Nonlinear Methods

Stuart B. Heinrich
The MITRE Corporation
sheinrich@mitre.org

April 13, 2015

1 Problem

Given a function $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$ with a desired output of $\mathbf{y}$, the residual error vector is $\epsilon = f(\mathbf{x}) - \mathbf{y}$ and the squared error is $s(\mathbf{x}) = ||\epsilon||^2$.

2 Method of gradient descent

The gradient of $s(\mathbf{x})$, denoted $\nabla s(\mathbf{x})$, is the direction of steepest ascent (i.e., the direction vector in $\mathbb{R}^N$ that most rapidly decreases $s(\mathbf{x})$). The method of gradient descent is to take steps of some size α in the direction opposite the gradient,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \nabla s(\mathbf{x}). \tag{1}$$

In order to calculate$\nabla s(\mathbf{x})$, we first expand the squared error terms,

$$\begin{aligned}
s(\mathbf{x}) &= ||\epsilon||^2 && (2)\\
&= ||f(\mathbf{x}) - \mathbf{y}||^2 && (3)\\
&= (f(\mathbf{x}) - \mathbf{y})^\mathsf{T} (f(\mathbf{x}) - \mathbf{y}) && (4)\\
&= f(\mathbf{x})^\mathsf{T} f(\mathbf{x}) - f(\mathbf{x})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T} f(\mathbf{x}) + \mathbf{y}^\mathsf{T}\mathbf{y} && (5)\\
&= f(\mathbf{x})^\mathsf{T} f(\mathbf{x}) - 2f(\mathbf{x})^\mathsf{T}\mathbf{y} + \mathbf{y}^\mathsf{T}\mathbf{y}, && (6)
\end{aligned}$$

where we use the fact that $\mathbf{x}^\mathsf{T}\mathbf{y}$ is a scalar, equal to $\mathbf{y}^\mathsf{T}\mathbf{x}$.

Then, the gradient is computed by taking the partial derivative,

$$\begin{aligned}
\nabla s(\mathbf{x}) &= \frac{\partial s(\mathbf{x})}{\partial \mathbf{x}} && (7)\\
&= \frac{\partial}{\partial \mathbf{x}} \left(f(\mathbf{x})^\mathsf{T} f(\mathbf{x}) - 2f(\mathbf{x})^\mathsf{T}\mathbf{y} + \mathbf{y}^\mathsf{T}\mathbf{y} \right) && (8)\\
&= \frac{\partial f(\mathbf{x})^\mathsf{T}}{\partial \mathbf{x}} f(\mathbf{x}) + f(\mathbf{x})^\mathsf{T} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} - 2\frac{\partial f(\mathbf{x})^\mathsf{T}}{\partial \mathbf{x}}\mathbf{y} && (9)\\
&= 2\frac{\partial f(\mathbf{x})^\mathsf{T}}{\partial \mathbf{x}} f(\mathbf{x}) - 2\frac{\partial f(\mathbf{x})^\mathsf{T}}{\partial \mathbf{x}}\mathbf{y} && (10)\\
&= 2\frac{\partial f(\mathbf{x})^\mathsf{T}}{\partial \mathbf{x}} \left(f(\mathbf{x}) - \mathbf{y} \right) && (11)\\
&= 2\mathbf{J}^\mathsf{T}\epsilon, && (12)
\end{aligned}$$

where $\mathbf{J} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian of $f(\mathbf{x})$.

Using a very small constant value for α would provide the most reliable convergence to the nearest local minima, but would be extremely slow. A better method would be to apply a line-search in the direction of the gradient, or use a concept of momentum to adjust α after each successful reduction in error, and decrease it otherwise.

3 Newton's method

In one dimensional problems, Newton's method is an efficient routine for finding the roots of a function $f(x)$. It uses the derivative at the current point to estimate the location of the root (Fig. 1),

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{13}$$

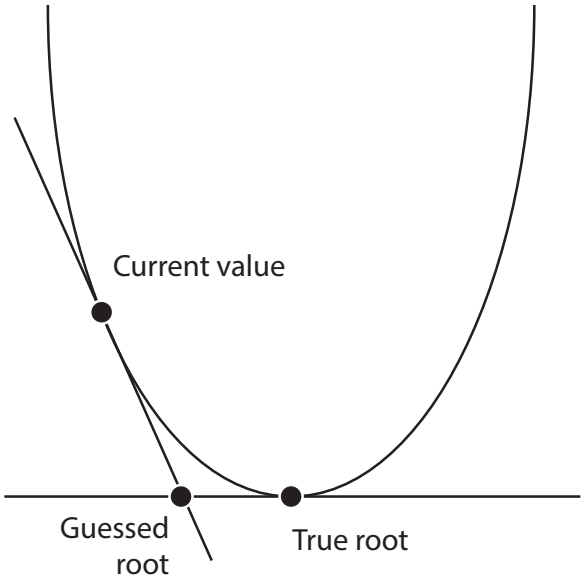

Figure 1. Newton's method for root finding

Note that the $f(x)$ in (13) is not comparable to the $f(\mathbf{x})$ discussed in Section 1, because $f(\mathbf{x})$ was defined as a transfer function, and the objective was to *minimize* $s(\mathbf{x})$, not to find roots.

