Trust Region Methods

Lecture 6, Continuous Optimisation
Oxford University Computing Laboratory, HT 2006
Notes by Dr Raphael Hauser (hauser@comlab.ox.ac.uk)

Trust region methods constitute a second fundamental class of algorithms.

- In iteration k, replace $f(x)$ by a locally valid quadratic model function $m_k(x)$ (recall that we already encountered this idea in the context of quasi-Newton methods).

- Choose a neighbourhood R_k of the current iterate x_k in which $m_k(x)$ can be trusted to approximate f well (we do not care about how well m_k approximates f outside R_k).

- The next iterate x_{k+1} is found by approximately minimising the model function over the trust region,

$$x_{k+1} \approx \arg\min_{x \in R_k} m_k(x).$$

All unconstrained optimisation methods we discussed so far in this course are based on line-searches

$$\min_{\alpha > 0} f(x_k + \alpha d_k),$$

where d_k is a descent direction.

In each iteration one replaces the n-dimensional minimisation problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

by a simpler one-dimensional minimisation problem.

Note: we replace the *unconstrained* optimisation problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

by the *constrained trust region subproblem* (to be approximately solved)

$$x_{k+1} \approx \arg\min_{x \in R_k} m_k(x). \tag{1}$$

This is worthwhile because (1) can be solved cheaply when

$$m_k(x) = f(x_k) + \nabla f(x_k)^{\mathsf{T}}(x - x_k) + \frac{1}{2}(x - x_k)^{\mathsf{T}} B_k (x - x_k) \tag{2}$$

is a quadratic function, see Lecture 7.

The linear part of $m_k(x)$ coincides with the first order Taylor approximation of $f(x)$.

$m_k(x)$ will closely match the second order Taylor approximation of $f(x)$ when $B_k \approx D^2 f(x_k)$.

To make the method work, we will thus have to worry about how to update B_k cheaply.

But note that the quasi-Newton Hessian approximations discussed in Lecture 5 are perfect for this job!

Accepting and Rejecting Updates:

Let y_{k+1} be the approximate minimiser of the trust region subproblem.

In principle, this is the point we would like to select as our next iterate x_{k+1}.

However, y_{k+1} is computed on the basis of the model function m_k, and it could happen that moving to y_{k+1} leads to an increase rather than decrease in of the *true* objective function f.

Trust-region methods therefore accept y_{k+1} only if the decrease achieved in f is at least a fixed proportion of the decrease "promised" by m_k,

$$x_{k+1} = \begin{cases} y_{k+1} \text{ if } \frac{f(x_k)-f(y_{k+1})}{m_k(x_k)-m_k(y_{k+1})} > \eta, \\ x_k \text{ otherwise,} \end{cases} \tag{3}$$

where $\eta \in (0, 1/4)$ is fixed.

Note that rejecting the update does not imply that the algorithm will stall, because we can still shrink the trust region so that $y_{k+2} \neq y_{k+1}$.

Updating the Trust Region:

The easiest way to define a trust region R_k is to choose the closed ball of radius Δ_k around x_k in some norm $\|\cdot\|$,

$$R_k = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}.$$

For simplicity, we will assume that $\|\cdot\|$ is the Euclidean norm. Δ_k is called the *trust region radius*.

In order to define a new trust region R_{k+1} around x_{k+1}, it suffices to fix a rule on how to select Δ_{k+1}.

The following rule is a popular choice, where y_{k+1} is as above:

$$\Delta_{k+1} = \begin{cases} \frac{\Delta_k}{4} \text{ if } \frac{f(x_k)-f(y_{k+1})}{m_k(x_k)-m_k(y_{k+1})} < \frac{1}{4}, \\ \min(2\Delta_k, \Delta_{\max}) \text{ if } \frac{f(x_k)-f(y_{k+1})}{m_k(x_k)-m_k(y_{k+1})} > \frac{3}{4}, \\ \Delta_k \text{ otherwise.} \end{cases} \tag{4}$$

Algorithm 1: Generic Trust region Method. Choose $\Delta_{\max} > 0$, $\Delta_0 \in (0, \Delta_{\max})$, $\eta \in (0, 1/4)$, $x_0 \in \mathbb{R}^n$, B_0, $\epsilon > 0$.

While $\|\nabla f(x_k)\| \geq \epsilon$ repeat

Compute y_{k+1} as the approximate minimiser of (1).

Determine x_{k+1} via (3).

Compute Δ_{k+1} using (4).

Build a new model function $m_{k+1}(x)$.

$k \leftarrow k + 1$.

end

- Δ_k never exceeds $\Delta_{\max}$.

- If the actual decrease $f(x_k) - f(y_{k+1})$ was below our expectations $m_k(x_k) - m_k(y_{k+1})$, this indicates that m_k should be regarded as a more local model than before. We thus find a reasonable Δ_{k+1} by shrinking Δ_k.

- If the actual decrease was above our expectations, we feel confident to expand the trust region by selecting Δ_{k+1} as an expansion of Δ_k.

