AI505

Optimization

Second-Order Methods

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Descent Direction Methods

How to select the descent direction?

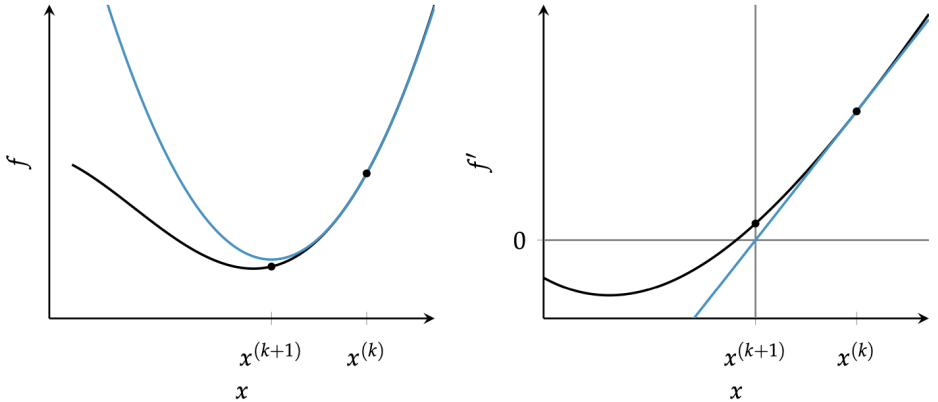
- first-order methods that rely on gradient
- second-order methods that rely on Hessian information

Advantages of second order methods in descent algorithms:

- way of accelerating the iteration [Davidon mid 1950s]
- additional information that can help improve the local model for informing the selection of
  - directions and
  - step lengths

Second-Order Methods

- Locally approximate function as quadratic
- Comparison of first-order and second order approximations



Outline

Newton's Method – Univariate

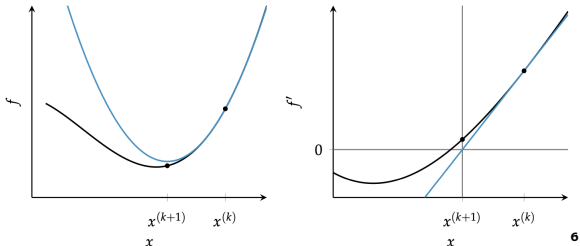
- Approximate a function using second-order Taylor series expansion
- analytically obtain the location where a quadratic approximation has a zero gradient.
- use that location as the next iteration to approach a local minimum.
- Univariate function

$$q(x) = f(x_k) + (x - x_k)f'(x_k) + \frac{(x - x_k)^2}{2}f''(x_k)$$

$$\frac{\mathrm{d}q(x)}{\mathrm{d}x} = f'(x_k) + (x - x_k)f''(x_k) = 0$$

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

$\equiv$ finding roots of derivative function



Newton's Method - Multivariate

- Multivariate function

$$f(\boldsymbol{x}) \approx q(\boldsymbol{x}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T H_k(\boldsymbol{x} - \boldsymbol{x}_k)$$

- Multivariate update rule

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - H_k^{-1}\nabla f(\boldsymbol{x}_k)$$

- H is the Hessian matrix
- (If f is quadratic and its Hessian is positive definite, then the update converges to the global minimum in one step.)

Algorithm

Input: $\nabla f, H, \boldsymbol{x}_0, \epsilon, k_{max}$
Output: $\boldsymbol{x}^*$
Set $k = 0, \Delta = 1, \boldsymbol{x} = \boldsymbol{x}_0$;
while $\|\Delta\| > \epsilon$ and $k \leq k_{max}$ **do**
    $\Delta = H(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x})$;
    $\boldsymbol{x} = \boldsymbol{x} - \Delta$;
    $k = k + 1$;

It can be modified to only give a descent direction $\boldsymbol{d} = -H(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x})$ and leave the step size to be determined with line search.

