Steepest Descent, Quasi-Newton methods

Niclas Börlin

5DA001 Non-linear Optimization

The Newton method and the Hessian

- The Newton method requires that the Hessian has to be
  - derived and implemented (takes time, may introduce blunders),
  - calculated (requires $\mathcal{O}(n^2)$ operations),
  - stored (requires storage for $n^2/2$ elements), and
  - "inverted" (requires $\mathcal{O}(n^3)$ operations).

Hessian approximations

- However, provided we use a global stratey, we may replace the Hessian in the Newton equation
$$\nabla^2 f(x_k)p = -\nabla f(x_k)$$
with *any* positive definite matrix B_k, i.e.
$$B_k p = -\nabla f(x_k).$$
and still get global convergence.
- Some methods calculate Hessian approximations based on first- or second-order derivatives, e.g.

Gauss-Newton	$B_k = J(x_k)^T J(x_k),$
Trust-region	$B_k = \nabla^2 f(x_k) + \lambda I,$
Levenberg-Marquardt	$B_k = J(x_k)^T J(x_k) + \lambda I.$

- Other methods calculate their Hessian approximations without any derivative information. Two such methods are called Steepest Descent and Quasi-Newton.

Steepest Descent

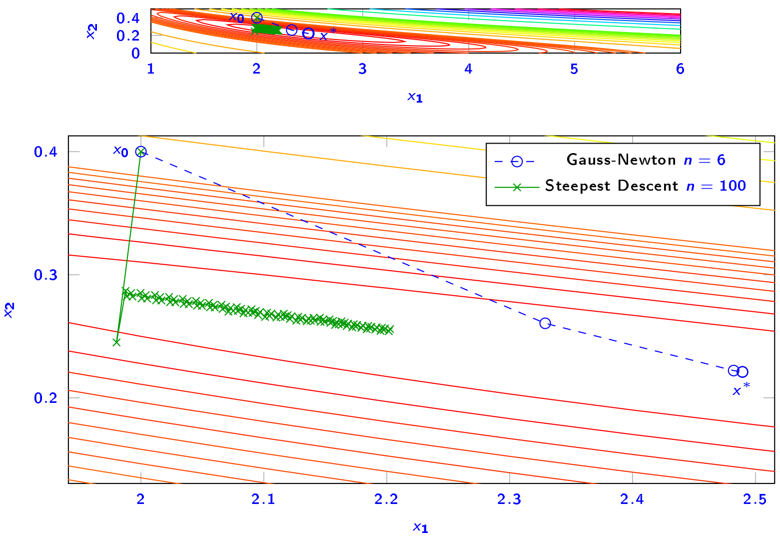
- The steepest descent method uses the simplest Hessian approximation
$$B_k = I \quad \Rightarrow \quad p_k = -\nabla f(x_k).$$
- The "calculation" of the search direction is cheap.
- However, the convergence rate is linear with a convergence constant C bounded from above by
$$C_{sup} = \left(\frac{\kappa(Q)-1}{\kappa(Q)+1}\right)^2,$$
where $Q = \nabla^2 f(x^*)$ is the true Hessian and $\kappa(Q)$ is the condition number of Q.
- The convergence rate will be poor even for moderate condition numbers.

$\kappa(Q)$	1	10	100	1000	10000
C	0	0.6694	0.9608	0.9960	0.9996

- **The steepest-descent method should only be used for problems that are known to be well-conditioned.**

Example

- Gauss-Newton and the Steepest-descent method with linesearch $\mu = 0.1$ for the antelope problem ($\kappa(Q) \approx 500$):



Quasi-Newton methods

- Quasi-Newton methods use a sequence of symmetric positive definite matrices that approximate the Hessian (or the inverse Hessian).
- For every iteration, the next (inverse) Hessian approximation is calculated by updating the current approximation.
- Each update is constructed to include the curvature information computed in the last step, i.e. the next (inverse) Hessian should behave as the true (inverse) Hessian over the last step.
- This condition is called the Secant Condition.

The Secant Condition

- The one-dimensional Secant method uses the approximation

$$\begin{aligned} f''(x_k) &\approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}, \\ f''(x_k)(x_k - x_{k-1}) &\approx f'(x_k) - f'(x_{k-1}). \end{aligned}$$

- In multiple dimensions this corresponds to

$$\nabla^2 f(x_k)(x_k - x_{k-1}) \approx \nabla f(x_k) - \nabla f(x_{k-1}).$$

- From this we obtain the Secant equation

$$B_k(x_k - x_{k-1}) = \nabla f(x_k) - \nabla f(x_{k-1}).$$

The Secant Condition

Cont'd

- With substitutions

$$s_k = x_{k+1} - x_k, y_k = \nabla f(x_{k+1}) - \nabla f(x_k),$$

we obtain

$$B_{k+1} s_k = y_k.$$

- If $H_k = B_k^{-1}$, the secant equation becomes

$$H_{k+1} y_k = s_k.$$

- The Hessian approximation B_k has $n^2/2$ independent elements, but the secant equation has only n equations.
- Thus, several updating schemes are possible.

