Quasi-Newton Methods

- Introduction

Damped Newton method:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{F}(\mathbf{x}^{(k)})^{-1}\mathbf{g}^{(k)}.$$

To avoid the computation of $\mathbf{F}(\mathbf{x}^{(k)})^{-1}$, the quasi-Newton methods use $\mathbf{H}_k$ to approximate $\mathbf{F}(\mathbf{x}^{(k)})^{-1}$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \mathbf{H}_k \mathbf{g}^{(k)}.$$

$\mathbf{H}_k$ should exhibit some properties of $\mathbf{F}(\mathbf{x}^{(k)})^{-1}$ and be easily updated. Expanding f about $\mathbf{x}^{(k)}$ yields

$$\begin{aligned} f(\mathbf{x}^{(k+1)}) &= f(\mathbf{x}^{(k)}) + \mathbf{g}^{(k)T}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) + o(\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|) \\ &= f(\mathbf{x}^{(k)}) - \alpha \mathbf{g}^{(k)T}\mathbf{H}_k\mathbf{g}^{(k)} + o(\|\mathbf{H}_k\mathbf{g}^{(k)}\|). \end{aligned}$$

Thus, to guarantee a decrease in f for small α, we have to have

$$\mathbf{g}^{(k)T}\mathbf{H}_k\mathbf{g}^{(k)} > 0. \quad \Longleftarrow \mathbf{H}_k \succ \mathbf{0}$$

Quasi-Newton Methods

- Approximating the inverse Hessian

Suppose first that the Hessian matrix $\mathbf{F}(\mathbf{x})$ of the objective function f is constant and independent of $\mathbf{x}$. In other words, the objective function is quadratic, with Hessian $\mathbf{F}(\mathbf{x}) = \mathbf{Q}$ for all $\mathbf{x}$, where $\mathbf{Q} = \mathbf{Q}^T$. Then,

$$\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)} = \mathbf{Q}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}).$$

Let

$$\Delta\mathbf{g}^{(k)} \triangleq \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)},$$

and

$$\Delta\mathbf{x}^{(k)} \triangleq \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}.$$

Then, we may write

$$\Delta\mathbf{g}^{(k)} = \mathbf{Q}\Delta\mathbf{x}^{(k)}.$$

Quasi-Newton Methods

- Approximating the inverse Hessian

We start with a real symmetric positive definite matrix $\mathbf{H}_0$. Note that given k, the matrix $\mathbf{Q}^{-1}$ satisfies

$$\mathbf{Q}^{-1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, 0 \leq i \leq k.$$

Therefore, we also impose the requirement that the approximation $\mathbf{H}_{k+1}$ of the Hessian satisfy

$$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, 0 \leq i \leq k.$$

$$\mathbf{H}_n = \mathbf{Q}^{-1}.$$

$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \alpha_n \mathbf{H}_n \mathbf{g}^{(n)}$ solves the quadratic problem.

$\alpha_n = 1!$

Quasi-Newton Methods

- Quasi-Newton algorithms

$$\begin{aligned}\mathbf{d}^{(k)} &= -\mathbf{H}_k \mathbf{g}^{(k)} \\ \alpha_k &= \arg\min_{\alpha \geq 0} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},\end{aligned}$$

where the matrices $\mathbf{H}_0, \mathbf{H}_1, \cdots$ are symmetric. In the quadratic case, the above matrices are required to satisfy

$$\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}, 0 \leq i \leq k,$$

where $\Delta \mathbf{x}^{(i)} = \mathbf{x}^{(i+1)} - \mathbf{x}^{(i)} = \alpha_i \mathbf{d}^{(i)}$, and $\Delta \mathbf{g}^{(i)} = \mathbf{g}^{(i+1)} - \mathbf{g}^{(i)} = \mathbf{Q} \Delta \mathbf{x}^{(i)}$.

Quasi-Newton Methods

- Quasi-Newton algorithms

It turns out that quasi-Newton methods are also conjugate direction methods, as stated in the following.

Theorem 1. *Consider a quasi-Newton algorithm applied to a quadratic function with Hessian* $\mathbf{Q} = \mathbf{Q}^T$*, such that for* $0 \leq k \leq n-1$*,*

$$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, 0 \leq i \leq k,$$

where $\mathbf{H}_{k+1} = \mathbf{H}_{k+1}^T$*. If* $\alpha_i \neq 0, 0 \leq i \leq k+1$*, then* $\mathbf{d}_0, \cdots, \mathbf{d}_{k+1}$ *are* $\mathbf{Q}$*-conjugate.*

By Theorem 1 we conclude that a quasi-Newton algorithm solves a quadratic of n variables in at most n steps. Note that the equations that the matrices $\mathbf{H}_k$ are required to satisfy do not determine those matrices uniquely. Thus, we have some freedom in the way we compute the $\mathbf{H}_k$. In the methods we describe, we compute $\mathbf{H}_{k+1}$ by adding a correction to $\mathbf{H}_k$. We consider three specific updating formulas.