Newton's method – Example

Minimize **Booth's function**:

$$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

- $\boldsymbol{x}_0 = [9, 8]$
- The gradient of Booth's function is:

$$\nabla f(x) = [10x_1 + 8x_2 - 34, 8x_1 + 10x_2 - 38]$$

- The Hessian of Booth's function is:

$$H(x) = \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix}$$

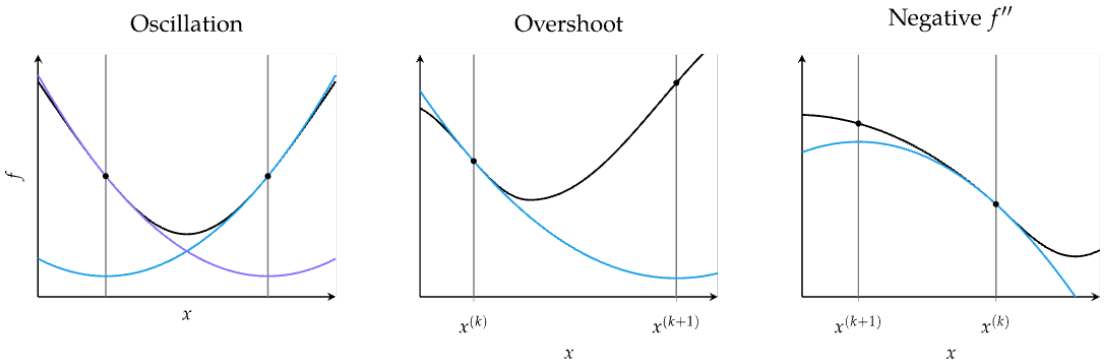
- The first iteration of Newton's method yields:

$$\begin{aligned} \boldsymbol{x}_1 &= \boldsymbol{x}_0 - H(\boldsymbol{x}_0)^{-1} \nabla f(\boldsymbol{x}_0) \\ &= \begin{bmatrix} 9 \\ 8 \end{bmatrix} - \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 10 \cdot 9 + 8 \cdot 8 - 34 \\ 8 \cdot 9 + 10 \cdot 8 - 38 \end{bmatrix} = \begin{bmatrix} 9 \\ 8 \end{bmatrix} - \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 120 \\ 114 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \end{aligned}$$

- Second iteration: The gradient at $\boldsymbol{x}_1$ is zero, so we have converged after a single iteration. The Hessian is positive definite everywhere, so $\boldsymbol{x}_1$ is the global minimum.

Newton's Method

Common causes of error in Newton's method



Newton's Method

- has quadratic convergence, meaning the difference between the minimum and the iterate is approximately squared with every iteration.

- This rate of convergence holds for Newton's method starting from x_0 within an interval $I = [x^* - \delta, x^* + \delta]$, for a root x^*, if

  1. $f''(x) \neq 0$ for all points in I,

  2. $f'''(x)$ is continuous on I, and

  3. $\frac{1}{2}\left|\frac{f'''(x_0)}{f''(x_0)}\right| < c\left|\frac{f'''(x^*)}{f''(x^*)}\right|$ for some $c < \infty$
     sufficient closeness condition, ensuring that the function is sufficiently approximated by the Taylor expansion and no overshoot.

Outline

Secant Method – Univariate

- For univariate functions, if the second derivative is unknown, it can be approximated using the secant method

$$f''(x_k) = \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

- Update equation

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f'(x_k) - f'(x_{k-1})} f'(x_k)$$

- It requires an additional initial design point and suffers from the same problems as Newton's method and may take more iterations to converge due to approximating the second derivative.

Outline

Quasi-Netwon Methods – Multivariate

- Automatic differentiation tools may not be applicable in many situations, and it may be much more costly to work with second derivatives in automatic differentiation software than with the gradient.
- Quasi-Newton methods, like steepest descent, require only the gradient of the objective function to be supplied at each iterate.
- By measuring the changes in gradients, they construct a model of the objective function that is good enough to produce superlinear convergence.
- The improvement over steepest descent is dramatic, especially on difficult problems.

