Notes on Optimization Algorithms

March 7, 2018

Nelder-Mead Simplex Algorithm

The algorithm iterates on the following steps:[1]

1. Create a simplex of dimension $K+1$ (where there are K parameters to be estimated) around the starting value x_0.
2. Modify the simplex based on how the function looks at different trial modifications. In particular, "order the points in the simplex from lowest function value $f(x(1))$ to highest $f(x(K+1))$. At each step in the iteration, the algorithm discards the current worst point $x(K+1)$, and accepts another point into the simplex" (Matlab website). Possible simplex modifications are:
   (a) Reflect
   (b) Expand
   (c) Contract outside
   (d) Contract inside
   (e) Shrink
3. Repeat steps 1-2 until the diameter of the simplex is smaller than the tolerance (default is 10^{-4})

BFGS Medium-Scale algorithm: line search — No user-supplied gradient or hessian

The medium-scale algorithm iterates on the following four steps for the kth iteration:

1. compute the search direction p_k by solving

$$\begin{aligned} \hat{H}_k p_k &= -\nabla f(x_k), \text{ or} \\ p_k &= -\hat{H}_k^{-1} \nabla f(x_k) \end{aligned}$$

   where $\hat{H}_k$ is an approximation of the hessian.

[1] For more details, see `http://www.mathworks.com/help/techdoc/math/bsotu2d.html#bsgpq6p-11`

2. Choose the step size α_k by solving

$$\min_{\alpha} f(x_k + \alpha_k p_k)$$

3. Update $x_{k+1} = x_k + \alpha_k p_k$

4. Update $\hat{H}_{k+1}$, letting $s_k = \alpha_k p_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$:

$$\hat{H}_{k+1} = \hat{H}_k + \frac{y_k y_k'}{y_k' s_k} - \frac{\hat{H}_k s_k s_k' \hat{H}_k}{s_k' \hat{H}_k s_k}$$

   where $'$ refers to matrix transpose, not differentiation.

5. Continue steps 1-4 until $\|\nabla f(x_k)\| <$ tolerance, where tolerance $= 10^{-6}$ by default.

BFGS Large-Scale algorithm: trust region method — user-supplied gradient and/or hessian

The trust region method operates by dividing the optimization into a series of 2-dimensional trust region subproblems. "The basic idea is to approximate f with a simpler function q, which reasonably reflects the behavior of function f in a neighborhood N around the point x. This neighborhood is the **trust region**" (Matlab website, emphasis added). The algorithm iterates on the following steps:

1. Divide the problem into a series of 2-dimensional trust-region subproblems

2. Find the optimal step size s by solving the following equation:

$$\min_{s} \frac{1}{2} s' H s + s' g \text{ subject to } \|Ds\| < \Delta$$

   where H is the hessian, g is the gradient, D is a diagonal scaling matrix, and Δ is a positive scalar. $\|\cdot\|$ is the 2-norm.

3. If $f(x+s) < f(x)$ then $x_{k+1} = x_k + s$. If not, then adjust Δ.

4. Continue steps 1-3 until $\|\nabla f(x_k)\| <$ tolerance, or $f(x_{k+1}) - f(x_k) <$ tolerance, or $s <$ tolerance(each 10^{-6} by default).