Notes on line search, trust regions, and the BFGS methods

Marco Sutti *

November 1, 2022

In these notes, we present the classical, standard (i.e., Euclidean) BFGS method, mainly following the presentation given in [NW06]. In the upcoming seminar of Thursday 10th November, we will discuss how to generalize the standard BFGS to Riemannian manifolds, according to the formulation of [RW12].

1 Preliminaries

We first review some concepts about line search, steepest descent, and trust region algorithms. Our purpose is to prepare the ground for the introduction of the BFGS quasi-Newton method. In numerical optimization algorithms, there are two main strategies: line search and trust regions.

1.1 Line-search methods

In classical optimization (minimization), line-search methods are widely used. They update the iterate x_k by choosing a search direction p_k and then adding a multiple α_k of this direction to the old iterate to obtain x_{k+1}. Algorithm 1 summarizes the general form of line-search algorithms.

Algorithm 1: Line-search minimization.

Given $f\colon \mathbb{R}^n \to \mathbb{R}$, starting point $x_0 \in \mathbb{R}^n$;
$k \leftarrow 0$;
repeat
  Compute a descent direction p_k;
  Set $x_{k+1} = x_k + \alpha_k p_k$, where α_k is computed from a line search procedure to satisfy the Wolfe conditions (1.1);
  $k \leftarrow k + 1$;
until x_{k+1} *sufficiently minimizes* f;

*Mathematics Division, National Center for Theoretical Sciences, Taipei, Taiwan (msutti@ncts.tw).

If we choose as descent direction $p_k = -\nabla f_k$, then we obtain the **steepest descent method**. We refer the reader to [NW06, Ch. 3] for more details on line-search techniques. If you have time, you can also have a look at these slides.

The usual stopping criterion for line search is the weak Wolfe conditions, which we recall here. Let f be a differentiable objective function. Let x_k be the current iterate, $\nabla f_k = \nabla f(x_k)$ the gradient, and p_k the search direction. The weak Wolfe conditions for the step size $\alpha_k > 0$ are defined by

$$f(x_k + \alpha_k p_k) - f(x_k) \leqslant c_1 \alpha_k \, p_k^\top \nabla f_k, \qquad p_k^\top \nabla f(x_k + \alpha_k p_k) \geqslant c_2 \, p_k^\top \nabla f_k, \tag{1.1}$$

with $0 < c_1 < c_2 < 1$. The first inequality is known as *sufficient decrease*, or *Armijo, condition*, while the second represents a *curvature condition*. One can reformulate the weak Wolfe conditions in terms of $\phi(\alpha_k) = f(x_k + \alpha_k p_k)$ as follows:

$$c_1 \, \phi'(0) \geqslant \frac{\phi(\alpha_k) - \phi(0)}{\alpha_k}, \qquad \phi'(\alpha_k) \geqslant c_2 \, \phi'(0),$$

with $0 < c_1 < c_2 < 1$. Since we are moving along a descent direction for f, we observe that the slope $\phi'(0)$ and the difference $\phi(\alpha_k) - \phi(0)$ are negative. The first inequality is thus asking for the decrease in ϕ at α_k to be larger than $c_1 \, \phi'(0)$. The second inequality is asking for the slope of ϕ at α_k to be larger than $c_2 \, \phi'(0)$. Figure 1 illustrates the weak Wolfe conditions in terms of ϕ.

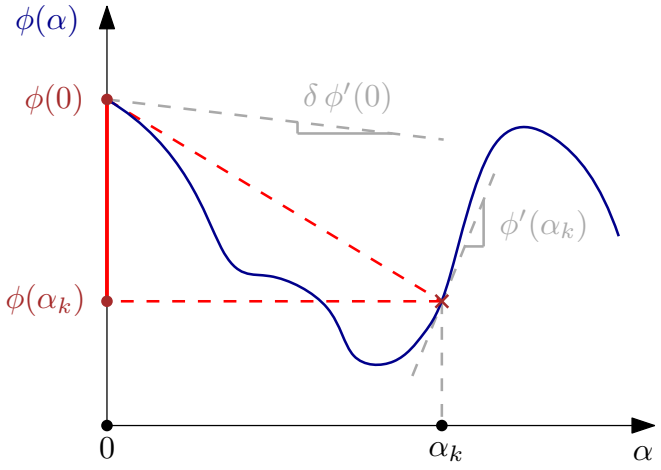

Figure 1: Illustration of the weak Wolfe conditions in terms of ϕ. From [Sut20, Fig. 5.1].

