Quasi-Newton Method

Yuqian Zhang

Rutgers University

yqz.zhang@rutgers.edu

March 1, 2021

Overview

Descent algorithms

- **Gradient Descent**

$$d^{(k)} = -\nabla f(x^{(k-1)})$$

  - This is the direction of steepest descent in ℓ^2
  - Gradient descent iterations are cheap, but typically many iterations are required for convergence.

- **Newton's method**

$$d^{(k)} = -(\nabla^2 f(x^{(k-1)}))^{-1} \nabla f(x^{(k-1)})$$

  - tend to be expensive (as they require a system solve), but they typically converge in far fewer iterations than gradient descent

Newton's method

$$d^{(k)} = -(\nabla^2 f(x^{(k-1)}))^{-1} \nabla f(x^{(k-1)})$$

- compute the gradient, a n-dim vector
- compute the Hessian, a $n \times n$-dim matrix
- invert the Hessian and apply the inverse to the gradient

Typically, computing the gradient is reasonable (maybe $O(n^2)$ or $O(n)$ flops and storage). Computing and inverting the Hessian might be harder; in general, these operations take $O(n^3)$ flops ... and that is for every iteration.

Quasi-Newton Method

- Estimate the Hessian, instead of calculating (and inverting) the Hessian at every point
- Approximate the Hessian (the second derivative) by measuring how the gradients (the first derivative) changes
- These Hessian estimates and their inverses can be quickly updated from one iteration to the next, thus avoiding the (extremely) expensive matrix inversion.

Low rank updates

- given the inverse $\boldsymbol{P}^{-1}$ of symmetric matrix $\boldsymbol{P}$
- adding a rank-r symmetric matrix $\boldsymbol{L}$ to $\boldsymbol{p}$
- the inverse $(\boldsymbol{P}+\boldsymbol{L})^{-1}$ can be computed in $O(rn^2)$
- suppose $\boldsymbol{L}=\boldsymbol{v}\boldsymbol{v}^T$ is rank-1

$$\left(\boldsymbol{P}+\boldsymbol{v}\boldsymbol{v}^T\right)^{-1}=\boldsymbol{P}^{-1}-\frac{1}{1+\boldsymbol{v}^T\tilde{\boldsymbol{v}}}\tilde{\boldsymbol{v}}\tilde{\boldsymbol{v}}^T, \quad \tilde{\boldsymbol{v}}=\boldsymbol{P}^{-1}\boldsymbol{v}.$$

Sherman-Morrison-Woodbury identity

$$\left(\boldsymbol{P}+\boldsymbol{U}\boldsymbol{V}^T\right)^{-1}=\boldsymbol{P}^{-1}-\tilde{\boldsymbol{U}}\left(\boldsymbol{I}+\boldsymbol{V}^T\tilde{\boldsymbol{U}}\right)^{-1}\tilde{\boldsymbol{V}}^T$$

where $\tilde{\boldsymbol{U}}=\boldsymbol{P}^{-1}\boldsymbol{U}$ and $\tilde{\boldsymbol{V}}=\boldsymbol{P}^{-1}\boldsymbol{V}$

Newton's method

- form a quadratic model around the current iterate $\boldsymbol{x}^{(k)}$

$$\tilde{f}_k(\boldsymbol{x}^{(k)} + \boldsymbol{v}) = f_k(\boldsymbol{x}^{(k)}) + \langle \boldsymbol{v}, \boldsymbol{a}_k \rangle + \frac{1}{2}\boldsymbol{v}^t \boldsymbol{P}_k \boldsymbol{v}$$

By Taylor's theorem, the particular choices of

$$\boldsymbol{a}_k = \nabla f(\boldsymbol{x}^{(k)}), \quad \boldsymbol{P}_k = \nabla^2 f(\boldsymbol{x}^{(k)})$$

- minimize the surrogate functional above to compute the step direction

$$\boldsymbol{d}^{(k+1)} = -\boldsymbol{P}_k^{-1} \boldsymbol{a}_k$$

Newton's method

- choosing a step size t_{k+1} and update

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + t_{k+1}\boldsymbol{d}^{(k+1)}$$

- repeat with new quadratic model

$$\tilde{f}_{k+1}(\boldsymbol{x}^{(k+1)} + \boldsymbol{v}) = f_k(\boldsymbol{x}^{(k+1)}) + \langle \boldsymbol{v}, \boldsymbol{a}_{k+1} \rangle + \frac{1}{2}\boldsymbol{v}^t \boldsymbol{P}_{k+1}\boldsymbol{v}$$

- Quasi-Newton methods operate in the same general framework

Quasi-Newton methods

$$\boldsymbol{d}^{(k+1)} = -\boldsymbol{P}_k^{-1}\boldsymbol{a}_k$$

- keep the linear term $\boldsymbol{a}_k = \nabla f(\boldsymbol{x}^{(k)})$
- find quadratic model $\boldsymbol{P}_k \succ 0$, which approximates $\nabla^2 f(\boldsymbol{x}^{(k)})$
  - use only gradient information
  - achieve super-linear convergence