Quasi-Newton Methods

- The Rank One Correction Formula

$$\mathbf{H}_{k+1} = \mathbf{H}_k + a_k \mathbf{z}^{(k)} \mathbf{z}^{(k)T}.$$

Determine a_k and $\mathbf{z}^{(k)}$, given $\mathbf{H}_k$, $\Delta\mathbf{g}^{(k)}$, $\Delta\mathbf{x}^{(k)}$, so that the required relationship discussed in the previous section is satisfied, namely, $\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, i = 1, \cdots, k$. To begin, let us first consider the condition $\mathbf{H}_{k+1}\Delta\mathbf{g}^{(k)} = \Delta\mathbf{x}^{(k)}$.

$$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(k)} = (\mathbf{H}_k + a_k \mathbf{z}^{(k)} \mathbf{z}^{(k)T})\Delta\mathbf{g}^{(k)} = \Delta\mathbf{x}^{(k)}.$$

$$\mathbf{z}^{(k)} = \frac{\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)}}{a_k \mathbf{z}^{(k)T}\Delta\mathbf{g}^{(k)}}.$$

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})^T}{a_k(\mathbf{z}^{(k)T}\Delta\mathbf{g}^{(k)})^2}.$$

Quasi-Newton Methods

- The Rank One Correction Formula

Multiply $\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)} = (a_k\mathbf{z}^{(k)T}\Delta\mathbf{g}^{(k)})\mathbf{z}^{(k)}$ by $\Delta\mathbf{g}^{(k)T}$ to obtain

$$\Delta\mathbf{g}^{(k)T}\Delta\mathbf{x}^{(k)} - \Delta\mathbf{g}^{(k)T}\mathbf{H}_k\Delta\mathbf{g}^{(k)} = \Delta\mathbf{g}^{(k)T}a_k\mathbf{z}^{(k)}\mathbf{z}^{(k)T}\Delta\mathbf{g}^{(k)}.$$

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})^T}{\Delta\mathbf{g}^{(k)T}(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})}.$$

Quasi-Newton Methods

- The Rank One Correction Formula

Rank One Algorithm

1. Set $k := 0$; select $\mathbf{x}^{(0)}$, and a real symmetric positive definite $\mathbf{H}_0$.
2. If $\mathbf{g}^{(k)} = \mathbf{0}$, stop; else $\mathbf{d}^{(k)} = -\mathbf{H}_k \mathbf{g}^{(k)}$.
3. Compute
$$\alpha_k = \arg\min_{\alpha \geq 0} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$$
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}.$$
4. Compute
$$\Delta \mathbf{x}^{(k)} = \alpha_k \mathbf{d}^{(k)}$$
$$\Delta \mathbf{g}^{(k)} = \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}$$
$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})(\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})^T}{\Delta \mathbf{g}^{(k)T}(\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})}.$$
5. Set $k := k + 1$; go to step 2.

Quasi-Newton Methods

- The Rank One Correction Formula

The rank one algorithm is based on satisfying the equation

$$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(k)} = \Delta\mathbf{x}^{(k)}.$$

However, what we want is

$$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, i = 0, 1, \cdots, k.$$

It turns out that the above is, in fact, automatically true, as stated in the following theorem.

Theorem 2. *For the rank one algorithm applied to the quadratic with Hessian* $\mathbf{Q} = \mathbf{Q}^T$, *we have* $\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, 0 \leq i \leq k.$

Quasi-Newton Methods

- The Rank One Correction Formula

For a nonquadratic objective function, $\mathbf{H}_{k+1}$ may not be positive definite and thus $\mathbf{d}^{(k+1)}$ may not be a descent direction. Furthermore, if

$$\Delta\mathbf{g}^{(k)T}(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})$$

is close to zero, then there may be numerical problems in evaluating $\mathbf{H}_{k+1}$. Example: Assume that $\mathbf{H}_k \succ \mathbf{0}$. It turns out that if $\Delta\mathbf{g}^{(k)T}(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)}) > 0$, then $\mathbf{H}_{k+1} \succ \mathbf{0}$. However, if $\Delta\mathbf{g}^{(k)T}(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)}) < 0$, then $\mathbf{H}_{k+1}$ may not be positive definite.

$$f(\mathbf{x}) = \frac{x_1^4}{4} + \frac{x_2^2}{2} - x_1x_2 + x_1 - x_2$$

$$\mathbf{x}^{(0)} = [0.59607, 0.59607]^T,$$

$$\mathbf{H}_0 = \begin{bmatrix} 0.94913 & 0.14318 \\ 0.14318 & 0.59702 \end{bmatrix}.$$

Quasi-Newton Methods

- The Rank One Correction Formula

Note that $\mathbf{H}_0 \succ \mathbf{0}$. We have

$$\Delta \mathbf{g}^{(0)T}(\Delta \mathbf{x}^{(0)}) - \mathbf{H}_0 \Delta \mathbf{g}^{(0)}) = -0.03276$$

and

$$\mathbf{H}_1 = \begin{bmatrix} 0.94481 & 0.23324 \\ 0.23324 & -1.2788 \end{bmatrix},$$

It is easy to check that $\mathbf{H}_1$ is not positive definite.

