02610
Optimization and Data Fitting
Week 4: Quasi-Newton Methods

Yiqiu Dong

DTU Compute
Technical University of Denmark

Steepest descent and Newton's methods

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}) \qquad f : \mathbb{R}^n \to \mathbb{R},\ f \in \mathcal{C}^2(\mathbb{R}^n)$$

- **Steepest descent method:**

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k, \qquad \boldsymbol{g}_k = \nabla f(\boldsymbol{x}_k)$$

  - **Pros:** Simple (only need the gradient)
  - **Cons:** Slow (linear convergence)

- **Newton's method:**

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla^2 f(\boldsymbol{x}_k)^{-1} \boldsymbol{g}_k$$

  - **Pros:** Fast (local quadratic convergence)
  - **Cons:** Expensive (need the Hessian and solution of linear system)

Quasi-Newton method

Idea: Similar as Newton's method, but we use a matrix B_k to approximate the Hessian $\nabla^2 f(\boldsymbol{x}_k)$. The matrix B_k should be easy to compute, $B_k \boldsymbol{p}_k = -\boldsymbol{g}_k$ should be easy to solve, and the method should still keep good convergence rate.

- **Quasi-Newton direction** is defined by

$$B_k \boldsymbol{p}_k = -\boldsymbol{g}_k \quad \text{or} \quad \boldsymbol{p}_k = -H_k \boldsymbol{g}_k,$$

where H_k is an inverse Hessian approximation.

- **Quasi-Newton iteration** is defined as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k.$$

Quasi-Newton method

Algorithm

Set $\boldsymbol{x}_0$ and $B_0 \succ 0$.
loop
  Solve $B_k \boldsymbol{p}_k = -\boldsymbol{g}_k$;
  Find the step length α_k;
  Update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$;
  Compute B_{k+1} from B_k
end loop

- We can also use an inverse Hessian approximation H_k instead of B_k, i.e., in Step 1 to compute $\boldsymbol{p}_k = -H_k \boldsymbol{g}_k$.
- Basic idea for updating B_k: Since B_k should already contain information on the Hessian, we only need update it accordingly.
- Different quasi-Newton method updates B_k differently.

Secant equation

Consider the second-order Taylor expansion as an approximation of $f(\boldsymbol{x})$, i.e.

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

Compute the gradient on $\boldsymbol{x}$, and obtain

$$\nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) \approx \boldsymbol{g} - \boldsymbol{g}_k.$$

Set $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$. We choose a Hessian approximation B_{k+1} or an inverse Hessian approximation H_{k+1} satisfy the **secant equation:**

$$B_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k \quad \text{or} \quad H_{k+1}\boldsymbol{y}_k = \boldsymbol{s}_k.$$

Secant equation

Consider a Hessian approximation B_{k+1} satisfying the secant equation:

$$B_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k,$$

where $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$.

For $f \in \mathcal{C}^2(\mathbb{R})$, according to the secant equation we have

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

- B_{k+1} is the slope of the secant line from $(x_k, f(x_k))$ and $(x_{k+1}, f(x_{k+1}))$.
- B_{k+1} is an approximation of $f''(x_k)$.
- In this case, with a unit step length the quasi-newton method is the same as the secant method for solving $f'(x) = 0$.

Secant equation

We define a quadratic model of the form

$$m_{k+1}(\boldsymbol{x}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T B_{k+1}(\boldsymbol{x} - \boldsymbol{x}_k),$$

which satisfies

$$\nabla m_{k+1}(\boldsymbol{x}_k) = \boldsymbol{g}_k, \qquad \nabla m_{k+1}(\boldsymbol{x}_{k+1}) = \boldsymbol{g}_{k+1}.$$

- The quasi-Newton method is basically using Newton direction of this quadratic approximation as the search direction.
- The second condition is equivalent to the secant equation.

