step of Gauss-Newton replaces the general nonlinear $f(x)$ with a model linear function (the first two terms of the Taylor series):

$$f(x^{(k)} + \Delta x) \approx f(x^{(k)}) + J(x^{(k)})\Delta x$$

Here J is the Jacobian, a rectangular $n \times k$ matrix with all the first partial derivatives of f:

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_k} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_k} \end{pmatrix}$$

This leads to a model linear residual $r(x^{(k)} + \Delta x) \approx b - f(x^{(k)}) - J\Delta x$, which we can now minimize in the least squares sense to get the next guess. This is now a regular linear least squares problem:

$$\min_{\Delta x} \left\| \left(b - f(x^{(k)}) \right) - J\Delta x \right\|^2$$

We can solve this with an appropriate technique, such as QR factorization of J, and then update $x^{(k+1)} = x^{(k)} + \Delta x$.

Note that this does fit the bill as a method which, when given a truly linear least squares problem, does allow the usual least squares methods. Furthermore, we can expect very good convergence if $f(x)$ is "close" to linear.

4.2 Connection to Newton

It's instructive to make a comparison between Gauss-Newton and regular Newton. The Newton system is based partly on the gradient of the objective $\|r(x)\|^2$:

$$\begin{aligned}(\nabla \|r(x)\|^2)_i &= \frac{\partial \|r(x)\|^2}{\partial x_i} \\ &= \sum_{s=1}^{k} 2 r_s(x) \frac{\partial r_s(x)}{\partial x_i} \\ &= \sum_{s=1}^{k} 2\left(b_s - f_s(x)\right)\left(-\frac{\partial f_s(x)}{\partial x_i}\right) \\ &= \sum_{s=1}^{k} 2\left(b_s - f_s(x)\right)\left(-J_{si}(x)\right) \\ &= 2\left[-J(x)^T(b - f(x))\right]_i\end{aligned}$$

and also on the Hessian:

$$\begin{aligned}H_{ij}(x) &= \frac{\partial^2 \|r(x)\|^2}{\partial x_i \partial x_j} \\ &= \frac{\partial}{\partial x_j}\left(\sum_{s=1}^{k} 2\left(b_s - f_s(x)\right)\left(-\frac{\partial f_s(x)}{\partial x_i}\right)\right) \\ &= \sum_{s=1}^{k} -2\frac{f_s(x)}{x_j}\left(-\frac{\partial f_s(x)}{\partial x_i}\right) + \sum_{s=1}^{k} 2\left(b_s - f_s(x)\right)\left(-\frac{\partial^2 f_s(x)}{\partial x_i \partial x_j}\right) \\ &= 2(J^T J)_{ij} - 2r \cdot H\end{aligned}$$

where H is the Hessian of f (technically a rank three tensor, not a matrix any more). The Newton linear system simplifies to:

$$(J^T J - r \cdot H)\Delta x = J^T(b - f(x^{(k)}))$$

Note this only differs from the normal equations $J^T J \Delta x = J^T(b - f(x^{(k)}))$ for Gauss-Newton by an additional term $-r \cdot H$ in the matrix which contains the second derivatives of f.

We can thus expect that Gauss-Newton will have good convergence properties like Newton, though unless the second derivatives of f are zero (i.e. f is linear) or the residual is zero at the final solution the convergence rate isn't quite as good. However, unlike Newton, there's never a danger of a negative eigenvalue spoiling the chances of a descent direction in Gauss-Newton: $J^T J$ has to be positive semi-definite.

Also note that Gauss-Newton makes a pretty good approximation to Newton despite not involving any second derivatives. This can be an important benefit, since apart from the very special case of polynomials, taking a derivative is generally both expensive and numerically worrisome (derivatives are generally not as smooth as the original function, with higher derivatives even worse). Next time we'll look at a more general class called **Quasi-Newton** methods which approximate Newton without need for second derivatives.

4.3 Making Gauss-Newton Robust

Far from convergence, Gauss-Newton makes no guarantee that the step with Δx will actually lead to an improvement. Just like Newton, we can make the method more robust with a few extra tweaks.

The first path we'll consider is adding line search: find a step size α so that $x^{(k)} + \alpha\Delta x$ leads to a good enough reduction in $\|r\|^2$. We also might run into issues where J is (nearly) rank-deficient, which might lead to troubles in solving the linear least squares problem; we can fix that by regularizing the model problem as before, minimizing

$$\min_{\Delta x} \left\| \left(b - f(x^{(k)}) \right) - J\Delta x \right\|^2 + \mu\|\Delta x\|^2$$

The normal equations for this problem end up with a matrix $J^T J + \mu I$, which shouldn't be a surprise. The parameter $\mu \geq 0$ should be adjusted up if the regular Gauss-Newton step is intractable.

The other path is to turn Gauss-Newton into a trust region method. This can be boiled down to the celebrated Levenberg-Marquardt algorithm, which unsurprisingly features almost the same linear system (if solved with the normal equations). I'll leave the details for you to look up if you end up facing nonlinear least squares problems down the road.