Some Optimization Methods that Use a Limited Number of Second Derivatives

Stephen P. Smith

Abstract. This paper describes two optimization methods that use all first derivatives, and a subset of second derivatives, all of which are available with backward differentiation. The first method is Newton's method on a direction set that changes dynamically during iteration. The second method is a quasi-Newton method that approximates the inverse Hessian matrix using a subset of second derivatives.

1. Introduction

Optimization methods that depend on first and second derivative have been advanced due to efficient algorithms developed for forward and backward differentiation (Griewank 2000). Prior[1] to these developments other methods had been promoted precisely to avoid calculating derivatives. Among those are the direction set methods that involve line searches and may not use any derivatives (Fletcher 1987, Section 4.2). And also the quasi-Newton methods that use finite differences of first derivatives to approximate second derivatives (Fletcher 1987, Section 3.2).

While calculating the entire gradient vector, or all first derivatives, can be accomplished in computing time proportional to the time needed to calculate the objective function, calculating all second derivatives may still pose a challenge. Nevertheless, a subset of the second derivatives, or isolated columns out of the Hessian matrix, can be evaluated efficiently.

Rather than leaving direction set methods, and quasi-Newton methods, as originally proposed tools meant to avoid derivative calculations, it is possible today to upgrade these methods so that information on all first and some second derivatives can be used. In particular, these upgrades offer possible advantages when only a subset of second derivatives are available because it may be too difficult to calculate all second derivatives, or because a robust algorithm that's less sensitive to starting values is sought, etc. Section 2 describes straight Newton's method while restricting the search to direction sets that change dynamically during iteration. Section 3 describes quasi-Netwon iteration when using only some second derivatives.

Perhaps these adaptations are not surprising given the history of optimization, and given that the mathematics is not complicated. In this vein, this paper is not so much original. Nevertheless, its important to see how these adaptations are now permitted given backward differentiation, which is the only point of this paper.

[1]1990 is about when methods based on automatic differentiation started to become better known.

2. Direction Set Optimization Using Selected Second Derivatives

To maximize, or minimize, $f(\theta)$ with respect to n parameters, θ_1, θ_2 ... θ_n, Newton's method is available:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{H}(\boldsymbol{\theta}_k)^{-1}\mathbf{g}(\boldsymbol{\theta}_k)$$

where

$$H_{ij} = \frac{\partial^2 f(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_i \partial\boldsymbol{\theta}_j}$$

$$g_i = \frac{\partial f(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_i}$$

Based on first principles, to apply Newton's method for a direction set only a smaller number of second derivatives are needed, and first derivatives are needed only for particular directions represented by a rectangular n×m matrix, **U**. The gradient vector, **g**, can be one of the vectors of **U**, and this then resembles a method of steepest ascent. Moreover, the gradient vector of the "second kind"[2] is defined by **s** = **Hg**, can also be in **U** and this makes the n×2 matrix **U**=[**g**, **s**]. At any particular iterate k+1, a new parameterization involving the m×1 vector **b** is introduced (but only implicitly because it is never calculated directly), where m<n, and such that:

$$\boldsymbol{\theta}(\mathbf{b}) = \boldsymbol{\theta}_k + \mathbf{U}\mathbf{b}$$

With **U** and **b** only defined for iterate k+1 (because they change dynamically across iterations), initializing **b** to zero and applying one round of Newton's method but limited to **b** to return **b*** then generates the desired updating equations for $\theta_{k+1}=\theta(\mathbf{b}^*)$:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{U}\left[\mathbf{U}^{\mathbf{T}}\mathbf{H}(\boldsymbol{\theta}_\mathbf{k})\mathbf{U}\right]^{-1}\mathbf{U}^{\mathbf{T}}\mathbf{g}(\boldsymbol{\theta}_\mathbf{k}) \qquad (1)$$

The gradient vector **g** is evaluated cheaply by backward differentiation, but it's the Hessian matrix of second derivatives that requires n passes through the recursion list which can be expensive. However, the required elements in update equation (1) can be evaluated much cheaper with on only m passes through the recursions: simply initialize the Q-matrix (at step-1)[3] with the columns of **U** in turn, and out (at step 4)[3] will come the particular column of **HU**.

[2]Please excuse my original coinage, but this vector is intended to measure how the gradient direction changes.

[3]In Section 3.4 of Smith (2000).

3. Quasi-Newton Optimization Using Selected Second Derivatives

Some types of Quasi-Newton optimization, such as the popular BFGS method (see Fletcher 1987, page 55), uses finite differences to approximate the Hessian matrix **H** (and its inverse), and hence does not evaluate any second derivatives directly. At iterate k+1, a proxy for the inverse Hessian matrix is used and is denoted by $\mathbf{J}_{k+1}$, and iteration follows as indicated below that offers an alternative to (1).

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{J}_{k+1}\mathbf{g}(\boldsymbol{\theta}_k)$$

In the BFGS method, low rank updates are available that are applied to $\mathbf{J}_k$ that generates $\mathbf{J}_{k+1}$. Because the matrix $\mathbf{J}_{k+1}$ is calculated from finite differences, it has no direct connection to **H** or $\mathbf{H}^{-1}$. The matrix $\mathbf{J}_k$ can also be rank deficient and still be useful as demonstrated in Section 2, which means its inverse (or $\mathbf{J}_k^{-1}$) does not exists.

In this paper, a different updating formula is given for calculating $\mathbf{J}_{k+1}$ that is based on a subset of second directional derivatives. Backward differentiation provides an efficient calculation for this subset of directional derivatives, and hence the method described here will compete with the BFGS approach that does not use second derivatives.

