The Advantages and Disadvantages of BFGS, a Quasi-Newton Method

Mitchell T. Scott

Emory University
Department of Mathematics

October 17, 2023

References

- *BFGS in a Nutshell: An Introduction to Quasi-Newton Methods | by Adrian Lam | Towards Data Science*.
- *Large-Scale Unconstrained Optimization*, in Numerical Optimization, J. Nocedal and S. J. Wright, eds., Springer Series in Operations Research and Financial Engineering, Springer, New York, NY, 2006, pp. 164–192.
- *Quasi-Newton Methods*, in Numerical Optimization, J. Nocedal and S. J. Wright, eds., Springer Series in Operations Research and Financial Engineering, Springer, New York, NY, 2006, pp. 135–163.
- M. T. Heath, *Scientific Computing*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2018. _eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611975581.

Notation

- [3] is a great book, but it is universally agreed upon that the notation can be confusing (n=2).
- Instead, we will be using more of the notation used in [4]. The main players are below:
  - $\mathbf{H}_k$ is the full Hessian at the k^{th} step.
  - $\mathbf{B}_k$ is the approximation of the Hessian at the k^{th} step.
  - $\mathbf{B}_k^{-1}$ is the approximation of the inverse Hessian at the k^{th} step.

M. Chung, *private communication*, Emory University, Atlanta, GA., 2023

Newton's Method for Minimization I

The question is to minimize $f : \mathbb{R}^d \to \mathbb{R}$, where $f \in \mathcal{C}^2$ over the entire domain, an *unconstrained* optimization problem.

$$\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$$

We will Taylor expand this function

$$f(\mathbf{x}+\mathbf{s}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \mathbf{s} + \frac{1}{2}\mathbf{s}^\top \mathbf{H}_f(\mathbf{x})\mathbf{s} + \mathcal{O}(\mathbf{s}^3)$$

where $\mathbf{H}_f(\mathbf{x})$ is the *Hessian matrix* of second order partials of f.

Newton's Method for Minimization II

This function is minimized in $\mathbf{s}$ when

$$\mathbf{H}_f(\mathbf{x})\mathbf{s} = -\nabla f(\mathbf{x})$$

Recall the Hessian is the Jacobian of the gradient, so writing $\mathbf{g} := \nabla f(\mathbf{x})$, we get

$$\mathbf{J}_g(\mathbf{x})\mathbf{s} = -\mathbf{g}(\mathbf{x}),$$

which is a Newton step for $\mathbf{g} = \nabla f(\mathbf{x}) = \mathbf{0}$. Essentially, Newton's method for optimization is a root finding algorithm for the stationary points of a function.

Just use Newton? That's quasi-correct!

1. **Pros:**
   1. Quadratic Convergence near the solution
   2. $\mathbf{H}$ is SPD near the solution
2. **Cons:**
   1. Assuming dense $\mathbf{H}$, $\mathcal{O}\left(n^2\right)$ scalar function evals, and $\mathcal{O}\left(n^3\right)$`flops` per iteration
   2. Requires second derivatives of f.

Enter *quasi-Newton methods* that work with $\mathbf{B}_k$, an approximation of $\mathbf{H}$, the true Hessian!

1. **Pros:**
   1. Doesn't require second derivatives.
   2. $\mathbf{B}$ is always SPD.
   3. Require only one gradient evaluation.
   4. Update the approximation and solve linear system in $\mathcal{O}\left(n^2\right)$
2. **Cons:**
   1. Superlinear convergence

Enter BFGS!

Figure: The founders of the BFGS alogirthm. From left to right: Broyden, Fletcher, Goldfarb, and Shanno.[1]

Pseudocode for BFGS Implementation

Require:

$\mathbf{x}_0$ = initial guess,

$\mathbf{B}_0$ = initial Hessian approximation

`tol` = convergence requirement

while convergence requirement not met **do**

Solve $\mathbf{B}_k \mathbf{s}_k = -\nabla f(\mathbf{x}_k)$ for $\mathbf{s}_k$

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{s}_k$

$\mathbf{y}_k \leftarrow \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$

$\mathbf{B}_{k+1} \leftarrow \mathbf{B}_k - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^\top \mathbf{B}_k}{\mathbf{s}_k^\top \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k}$

$k \leftarrow k + 1$

end while

Pseudocode for BFGS Implementation (Inverse Problem)

Require:

$\mathbf{x}_0$ = initial guess,

$\mathbf{B}_0^{-1}$ = initial inverse Hessian approximation

`tol` = convergence requirement

while convergence requirement not met **do**

$\mathbf{p}_k \leftarrow -\mathbf{B}_k^{-1}\nabla f(\mathbf{x}_k)$

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$

$\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$

$\mathbf{y}_k \leftarrow \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$

$\mathbf{B}_{k+1}^{-1} \leftarrow \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^\top \mathbf{s}_k}\right) \mathbf{B}_k^{-1} \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k}$

$k \leftarrow k + 1$

end while

Rank-2 updates

Recall that $\mathbf{s}_k := \mathbf{x}_{k+1} - \mathbf{x}_k, \mathbf{y}_k := \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$.