- If there is neither reason for gloom nor euphoria, we stick to the previous value $\Delta_{k+1} = \Delta_k$.

The Cauchy Point:

In step **S1** of the algorithm, the approximate minimiser y_{k+1} can be computed in many different ways. Some of these methods will be discussed in Lecture 7.

To derive a convergence result for Algorithm 1, we need to assume that the method chosen for computing y_{k+1} compares favourably to a specific benchmark.

The *Cauchy point* is obtained when a steepest descent line-search is applied to m_k at x_k and is restricted to R_k.

An unrestricted line-search in the direction $-\nabla f(x_k)$ yields the step-length multiplier

$$\begin{aligned}\alpha_k^u &:= \arg\min_{\alpha\geq 0} m_k(x_k - \alpha\nabla f(x_k))\\ &= \arg\min_{\alpha\geq 0} f(x_k) - \alpha\nabla f(x_k)^{\mathsf{T}}\nabla f(x_k) + \frac{\alpha^2}{2}\nabla f(x_k)^{\mathsf{T}} B_k \nabla f(x_k)\\ &= \begin{cases} +\infty \text{ if } \nabla f(x_k)^{\mathsf{T}} B_k \nabla f(x_k) \leq 0,\\ \frac{\nabla f(x_k)^{\mathsf{T}}\nabla f(x_k)}{\nabla f(x_k)^{\mathsf{T}} B_k \nabla f(x_k)} \text{ otherwise.}\end{cases}\end{aligned}$$

If we want to stay within R_k we have to "clip" α_k^u to a constrained step-length multiplier α_k^c.

Note that $\alpha \mapsto m_k(x_k - \alpha\nabla f(x_k))$ is strictly decreasing on $[0, \alpha_k^u)$.

Moreover, the radius $\|x_k - \alpha\nabla f(x_k)\|$ is strictly increasing over the same interval.

Therefore, the correct clipping rule is given by

$$\alpha_k^c = \min\left(\frac{\Delta_k}{\|\nabla f(x_k)\|}, \alpha_k^u\right) \tag{5}$$

and the Cauchy point is

$$y_k^c := x_k - \alpha_k^c \nabla f(x_k).$$

Theorem 1: Global Convergence of Algorithm 1. Let Algorithm 1 be applied to the minimisation of $f \in C^2(\mathbb{R}^n, \mathbb{R})$, and for all k let y_{k+1} be computed such that $m_k(y_{k+1}) \leq m_k(y_k^c)$ holds.

Let there exist $\beta > 0$ such that for all k, $\|B_k\|, \|D^2 f(x_k)\| \leq \beta$, and finally, let $\Delta_0 \geq \epsilon/(14\beta)$.

Then exactly one of two following alternatives occurs:

(i) The algorithm does not terminate, but $\lim_{k\to\infty} f(x_k) = -\infty$ and f is unbounded below.

(ii) The algorithm terminates in finite time, returning an approximate minimiser.

Proof: If $\|\nabla f(x_k)\| < \epsilon$ occurs for some $k \in \mathbb{N}$ then (ii) occurs.

We may therefore assume that $\|\nabla f(x_k)\| \geq \epsilon$ for all k and need to show that $f(x_k) \to -\infty$.

The following claims will be proven in the notes and exercises.

C1: The update is accepted, i.e., $x_{k+1} = y_{k+1}$ in (3), for infinitely many k.

C2: Whenever $x_{k+1} = y_{k+1}$ occurs, we have

$$f(x_{k+1}) - f(x_k) \leq -\eta\epsilon^2/(28\beta).$$

Furthermore, the updating rule (3) guarantees that

$$f(x_{k+1}) - f(x_k) \le 0$$

for all k.

Therefore, Claim 1 and 2 imply

$$\lim_{k\to\infty} f(x_k) = \sum_{k=0}^{\infty} f(x_{k+1}) - f(x_k) = -\infty. \quad \square$$

Lemma 2: There are at most $\lfloor \log_4 \frac{\Delta_{\max} 7\beta}{2\epsilon} \rfloor$ rejected updates between successive accepted updates.

Proof: Suppose to the contrary that all updates y_{k+1} for $k = k_0, k_0 + 1, \ldots, k_0 + \lceil \log_4 \frac{\Delta_{\max} 7\beta}{2\epsilon} \rceil =: k_1$ are rejected.

Then

$$\Delta_{k_1} = \Delta_{k_0} 4^{-(k_1 - k_0)} \le \frac{2\epsilon}{7\beta},$$

By Lemma 1 y_{k_1+1} is not rejected, contradicting the above assumption. $\square$

Claim 1 is an immediate consequence of Lemma 2.

Lemma 1: Let $\|\nabla f(x_k)\| \ge \epsilon$ and $\Delta_k < 2\epsilon/(7\beta)$. Then

$$\frac{f(y_{k+1}) - f(x_k)}{m_k(y_{k+1}) - m_k(x_k)} > \frac{1}{4}.$$

Proof: See Lecture Note 6. $\square$

Reading Assignment: Lecture-Note 6.