Properties of Quasi-Newton methods

- An example of a Quasi-Newton update formula is

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$$

- This formula illustrates several key features with quasi-Newton approximations.
  - The new approximation B_{k+1} is found by updating the old approximation B_k.
  - As a starting approximation $B_0 = I$ may be used, but if a better approximation is available at a small cost, it should be used.

Properties of Quasi-Newton methods
Cont'd

-
  - The secant condition will be satisfied indepedently of how B_k is chosen:

$$\begin{aligned} B_{k+1}s_k &= B_k s_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k} s_k \\ &= B_k s_k + \frac{(y_k - B_k s_k)((y_k - B_k s_k)^T s_k)}{(y_k - B_k s_k)^T s_k} \\ &= B_k s_k + (y_k - B_k s_k) = y_k. \end{aligned}$$

  - The new approximation B_{k+1} can be obtained using $\mathcal{O}(n^2)$ operations since the update only involves vector products.

Properties of Quasi-Newton methods
Cont'd

- The number of operations needed to calculate the search direction depends on the choice of update formula.
  - If a B_k update formula is used, the solution of the secant equation needs $\mathcal{O}(n^3)$ operations.
  - Update formulas for the cholesky factors $LL^T = B_k$ exist that reduce the number of operations to $\mathcal{O}(n^2)$.
  - Another $\mathcal{O}(n^2)$ solution is if we use an update formula for H_k. The search direction may then be calculated from

$$p_k = H_k(-\nabla f_k).$$

- Thus, the total time complexity of one iteration of a Quasi-Newton method can be reduced to $\mathcal{O}(n^2)$ compared to $\mathcal{O}(n^3)$ for e.g. Newton and Trust-region methods.

Common Quasi-Newton methods
BGFS

- The most widely used Quasi-Newton formula is known as BFGS (Broyden-Fletcher-Goldfarb-Shanno):

$$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}.$$

- BFGS preserves symmetry and positive definiteness if $y_k^T s_k > 0$, which may be satisfied with a line search using the Wolfe condition.
- The corresponding update formula for H_k is

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T,$$

where $\rho_k = 1/(y_k^T s_k)$.

Common Quasi-Newton methods
DFP

- One of the first Quasi-Newton formulas was DFP (Davidon-Fletcher-Powell):

$$B_{k+1} = (I - \rho_k y_k s_k^T) B_k (I - \rho_k s_k y_k^T) + \rho_k y_k y_k^T,$$

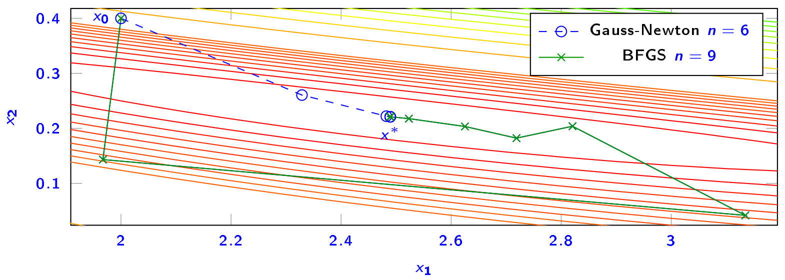
and

$$H_{k+1} = H_k - \frac{(H_k y_k)(H_k y_k)^T}{y_k^T H_k y_k} + \frac{s_k s_k^T}{y_k^T s_k}.$$

where $\rho_k = 1/(y_k^T s_k)$.

Example

- Gauss-Newton and BFGS on the antelope problem, no line search.



$$B_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, B_{1..8} = \left\{ \begin{bmatrix} 15 & 105 \\ 105 & 776 \end{bmatrix}, \begin{bmatrix} 6 & 31 \\ 31 & 289 \end{bmatrix}, \begin{bmatrix} 8 & 45 \\ 45 & 398 \end{bmatrix}, \begin{bmatrix} 9 & 66 \\ 66 & 671 \end{bmatrix}, \begin{bmatrix} 10 & 72 \\ 72 & 621 \end{bmatrix}, \begin{bmatrix} 10 & 80 \\ 80 & 700 \end{bmatrix}, \begin{bmatrix} 10 & 78 \\ 78 & 691 \end{bmatrix}, \begin{bmatrix} 10 & 75 \\ 75 & 664 \end{bmatrix} \right\}, B_9 = \begin{bmatrix} 10 & 74 \\ 74 & 652 \end{bmatrix}, \nabla^2 f(x^*) = \begin{bmatrix} 10 & 74 \\ 74 & 652 \end{bmatrix}.$$

Convergence properties

- The BFGS method has super-linear convergence, i.e. faster than linear but slower than quadratic.
- The deficit to quadratically convergent methods usually shows only in the few last iterations.
- Thus, for many practical applications, BFGS converges as fast as a Newton method, i.e. it requires approximately the same number of iterations.
- Since the BFGS search direction can be calculated in n^2 time vs. n^3 time for the Newton method, the required execution time is usually substantially less than Newton.