Symmetric rank-1 (SR1) method

Let's try an update of the form

$$B_{k+1} = B_k + \sigma \boldsymbol{v}\boldsymbol{v}^T, \qquad \sigma \in \{-1, 1\}.$$

The secant equation $B_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k$ yields

$$\boldsymbol{y}_k = B_k\boldsymbol{s}_k + (\sigma \boldsymbol{v}^T\boldsymbol{s}_k)\boldsymbol{v}.$$

This only holds if $\boldsymbol{v}$ is a multiple of $\boldsymbol{y}_k - B_k\boldsymbol{s}_k$. We set $\boldsymbol{v} = \delta(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)$ and substitute it into the above equation, we obtain

$$\sigma = \mathrm{sign}\big((\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^T\boldsymbol{s}_k\big), \qquad \delta^2 = |(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^T\boldsymbol{s}_k|^{-1}.$$

SR1 update:

$$B_{k+1} = B_k + \frac{(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^T}{(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^T\boldsymbol{s}_k}.$$

Symmetric rank-1 (SR1) method

According to the **Sherman-Morrison formula**:

$$(A + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\boldsymbol{u}\boldsymbol{v}^T A^{-1}}{1 + \boldsymbol{v}^T A^{-1}\boldsymbol{u}},$$

we obtain **SR1 inverse Hessian update:**

$$H_{k+1} = H_k + \frac{(\boldsymbol{s}_k - H_k\boldsymbol{y}_k)(\boldsymbol{s}_k - H_k\boldsymbol{y}_k)^T}{(\boldsymbol{s}_k - H_k\boldsymbol{y}_k)^T\boldsymbol{y}_k}.$$

- **Pros:** Simple and cheap.
- **Cons:**
  1. Does not preserve positive definiteness.
     Solution: Combining with trust-region method.
  2. Numerically unstable: when $(\boldsymbol{s}_k - H_k\boldsymbol{y}_k)^T\boldsymbol{y}_k$ is close to zero, it breaks down.
     Solution: Skipping the update if the denominator is small.

Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

Instead of rank-1 update, let's try a rank-2 update:

$$B_{k+1} = B_k + \sigma_1 \boldsymbol{u}\boldsymbol{u}^T + \sigma_2 \boldsymbol{v}\boldsymbol{v}^T.$$

The secant equation $B_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k$ yields

$$\boldsymbol{y}_k - B_k\boldsymbol{s}_k = (\sigma_1 \boldsymbol{u}^T\boldsymbol{s}_k)\boldsymbol{u} + (\sigma_2 \boldsymbol{v}^T\boldsymbol{s}_k)\boldsymbol{v}.$$

Setting $\boldsymbol{u} = \boldsymbol{y}_k$ and $\boldsymbol{v} = B_k\boldsymbol{s}_k$ and solving for σ_1, σ_2, we obtain **BFGS update**:

$$B_{k+1} = B_k - \frac{B_k\boldsymbol{s}_k\boldsymbol{s}_k^T B_k}{\boldsymbol{s}_k^T B_k\boldsymbol{s}_k} + \frac{\boldsymbol{y}_k\boldsymbol{y}_k^T}{\boldsymbol{y}_k^T\boldsymbol{s}_k}$$

BFGS method

According to the **Sherman-Morrison-Woodbury formula**:

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1} U)^{-1} V^T A^{-1}$$

we obtain **BFGS inverse Hessian update**:

$$H_{k+1} = (I - \rho_k \boldsymbol{s}_k \boldsymbol{y}_k^T) H_k (I - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^T) + \rho_k \boldsymbol{s}_k \boldsymbol{s}_k^T$$

where $\rho_k = 1/(\boldsymbol{y}_k^T \boldsymbol{s}_k)$.

- The BFGS update is still very cheap, only $O(n^2)$ per update.

BFGS method

Algorithm

Set $\boldsymbol{x}_0$ and $B_0 \succ 0$ OR $H_0 \succ 0$.

loop

Compute search direction by solving $B_k \boldsymbol{p}_k = -\boldsymbol{g}_k$ OR $\boldsymbol{p}_k = -H_k \boldsymbol{g}_k$;

Update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$, where α_k satisfies the Wolfe conditions;

Define $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$;

Update B_{k+1} from B_k OR H_{k+1} from H_k according to the BFGS updates;

end loop

Positive definiteness

If $\boldsymbol{s}_k^T \boldsymbol{y}_k > 0$ (**curvature condition**), the BFGS update preserves positive definiteness of H_k.

