Numerical Optimization

Lecture Notes #18
Quasi-Newton Methods — The BFGS Method

Peter Blomgren,
⟨blomgren.peter@gmail.com⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Fall 2018

Outline

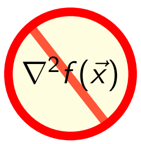
Quasi-Newton Methods — The Sales Pitch!

Quasi-Newton methods require **only the gradient** (like steepest descent) of the objective to be computed at each iterate.

By successive measurements of the gradient, Quasi-Newton methods build a quadratic model of the objective function which is sufficiently good that **superlinear** convergence is achieved.

Quasi-Newton methods are much faster than steepest descent (and coordinate descent) methods.

Since second derivatives (the Hessian) are not required, quasi-Newton methods are **sometimes** more efficient (as measured by total work / "wall-clock computational time") than Newton methods, especially when Hessian evaluation is slow/expensive.



The BFGS Method: Introduction — 1 of 3

The BFGS method is named for its discoverers: BROYDEN-FLETCHER-GOLDFARB-SHANNO, and is the most popular quasi-Newton method.

We first derive the DFP method (a close relative; named after DAVIDON-FLETCHER-POWELL) and then the BFGS method; and look at some properties and practical implementation details.

The derivation starts with the quadratic model

$$m_k(\bar{\mathbf{p}}) = f(\bar{\mathbf{x}}_k) + \nabla f(\bar{\mathbf{x}}_k)^T \bar{\mathbf{p}} + \frac{1}{2}\bar{\mathbf{p}}^T B_k \bar{\mathbf{p}}$$

at the current iterate $\bar{\mathbf{x}}_k$. B_k is a symmetric positive definite matrix (model Hessian) that will be **updated** in every iteration.

The BFGS Method: Introduction

Standard Stuff

Given this convex quadratic model, we can write down the minimizer $\bar{\mathbf{p}}_k$ explicitly as

$$\bar{\mathbf{p}}_k = -B_k^{-1}\nabla f(\bar{\mathbf{x}}_k).$$

We can compute the search direction $\bar{\mathbf{p}}_k$ using *e.g.* the Cholesky factorization, or a (P)CG-iteration; once we have $\bar{\mathbf{p}}_k$ we find the new iterate:

$$\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_k + \alpha_k \bar{\mathbf{p}}_k,$$

where we require that the step length α_k satisfies *e.g.* the Wolfe conditions:

$$\begin{array}{rcll} f(\bar{\mathbf{x}}_k + \alpha\bar{\mathbf{p}}_k) & \leq & f(\bar{\mathbf{x}}_k) + c_1\alpha\bar{\mathbf{p}}_k^T\nabla f(\bar{\mathbf{x}}), & c_1 \in (0,1) \\ \bar{\mathbf{p}}_k^T\nabla f(\bar{\mathbf{x}}_k + \alpha\bar{\mathbf{p}}_k) & \geq & c_2\bar{\mathbf{p}}_k^T\nabla f(\bar{\mathbf{x}}_k), & c_2 \in (c_1, 1). \end{array}$$

SAN DIEGO STATE UNIVERSITY

The BFGS Method: Introduction

So far we have not really done anything new — the key difference compared with the linesearch Newton method is that we are using an approximate Hessian $B_k \neq \nabla^2 f(\bar{\mathbf{x}}_k)$.

Instead to computing a completely new B_k in each iteration, we will update

$$B_{k+1} = B_k + \textbf{“something,”}$$

using partial about the curvature at step $\#k$. Thus we get a new model

$$m_{k+1}(\bar{\mathbf{p}}) = f(\bar{\mathbf{x}}_{k+1}) + \nabla f(\bar{\mathbf{x}}_{k+1})^T\bar{\mathbf{p}} + \frac{1}{2}\bar{\mathbf{p}}^T B_{k+1}\bar{\mathbf{p}}.$$

Clearly, for this to make sense we must impose some conditions on the update.

SAN DIEGO STATE UNIVERSITY

The BFGS Method: Conditions on B_{k+1}

We impose two conditions on the new model $m_{k+1}(\bar{\mathbf{p}})$:

[1,2] $m_{k+1}(\bar{\mathbf{p}})$ must match the gradient of the objective function in $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{x}}_{k+1}$.