With that said, at the minimum of any function, the derivative will be zero, so Newton's method can also be used to find the minimum of an arbitrary one-dimensional function by simply differentiating again. In this case, the geometrical interpretation is that it fits a local parabola to the function around the point, and moves to the mimum of the parabola (Fig. 2)

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} \tag{14}$$

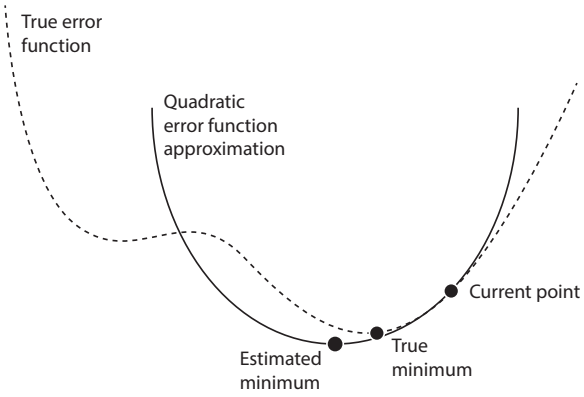

Figure 2. Newton's method for error minimization

Thus, in the one-dimensional case, (14) could be used to minimize $s(\mathbf{x})$, and it would be the equivalent of gradient descent with a very smart way of choosing α. Geometrically, α is chosen by finding where the locally approximated tangent line intersects zero. As opposed to gradient descent, this is a second-order method of optimization because it relies on second order derivatives, meaning it can find the optimum in fewer iterations.

Newton's method can be generalized to higher dimensional problems by replacing the derivative $f'(x_n)$ with the gradient $\nabla f(\mathbf{x}_n)$, and by replacing the reciprocal of the second derivative $1/f''(x_n)$ by $\left(\nabla^2 f(\mathbf{x}_n)\right)^{-1}$, the inverse of the matrix of second order partial derivatives of f, where $\nabla^2 f(\mathbf{x}_n) = \mathbf{H}_f$ is often called the Hessian matrix. This yields a multi-dimensional version of Newton's method,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \left(\nabla^2 f(\mathbf{x}_n)\right)^{-1} \nabla f(\mathbf{x}_n) \tag{15}$$

Rather than explicitly computing an inverse (which is inefficient and numerically unstable), (15) may be rearranged into a linear least squares problem,

$$\nabla^2 f(\mathbf{x}_n)(\mathbf{x}_{n+1} - \mathbf{x}_n) = -\nabla f(\mathbf{x}_n) \tag{16}$$

Dropping the subscripts and denoting $\delta = \mathbf{x}_{n+1} - \mathbf{x}_n$, this may be written as

$$\mathbf{H}_f \delta = -\nabla f \tag{17}$$

If we apply (17) to the minimization of $s(\mathbf{x})$ (squared error) from Section 2, then we can substitute the gradient from (12) to get

$$\mathbf{H}_s \delta = -2\mathbf{J}^\mathsf{T} \epsilon \tag{18}$$

The geometrical interpretation of this method is that the squared error function $s(\mathbf{x})$ is locally approximated as a quadratic surface,

$$s(\mathbf{x} + \delta) \approx s(\mathbf{x}) + \nabla s(\mathbf{x})^\mathsf{T} \delta + \frac{1}{2} \delta^\mathsf{T} \nabla^2 s(\mathbf{x}) \delta \tag{19}$$

and the solved update vector δ is the minimum of that quadratic surface approximation. This is one of the most efficient known method for minimzing a convex function (in terms of iteration count).