Hessian approximation

Consider quadratic model

$$\begin{aligned}\tilde{f}_{k+1}(\boldsymbol{x}) = & f_k(\boldsymbol{x}) + \left\langle \boldsymbol{x} - \boldsymbol{x}^{(k+1)}, \boldsymbol{a}_{k+1} \right\rangle \\ & + \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x}^{(k+1)}\right)^T \boldsymbol{P}_{k+1}\left(\boldsymbol{x} - \boldsymbol{x}^{(k+1)}\right)\end{aligned}$$

then

$$\nabla \tilde{f}_{k+1}(\boldsymbol{x}) = \boldsymbol{a}_{k+1} + \boldsymbol{P}_{k+1}\left(\boldsymbol{x} - \boldsymbol{x}^{(k+1)}\right)$$

Gradient Matching Criterion for the most recent two iterates:

$$\begin{aligned}\nabla \tilde{f}_{k+1}(\boldsymbol{x}^{(k+1)}) &= \nabla f(\boldsymbol{x}^{(k+1)}) \\ \nabla \tilde{f}_{k+1}(\boldsymbol{x}^{(k)}) &= \nabla f(\boldsymbol{x}^{(k)})\end{aligned}$$

Hessian approximation

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

- Using the gradients for the $\boldsymbol{a}_{k+1}$ in the linear terms, the first condition above is automatic no matter what we choose for $\boldsymbol{P}_{k+1}$
- choose $\boldsymbol{P}_{k+1}$ so that the second condition above holds.

$$\nabla \tilde{f}_{k+1}(\boldsymbol{x}^{(k+1)} - t_{k+1}\boldsymbol{d}^{(k+1)}) = \nabla f(\boldsymbol{x}^{(k)})$$

Hessian approximation

$$\nabla \tilde{f}_{k+1}(\boldsymbol{x}^{(k+1)} - t_{k+1}\boldsymbol{d}^{(k+1)}) = \nabla f(\boldsymbol{x}^{(k)})$$

- choose $\boldsymbol{P}_{k+1}$ so that the second condition above holds

$$t_{k+1}\boldsymbol{P}_{k+1}\boldsymbol{d}^{(k+1)} = \nabla f(\boldsymbol{x}^{(k+1)}) - \nabla f(\boldsymbol{x}^{(k)})$$

- since $t_{k+1}\boldsymbol{d}^{(k+1)} = \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}$

$$\boldsymbol{P}_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k$$

with $\boldsymbol{s}_k = \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}$ and $\boldsymbol{y}_k = \nabla f(\boldsymbol{x}^{(k+1)}) - \nabla f(\boldsymbol{x}^{(k)})$

Hessian approximation

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{P} - \boldsymbol{P}_k\|_M \\ \text{subject to} \quad & \boldsymbol{P}^T = \boldsymbol{P} \\ & \boldsymbol{P}\boldsymbol{s}_k = \boldsymbol{y}_k \end{aligned}$$

- Quasi-Newton methods choose the $\boldsymbol{P}_{k+1}$ that is closest to the last quadratic model $\boldsymbol{P}_k$
- $\|\cdot\|_M$ is some matrix norm - different norms lead to different quasi-Newton methods.

DFP

Davidon-Fletcher-Powell formula: The original quasi-Newton method, developed by Davidson in the 50s, then analyzed by Fletcher and Powell, is based on a using a weighted Frobenius norm for $\|\cdot\|_M$.

$$\boldsymbol{P}_{k+1} = \left(\boldsymbol{I} - \gamma_k \boldsymbol{y}_k \boldsymbol{s}_k^T\right) \boldsymbol{P}_k \left(\boldsymbol{I} - \gamma_k \boldsymbol{y}_k \boldsymbol{s}_k^T\right) + \gamma_k \boldsymbol{y}_k \boldsymbol{y}_k^T$$

where $\gamma_k = 1/\boldsymbol{y}_k^T \boldsymbol{s}_k$

- adding a rank-2 matrix — remove the parts of the row and column spaces of $\boldsymbol{P}_k$ and replace that with chosen rank-1 matrix
- This step corresponds to finding the matrix that is closest to Pk in a certain norm under the constraint $\boldsymbol{P}\boldsymbol{s}_k = \boldsymbol{y}_k$

DFP

Let $\boldsymbol{Q}_k = \boldsymbol{P}_k^{-1}$ and apply Woodbury formula, then the Hessian inverse can be calculated via

$$\boldsymbol{Q}_{k+1} = \boldsymbol{Q}_k - \frac{1}{\boldsymbol{y}_k^T \tilde{\boldsymbol{y}}_k} \tilde{\boldsymbol{y}}_k \tilde{\boldsymbol{y}}_k^T + \frac{1}{\boldsymbol{y}_k^T \boldsymbol{s}_k} \boldsymbol{s}_k \boldsymbol{s}_k^T$$

where $\tilde{\boldsymbol{y}}_k = \boldsymbol{Q} \boldsymbol{y}_k$.