1.2 Trust-region methods

In trust region methods, the information gathered about f is used to construct a model m_k whose behavior around the current iterate x_k approximates that of the actual objective function f. We restrict the search for a minimizer of m_k to some **region** around x_k where the suitability of the model m_k can be **trusted** (hence the name of this algorithmic strategy, **trust-regions**).

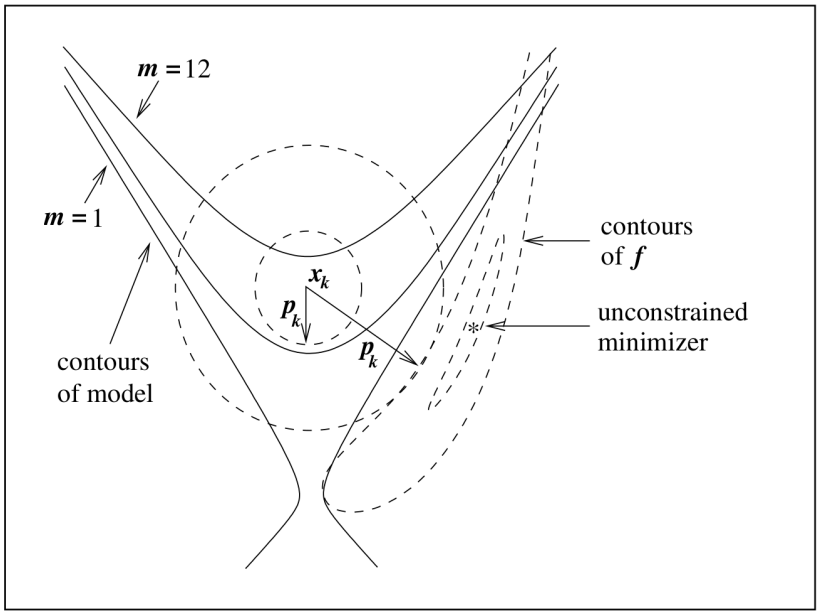

Figure 2: Two possible trust regions (circles) and their corresponding steps p_k. The solid lines are contours of the model function m_k. Illustration from [NW06, Fig. 2.4].

Problem statement: Find the (candidate) direction p by solving the subproblem

$$\min_p m_k(x_k + p), \qquad \text{where } x_k + p \text{ lies inside the trust region.}$$

If the candidate solution does not produce a sufficient decrease in f, we conclude that the trust region is too large, so we shrink it and re-solve the subproblem. Usually, the trust-region is a ball defined by $\|p\|_2 \leqslant \Delta$, where the scalar Δ is called *trust-region radius*. The model m_k is usually defined to be a quadratic function of the form

$$m_k(x_k + p) = f_k + p^\top \nabla f_k + \tfrac{1}{2} p^\top B_k p,$$

where $f_k \coloneqq f(x_k)$, $\nabla f_k \coloneqq \nabla f(x_k)$ the matrix B_k is either the Hessian $\nabla^2 f_k$ or some approximation to it (like in the case of BFGS, see section 2.1).

Each time we decrease the size of the trust region after failure of a candidate iterate, the step from x_k to the new candidate will be shorter, and it usually points in a different direction from the previous candidate, see Figure 2. In this sense, the trust-region strategy differs from line search. Observe and compare:

- Line search: starts by choosing a direction p_k, then computes an acceptable step size α_k (e.g., using Wolfe conditions (1.1));
- Trust region: starts by choosing a maximum distance (i.e., the trust region radius Δ_k) and then compute a direction and step that gives the best improvement possible subject to the distance constraint $\|p\|_2 \leqslant \Delta$.

We refer the reader to [NW06, Ch. 4] for more details on trust-region methods.

1.3 Search directions for line-search methods

In this section, we recall the three main directions used in classical optimization algorithms.

1. **Steepest descent direction:** the direction of steepest descent is the direction of the negative gradient: $-\nabla f_k$, see Figure 3. We do not discuss it further here, since our focus is on quasi-Newton methods.