4 Gauss-Newton method

Unfortunately, it is usually not feasible or computationally practical to compute the Hessian, so (18) cannot be used. The Gauss-Newton method provides a more practical alternative by using a local first order approximation of $f(\mathbf{x})$,

$$f(\mathbf{x}+\delta) \approx f(\mathbf{x}) + \mathbf{J}\delta \tag{20}$$

Substituting (20) for $f(\mathbf{x})$ in $s(\mathbf{x})$, we obtain an estimate of the squared error at an offset point,

$$\begin{aligned} s(\mathbf{x}+\delta) &\approx ||f(\mathbf{x}) + \mathbf{J}\delta - \mathbf{y}||^2 && (21)\\ &= ||\epsilon + \mathbf{J}\delta||^2 && (22)\\ &= (\epsilon + \mathbf{J}\delta)^\mathsf{T}(\epsilon + \mathbf{J}\delta) && (23)\\ &= \epsilon^\mathsf{T}\epsilon + \epsilon^\mathsf{T}\mathbf{J}\delta + \delta^\mathsf{T}\mathbf{J}^\mathsf{T}\epsilon + \delta^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{J}\delta && (24)\\ &= \epsilon^\mathsf{T}\epsilon + 2\delta^\mathsf{T}\mathbf{J}^\mathsf{T}\epsilon + \delta^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{J}\delta && (25) \end{aligned}$$

This approximation is analog to (19) used in Newton's method. Noting that $s(\mathbf{x}) = \epsilon^\mathsf{T}\epsilon$ and $\nabla s(\mathbf{x}) = 2\mathbf{J}^\mathsf{T}\epsilon$, the only difference is that $2\mathbf{J}^\mathsf{T}\mathbf{J}$ is used in place of $\nabla^2 s(\mathbf{x})$.

At the minimum (or maximum) of $s(\mathbf{x}+\delta)$, all the partial derivatives will be zero. Therefore, to find the value of δ that will take us to the minimum, we need to solve

$$\begin{aligned} 0 &= \frac{\partial s(\mathbf{x}+\delta)}{\partial \delta} && (26)\\ &= \frac{\partial}{\partial \delta}\left(\epsilon^\mathsf{T}\epsilon + 2\delta^\mathsf{T}\mathbf{J}^\mathsf{T}\epsilon + \delta^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{J}\delta\right) && (27) \end{aligned}$$

for δ. Distributing the partial derivative, and applying the product rule to the last term, this expands to

$$0 = 2\left(\frac{\partial}{\partial \delta}\delta^\mathsf{T}\mathbf{J}^\mathsf{T}\epsilon\right) + \left(\frac{\partial}{\partial \delta}\delta^\mathsf{T}\right)(\mathbf{J}^\mathsf{T}\mathbf{J}\delta) + (\delta^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{J})\left(\frac{\partial}{\partial \delta}\delta\right) \tag{28}$$

It is known that

$$\frac{\partial}{\partial \mathbf{x}}\left(\mathbf{x}^\mathsf{T}\mathbf{A}\right) = \mathbf{A}^\mathsf{T}\frac{\partial}{\partial \mathbf{x}}\mathbf{x} \tag{29}$$

Using (29), (28) reduces to

$$\begin{aligned} 0 &= 2\epsilon^\mathsf{T}\mathbf{J} + \delta^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{J} + \delta^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{J} && (30)\\ &= 2\epsilon^\mathsf{T}\mathbf{J} + 2\delta^\mathsf{T}\mathbf{J}^\mathsf{T}\mathbf{J} && (31)\\ & && (32) \end{aligned}$$

Transposing and rearranging, we obtain the so-called 'normal equation,'

$$\begin{aligned} 2\mathbf{J}^\mathsf{T}\mathbf{J}\delta &= -2\mathbf{J}^\mathsf{T}\epsilon && (33)\\ \mathbf{J}^\mathsf{T}\mathbf{J}\delta &= -\mathbf{J}^\mathsf{T}\epsilon && (34) \end{aligned}$$

This is a linear least squares problem which can be solved for δ. Notice that this equation is identical to (18), except that the Hessian is replaced by $2\mathbf{J}^\mathsf{T}\mathbf{J}$. For this reason, $2\mathbf{J}^\mathsf{T}\mathbf{J}$ is known as the *Gauss-Newton approximation of the Hessian.*

Geometrically, the Gauss-Newton method uses a local tangent plane approximation of $f(\mathbf{x})$ which results in a local quadratic approximation of $s(\mathbf{x})$. It then moves to the point which minimizes this quadratic approximation. In contrast, Newton's method directly approximates the error function $s(\mathbf{x})$ with a quadratic surface using second order derivatives, and chooses the update as the minimum of that surface. Thus, both methods compute the update as the minimum of a quadratic surface, but the Gauss-Newton method is not a true second order method. However, it is a good approximation, and faster than most other first order methods.