If we use a "rank two" update, then $\mathbf{H}_k$ is guaranteed to be positive definite for all k, **provided the line search is exact**.

Quasi-Newton Methods

- The Davidon –Fletcher-Powell (DFP) Algorithm

1. Set $k := 0$; select $\mathbf{x}^{(0)}$,and a real symmetric positive definite $\mathbf{H}_0$.
2. If $\mathbf{g}^{(k)} = \mathbf{0}$, stop; else $\mathbf{d}^{(k)} = -\mathbf{H}_k \mathbf{g}^{(k)}$.
3. Compute

$$\alpha_k = \arg\min_{\alpha \geq 0} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}.$$

4. Compute

$$\Delta \mathbf{x}^{(k)} = \alpha_k \mathbf{d}^{(k)}$$

$$\Delta \mathbf{g}^{(k)} = \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}$$

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\Delta \mathbf{x}^{(k)} \Delta \mathbf{x}^{(k)T}}{\Delta \mathbf{x}^{(k)T} \Delta \mathbf{g}^{(k)}} - \frac{[\mathbf{H}_k \Delta \mathbf{g}^{(k)}][\mathbf{H}_k \Delta \mathbf{g}^{(k)}]^T}{\Delta \mathbf{g}^{(k)T} \mathbf{H}_k \Delta \mathbf{g}^{(k)}}.$$

5. Set $k := k + 1$; go to step 2.

Quasi-Newton Methods

- The Davidon –Fletcher-Powell (DFP) Algorithm

We show that the DFP algorithm is a quasi-Newton method, in the sense that when applied to quadratic problems, we have $\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, 0 \leq i \leq k$.

Theorem 3. *In the DFP algorithm applied to the quadratic with Hessian* $\mathbf{Q} = \mathbf{Q}^T$, *we have* $\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, 0 \leq i \leq k$.

By Theorems 1 and 3 we conclude that the DFP algorithm is a conjugate direction algorithm.

Theorem 4. *Suppose that* $\mathbf{g}^{(k)} \neq \mathbf{0}$. *In the DFP algorithm, if* $\mathbf{H}_k$ *is positive definite, then so is* $\mathbf{H}_{k+1}$.

The DFP algorithm is superior to the rank one algorithm in that it preserves the positive definiteness of $\mathbf{H}_k$. However, it turns out that in the case of larger nonquadratic problems the algorithm has the tendency of sometimes getting "stuck." This phenomenon is attributed to $\mathbf{H}_k$ becoming nearly singular.

Quasi-Newton Methods

- The Broyden-Fletcher-Goldfarb-Shanno (BFGS) Algorithm

$$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, 0 \leq i \leq k,$$

is derived from $\Delta\mathbf{g}^{(i)} = \mathbf{Q}\Delta\mathbf{x}^{(i)}, 0 \leq i \leq k$. We then formulate update formulas for the approximations to the inverse of the Hessian matrix $\mathbf{Q}^{-1}$. An alternative to approximating $\mathbf{Q}^{-1}$ is to approximate $\mathbf{Q}$ itself. To do this let $\mathbf{B}_k$ be our estimate of $\mathbf{Q}$ at the kth step. We require $\mathbf{B}_{k+1}$ to satisfy

$$\Delta\mathbf{g}^{(i)} = \mathbf{B}_{k+1}\Delta\mathbf{x}^{(i)}, 0 \leq i \leq k.$$

Notice that the above set of equations is similar to the previous set of equations for $\mathbf{H}_{k+1}$, the only difference being that the roles of $\Delta\mathbf{x}^{(i)}$ and $\Delta\mathbf{g}^{(i)}$ are interchanged. Thus, given any update formula for $\mathbf{H}_k$, a corresponding update formula for $\mathbf{B}_k$ can be found by interchanging the roles of $\mathbf{B}_k$ and $\mathbf{H}_k$, and of $\Delta\mathbf{g}^{(k)}$ and $\Delta\mathbf{x}^{(k)}$. In particular, the BFGS update for $\mathbf{B}_k$ corresponds to the DFP update for $\mathbf{H}_k$. Formulas related in this way are said to be *dual* or *complementary*.

Quasi-Newton Methods

- The Broyden-Fletcher-Goldfarb-Shanno (BFGS) Algorithm

$$\mathbf{H}_{k+1}^{DFP} = \mathbf{H}_k + \frac{\Delta\mathbf{x}^{(k)}\Delta\mathbf{x}^{(k)T}}{\Delta\mathbf{x}^{(k)T}\Delta\mathbf{g}^{(k)}} - \frac{\mathbf{H}_k\Delta\mathbf{g}^{(k)}\Delta\mathbf{g}^{(k)T}\mathbf{H}_k}{\Delta\mathbf{g}^{(k)T}\mathbf{H}_k\Delta\mathbf{g}^{(k)}}.$$