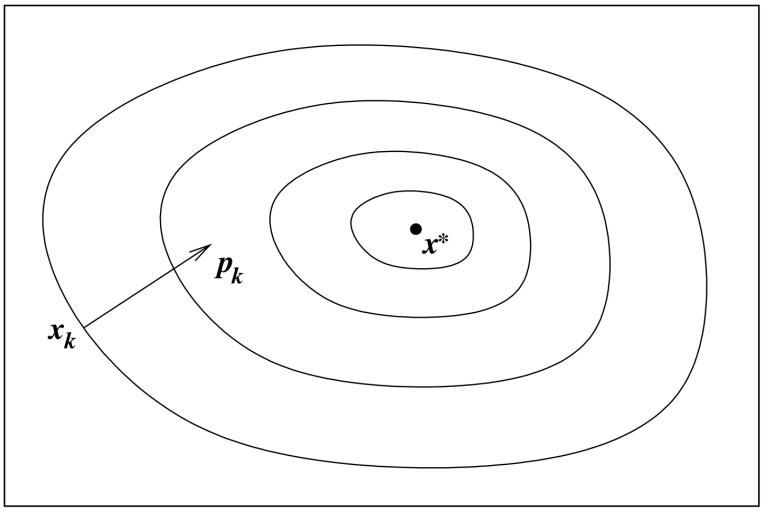

Figure 3: Steepest descent direction for a function of two variables. Illustration from [NW06, Fig. 2.5].

2. **Newton direction:** Consider the second-order Taylor series approximation to $f(x_k + p)$, i.e.,

$$f(x_k + p) \approx f_k + p^\top \nabla f_k + \tfrac{1}{2}\, p^\top \nabla^2 f_k\, p =: m_k(p),$$

Assuming $\nabla^2 f_k \succ 0$, we obtain the Newton direction by finding the vector p that minimizes $m_k(p)$. Setting the derivative of $m_k(p)$ w.r.t. p to 0, i.e.,

$$\nabla f_k + \nabla^2 f_k\, p = 0,$$

and solving for p, we find the Newton direction

$$p_k^{\mathrm{N}} = -\left(\nabla^2 f_k\right)^{-1} \nabla f_k. \tag{1.2}$$

Unlike the steepest descent direction, there is a "natural" step length of 1 associated with the Newton direction. Most line search implementations of Newton's method use the unit step $\alpha = 1$ where possible and adjust α only when it does not produce a satisfactory reduction in the value of f.

The main drawback of the Newton direction is the need for the Hessian, which can be computationally expensive.

3. **Quasi-Newton search directions:** do not require computation of the Hessian and yet still attain a **superlinear rate of convergence**.

   Instead of the exact Hessian $\nabla^2 f_k$, they use an approximation B_k in the Newton direction formula (1.2), i.e.,

   $$p_k = -B_k^{-1} \nabla f_k.$$

   The idea is that changes in the gradient provide information about the second derivative of f along the search direction.

   We choose the Hessian approximation B_k such that it satisfies the following condition, known as the **secant equation**:

   $$B_{k+1} s_k = y_k,$$

   where

   $$s_k = x_{k+1} - x_k, \qquad \text{and} \qquad y_k = \nabla f_{k+1} - \nabla f_k.$$

   Reason for this name: if $f : \mathbb{R} \to \mathbb{R}$, the ratio y_k / s_k is the slope of a secant line joining two points on a curve. See Figure 4.

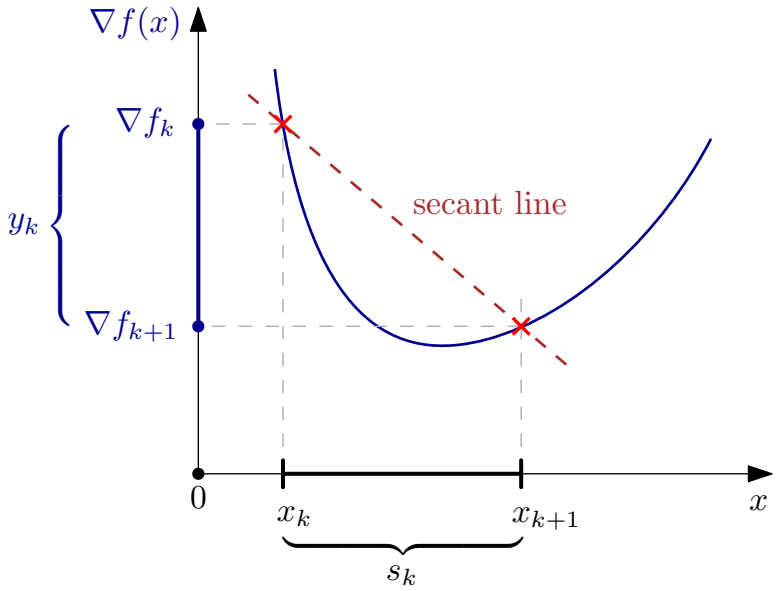

Figure 4: The meaning of the secant equation.