5 Levenberg-Marquardt method

The update vector computed by the Gauss-Newton method is a very large step based on extrapolating the local gradient far away from the point which the original planar function approximation was made. As such, it is not guaranteed to result in improvement (and could easily move from one local minima into a different local minima). Generally, we do not care if we move to another local minima as long as the error is reduced. To this end, Levenberd modified (34) by adding a damping term multiplied by a scalar λ,

$$\left(\mathbf{J}^\mathsf{T}\mathbf{J} + \lambda \mathtt{I}\right)\delta = -\mathbf{J}^\mathsf{T}\epsilon \tag{35}$$

A big advantage of this method is that, in the case that the original system of equations is singular or near-singular, the damping parameter can produce a stable well-conditioned numerical solution.

In the limit as λ goes to zero, (35) becomes equivalent to the basic Gauss-Newton method. However, in the limit as λ goes to infinity, the $\mathbf{J}^\mathsf{T}\mathbf{J}$ term becomes overshadowed by $\lambda\mathtt{I}$, and is closely approximated by

$$\lim_{\lambda\to\infty} \lambda\delta = -\mathbf{J}^\mathsf{T}\epsilon \tag{36}$$

Rearranging, we find that $\lim_{\lambda\to\infty}\delta = -\frac{1}{\lambda}\mathbf{J}^\mathsf{T}\epsilon$, which is equivalent to (1) using a very small step size of $\alpha = 1/\lambda$. In other words, by varying λ in (35), one smoothly transitions between the slow but reliable gradient descent update, and the fast but unreliable Gauss-Newton update. With this in mind, the idea behind Levenberg's method is to dynamically adjust λ over time – whenever an update successfully reduces the error, λ is

decreased (making the algorithm more aggressive), and whenever an update fails to reduce the error, λ is increased, making the algorithm more tentative and taking steps more similar to the gradient direction.

Marquardt provided the insight that each component of the gradient can be scaled according to the curvature so that there is larger movement along directions where the gradient is smaller, in order to avoid slow convergence in the directions which have smaller gradients. To this end, Marquardt replaced the damping matrix from (35) with the diagonal of $\mathbf{J}^\mathsf{T}\mathbf{J}$, yielding the popular Levenberg-Marquardt update equation,

$$\left(\mathbf{J}^\mathsf{T}\mathbf{J} + \lambda \operatorname{diag}(\mathbf{J}^\mathsf{T}\mathbf{J})\right)\delta = -\mathbf{J}^\mathsf{T}\epsilon \tag{37}$$

6 Trust region methods

A limitation of (35) or (37) is that if λ is too small, the matrix equations must be augmented and re-solved, which can be very expensive. An alternative is to recognize that the choice of λ effectively chooses a radius, and that if we can explicitly keep track of the radius within which the approximation is valid, and only solve for updates within the trust radius, then we can reduce the number of failed iterations.

In addition, by explicitly managing a trust radius, we are more likely to stay in the same basin of attraction that we started with than a highly aggressive method that pays no need to the step size.

Let $m(\delta)$ be the linearized approximation of $s(\mathbf{x}+\delta)$ at $\mathbf{x}$. Then given a trust radius Δ, the trust-radius constrained update is that which minimizes the error function constrained within the trust region,

$$\operatorname{argmin}_\delta m(\delta) \quad \text{subject to} \quad ||\delta|| \leq \Delta \tag{38}$$

Consider the solutoin of (34), which we denote as δ_{GN}. If $||\delta_{GN}|| < \Delta$, then δ_{GN} is the solution of (38). Otherwise, the solution must be obtained by solving the damped equation in (35) for some *unknown* value of λ. Unfortunately (35) is too expensive to re-solve iteratively in order to find the optimal λ.

6.1 Powell's dog-leg approximation

Consider the solution of (35) as a function of λ, written $\delta(\lambda)$. When $\lambda = 0$ the solution is given by δ_{GN}, and in the limit as $\lambda \to \infty$, the solution will be in the negative direction of the gradient. Thus, because the curve is dominated by the gradient direction in one part, and the direction of the Gauss-Newton solution in the other, this curve can be fairly well approximated as a piecewise linear curve with two segments, or "dog leg" (Fig. 3).