Sherman-Morrison-Woodbury identity

$$\left(\boldsymbol{P} + \boldsymbol{U}\boldsymbol{V}^T\right)^{-1} = \boldsymbol{P}^{-1} - \tilde{\boldsymbol{U}} \left(\boldsymbol{I} + \boldsymbol{V}^T \tilde{\boldsymbol{U}}\right)^{-1} \tilde{\boldsymbol{V}}^T$$

where $\tilde{\boldsymbol{U}} = \boldsymbol{P}^{-1}\boldsymbol{U}$ and $\tilde{\boldsymbol{V}} = \boldsymbol{P}^{-1}\boldsymbol{V}$

BFGS

— the most widely used and effective quasi-Newton methods

BFGS

Let $\|\boldsymbol{X}\|_M \doteq \left\|\boldsymbol{W}^{1/2}\boldsymbol{X}\boldsymbol{W}^{1/2}\right\|_F$ for any weight matrix $\boldsymbol{W}$ obeying $\boldsymbol{W}\boldsymbol{s}_t = \boldsymbol{y}_t$

$$\begin{array}{rl} \text{minimize} & \left\|\boldsymbol{W}^{1/2}(\boldsymbol{Q}-\boldsymbol{Q}_k)\boldsymbol{W}^{1/2}\right\|_F \\ \text{subject to} & \boldsymbol{Q} = \boldsymbol{Q}^k \\ & \boldsymbol{Q}\boldsymbol{y}_k = \boldsymbol{s}_k \end{array}$$

Close form solution

$$\boldsymbol{Q}_{k+1} = \left(\boldsymbol{I} - \gamma_k \boldsymbol{y}_k \boldsymbol{s}_k^T\right) \boldsymbol{Q}_k \left(\boldsymbol{I} - \gamma_k \boldsymbol{y}_k \boldsymbol{s}_k^T\right) + \gamma_k \boldsymbol{s}_k \boldsymbol{s}_k^T$$

where $\gamma_k = \frac{1}{\boldsymbol{y}_k^T \boldsymbol{s}_k}$

BFGS

Choosing among all inverse matrices that are closest to $\boldsymbol{P}_k^{-1}$ such k that $\boldsymbol{P}_k \boldsymbol{s}_k = \boldsymbol{y}_k$ is satisfied

$$\boldsymbol{Q}_{k+1} = \left(\boldsymbol{I} - \gamma_k \boldsymbol{y}_k \boldsymbol{s}_k^T\right) \boldsymbol{Q}_k \left(\boldsymbol{I} - \gamma_k \boldsymbol{y}_k \boldsymbol{s}_k^T\right) + \gamma_k \boldsymbol{s}_k \boldsymbol{s}_k^T, \quad \gamma_k = \frac{1}{\boldsymbol{y}_k^T \boldsymbol{s}_k}.$$

Conversely,

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \frac{1}{\boldsymbol{s}_k^T \tilde{\boldsymbol{s}}_k} \tilde{\boldsymbol{s}}_k \tilde{\boldsymbol{s}}_k^T + \frac{1}{\boldsymbol{y}_k^T \boldsymbol{s}_k} \boldsymbol{y}_k \boldsymbol{y}_k^T$$

where $\tilde{\boldsymbol{s}}_k = \boldsymbol{P}_k \boldsymbol{s}_k$

BFGS

BFGS Algorithm

1. for $k = 1, 2, \cdots$ do
2. $\quad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - t_{k+1}\boldsymbol{Q}_{k+1}\nabla f(\boldsymbol{x}^{(k)})$
3. $\quad \boldsymbol{Q}_{k+1} = \left(\boldsymbol{I} - \gamma_k \boldsymbol{y}_k \boldsymbol{s}_k^T\right) \boldsymbol{Q}_k \left(\boldsymbol{I} - \gamma_k \boldsymbol{y}_k \boldsymbol{s}_k^T\right) + \gamma_k \boldsymbol{s}_k \boldsymbol{s}_k^T$

- Initial $\boldsymbol{P}_0 = \boldsymbol{I}$ or estimate Hessian at the initial point
- Each iterate cost $O(n^2)$
- BFGS update maintains the positive-semidefiniteness of the $\boldsymbol{P}_k$ and $\boldsymbol{Q}_k$

Convergence of BFGS

- **Global convergence**: If f is strongly convex, then BFGS with backtracking converges to x^* from any starting point $x^{(0)}$ and initial quadratic model $\boldsymbol{Q}_0 \succ 0$.
- **Superlinear local convergence**:If f is strongly convex and $\nabla^2 f(x)$ is Lipschitz, then when we are close to the solution

$$\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\right\|_2 \leq c_k \left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\right\|_2$$

where $c_k \to 0$.

Convergence of descent algorithms

- Gradient Descent: f strongly convex

$$\left(\boldsymbol{x}^{(k+1)} - p^*\right) \leq \left(1 - \frac{m}{M}\right)\left(\boldsymbol{x}^{(k)} - p^*\right)$$

- Newton's Method: f strongly convex and Lipschitz Hessian

$$\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\right\|_2 \leq C \left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\right\|_2^2$$

- Quasi-Newton method: f strongly convex and Lipschitz Hessian

$$\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\right\|_2 \leq c_k \left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\right\|_2, \quad c_k \to 0$$