Quasi-Netwon Methods – Multivariate

Use an approximation $Q_k \approx H^{-1}(\boldsymbol{x}_k)$

Input: $\boldsymbol{x}_0$, convergence tolerance $\epsilon > 0$, Q_0 (typically the $n \times n$ identity matrix)
Output: x^*
Set $k \leftarrow 0$;
while $\|\nabla f(\boldsymbol{x}_k)\| > \epsilon$ **do**
  Compute search direction $\boldsymbol{d}(x_k) = -Q_k \nabla f(\boldsymbol{x}_k)$;
  Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}(\boldsymbol{x}_k)$ where α_k is computed from a line search procedure to satisfy the Wolfe conditions;
  Define $\boldsymbol{\delta}_{k+1} \stackrel{def}{=} \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{\gamma}_{k+1} \stackrel{def}{=} \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$;
  Compute Q_{k+1};
  $k \leftarrow k + 1$;

- Davidon-Fletcher-Powell (DFP) method
- Broyden-Fletcher-Goldfarb-Shanno (BFGS) method
- Limited-memory BFGS (L-BFGS) method

Davidon-Fletcher-Powell (DFP) method

$$Q_{k+1} = Q_k - \frac{Q_k \gamma_k \gamma_k^T Q_k}{\gamma_k^T Q_k \gamma_k} + \frac{\delta_k \delta_k^T}{\delta_k^T \gamma_k}$$

where all terms on the right hand side are evaluated at the same iteration k.

The update for Q in the DFP method has three properties:

- Q remains symmetric and positive definite.
- If $f(x) = \frac{1}{2}\boldsymbol{x}^T A \boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x} + c$, then $Q = A^{-1}$. Thus the DFP has the same convergence properties as the conjugate gradient method.
- For high-dimensional problems, storing and updating Q can be significant compared to other methods like the conjugate gradient method.

Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

$$Q_{k+1} = Q_k - \left(\frac{\delta_k \gamma_k^T Q_k + Q_k \gamma_k \delta_k^T}{\delta_k^T \gamma_k} \right) + \left(1 + \frac{\gamma_k^T Q_k \gamma_k}{\delta_k^T \gamma_k} \right) \frac{\delta_k \delta_k^T}{\delta_k^T \gamma_k}$$

BFGS better than DFP with approximate line search but still uses an $n \times n$ dense matrix.

Theorem: Suppose that f is twice continuously differentiable and that the iterates generated by the BFGS algorithm converge to a minimizer x^* at which the Hessian matrix G is Lipschitz continuous Suppose also that the sequence $\|x_k - x^*\|$ converges to zero rapidly enough that $\sum_{k=1}^{\infty} \|x_k - x^*\| < \infty$. Then x_k converges to x^* at a superlinear rate (ie, faster than linear).

Limited-memory BFGS (L-BFGS) method

For **large-scale unconstrained optimization**
It stores the last m values for δ and γ rather than the full inverse Hessian ($i = 1$ oldest, $i = m$ last).
Compute $\boldsymbol{d}$ at $\boldsymbol{x}$ as $\boldsymbol{d} = -\boldsymbol{z}_m$ using:

$$\boldsymbol{q}_m = \nabla f(\boldsymbol{x}_k) \qquad \boldsymbol{q}_i = \boldsymbol{q}_{i+1} - \frac{\boldsymbol{\delta}_{i+1}^T \boldsymbol{q}_{i+1}}{\boldsymbol{\gamma}_{i+1}^T \boldsymbol{\delta}_{i+1}} \boldsymbol{\gamma}_{i+1}, \quad i = m-1, \ldots, 1$$

$$\boldsymbol{z}_0 = \frac{\boldsymbol{\delta}_m \odot \boldsymbol{\delta}_m \odot \boldsymbol{q}_m}{\boldsymbol{\gamma}_m^T \boldsymbol{\gamma}_m} \qquad \boldsymbol{z}_i = \boldsymbol{z}_{i-1} + \boldsymbol{\delta}_{i-1} \left(\frac{\boldsymbol{\delta}_{i-1}^T \boldsymbol{q}_{i-1}}{\boldsymbol{\gamma}_{i-1}^T \boldsymbol{\delta}_{i-1}} - \frac{\boldsymbol{\gamma}_{i-1}^T \boldsymbol{z}_{i-1}}{\boldsymbol{\gamma}_{i-1}^T \boldsymbol{\gamma}_{i-1}} \right), \quad i = 1, ..., m$$