The second condition is satisfied by construction, since

$$\nabla m_{k+1}(\bar{\mathbf{0}}) = \nabla f(\bar{\mathbf{x}}_{k+1}).$$

The first condition gives us

$$\nabla m_{k+1}(-\alpha_k\bar{\mathbf{p}}_k) = \nabla f(\bar{\mathbf{x}}_{k+1}) - \alpha_k B_{k+1}\bar{\mathbf{p}}_k = \nabla f(\bar{\mathbf{x}}_k).$$

With a little bit of re-arrangement we get

$$\boldsymbol{\alpha_k \mathbf{B}_{k+1}\bar{\mathbf{p}}_k = \nabla \mathbf{f}(\bar{\mathbf{x}}_{k+1}) - \nabla \mathbf{f}(\bar{\mathbf{x}}_k)}.$$

SAN DIEGO STATE UNIVERSITY

The BFGS Method: Conditions on B_{k+1}

We clean up the notation by introducing ($\bar{\mathbf{x}}_k$, and $\bar{\mathbf{y}}_k$):

$$\begin{array}{rcll} \bar{\mathbf{s}}_k & = & \bar{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}_k & \equiv \alpha_k\bar{\mathbf{p}}_k \\ \bar{\mathbf{y}}_k & = & \nabla f(\bar{\mathbf{x}}_{k+1}) - \nabla f(\bar{\mathbf{x}}_k). & \end{array}$$

We can now express the condition on B_{k+1} in terms of $\bar{\mathbf{s}}_k$ and $\bar{\mathbf{y}}_k$:

Secant Equation

$$\mathbf{B}_{k+1}\bar{\mathbf{s}}_k = \bar{\mathbf{y}}_k.$$

By pre-multiplying the secant equation by $\bar{\mathbf{s}}_k^T$ we get the

Curvature Condition

$$\underbrace{\bar{\mathbf{s}}_k^T B_{k+1}\bar{\mathbf{s}}_k}_{>0} = \bar{\mathbf{s}}_k^T\bar{\mathbf{y}}_k \quad\Rightarrow\quad \bar{\mathbf{s}}_k^T\bar{\mathbf{y}}_k > \mathbf{0}.$$

SAN DIEGO STATE UNIVERSITY

The BFGS Method: Conditions on B_{k+1}

If we impose the Wolfe, or strong Wolfe condition on the line search procedure, the curvature condition will always hold, since

$$\nabla f(\bar{\mathbf{x}}_{k+1})^T \bar{\mathbf{s}}_k \geq c_2 \nabla f(\bar{\mathbf{x}}_k)^T \bar{\mathbf{s}}_k,$$

by the (curvature) Wolfe condition, and therefore

$$\bar{\mathbf{y}}_k^T \bar{\mathbf{s}}_k \geq (c_2 - 1)\alpha_k \nabla f(\bar{\mathbf{x}}_k)^T \bar{\mathbf{p}}_k,$$

where the right-hand-side is positive since $c_2 < 1$ and $\bar{\mathbf{p}}_k$ is a descent direction.

When the curvature condition is satisfied, the secant equation always has at least one solution B_{k+1}.

The BFGS Method: More Conditions on B_{k+1}

It turns out that there are infinitely many symmetric positive definite matrices B_{k+1} which satisfy the secant equation.

Degrees of Freedom	Conditions Imposed
$n(n+1)/2$ — Symmetric	n — The Secant Equation n — Principal minors positive (PD)

To determine B_{k+1} uniquely we must impose additional conditions — we will select the B_{k+1} that is closest to B_k in some sense:

Matrix-Minimization-Problem

$$B_{k+1} = \arg\min_B \|B - B_k\|_{\text{some-norm}}$$

$$\textbf{subject to} \quad B = B^T,\ B\bar{\mathbf{s}}_k = \bar{\mathbf{y}}_k.$$

The BFGS Method: More Conditions on B_{k+1}

Each choice of matrix norm in this matrix-minimization-problem (MMP) gives rise to a different quasi-Newton method.

The **weighted Frobenius norm**

$$\|A\|_W \;=\; \|W^{1/2} A W^{1/2}\|_F \;=\; \|C\|_F \;=\; \sqrt{\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}^2}$$

allows easy solution of the MMP, and gives rise to a scale-invariant optimization method.