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\Delta\mathbf{g}^{(k)}\Delta\mathbf{g}^{(k)T}}{\Delta\mathbf{g}^{(k)T}\Delta\mathbf{x}^{(k)}} - \frac{\mathbf{B}_k\Delta\mathbf{x}^{(k)}\Delta\mathbf{x}^{(k)T}\mathbf{B}_k}{\Delta\mathbf{x}^{(k)T}\mathbf{B}_k\Delta\mathbf{x}^{(k)}}.$$

$$\begin{aligned}\mathbf{H}_{k+1}^{BFGS} &= (\mathbf{B}_{k+1})^{-1} \\ &= \left(\mathbf{B}_k + \frac{\Delta\mathbf{g}^{(k)}\Delta\mathbf{g}^{(k)T}}{\Delta\mathbf{g}^{(k)T}\Delta\mathbf{x}^{(k)}} - \frac{\mathbf{B}_k\Delta\mathbf{x}^{(k)}\Delta\mathbf{x}^{(k)T}\mathbf{B}_k}{\Delta\mathbf{x}^{(k)T}\mathbf{B}_k\Delta\mathbf{x}^{(k)}}\right)^{-1}.\end{aligned}$$

Proposition 3 (The Sherman-Morrison formula)**.**

$$(\mathbf{A}+\mathbf{UCV}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1}+\mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1},$$

where $\mathbf{A}$ *and* $\mathbf{C}$ *are invertible and the sizes of* $\mathbf{U}$, $\mathbf{C}$, *and* $\mathbf{V}$ *are compatible.*

Quasi-Newton Methods

- The Broyden-Fletcher-Goldfarb-Shanno (BFGS) Algorithm

$$\mathbf{H}_{k+1}^{BFGS} = \mathbf{H}_k + \left(1 + \frac{\Delta\mathbf{g}^{(k)T}\mathbf{H}_k\Delta\mathbf{g}^{(k)}}{\Delta\mathbf{g}^{(k)T}\Delta\mathbf{x}^{(k)}}\right)\frac{\Delta\mathbf{x}^{(k)}\Delta\mathbf{x}^{(k)T}}{\Delta\mathbf{x}^{(k)T}\Delta\mathbf{g}^{(k)}} - \frac{\mathbf{H}_k\Delta\mathbf{g}^{(k)}\Delta\mathbf{x}^{(k)T} + \left(\mathbf{H}_k\Delta\mathbf{g}^{(k)}\Delta\mathbf{x}^{(k)T}\right)^T}{\Delta\mathbf{g}^{(k)T}\Delta\mathbf{x}^{(k)}}.$$

For the quadratic case, the DFP algorithm satisfies $\mathbf{H}_{k+1}^{DFP}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, 0 \leq i \leq k$. Therefore, the BFGS update for $\mathbf{B}_k$ satisfies $\mathbf{B}_{k+1}\Delta\mathbf{x}^{(i)} = \Delta\mathbf{g}^{(i)}, 0 \leq i \leq k$. By construction of the BFGS formula for $\mathbf{H}_{k+1}^{BFGS}$, we conclude that $\mathbf{H}_{k+1}^{BFGS}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, 0 \leq i \leq k$. Hence, the BFGS algorithm enjoys all the properties of quasi-Newton methods, including the conjugate directions property. Moreover, the BFGS algorithm also inherits the positive definiteness property of the DFP algorithm; that is, if $\mathbf{g}^{(k)} \neq \mathbf{0}$ and $\mathbf{H}_k \succ \mathbf{0}$, then $\mathbf{H}_{k+1}^{BFGS} \succ \mathbf{0}$.

The BFGS update is reasonably robust when the line searches are sloppy. This property allows us to save time in the line search part of the algorithm. The BFGS formula is often far more efficient than the DFP formula.

Quasi-Newton Methods

- The Broyden-Fletcher-Goldfarb-Shanno (BFGS) Algorithm

Example: Use the BFGS method to minimize

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{x}^T\mathbf{b} + \log(\pi), \text{ where } \mathbf{Q} = \begin{bmatrix} 5 & -3 \\ -3 & 2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

For nonquadratic problems, quasi-Newton algorithms will not usually converge in n steps. As in the case of the conjugate gradient methods, here too some modifications may be necessary to deal with nonquadratic problems. For example, we may reinitialize the direction vector to the negative gradient after every few iterations (e.g., n or $n+1$), and continue until the algorithm satisfies the stopping criterion.

Quasi-Newton Methods

- Limited-Memory Quasi-Newton Methods

Limited-memory quasi-Newton methods are useful for solving large problems whose Hessian matrices cannot be computed at a reasonable cost or are not sparse. These methods maintain simple and compact approximations of Hessian matrices: Instead of storing fully dense $n \times n$ approximations, they save only a few vectors of length n that represent the approximations implicitly. Despite these modest storage requirements, they often yield an acceptable (albeit linear) rate of convergence. Various limited-memory methods have been proposed; we focus mainly on an algorithm known as L-BFGS, which, as its name suggests,is based on the BFGS updating formula. The main idea of this method is to use curvature information from only the most recent iterations to construct the Hessian approximation. Curvature information from earlier iterations, which is less likely to be relevant to the actual behavior of the Hessian at the current iteration, is discarded in the interest of saving storage.