The direction set that defines the directional derivatives is represented by the rectangular n×m matrix **U** again, where m<n. The column vectors of **U** define the m directions. These columns can be populated by any selection of directions, and they can change during iteration. For m=2, a good selection of columns are **g**, and **s**=**Hg**, i.e., the gradient vector and the gradient vector of the second kind, which are highly influential directions that impact on the objective function and are easy to compute using backward differentiation. Both **U** and **W**=**HU** will specify the rank-m update of $\mathbf{J}_k$, in the discussion below. Like **s**, **W** is easy to calculate because it will involve only m<n passes through the recursion list for backward differentiation.[4]

Noting that **HU**=**W** implies $\mathbf{U}=\mathbf{H}^{-1}\mathbf{W}$, and that the motivation for calculating $\mathbf{J}_{k+1}$ is to approximate $\mathbf{H}^{-1}$, then the goal of updating is to find the n×m matrix **V** such that $[\mathbf{J}_k + \mathbf{V}\mathbf{V}^T]\mathbf{W}=\mathbf{U}$. However, this equation is hard to solve for the matrix **V**, and its easier to start by finding $\mathbf{V}\mathbf{V}^T$ as shown below.

[4]Initialize the Q-matrix at step-1 (in Section 3.4 of Smith, 2000) with a column of **U**, and out at step 4 will come the particular column of **W**.

$$\begin{aligned}\left[\mathbf{J}_k + \mathbf{V}\mathbf{V}^T\right]\mathbf{W} = \mathbf{U} \Rightarrow \mathbf{V}\mathbf{V}^T\mathbf{W} &= \mathbf{U} - \mathbf{J}_k\mathbf{W} \\ \Rightarrow \mathbf{V} &= \left[\mathbf{U} - \mathbf{J}_k\mathbf{W}\right]\left[\mathbf{V}^T\mathbf{W}\right]^{-1} \\ \Rightarrow \mathbf{V}\mathbf{V}^T &= \left[\mathbf{U} - \mathbf{J}_k\mathbf{W}\right]\left[\mathbf{V}^T\mathbf{W}\right]^{-1}\left[\mathbf{W}^T\mathbf{V}\right]^{-1}\left[\mathbf{U} - \mathbf{J}_k\mathbf{W}\right]^T \\ \Rightarrow \mathbf{V}\mathbf{V}^T &= \left[\mathbf{U} - \mathbf{J}_k\mathbf{W}\right]\left[\mathbf{W}^T\mathbf{V}\mathbf{V}^T\mathbf{W}\right]^{-1}\left[\mathbf{U} - \mathbf{J}_k\mathbf{W}\right]^T\end{aligned}$$

Both sides in the last equation (above) involve $\mathbf{VV}^T$, and so we are still not done. However, the top line gives the equation, $\mathbf{VV}^T\mathbf{W}=\mathbf{U}-\mathbf{J}_k\mathbf{W}$, and this equation can be pre-multiplied by $\mathbf{W}^T$ to give a solution for $\mathbf{W}^T\mathbf{VV}^T\mathbf{W}$ that does not involve $\mathbf{V}$:

$$\mathbf{W}^T\mathbf{V}\mathbf{V}^T\mathbf{W} = \mathbf{W}^T\mathbf{U} - \mathbf{W}^T\mathbf{J}_k\mathbf{W} = \mathbf{W}^T\left[\mathbf{H}^{-1} - \mathbf{J}_k\right]\mathbf{W}$$

The sought rank-m update is given uniquely by the following.

$$\mathbf{J}_{k+1} = \mathbf{J}_k + \left[\mathbf{U} - \mathbf{J}_k\mathbf{W}\right]\mathbf{A}^{-1}\left[\mathbf{U} - \mathbf{J}_k\mathbf{W}\right]^T$$
$$\textit{where}$$
$$\mathbf{A} = \mathbf{W}^T\mathbf{U} - \mathbf{W}^T\mathbf{J}_k\mathbf{W} = \mathbf{W}^T\left[\mathbf{H}^{-1} - \mathbf{J}_k\right]\mathbf{W}$$

This updating strategy is interesting, because $\mathbf{J}_0$ can be set to the null matrix, then $\mathbf{J}_1$ agrees with Section 2 and its direction set optimization that constitutes one step of Newton's method. When $\mathbf{J}_{k+1}$ is non-singular, then we also have the equation $\mathbf{J}_{k+1}{}^{-1}\mathbf{U}=\mathbf{W}$, which relates well to a hypothetical rank-m update that approximates the Hessian matrix (rather than its inverse). However, the present updating strategy works even when $\mathbf{J}_{k+1}$ is singular.

Rarely is iteration permitted without supervision, as things can go wrong. In minimization, a Hessian matrix (or its inverse) is sought that is positive definite, and in maximization a negative definite matrix is sought. In regard to the present updating strategy, there are several checks that can be made to make sure that $\mathbf{J}_{k+1}$ is trending appropriately. For the case of minimization, the following checks apply.

1. Accept $\mathbf{J}_{k+1}$ if both $\mathbf{W}^T\mathbf{U}$ and $\mathbf{A}$ are positive definite.
2. If $\mathbf{W}^T\mathbf{U}$ is positive definite, but not $\mathbf{A}$, set $\mathbf{J}_k$ to the null matrix and recalculate and accept $\mathbf{J}_{k+1}$.
3. If $\mathbf{W}^T\mathbf{U}$ is not positive definite, set $\mathbf{J}_{k+1}=\mathbf{J}_k$ or consider an alternative strategy.

References

Fletcher, R., 1987, *Practical Methods of Optimization*, Second Edition. John Wiley & Sons.

Griewank, A., 2000, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM.

Smith, S.P., 2000, A Tutorial on Simplicity and Computation Differentiation for Statisticians. Memo. viXra: 1702.0243.