The matrix W is chosen to be the inverse G_k^{-1} of the **average Hessian**

$$G_k = \int_0^1 \nabla^2 f(\bar{\mathbf{x}}_k + \tau\alpha_k \bar{\mathbf{p}}_k)\, d\tau.$$

Square Roots of SPD Matrices

- A positive semi-definite matrix, M has a unique positive semi-definite square root, $R = M^{1/2}$.
- When $M = X\Lambda X^{-1} \stackrel{\text{SPD}}{=} Q\Lambda Q^T$, let $R = QSQ^T$, and

  $$R^2 = (QSQ^T)^2 = QSQ^T QSQ^T = QSSQ^T = QS^2Q^T = M,$$

  showing that

  $$S = \Lambda^{1/2}, \quad \text{and therefore} \quad R = Q\Lambda^{1/2}Q^T$$

- $\exists$ other approaches.

The DFP Method

With this weighting matrix and norm, the unique solution of the MMP is

$$B_{k+1} = \left(I - \gamma_k \bar{\mathbf{y}}_k \bar{\mathbf{s}}_k^T\right) B_k \left(I - \gamma_k \bar{\mathbf{s}}_k \bar{\mathbf{y}}_k^T\right) + \gamma_k \bar{\mathbf{y}}_k \bar{\mathbf{y}}_k^T, \quad \gamma_k = \frac{1}{\bar{\mathbf{y}}_k^T \bar{\mathbf{s}}_k}.$$

Note that γ_k is a scalar, and $\bar{\mathbf{y}}_k \bar{\mathbf{s}}_k^T$, $\bar{\mathbf{s}}_k \bar{\mathbf{y}}_k^T$, and $\bar{\mathbf{y}}_k \bar{\mathbf{y}}_k^T$ are rank-one matrices.

This is the original Davidon-Fletcher-Powell (DFP) method suggested by W.C. Davidon in 1959.

The original paper describing this revolutionary idea — *the first quasi-Newton method* — was not accepted for publication. It later appeared in **1991** in the first issue the the *SIAM Journal on Optimization.*

Fletcher and Powell demonstrated that this algorithm was much faster and more reliable than existing methods (at the time). This revolutionized the field of non-linear optimization.

San Diego State University

The DFP Method: Cleaning Up

The inverse of B_k is useful for the implementation of the method, since it allows the search direction $\bar{\mathbf{p}}_k$ to be computed using a simple matrix-vector product. We let

$$H_k = B_k^{-1}$$

and use

Sherman-Morrison-Woodbury formula

If $A \in \mathbb{R}^{n \times n}$ is non-singular and $\bar{\mathbf{a}}, \bar{\mathbf{b}} \in \mathbb{R}^n$, and if

$$B = A + \bar{\mathbf{a}}\bar{\mathbf{b}}^T$$

then

$$B^{-1} = A^{-1} - \frac{A^{-1}\bar{\mathbf{a}}\bar{\mathbf{b}}^T A^{-1}}{1 + \bar{\mathbf{b}}^T A^{-1}\bar{\mathbf{a}}}.$$

San Diego State University

The DFP Method: Cleaning Up

With a little bit of linear algebra we end up with

$$H_{k+1} = H_k - \underbrace{\frac{H_k \bar{\mathbf{y}}_k \bar{\mathbf{y}}_k^T H_k}{\bar{\mathbf{y}}^T H_k \bar{\mathbf{y}}_k}}_{\text{Update \#1}} + \underbrace{\frac{\bar{\mathbf{s}}_k \bar{\mathbf{s}}_k^T}{\bar{\mathbf{y}}_k^T \bar{\mathbf{s}}_k}}_{\text{Update \#2}} .$$

Both the update terms are rank-one matrices; so that H_k undergoes a rank-2 modification in each iteration.

This is the **fundamental idea of quasi-Newton updating:** instead of recomputing the matrix (-inverse) from scratch each time around, we apply a simple modification which combines the more recently observed information about the objective with existing knowledge embedded in the current Hessian approximation.

San Diego State University

Improving on DFP — The BFGS Method

The DFP method is quite effective, but once the quasi-Newton idea was accepted by the optimization community is was quickly superseded by the BFGS method.

BFGS updating is derived by instead of imposing conditions on the Hessian approximations B_k, we impose conditions directly on the inverses H_k.

The updated approximation must be symmetric positive definite, and must satisfy the **secant equation** in the form

$$\mathbf{H_{k+1}\bar{y}_k = \bar{s}_k}, \qquad \textbf{compare:} \qquad B_{k+1}\bar{\mathbf{s}}_k = \bar{\mathbf{y}}_k.$$

We get a slightly different matrix minimization problem...