For minimization, the inverse Hessian Q must remain positive definite.
The initial Hessian is often set to the diagonal of

$$Q_0 = \frac{\boldsymbol{\gamma}_0 \boldsymbol{\delta}_0^T}{\boldsymbol{\gamma}_0^T \boldsymbol{\gamma}_0}$$

Computing the diagonal for the above expression and substituting the result into $\boldsymbol{z}_0 = Q_0 \boldsymbol{q}_0$ results in the equation for $\boldsymbol{z}_0$.

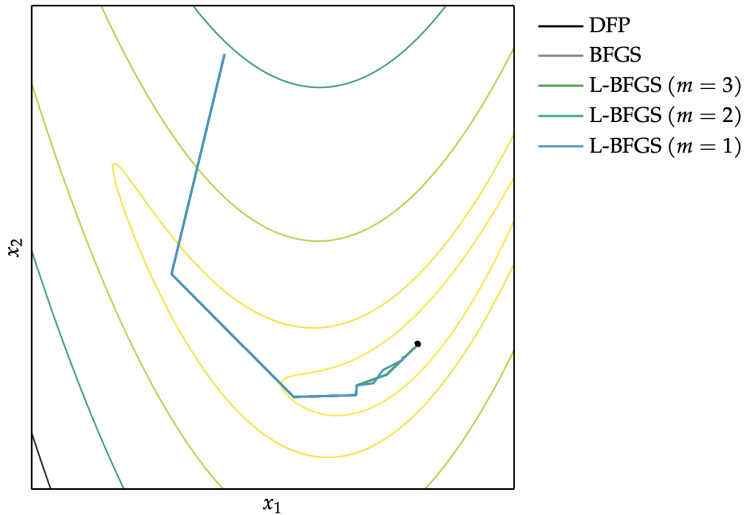
L-BFGS

(L-BFGS two-loop recursion)

Input:
Output: $Q_k \nabla f_k$
Set $\boldsymbol{q} \leftarrow \nabla f_k$;
for $i = k-1, k-2, \ldots, k-m$ **do**
  $\boldsymbol{z}_i \leftarrow \frac{\delta_i^T \boldsymbol{q}}{\gamma_i \cdot \delta_i^T}$;
  $\boldsymbol{q} \leftarrow \boldsymbol{q} - \alpha_i \cdot \gamma_i$;
$\boldsymbol{r} \leftarrow Q_0 \boldsymbol{q}$;
for $i = k-m, k-m+1, \ldots, k-1$ **do**
  $\beta \leftarrow \frac{\gamma_i^T \cdot \boldsymbol{r}}{\gamma_i \cdot \delta_i^T}$;
  $\boldsymbol{r} \leftarrow \boldsymbol{r} + \delta_i(\boldsymbol{z}_i - \beta)$;

Input:
Output: x^*
Choose starting point $\boldsymbol{x}_0$, integer $m > 0$;
$k \leftarrow 0$;
while not convergence **do**
  Set Q_0;
  Compute $\boldsymbol{d}_k \leftarrow -Q_k \nabla f_k$ from Algorithm on the left;
  Compute $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, where α_k is chosen to satisfy the Wolfe conditions;
  if $k > m$ **then**
    Discard the vector pair $\{\delta_{k-m}, \gamma_{k-m}\}$ from storage;
  Compute and save $\boldsymbol{\delta}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, $\gamma_k = \nabla f_{k+1} - \nabla f_k$;
  $k \leftarrow k+1$;

BFGS Methods - Comparison



Summary

- Incorporating second-order information in descent methods often speeds convergence.
- Newton's method is a root-finding method that leverages second-order information to quickly descend to a local minimum.
- The secant method and quasi-Newton methods approximate Newton's method when the second-order information is not directly available.
- In Python, methods implemented in the module scipy
  https://docs.scipy.org/doc/scipy/tutorial/optimize.html