Definition (Secant Equation)

We require that $\mathbf{B}_{k+1}$ satisfies $\mathbf{B}_{k+1}\mathbf{s}_k = \mathbf{y}_k$, which is a multidimensional *secant equation*. Similarly, we require $\mathbf{B}_{k+1}^{-1}\mathbf{y}_k = \mathbf{s}_k$ as an *inverse secant equation*.

Definition (Curvature Condition)

For $\mathbf{B}_{k+1}$ to be SPD, the *curvature condition* needs to be satisfied

$$\mathbf{s}_k^\top \mathbf{y}_k > 0.$$

coming from premultiplying the secant equation by $\mathbf{s}_k^\top$,

$$\mathbf{s}_k^\top \mathbf{B}_{k+1}\mathbf{s}_k = \mathbf{s}_k^\top \mathbf{y}_k > 0.$$

Initial Hessian approximation

How do we choose the initial Hessian or initial inverse Hessian?

Theorem (Preservation of SPD structure over iterations)

If $\mathbf{B}_k^{-1}$ is SPD, then both updates will produce an SPD $\mathbf{B}_{k+1}^{-1}$.

Proof.

Let $\mathbf{z}$ be a nonzero vector, then

$$\mathbf{z}^\top \mathbf{B}_{k+1}^{-1}\mathbf{z} = \left(\mathbf{z} - \frac{\mathbf{y}_k\left(\mathbf{s}_k^\top \mathbf{z}\right)}{\mathbf{y}_k^\top \mathbf{s}_k}\right)\mathbf{B}_k^{-1}\left(\mathbf{z} - \frac{\mathbf{y}_k\left(\mathbf{s}_k^\top \mathbf{z}\right)}{\mathbf{y}_k^\top \mathbf{s}_k}\right) + \frac{\left(\mathbf{z}^\top \mathbf{s}_k\right)^2}{\mathbf{y}_k^\top \mathbf{s}_k} \geq 0$$

□

Initial Hessian approximations

What are some SPD matrices that are used in practice?

1. $\mathbf{I}$
   1. Easy way to start off.
   2. First step is the vanilla steepest descent.
2. $\gamma\mathbf{I}$ where $\gamma \in \mathbb{R}^+$
   1. $\gamma = \delta \|g_0\|^{-1}$
   2. $\gamma_k = \frac{\mathbf{s}_{k-1}^\top \mathbf{y}_{k-1}}{\mathbf{y}_{k-1}^\top \mathbf{y}_{k-1}}$
3. $\mathbf{H}$, the true Hessian
   1. Starts the algorithm off better.
   2. Expensive to compute.
4. Something in between the two extremes like a finite difference approximation of $\mathbf{H}$.

BFGS example[4]

Let

$$f(\mathbf{x}) = 0.5x_1^2 + 2.5x_2^2, \text{ with } \mathbf{x}_0 = [5, 1]^\top$$

Clearly the gradient is given by

$$\nabla f(\mathbf{x}) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix}$$

Assume $\mathbf{B}_0 = \mathbf{I}$, which is equivalent to the first step being the steepest descent step, so

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{s}_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix} + \begin{bmatrix} -5 \\ -5 \end{bmatrix} = \begin{bmatrix} 0 \\ -4 \end{bmatrix}.$$

Exercise: Show that the approximate Hessian according to BFGS is

$$\mathbf{B}_1 = \begin{bmatrix} 0.667 & 0.333 \\ 0.333 & 4.667 \end{bmatrix}.$$

BFGS example, cont.

A new step is computed and the process continued. The resulting sequence of iterates are shown below.

k	$\mathbf{x}_k^\top$	$f(\mathbf{x}_k)$	$\nabla f(\mathbf{x}_k)^\top$
1	5.000 1.000	15.000	5.000 5.000
2	0.000 -4.000	40.000	0.000 -20.000
3	-2.222 0.444	2.963	-2.222 2.222
4	0.816 0.082	0.350	0.816 0.408
5	-0.009 -0.015	0.001	-0.009 -0.077
6	-0.001 0.001	0.000	-0.001 0.005

BFGS example, cont.

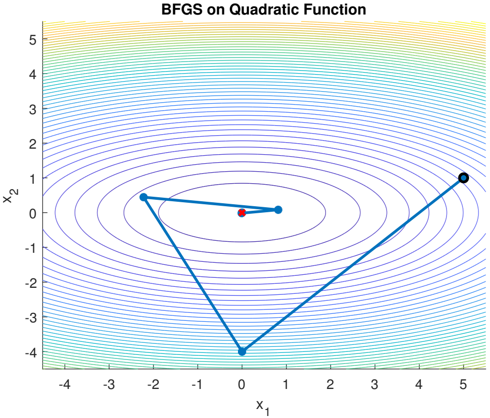

Figure: BFGS without linesearch converges superlinearly on $0.5x_1^2 + 2.5x_2^2$.

Iterative Method Showdown (BFGS vs. SD vs. Newton)

- Comparing the three methods we know (and love) on the Rosenbrock function, $\mathbf{x}_0 = [-1.2, 1]^\top$, with Wolfe conditions (why?)
- [3] has iterates below with
  - SD had 5264 iterations,
  - BFGS had 34 iterations,
  - Newton had 21 iterations.