San Diego State University

The BFGS Matrix Minimization Problem

Matrix-Minimization-Problem (BFGS)

$$H_{k+1} = \arg\min_{H} \|H - H_k\|_{\text{some-norm}}$$

$$\textbf{subject to} \quad H = H^T,\ H\bar{\mathbf{y}}_k = \bar{\mathbf{s}}_k$$

If we again choose the weighted Frobenius norm (with the same weight), then we get the unique update

$$H_{k+1} = \left(I - \rho_k \bar{\mathbf{s}}_k \bar{\mathbf{y}}_k^T\right) H_k \left(I - \rho_k \bar{\mathbf{y}}_k \bar{\mathbf{s}}_k^T\right) + \rho_k \bar{\mathbf{s}}_k \bar{\mathbf{s}}_k^T, \quad \rho_k = \frac{1}{\bar{\mathbf{y}}_k^T \bar{\mathbf{s}}_k},$$

which translated back to the Hessian approximation yields

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

BFGS vs. DFP Updates

BFGS Update —

$$H_{k+1} = \left(I - \rho_k \bar{\mathbf{s}}_k \bar{\mathbf{y}}_k^T\right) H_k \left(I - \rho_k \bar{\mathbf{y}}_k \bar{\mathbf{s}}_k^T\right) + \rho_k \bar{\mathbf{s}}_k \bar{\mathbf{s}}_k^T, \quad \rho_k = \frac{1}{\bar{\mathbf{y}}_k^T \bar{\mathbf{s}}_k},$$

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

DFP Update —

$$B_{k+1} = \left(I - \gamma_k \bar{\mathbf{y}}_k \bar{\mathbf{s}}_k^T\right) B_k \left(I - \gamma_k \bar{\mathbf{s}}_k \bar{\mathbf{y}}_k^T\right) + \gamma_k \bar{\mathbf{y}}_k \bar{\mathbf{y}}_k^T, \quad \gamma_k = \frac{1}{\bar{\mathbf{y}}_k^T \bar{\mathbf{s}}_k}.$$

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

The BFGS Method: Starting — $H_0 = \mathbf{???}$

The initial value for the iteration can be selected in different ways

- A finite difference approximation at $\bar{\mathbf{x}}_0$.
- $H_0 = I$, the identity matrix.
- $H_0 = \mathbf{diag}(s_1, s_2, \ldots, s_n)$, where $\bar{\mathbf{s}}$ captures the scaling of the variables (if known).

The BFGS Method: Algorithm

Algorithm: The BFGS Method

Given starting point $\bar{\mathbf{x}}_0$, convergence tolerance $\epsilon > 0$, and initial inverse Hessian approximation H_0:

k = 0
while($\|\nabla f(\bar{\mathbf{x}}_k)\| > \epsilon$)
  $\bar{\mathbf{p}}_k = -H_k \nabla f(\bar{\mathbf{x}}_k)$
  $\bar{\mathbf{x}}_{k+1} = \texttt{linesearch}(\bar{\mathbf{p}}_k, \ldots)$
  $\bar{\mathbf{s}}_k = \bar{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}_k$
  $\bar{\mathbf{y}}_k = \nabla f(\bar{\mathbf{x}}_{k+1}) - \nabla f(\bar{\mathbf{x}}_k)$
  $\rho_k = \frac{1}{\bar{\mathbf{y}}_k^T \bar{\mathbf{s}}_k}$
  $H_{k+1} = \left(I - \rho_k \bar{\mathbf{s}}_k \bar{\mathbf{y}}_k^T\right) H_k \left(I - \rho_k \bar{\mathbf{y}}_k \bar{\mathbf{s}}_k^T\right) + \rho_k \bar{\mathbf{s}}_k \bar{\mathbf{s}}_k^T$
  k = k + 1
end-while

The BFGS Method: Summary

The cost per iteration is

- $\mathcal{O}(n^2)$ arithmetic operations
- function evaluation
- gradient evaluation

The convergence rate is

- Super-linear

Newton's method converges quadratically, but the cost per iteration is higher — it requires the solution of a linear system. In addition Newton's method requires the calculation of second derivatives whereas the BFGS method does not.

The BFGS Method: Stability and Self-Correction

If at some point $\rho_k = 1/\bar{\mathbf{y}}_k^T\bar{\mathbf{s}}_k$ becomes large, *i.e.* $\bar{\mathbf{y}}_k^T\bar{\mathbf{s}}_k \sim 0$, then from the update formula

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

we see that H_{k+1} becomes large.