Quasi-Newton Methods

- Limited-Memory BFGS

BFGS:

$$\mathbf{H}_{k+1} = \mathbf{V}_k^T \mathbf{H}_k \mathbf{V}_k + \rho_k \Delta\mathbf{x}_k \Delta\mathbf{x}_k^T, \tag{1}$$

where

$$\rho_k = \frac{1}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k}, \quad \mathbf{V}_k = \mathbf{I} - \rho_k \Delta\mathbf{g}_k \Delta\mathbf{x}_k^T. \tag{2}$$

for updating $\mathbf{x}$

Since the inverse Hessian approximation $\mathbf{H}_k$ will generally be dense, the cost of storing and manipulating it is prohibitive when the number of variables is large. To circumvent this problem, we store a modified version of $\mathbf{H}_k$ implicitly, by storing a certain number (say, m) of the vector pairs $\{\Delta\mathbf{x}_i, \Delta\mathbf{g}_i\}$ used in the formulae (1)-(2). The product $\mathbf{H}_k\mathbf{g}_k$ can be obtained by performing a sequence of inner products and vector summations involving $\mathbf{g}_k$and the pairs $\{\Delta\mathbf{x}_i, \Delta\mathbf{g}_i\}$. After the new iterate is computed, the oldest vector pair in the set of pairs $\{\Delta\mathbf{x}_i, \Delta\mathbf{g}_i\}$ is replaced by the new pair $\{\Delta\mathbf{x}_k, \Delta\mathbf{g}_k\}$. In this way, the set of vector pairs includes curvature information from the m most recent iterations. Practical experience has shown that modest values of m (between 3 and 20, say) often produce satisfactory results.

Quasi-Newton Methods

- Limited-Memory BFGS

More details: At iteration k, the current iterate is $\mathbf{x}_k$ and the set of vector pairs is given by $\{\Delta\mathbf{x}_i, \Delta\mathbf{g}_i\}$ for $i = k-m, \ldots, k-1$. We first choose some initial Hessian approximation $\mathbf{H}_k^0$ (in contrast to the standard BFGS iteration, this initial approximation is allowed to vary from iteration to iteration) and find by repeated application of the formula (1) that the L-BFGS approximation $\mathbf{H}_k$ satisfies the following formula:

$$\begin{aligned}\mathbf{H}_k =&(\mathbf{V}_{k-1}^T \cdots \mathbf{V}_{k-m}^T)\mathbf{H}_k^0(\mathbf{V}_{k-m} \cdots \mathbf{V}_{k-1}) \\ &+ \rho_{k-m}(\mathbf{V}_{k-1}^T \cdots \mathbf{V}_{k-m+1}^T)\Delta\mathbf{x}_{k-m}\Delta\mathbf{x}_{k-m}^T(\mathbf{V}_{k-m+1} \cdots \mathbf{V}_{k-1}) \\ &+ \rho_{k-m+1}(\mathbf{V}_{k-1}^T \cdots \mathbf{V}_{k-m+2}^T)\Delta\mathbf{x}_{k-m+1}\Delta\mathbf{x}_{k-m+1}^T(\mathbf{V}_{k-m+2} \cdots \mathbf{V}_{k-1}) \\ &+ \cdots \\ &+ \rho_{k-1}\Delta\mathbf{x}_{k-1}\Delta\mathbf{x}_{k-1}^T. \end{aligned} \tag{3}$$

From this expression we can derive a recursive procedure to compute the product $\mathbf{H}_k\mathbf{g}_k$ efficiently.

Quasi-Newton Methods

- Limited-Memory BFGS

Algorithm 1 L-BFGS two-loop recursion

$\mathbf{q} \leftarrow \mathbf{g}_k$;
for $i = k-1, k-2, \ldots, k-m$ **do**
  $\alpha_i \leftarrow \rho_i \Delta\mathbf{x}_i^T \mathbf{q}$;
  $\mathbf{q} \leftarrow \mathbf{q} - \alpha_i \Delta\mathbf{g}_i$;
end for
$\mathbf{p} \leftarrow \mathbf{H}_k^0 \mathbf{q}$;
for $i = k-m, k-m+1, \ldots, k-1$ **do**
  $\beta \leftarrow \rho_i \Delta\mathbf{g}_i^T \mathbf{p}$;
  $\mathbf{p} \leftarrow \mathbf{p} + (\alpha_i - \beta)\Delta\mathbf{x}_i$;
end for
stop with result $\mathbf{H}_k \mathbf{g}_k = \mathbf{p}$.