Steepest Descent	BFGS	Newton
1.827e-04	1.70e-03	3.48e-02
1.826e-04	1.17e-03	1.44e-02
1.824e-04	1.34e-04	1.82e-04
1.823e-04	1.01e-06	1.17e-08

The Showdown Continues

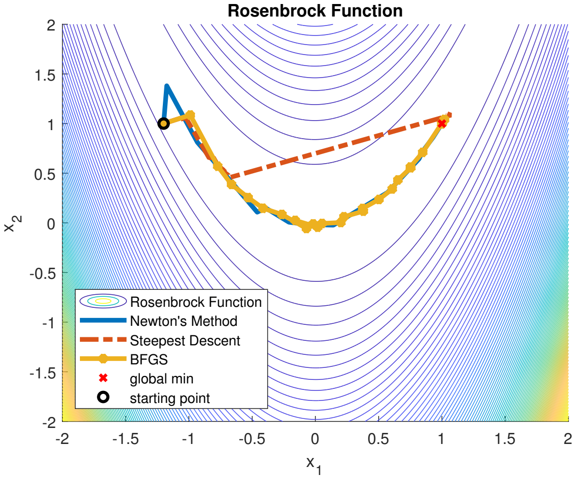

Figure: SD maxed out at 500 iterations, BFGS had 29, and Newton 17.

BFGS converges globally

Theorem (Global Convergence of BFGS,[3])

Let $\mathbf{B}_0$ be any symmetric positive definite initial matrix. Let x_0 be a starting point where

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

$$m\|\mathbf{z}\|^2 \leq \mathbf{z}^\top \nabla^2 f(\mathbf{x})\mathbf{z} \leq M\|\mathbf{z}\|^2, \forall \mathbf{z} \in \mathbb{R}^n, \mathbf{x} \in \mathcal{L}$$

Then the sequence $\{\mathbf{x}_k\}$ generated by the BFGS algorithm (with $\mathtt{tol} = 0$) converges to the minimizer $\mathbf{x}^*$ of f.

BFGS converges superlinearly

Theorem (Superlinear Convergence of BFGS,[3])

Suppose that f is twice continuously differentiable and that the iterates generated by the BFGS algorithm, converges to a minimizer $\mathbf{x}^*$ at which the Hessian matrix $\mathbf{H}$ is Lipschitz continuous at $\mathbf{x}^*$, that is,

$$\|\mathbf{H}(\mathbf{x}) - \mathbf{H}(\mathbf{x}^*)\| \leq L\|\mathbf{x} - \mathbf{x}^*\|, \forall \mathbf{x} \text{ near } \mathbf{x}^*, L > 0.$$

Suppose also that

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

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

What were we talking about? (Limited Memory BFGS)

- What happens if your problem is large scale, resulting in the storage of a large dense $\mathbf{B}_k^{-1}$?
- Instead, we store a modified version of $\mathbf{B}_k^{-1}$ by storing some vector pairs $\{\mathbf{s}_i, \mathbf{y}_i\}$ and doing inner products and vector sums.
- After the new iterate is computed, we discard the oldest vector pair, assuming the curvature information it encodes is not as valuable.

L-BFGS Two-loop recursion

$\mathbf{q} \leftarrow \nabla f_k$
for $i = k-1 : -1 : k-m$ **do**
  $\alpha_i \leftarrow \rho_i \mathbf{s}_i^\top \mathbf{q}$
  $\mathbf{q} \leftarrow \mathbf{q} - \alpha_i \mathbf{y}_i$
end for
$\mathbf{r} \leftarrow \left(\mathbf{B}_k^{-1}\right)^0 \mathbf{q}$
for $i = k-m : k-1$ **do**
  $\beta \leftarrow \rho_i \mathbf{y}_i^\top \mathbf{r}$
  $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{s}_i \left(\alpha_i - \beta_i\right)$
end for
  return $\mathbf{B}_k^{-1} \nabla f_k = \mathbf{r}_k$

L-BFGS Implementation

Require:

$\mathbf{x}_0 =$ initial guess,

$m \in \mathbb{Z}^+$,number of kept vector pairs ($m = 3 - 20$ in practice)

$k \leftarrow 0$

while Not Converged **do**

Choose $\left(\mathbf{B}_k^{-1}\right)^0$, could be $\frac{\langle \mathbf{s}_{k-1}, \mathbf{y}_{k-1} \rangle}{\langle \mathbf{y}_{k-1}, \mathbf{y}_{k-1} \rangle}\mathbf{I}$

Compute $\mathbf{p}_k \leftarrow \mathbf{B}_k^{-1} \nabla f_k = \mathbf{r}_k$

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$ ▷ Wolfe-Powell Conditions

if $k > m$ **then**

Delete $\{\mathbf{s}_{k-m}, \mathbf{y}_{k-m}\}$

$\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$

$\mathbf{y}_k \leftarrow \nabla f_{k+1} - \nabla f_k$

end if

$k \leftarrow k + 1$

end while