If for this, or some other, reason H_k becomes a poor approximation of $\left[\nabla^2 f(\bar{\mathbf{x}}_k)\right]^{-1}$ for some k, is there any hope of correcting it?

It has been shown that the BFGS method has **self-correcting properties**. — If H_k incorrectly estimates the curvature of the objective function, and if this estimate slows down the iteration, then the Hessian approximation will tend to correct itself within a few steps.

The BFGS Method: Stability and Self-Correction

The self-correcting properties stand and fall with the quality of the line search! — The Wolfe conditions ensure that the model captures appropriate curvature information.

The DFP method is less effective at self-correcting bad Hessian approximations.

Practical Implementation Details:

- The linesearch should always test $\alpha = 1$ first, because this step length will eventually be accepted, thus creating super-linear convergence.
- The linesearch can be somewhat "sloppy:" $c_1 = 10^{-4}$ and $c_2 = 0.9$ are commonly used values in the Wolfe conditions.
- The initial matrix H_0 should not be too large, if $H_0 = \beta I$, then the first step is $\bar{\mathbf{p}}_0 = -\beta\nabla f(\bar{\mathbf{x}}_0)$ which may be too long if β is large, often H_0 is rescaled before the update H_1 is computed:

$$H_0 \leftarrow \frac{\bar{\mathbf{y}}_k^T \bar{\mathbf{s}}_k}{\bar{\mathbf{y}}_k^T \bar{\mathbf{y}}_k} I.$$

L-BFGS

Forming the $n \times n$ dense matrix H_k can be quite expensive for large problems. `L-BFGS` stores a limited history of the BFGS update vectors $\bar{\mathbf{s}}_k$ and $\bar{\mathbf{y}}_k$ (which are size n), and use these to "implicitly" form the matrix operations.

In standard BFGS, the current H_k contains updates all the way back to initial step $\{\bar{\mathbf{s}}_j, \bar{\mathbf{y}}_j\}_{j=0}^{k-1}$, whereas `L-BFGS` only uses a limited number of "recent" updates; so that the action of $\tilde{H}_k$ is formed by application of $\{\bar{\mathbf{s}}_j, \bar{\mathbf{y}}_j\}_{j=\max(0,k-m)}^{k-1}$.

L-BFGS — "Two Loop Recursion"

Given a local initial positive definite model for the Hessian, $\tilde{H}_k$:

1. $\bar{\mathbf{v}} = \nabla f(\bar{\mathbf{x}}_k)$
2. $\alpha_j = \rho_j \bar{\mathbf{s}}_j^T \bar{\mathbf{v}}, \quad \bar{\mathbf{v}} = \bar{\mathbf{v}} - \alpha_j \bar{\mathbf{y}}_j, \quad j = k-1, \dots, k-m.$
3. $\bar{\mathbf{w}} = \tilde{H}_k \bar{\mathbf{v}}$
4. $\beta_j = \rho_j \bar{\mathbf{y}}_j^T \bar{\mathbf{w}}, \quad \bar{\mathbf{w}} = \bar{\mathbf{w}} + \bar{\mathbf{s}}_j(\alpha_j - \beta_j), \quad j = k-m, \dots, k-1$
5. Now, use $\bar{\mathbf{p}}_k = -\bar{\mathbf{w}} \quad (\approx -H_k \nabla f(\bar{\mathbf{x}}_k))$.

References:

1. Matthies, H.; Strang, G. (1979). *"The solution of non linear finite element equations."* International Journal for Numerical Methods in Engineering 14 (11): 16131626. doi:10.1002/nme.1620141104
2. Nocedal, J. (1980). *"Updating Quasi-Newton Matrices with Limited Storage."* Mathematics of Computation 35 (151): 773782. doi:10.1090/S0025-5718-1980-0572855-7

Index

Reference(s):

D1991 Davidon, William C. *"Variable metric method for minimization."* SIAM Journal on Optimization 1, no. 1 (1991): 1-17.

MS1979 Matthies, H.; Strang, G. (1979). *"The solution of non linear finite element equations."* International Journal for Numerical Methods in Engineering 14 (11): 16131626. doi:10.1002/nme.1620141104

N1980 Nocedal, J. (1980). *"Updating Quasi-Newton Matrices with Limited Storage."* Mathematics of Computation 35 (151): 773782. doi:10.1090/S0025-5718-1980-0572855-7