Proof: According to BFGS inverse Hessian update, for any $\boldsymbol{u} \in \mathbb{R}^n$ we have

$$\boldsymbol{u}^T H_{k+1} \boldsymbol{u} = \left(\boldsymbol{u} - \rho_k(\boldsymbol{s}_k^T \boldsymbol{u})\boldsymbol{y}_k\right)^T H_k \left(\boldsymbol{u} - \rho_k(\boldsymbol{s}_k^T \boldsymbol{u})\boldsymbol{y}_k\right) + \rho_k(\boldsymbol{s}_k^T \boldsymbol{u})^2.$$

- If $H_k \succ 0$, then both terms in the right-hand side are nonnegative.
- The second term is zero only if $\boldsymbol{s}_k^T \boldsymbol{u} = 0$, and in this case the first term is zero only if $\boldsymbol{u} = 0$.

Davidon-Fletcher-Powell (DFP) method

Alternatively, we can compute a rank-2 update on the inverse Hessian approximate H_k:

$$H_{k+1} = H_k + \sigma_1 \boldsymbol{u}\boldsymbol{u}^T + \sigma_2 \boldsymbol{v}\boldsymbol{v}^T.$$

The secant equation $\boldsymbol{s}_k = H_{k+1}\boldsymbol{y}_k$ yields

$$\boldsymbol{s}_k - H_k\boldsymbol{y}_k = (\sigma_1 \boldsymbol{u}^T\boldsymbol{y}_k)\boldsymbol{u} + (\sigma_2 \boldsymbol{v}^T\boldsymbol{y}_k)\boldsymbol{v}.$$

Setting $\boldsymbol{u} = \boldsymbol{s}_k$ and $\boldsymbol{v} = H_k\boldsymbol{y}_k$ and solving for σ_1, σ_2, we obtain **DFP inverse Hessian update**:

$$H_{k+1} = H_k - \frac{H_k\boldsymbol{y}_k\boldsymbol{y}_k^T H_k}{\boldsymbol{y}_k^T H_k \boldsymbol{y}_k} + \frac{\boldsymbol{s}_k\boldsymbol{s}_k^T}{\boldsymbol{y}_k^T\boldsymbol{s}_k}$$

DFP method

Similar as BFGS, according to the Sherman-Morrison-Woodbury formula, we obtain **DFP Hessian update**:

$$B_{k+1} = (I - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^T) B_k (I - \rho_k \boldsymbol{s}_k \boldsymbol{y}_k^T) + \rho_k \boldsymbol{y}_k \boldsymbol{y}_k^T$$

where $\rho_k = 1/(\boldsymbol{y}_k^T \boldsymbol{s}_k)$.

- Same as BFGS, DFP update is cheap, only $O(n^2)$ per update.
- Same as BFGS, DFP preserves positive definiteness.
- Sometimes numerical unstable.

Broyden class

BFGS, DFP and SR1 are only 3 examples of numerous quasi-Newton updating formulae. Now we define a more general formula, **Broyden class**:

$$B_{k+1} = B_k - \frac{B_k \boldsymbol{s}_k \boldsymbol{s}_k^T B_k}{\boldsymbol{s}_k^T B_k \boldsymbol{s}_k} + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} + \phi_k (\boldsymbol{s}_k^T B_k \boldsymbol{s}_k) \boldsymbol{v}_k \boldsymbol{v}_k^T,$$

where ϕ_k is a scalar parameter and

$$\boldsymbol{v}_k = \frac{\boldsymbol{y}_k}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{B_k \boldsymbol{s}_k}{\boldsymbol{s}_k^T B_k \boldsymbol{s}_k}.$$

- $\phi_k = 0$, we get BFGS;
- $\phi_k = 1$, we get DFP;
- $\phi_k = \boldsymbol{y}_k^T \boldsymbol{s}_k / (\boldsymbol{y}_k^T \boldsymbol{s}_k - \boldsymbol{s}_k^T B_k \boldsymbol{s}_k)$, we get SR1.