Quasi-Newton Methods

- Limited-Memory BFGS

Without considering the multiplication $\mathbf{H}_k^0\mathbf{q}$, the two-loop recursion scheme requires $4mn$ multiplications; if $\mathbf{H}_k^0$ is diagonal, then n additional multiplications are needed. Apart from being inexpensive, this recursion has the advantage that the multiplication by the initial matrix $\mathbf{H}_k^0$ is isolated from the rest of the computations, allowing this matrix to be chosen freely and to vary between iterations. We may even use an implicit choice of $\mathbf{H}_k^0$ by defining some initial approximation $\mathbf{B}_k^0$ to the Hessian (not its inverse) and obtaining $\mathbf{p}$ by solving the system $\mathbf{B}_k^0\mathbf{p} = \mathbf{q}$.

A method for choosing $\mathbf{H}_k^0$ that has proved effective in practice is to set $\mathbf{H}_k^0 = \gamma_k\mathbf{I}$, where

$$\gamma_k = \frac{\Delta\mathbf{x}_{k-1}^T\Delta\mathbf{g}_{k-1}}{\Delta\mathbf{g}_{k-1}^T\Delta\mathbf{g}_{k-1}}. \tag{4}$$

γ_k is the scaling factor that attempts to estimate the size of the true Hessian matrix along the most recent search direction. This choice helps to ensure that the search direction $\mathbf{p}_k$ is well scaled, and as a result the step length $\alpha_k = 1$ is accepted in most iterations.

Quasi-Newton Methods

- Limited-Memory BFGS

It is important that the line search be based on the Wolfe conditions:

$$\begin{aligned} f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) &\leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f_k^T \mathbf{p}_k, \\ \nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k &\geq c_2 \nabla f_k^T \mathbf{p}_k, \end{aligned} \tag{5}$$

with $0 < c_1 < c_2 < 1$, or strong Wolfe conditions

$$\begin{aligned} f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) &\leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f_k^T \mathbf{p}_k, \\ |\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k| &\leq c_2 |\nabla f_k^T \mathbf{p}_k|, \end{aligned} \tag{6}$$

so that BFGS updating is stable.

Quasi-Newton Methods

- Limited-Memory BFGS

Algorithm 2 L-BFGS

Choose starting point $\mathbf{x}_0$, integer $m > 0$;
$k \leftarrow 0$;
repeat
  Choose $\mathbf{H}_k^0$ (for example, by using (4));
  Compute $\mathbf{p}_k \leftarrow -\mathbf{H}_k \nabla f_k$ from Algorithm 1;
  Compute $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$, where α_k is chosen to satisfy the Wolfe conditions (5);
  if $k > m$ **then**
    Discard the vector pair $\{\Delta\mathbf{x}_{k-m}, \Delta\mathbf{g}_{k-m}\}$ from storage;
  end if
  Compute and save $\Delta\mathbf{x}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$, $\Delta\mathbf{g}_k = \nabla f_{k+1} - \nabla f_k$;
  $k \leftarrow k + 1$;
until convergence.

Quasi-Newton Methods

- Limited-Memory BFGS

The strategy of keeping the m most recent correction pairs $\{\Delta\mathbf{x}_i, \Delta\mathbf{g}_i\}$ works well in practice; indeed no other strategy has yet proved to be consistently better. During its first $m-1$ iterations, Algorithm 2 is equivalent to the BFGS algorithm if the initial matrix $\mathbf{H}_0$ is the same in both methods, and if L-BFGS chooses $\mathbf{H}_k^0 = \mathbf{H}_0$ at each iteration.

Quasi-Newton Methods

- Limited-Memory BFGS

Table 1 presents results illustrating the behavior of Algorithm 2 for various levels of memory m. The table shows that the algorithm tends to be less robust when m is small. As the amount of storage increases, the number of function evaluations tends to decrease; but since the cost of each iteration increases with the amount of storage, the best CPU time is often obtained for small values of m. Clearly, the optimal choice of m is problem dependent.

Table 1: Performance of Algorithm 2

Problem	n	L-BFGS $m = 3$		L-BFGS $m = 5$		L-BFGS $m = 17$		L-BFGS $m = 29$	
		nfg	time	nfg	time	nfg	time	nfg	time
DIXMAANL	1500	146	16.5	134	17.4	120	28.2	125	44.4
EIGENALS	110	821	21.5	569	15.7	363	16.2	168	12.5
FREUROTH	1000	> 999	—	> 999	—	69	8.1	38	6.3
TRIDIA	1000	876	46.6	611	41.4	531	84.6	462	127.1

Quasi-Newton Methods

- Limited-Memory BFGS

Because some rival algorithms are inefficient, Algorithm 2 is often the approach of choice for large problems in which the true Hessian is not sparse. In particular, a Newton method in which the exact Hessian is computed and factorized is not practical in such circumstances. The L-BFGS approach may also outperform Hessian-free Newton methods such as Newton-CG approaches, in which Hessian-vector products are calculated by finite differences or automatic differentiation. The main weakness of the L-BFGS method is that it converges slowly on ill-conditioned problems. On certain applications, the nonlinear conjugate gradient methods are competitive with limited-memory quasi-Newton methods.