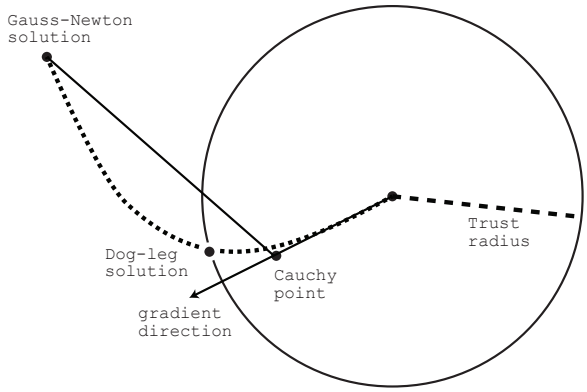

Figure 3. Example dog-leg intersected with the turst radius

The first linear segment connects $\mathbf{x}$ to the point in the direction of the gradient that minimizes $m(\delta)$, called the Cauchy point, given by

$$\delta_C = -\frac{\mathbf{g}^\mathsf{T}\mathbf{g}}{\mathbf{g}^\mathsf{T}\mathbf{B}\mathbf{g}}\mathbf{g}, \tag{39}$$

where $\mathbf{g} = \mathbf{J}^\mathsf{T}\epsilon$ is (half) the gradient, and $\mathbf{B} = \mathbf{J}^\mathsf{T}\mathbf{J}$ is (half) the Gauss-Newton approximation of the Hessian.

The second leg connects δ_C to δ_{GN}, the unconstrained solution to the Gauss-Newton equations. Thus, the dog-leg curve can be written piecewise as

$$\delta(\tau) = \begin{cases} \tau\delta_C & 0 \leq \tau \leq 1 \\ \delta_C + (\tau-1)(\delta_{GN} - \delta_C) & 1 \leq \tau \leq 2 \end{cases}. \tag{40}$$

It is easy to compute the intersection of the dog-leg curve with the trust region, and the update vector δ at this point can be computed as a simple linear interpolation between δ_C and δ_{GN}.

If $||\delta_C|| > \Delta$, then the trust radius is intersected by the first gradient segment, and the solution of (38) is given by

$$\delta = \frac{\delta_C}{||\delta_C||}\Delta. \tag{41}$$

Otherwise, we must find the intersection between the sphere of the trust region

$$||\delta - \mathbf{x}||^2 = \Delta, \tag{42}$$

and the second line segment of the dog-leg,

$$\delta = \delta_C + d\frac{\delta_{GN} - \delta_C}{||\delta_{GN} - \delta_C||}. \tag{43}$$

In this case, the solution is given by the positive root of a quadratic equation,

$$d = -e + \sqrt{e^2 - ||\delta_{GN} - \mathbf{x}||^2 + \Delta^2}, \tag{44}$$

using

$$e = \left(\frac{\delta_{GN} - \delta_C}{||\delta_{GN} - \delta_C||} \right)^{\mathsf{T}} (\delta_C - \mathbf{x}). \tag{45}$$

Finally, the δ that minimizes (38) is given by substituting (44) back into (43).

To summarize, the dog-leg update is given by one of three cases:

1. First, solve (34) for δ_{GN}. If $||\delta_{GN}|| < \Delta$, then $\delta = \delta_{GN}$ is the update.
2. Otherwise, compute the Cauchy point δ_C. If $\delta_C > \Delta$, then the update is given by (41).
3. Otherwise, the solution is given by substituting (44) into (43).

6.2 Updating the trust region

After solving for the dogleg update, the trust region size is increased or decreased adaptively based on a measure of how good the approximation is. We can use the following ratio as a heuristic for accuracy of the current quadratic approximation,

$$\rho = \frac{s(\mathbf{x}) - s(\mathbf{x} + \delta)}{m(0) - m(\delta)} \tag{46}$$

If the approximation was perfectly accurate, then $\rho = 1$. If $\rho > 1$ then the error was reduced more than expected by the quadratic model, indicating that the model is working well and the trust region can be expanded. In contrast, if $\rho < 1$ then the model predicted a larger decrease in error than was actually observed, indicating that the trust region may need to be reduced.

Based on these principles, a common heuristic update scheme is

$$\Delta_{k+1} = \begin{cases} \Delta_k/4 & \rho_k < 1/4 \\ \min(2\Delta_k, \Delta_{max}) & \rho_k > 3/4 \text{ and } ||\mathbf{p}|| = \Delta_k \\ \Delta_k & \text{otherwise} \end{cases} \tag{47}$$

6.3 Initialization and convergence

A common rule of thumb is to initialize $\Delta_0 = ||\mathbf{x}_0||$. We must also choose values for Δ_{max}. Convergence may be detected when the reduction in error is below some threshold, $f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{p}_k) < \epsilon$.

6.4 Summary

A limitation of Dog-Leg is that if the linear system in (34) is singular or poorly conditioned, it may not be possible to compute a good update step. This means that minimal parameterizations must be used to avoid singularity. Also, it is recommended to use a method such as perturbed cholesky when solving (34) in order to help avoid singular conditions. Another limitation is that the radius Δ does not scale with the different dimensions (as in the case of Levenberg-Marquardt), which can lead to slow convergence in the direction of small gradient directions.