Levenberg-Marquardt algorithms
vs
Trust Region algorithms

Frank Vanden Berghen
IRIDIA, Université Libre de Bruxelles
fvandenb@iridia.ulb.ac.be

November 12, 2004

For an in-depth explanation and more references about this subject, see my thesis, section 2.1, available at: **http://iridia.ulb.ac.be/~fvandenb/mythesis/index.html**

Let's assume that we want to find x^*, the minimum of the objective function $\mathcal{F}(x) : \Re^n \rightarrow \Re$.

Let us write the Taylor development limited to the degree 2 of $\mathcal{F}$ around x_k:

$$\mathcal{F}(x_k + \delta) \approx \mathcal{Q}_k(\delta) = \mathcal{F}(x_k) + g_k^t\delta + \frac{1}{2}\delta^t B_k \delta$$

with $\mathcal{Q}_k(\delta)$, the quadratical approximation of $\mathcal{F}(x)$ around x_k.
g_k, the gradient of $\mathcal{F}(x)$ computed at x_k.
B_k, an approximation of the real hessian matrix H_k of $\mathcal{F}(x)$ at x_k.
H_k, the real hessian matrix of $\mathcal{F}(x)$ at x_k.

For the moment, we will assume that $B_k := H_k$.

The unconstrained minimum δ_k of $\mathcal{Q}(\delta)$ is:

$$\begin{aligned} \nabla\mathcal{Q}(\delta_k) = g_k + B_k\delta_k &= 0 \\ \Longleftrightarrow B_k\delta_k &= -g_k \end{aligned} \tag{1}$$

Equation 1 is called the equation of the *Newton Step* δ_k.

So, the Newton's method to find the minimum x^* of $\mathcal{F}(x)$ is:

1. Set $k = 0$. Set $x_0 = x_{start}$.
2. solve $B_k\delta_k = -g_k$ (go to the minimum of the current quadratical approximation of $\mathcal{F}$).
3. set $x_{k+1} = x_k + \delta_k$
4. Increment k. Stop if $g_k \approx 0$ otherwise, go to step 2.

Newton's method is VERY fast: when x_k is close to x^* (when we are near the optimum) this method has quadratical convergence speed:

$$\|x_{k+1} - x^*\| < \epsilon \|x_k - x^*\|^2$$

with $\epsilon < 1$. Unfortunately, it does NOT always converge to the minimum x^* of $\mathcal{F}(x)$. To have convergence, we need to be sure that B_k is always positive definite, ie. that the curvature of $\mathcal{F}(x)$ is always positive.

PROOF: B_k must be positive definite to have convergence.

We want the search direction δ_k to be a descent direction

$$\Longrightarrow \delta^T g < 0 \tag{2}$$

Taking the value of g from (1) and putting it in (2), we have:

$$\begin{aligned} & -\delta^T B\delta < 0 \\ \Leftrightarrow \quad & \delta^T B\delta > 0 \end{aligned} \tag{3}$$

The Equation (3) says that B_k must always be positive definite.

END OF PROOF.

So, we must always construct the B_k matrix so that it is a positive definite approximation of H_k, the real Hessian matrix of $\mathcal{F}(x)$. The Newton's Algorithm cannot use negative curvature (when H_k negative definite) inside $\mathcal{F}(x)$. See figure 1 for an illustration about positive/negative curvature.

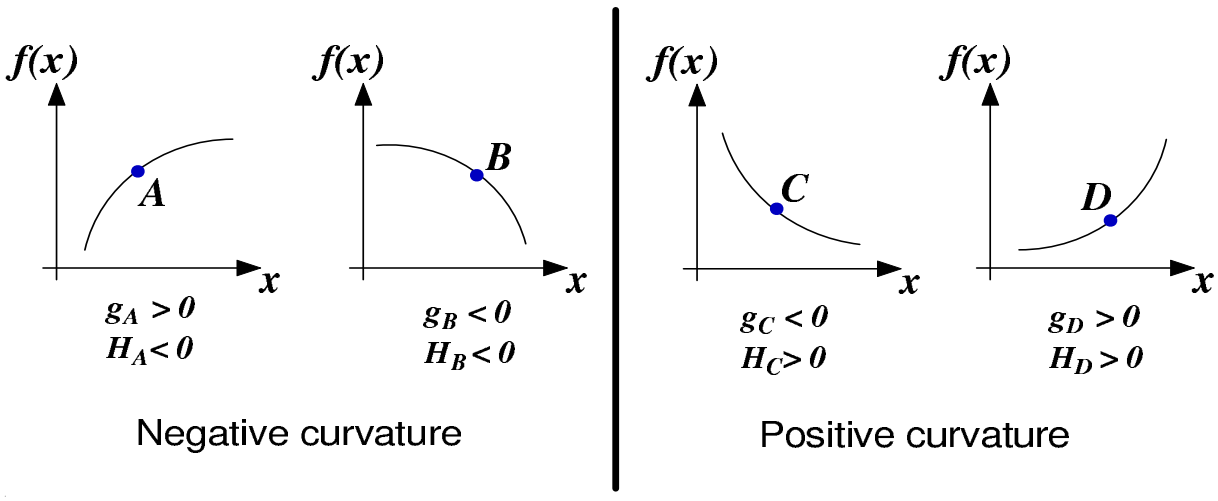

Figure 1: positive/negative curvature of a function $f(x) : \Re \rightarrow \Re$

One possibility to solve this problem is to take $B_k = I$ (I=identity matrix), which is a very bad approximation of the Hessian H but which is always positive definite. We will simply have $\delta_k = -g_k$. We will simply follow the slope. This algorithm is called the "steepest descent algorithm". It is very slow. It has linear speed of convergence: $\|x_{k+1} - x^*\| < \epsilon \|x_k - x^*\|$ with

$\epsilon < 1$ (problem is when $\epsilon = 0.99$).