Broyden class

Another form:

$$B_{k+1} = (1-\phi_k)B_{k+1}^{\mathrm{BFGS}} + \phi_k B_{k+1}^{\mathrm{DFP}}.$$

- All members of the Broyden class satisfy the secant equation.
- If $0 \le \phi_k \le 1$, when $\boldsymbol{s}_k^T \boldsymbol{y}_k > 0$, the Broyden class preserves positive definiteness.

Global convergence of BFGS

Assume

1. f is twice continuously differentiable.
2. The level set $\mathcal{L} = \{\boldsymbol{x} \in \mathbb{R}^n | f(\boldsymbol{x}) \le f(\boldsymbol{x}_0)\}$ is convex, and there exist positive constants m and M such that

$$m\|\boldsymbol{z}\|_2^2 \le \boldsymbol{z}^T \nabla^2 f(\boldsymbol{x})\boldsymbol{z} \le M\|\boldsymbol{z}\|_2^2$$

   for all $\boldsymbol{z} \in \mathbb{R}^n$ and $\boldsymbol{x} \in \mathcal{L}$.

Theorem

Let B_0 be any symmetric positive definite initial matrix, and let $\boldsymbol{x}_0$ be a starting point for which both assumptions are satisfied. Then the sequence $\{\boldsymbol{x}_k\}$ generated by the BFGS method converges to the minimizer $\boldsymbol{x}^*$ of f.

- This result holds for the Broyden class with $\phi_k \in [0, 1)$.

Superlinear convergence of BFGS

Assume

3. The Hessian matrix $\nabla^2 f$ is Lipschitz continuous at $\boldsymbol{x}^*$, that is,

$$\|\nabla^2 f(\boldsymbol{x}) - \nabla^2 f(\boldsymbol{x}^*)\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{x}^*\|_2,$$

   for all $\boldsymbol{x}$ near $\boldsymbol{x}^*$, where L is a positive constant.

Theorem

Suppose that f is twice continuously differentiable and that the iterates generated by the BFGS method converge to a minimizer $\boldsymbol{x}^*$ at which the above assumption holds. Suppose also that

$$\sum_{k=1}^{\infty} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 < \infty$$

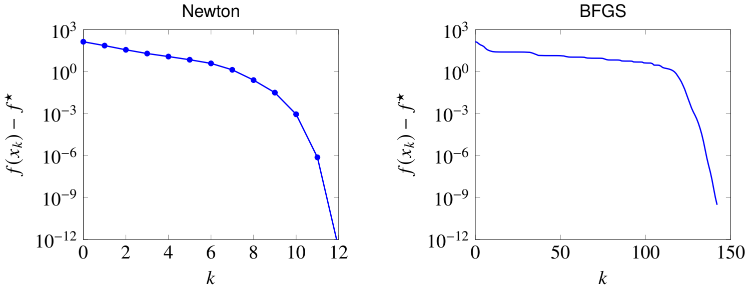
holds. Then $\boldsymbol{x}_k$ converges to $\boldsymbol{x}^*$ at a superlinear rate.

Example

Example from Vandenberghe's lecture notes:

$$\min_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x} - \sum_{i=1}^{m} \log(b_i - \boldsymbol{a}_i^T \boldsymbol{x})$$

with $n = 100$ and $m = 500$.



- Cost per Newton iteration: $O(n^3)$ plus computing the Hessian.
- Cost per BFGS iteration: $O(n^2)$.

scipy.optimize.minimize

scipy.optimize.

minimize

```
minimize(fun, x0, args=(), method=None, jac=None, hess=None, hessp=None,
bounds=None, constraints=(), tol=None, callback=None, options=None)
```

Minimization of scalar function of one or more variables. [source]

Inputs:

- `fun`: the objective function
- `x0`: an starting point;
- `method`: to specify the choice of the optimization methods, e.g. 'BFGS', 'dogleg', etc;
- `jac`: to specify the method to compute the gradient;
- `tol`: tolerance for termination;
- `options`: a dictionary of solver options. For example:
  - Set `'maxiter': 200` to change the maximum number of iterations to 200;
  - Set `'disp': True` to print the convergence messages.