Quasi-Newton Methods

- Relationship with Conjugate Gradient Methods

We start by considering the Hestenes-Stiefel form of the nonlinear conjugate gradient method. Recalling that $\Delta\mathbf{x}_k = \alpha_k \mathbf{p}_k$, the search direction for this method is given by

$$\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \frac{\mathbf{g}_{k+1}^T \Delta\mathbf{g}_k}{\Delta\mathbf{g}_k^T \mathbf{p}_k}\mathbf{p}_k = -\left(\mathbf{I} - \frac{\Delta\mathbf{x}_k \Delta\mathbf{g}_k^T}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k}\right)\mathbf{g}_{k+1} \equiv -\hat{\mathbf{H}}_{k+1}\mathbf{g}_{k+1}. \quad (7)$$

This formula resembles a quasi-Newton iteration, but the matrix $\hat{\mathbf{H}}_{k+1}$ is neither symmetric nor positive definite. We could symmetrize it as $\hat{\mathbf{H}}_{k+1}^T \hat{\mathbf{H}}_{k+1}$, but this matrix does not satisfy the secant equation $\hat{\mathbf{H}}_{k+1}\Delta\mathbf{g}_k = \Delta\mathbf{x}_k$ and is, in any case, singular. An iteration matrix that is symmetric, positive definite, and satisfies the secant equation is given by

$$\mathbf{H}_{k+1} = \left(\mathbf{I} - \frac{\Delta\mathbf{x}_k \Delta\mathbf{g}_k^T}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k}\right)\left(\mathbf{I} - \frac{\Delta\mathbf{g}_k \Delta\mathbf{x}_k^T}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k}\right) + \frac{\Delta\mathbf{x}_k \Delta\mathbf{x}_k^T}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k}. \quad (8)$$

Quasi-Newton Methods

- Relationship with Conjugate Gradient Methods

This matrix is exactly the one obtained by applying a single BFGS update (1) to the identity matrix. Hence, an algorithm whose search direction is given by $\mathbf{p}_{k+1} = -\mathbf{H}_{k+1}\mathbf{g}_{k+1}$, with $\mathbf{H}_{k+1}$ defined by (8), can be thought of as a "memoryless" BFGS method, in which the previous Hessian approximation is always reset to the identity matrix before updating it and where only the most recent correction pair $(\Delta\mathbf{x}_k, \Delta\mathbf{g}_k)$ is kept at every iteration. Alternatively, we can view the method as a variant of Algorithm 2 in which $m = 1$ and $\mathbf{H}_k^0 = \mathbf{I}$ at each iteration.

Quasi-Newton Methods

- Relationship with Conjugate Gradient Methods

A more direct connection with conjugate gradient methods can be seen if we consider the memoryless BFGS formula (8) in conjunction with an exact line search, for which $\mathbf{g}_{k+1}^T \mathbf{p}_k = 0$ for all k. We then obtain

$$\mathbf{p}_{k+1} = -\mathbf{H}_{k+1}\mathbf{g}_{k+1} = -\mathbf{g}_{k+1} + \frac{\mathbf{g}_{k+1}^T \Delta\mathbf{g}_k}{\Delta\mathbf{g}_k^T \mathbf{p}_k}\mathbf{p}_k, \tag{7}$$

which is none other than the Hestenes-Stiefel conjugate gradient method. Moreover, it is easy to verify that when $\mathbf{g}_{k+1}^T \mathbf{p}_k = 0$, the Hestenes-Stiefel formula reduces to the Polak-Ribiere formula. Even though the assumption of exact line searches is unrealistic, it is intriguing that the BFGS formula is related in this way to the Polak-Ribiere and Hestenes-Stiefel methods.

Majorization Minimization

- Basic framework

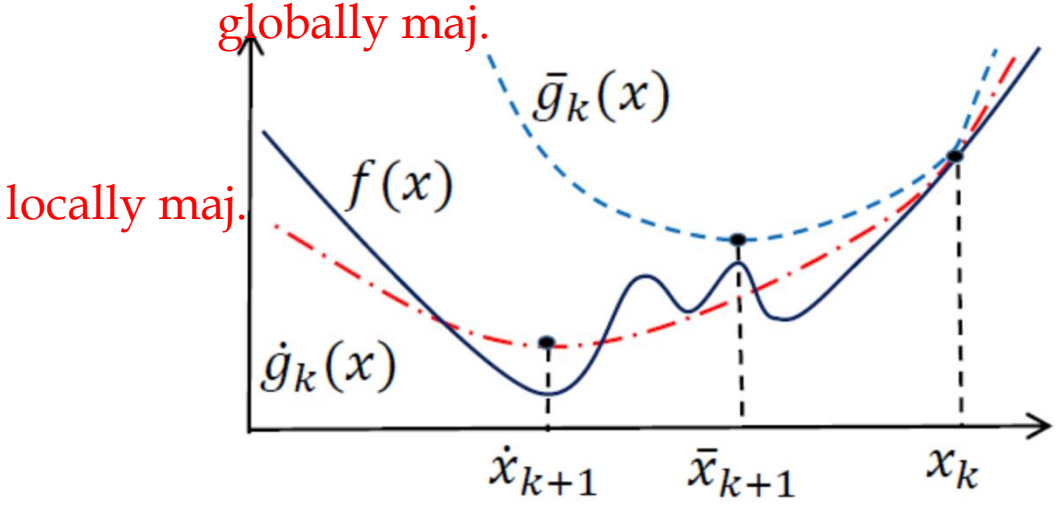
$$\min_{\mathbf{x}} \; f(\mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C}.$$

$$\min_{\mathbf{x}} \; g_k(\mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C}.$$