Another possibility, if we don't have H_k positive definite, is to use instead $B_{new,k} = H_k + \lambda I$ with λ being a very big number, such that $B_{new,k}$ is positive definite. Then we solve, as usual, the Newton Step equation (see equation (1)):

$$B_{new,k}\ \delta_k = -g_k \quad \Leftrightarrow \quad (H_k + \lambda I)\delta_k = -g_k \tag{4}$$

Choosing a high value for λ has 2 effects:

1. H_k (inside equation $B_{new,k} = H_k + \lambda I$) will become negligible and we will find, as search direction, "the steepest descent step".

2. The step size $\|\delta_k\|$ is reduced.

In reality, only the above second point is important. It can be proven that, if we impose a proper limitation on the step size $\|\delta_k\| < \Delta_k$, we maintain global convergence even if B_k is an indefinite matrix. Trust region algorithms are based on this principle (Δ_k is called the trust region radius). In trust region algorithm the steps δ_k are:

$$\delta_k \text{ is the solution of } \mathcal{Q}(\delta_k) = \min_{\delta} Q(\delta) \text{ subject to } \|\delta\| < \Delta_k \tag{5}$$

The old Levenberg-Marquardt algorithm uses a technique which adapts the value of λ during the optimization. If the iteration was successful ($\mathcal{F}(x_k + \delta_k) < \mathcal{F}(\delta_k)$), we decrease λ to exploit more the curvature information contained inside H_k. If the previous iteration was unsuccessful ($\mathcal{F}(x_k + \delta_k) > \mathcal{F}(\delta_k)$), the quadratic model don't fit properly the real function. We must then only use the "basic" gradient information. We will increase λ in order to follow closely the gradient ("steepest descent algorithm"). This old algorithm is the base for the explanation of the update of the trust region radius Δ_k in Trust Region Algorithms.

For intermediate value of λ, we will thus follow a direction which is a mixture of the "steepest descent step" and the "Newton Step". This direction is based on a perturbated Hessian matrix $B_{new,k}$ and can sometime be disastrous (There is no geometrical meaning of the perturbation λI on H_k).

When a negative curvature is encountered (H_k negative definite):

- Newton's Method fail.
- Levenberg-Marquardt algorithms are following a perturbated and approximative direction of research δ_k based on an arbitrary perturbation of H_k (δ_k is the solution of equation (4): $(H_k + \lambda I)\delta_k = -g_k$).
- Trust region algorithms will perform a long step ($\|\delta_k\| = \Delta_k$) and "move" quickly to a more interesting area (see equation (5))

Trust Region algorithm will thus exhibit better performances each time a negative curvature is encountered and have thus better performances than all the Levenberg-Marquardt algorithms. Unfortunately, the computation of δ_k for Trust Region algorithm involves a constrained minimization of a quadratic subject to one non-linear constraint (see equation (5)). This is not a trivial problem to solve at all. The algorithmic complexity of Trust region algorithms is much higher. This explains why they are not very often encountered despite their better performances.

The solution of equation (5) can be computed very efficiently using the fast algorithm from Moré and Sorensen.

There is one last point which must still be taken into account: How can we obtain H_k? Usually, we don't have the analytical expression of H_k. H_k must thus be approximated numerically. H_k is usually constructed iteratively based on information gathered from old evaluations $\mathcal{F}(x_j)$ for $j = 1, \ldots, k$. Iterative construction of H_k can be based on:

- The well-known BFGS formulae:
  Each update is fast to compute but we get poor approximation of H_k.
- Multivariate polynomial interpolation:
  Each update is very time consuming. We get very precise H_k.
- Finite difference approximation:
  Very poor quality: numerical instability occurs very often.

To summarize:

- Levenberg-Marquardt algorithms and Trust region algorithms are both Newton Step-based methods (they are called "Restricted Newton Step methods"). Thus they both exhibits quadratical speed of convergence near x^*.
- When we are far from the solution (x_k far from x^*), we can encounter a negative curvature (H_k negative definite). If this happens, Levenberg-Marquardt algorithms will slow down dramatically. In opposition, Trust Region Methods will perform a very long step δ_k and "move" quickly to a more interesting area.

Old Levenberg-Marquardt algorithms were updating iteratively H_k only on iterations k where a good value for λ has been found (This is because on old computers the update of H_k is very time consuming, so we want to avoid it). Modern Levenberg-Marquardt algorithms are updating iteratively H_k at every iterations k but they are still enable to follow a negative curvature inside the function $\mathcal{F}(x)$. The steps δ_k remains thus of poor quality compared to trust region algorithms.

To summarize again: Trust Region Methods are an evolution of the Levenberg-Marquardt algorithms. Trust Region Methods are able to follow the negative curvature of the objective function. Levenberg-Marquardt algorithms are NOT able to do so and are thus slower.