Then, we usually impose additional conditions on B_k, such as:

- **symmetry** (this is motivated by the symmetry of the exact Hessian);
- a requirement that the difference between successive approximations B_k and B_{k+1} have **low rank**.

The most popular formula for updating the Hessian approximation B_k is the **BFGS formula** (Broyden, Fletcher, Goldfarb and Shanno), defined by:

$$B_{k+1} = B_k \underbrace{- \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}}_{\text{rank-2 matrix}}. \tag{1.3}$$

Note that the difference between B_k and B_{k+1} is a rank-two matrix. This direction B_{k+1} satisfies the secant condition and maintains symmetry.

We refer the reader to [NW06, Ch. 2] for more details on search directions in line-search methods.

2 Quasi-Newton methods

We refer the reader to [NW06, Ch. 2] for all the details on quasi-Newton methods. Here, we only give the main concepts.

The search direction has the form:

$$p_k = -B_k^{-1} \nabla f_k.$$

We assume here that the step length α_k is computed by an inexact line search that satisfies the Wolfe or strong Wolfe conditions, with the same proviso mentioned above for Newton's method: The line search algorithm will always try the step length $\alpha = 1$ first, and will accept this value if it satisfies the Wolfe conditions. This implementation detail turns out to be crucial in obtaining a fast rate of convergence.

Like steepest descent, quasi-Newton methods require only first-order information (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**.

2.1 BFGS

We begin the derivation by forming the following quadratic model m_k of the objective function at the current iterate x_k:

$$m_k(p) = f_k + p^\top \nabla f_k + \tfrac{1}{2} p^\top B_k \, p,$$

Again, B_k is an approximation to the Hessian, it is an $n \times n$ symmetric positive definite matrix that will be updated at every iteration k. The minimizer of this convex quadratic model

$$p_k = -B_k^{-1} \nabla f_k \tag{2.1}$$

is used as **search direction**, and the new iterate is given by

$$x_{k+1} = x_k + \alpha_k p_k,$$

where α_k is chosen to satisfy the Wolfe conditions (1.1). Therefore, the algorithmic strategy is exactly like the one outlined in Algorithm 1, with the only difference being that here the search direction is chosen according to (2.1).

A main feature of BFGS is that instead of computing B_k afresh at every iteration, Davidon [Dav91] proposed to update it to account for the **curvature** measured during the most recent step.

Suppose that we have generated a new iterate x_{k+1} and wish to construct a new quadratic model, of the form

$$m_{k+1}(p) = f_{k+1} + p^\top \nabla f_{k+1} + \tfrac{1}{2}\, p^\top B_{k+1}\, p,$$

What requirements should we impose on B_{k+1}, based on the knowledge gained during the latest step? The first condition is known as **secant equation** [NW06, eq. (6.6)]:

$$B_{k+1} s_k = y_k, \tag{2.2}$$

where

$$s_k = x_{k+1} - x_k, \qquad \text{and} \qquad y_k = \nabla f_{k+1} - \nabla f_k.$$

The secant equation requires that the symmetric positive definite matrix B_{k+1} map s_k into y_k. This is possible only if s_k and y_k satisfy the **curvature condition**

$$s_k^\top y_k > 0. \tag{2.3}$$

When f is strongly convex (which means $\nabla^2 f \succ 0$ for a differentiable f), this inequality is satisfied for any two points x_k and x_{k+1}. However, this condition will not always hold for nonconvex functions, and in this case we need to enforce (2.3) explicitly, by imposing restrictions on the line search procedure that chooses the step length α. In fact, the condition (2.3) is guaranteed to hold if we impose the Wolfe (1.1) or strong Wolfe conditions [NW06, eq. (3.7)] on the line search. When the curvature condition is satisfied, the secant equation (2.2) always has a solution B_{k+1}.

To determine B_{k+1} uniquely, we impose the additional condition that among all symmetric matrices satisfying the secant equation, B_{k+1} is, in some sense, closest to the current matrix B_k. In other words, we solve the problem:

$$\min_B \|B - B_k\|,$$

$$\text{subject to} \quad B = B^\top, \quad Bs_k = y_k.$$

The solution to this optimization problem yields the DFP updating formula (Davidon, 1959), that we do not further discuss here. See [Dav91, NW06] for more details.

The BFGS updating formula can be derived if instead of imposing conditions on the Hessian approximations B_k, we impose similar conditions on their inverses $H_k = B_k^{-1}$. The updated approximation H_{k+1} must be symmetric ($H_{k+1} = H_{k+1}^\top$) and positive definite ($H_{k+1} \succ 0$), and must satisfy the secant equation, now written as

$$H_{k+1}\, y_k = s_k.$$

The condition of closeness to H_k is now specified by the following optimization problem

$$\min_H \|H - H_k\|,$$

$$\text{subject to} \quad H = H^\top, \quad Hy_k = s_k.$$

The unique solution H_{k+1} is [NW06, eq. (6.17)]

$$H_{k+1} = \left(I - \rho_k s_k y_k^\top\right) H_k \left(I - \rho_k y_k s_k^\top\right) + \rho_k s_k s_k^\top, \tag{2.4}$$

with $\rho_k := 1/(y_k^\top s_k)$.

2.1.1 The algorithm

A last aspect to consider is the choice of the initial approximation H_0. There is no general formula that works well in all cases. One can use specific information about the problem, for instance by setting it to the inverse of an approximate Hessian calculated by finite differences at x_0. Otherwise, we can simply set it to be the identity matrix, or a multiple of the identity matrix $H_0 = \beta I$, where the multiple is chosen to reflect the scaling of the variables; but there is no good general strategy for choosing β.

Algorithm 2: BFGS method

Given starting point x_0, convergence tolerance $\varepsilon > 0$, inverse Hessian approximation H_0;
$k \leftarrow 0$;
while $\|\nabla f_k\| > \varepsilon$ **do**
  Compute search direction $p_k = -H_k \nabla f_k$;
  Set $x_{k+1} = x_k + \alpha_k p_k$, where α_k is computed from a line search procedure to satisfy the Wolfe conditions;
  Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$;
  Compute H_{k+1} by means of (2.4);
  $k \leftarrow k + 1$;
end

Each iteration can be performed at a cost of $\mathcal{O}(n^2)$ arithmetic operations (plus the cost of function and gradient evaluations); there are no $\mathcal{O}(n^3)$ operations such as linear system solves or matrix–matrix operations. The algorithm is robust, and its rate of convergence is superlinear, which is fast enough for most practical purposes. Even though Newton's method converges more rapidly (that is, quadratically, see talk of 2022.10.27), its cost per iteration usually is higher, because of its need for second derivatives and solution of a linear system.

We can derive a version of the BFGS algorithm that works with the Hessian approximation B_k rather than H_k. The update formula for B_k is obtained by simply applying the Sherman–Morrison–Woodbury formula to (2.4) to obtain (1.3)

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}.$$

A naive implementation of this variant is not efficient for unconstrained minimization, because it requires the system $B_k p_k = -\nabla f_k$ to be solved for the step p_k, thereby increasing the cost of the step computation to $\mathcal{O}(n^3)$.

2.1.2 Implementation

The line search, which should satisfy either the Wolfe conditions (1.1) or the strong Wolfe conditions [NW06, eq. (3.7)], should always try the step length $\alpha_k = 1$ first, because this step length will eventually always be accepted (under certain conditions), thereby producing superlinear convergence of the overall algorithm.

2.1.3 Convergence analysis

Global and local convergence results for BFGS can be found in [NW06, §6.4].

There are no truly global convergence results for general nonlinear objective functions. In other words, we cannot prove that the iterates of these quasi-Newton methods approach a stationary point of the problem from any starting point and any (suitable) initial Hessian approximation. Conversely, there are well-known local, **superlinear convergence** results that are true under reasonable assumptions.

Here, we only report the local convergence result of BFGS. Please refer to [NW06, Ch. 2] for the details.

Theorem 2.1 ([NW06, Thm. 6.6])**.** *Suppose that f is twice continuously differentiable and that the iterates generated by the BFGS algorithm 2 converge to a minimizer x^* at which the Hessian matrix $\nabla^2 f(x)$ is Lipschitz continuous [NW06, Assumption 6.2]. Suppose also that*

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

holds. Then x_k converges to x^* at a ***superlinear rate****.*

References

[Dav91] Davidon, W. C. Variable Metric Method for Minimization. *SIAM J. Optim.*, 1(1):1–17, 1991.

[NW06] Nocedal, J. and Wright, S. J. *Numerical Optimization.* Springer New York, NY, second edition, 2006.

[RW12] Ring, W. and Wirth, B. Optimization Methods on Riemannian Manifolds and Their Application to Shape Space. *SIAM J. Optim.*, 22(2):596–627, 2012.

[Sut20] Sutti, M. *Riemannian algorithms on the Stiefel and the fixed-rank manifold.* PhD thesis, Section de mathématiques, University of Geneva, Dec 2020.