1. $f(\mathbf{x}) \leq g_k(\mathbf{x}), \forall \mathbf{x} \in \mathcal{C};$ globally majorant

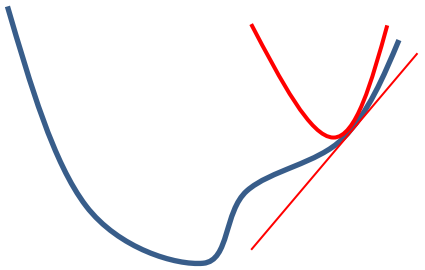
2. $f(\mathbf{x}_k) = g_k(\mathbf{x}_k).$

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}} g_k(\mathbf{x}) \Longrightarrow f(\mathbf{x}_{k+1}) \leq g_k(\mathbf{x}_{k+1}) \leq g_k(\mathbf{x}_k) = f(\mathbf{x}_k).$$



Majorization Minimization

- How to choose the majorant function?



Lipschitz Gradient Surrogate:

$$\min_{\mathbf{x}} \; g_k(\mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C}.$$

$$g_k(\mathbf{x}) = f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}_k\|^2.$$

$$\mathbf{x}_{k+1} = \mathcal{P}_{\mathcal{C}}(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)).$$

projected gradient descent

$$(\mathcal{C} = \mathbb{R}^n)$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k).$$

gradient descent

Majorization Minimization

- How to choose the majorant function?
  - How to choose α?

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)) < f(\mathbf{x}_k).$$ locally majorant

$$\Downarrow$$

If not satisfied $\alpha \leftarrow \mu\alpha$. $(\mu \in (0,1))$ backtracking

If f is L-smooth, i.e., $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$, then we may choose

$$\alpha = L^{-1}.$$

globally majorant

$$f(\mathbf{x}) \leq f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}_k\|^2 \triangleq g_k(\mathbf{x}).$$

Majorization Minimization

- How to choose the majorant function?

Quadratic Surrogate:

$$g_k(\mathbf{x}) = f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T \mathbf{H}(\mathbf{x} - \mathbf{x}_k), \text{ where } \mathbf{H} \succ \nabla^2 f.$$

Majorization Minimization

- How to choose the majorant function?

$$\min_{\mathbf{x}} \; g_k(\mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C}.$$

$$g_k(\mathbf{x}) = f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}_k\|^2.$$

asymptotic smoothness

$g_k(\mathbf{x}) - f(\mathbf{x})$ is smooth.

$g_k(\mathbf{x}) \geq f(\mathbf{x}), \ \forall \mathbf{x}$

globally majorant

$$\lim_{k\to\infty} \nabla g_k(\mathbf{x}_k, \mathbf{d}) - \nabla f(\mathbf{x}_k; \mathbf{d}) = 0.$$

$$g_k(\mathbf{x}_{k+1}) \geq f(\mathbf{x}_{k+1}).$$

locally majorant

Relaxed Majorization Minimization

Robust Matrix Factorization: $\min_{\mathbf{U}\in\mathcal{C}_U, \mathbf{V}\in\mathcal{C}_V} \|\mathbf{W}\odot(\mathbf{M}-\mathbf{U}\mathbf{V}^T)\|_1 + R_u(\mathbf{U}) + R_v(\mathbf{V}).$

Majorization Minimization

- How to choose the majorant function?

$$\min_{\mathbf{x}} \; f(\mathbf{x}) \triangleq \tilde{f}(\boldsymbol{\theta}^T \mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C}. \qquad \boxed{\tilde{f}(x) \text{ is convex}}$$

Jensen Surrogate:

$$g_k(\mathbf{x}) = \sum_{i=1}^{n} w_i \tilde{f}\left(\frac{\theta_i}{w_i}(x_i - x_{k,i}) + \boldsymbol{\theta}^T \mathbf{x}_k \right),$$

where $\mathbf{w} \in \mathbb{R}_+^n$, $\|\mathbf{w}\|_1$ and $w_i \neq 0$ whenever $\theta_i \neq 0$.

Majorization Minimization

- How to choose the majorant function?

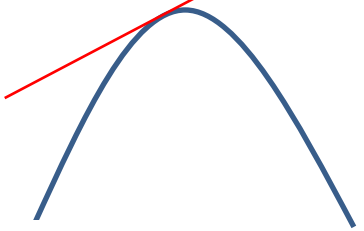
$$\min_{\mathbf{x}} \; f(\mathbf{x}) + h(\mathbf{x}), \quad s.t. \; \mathbf{x} \in \mathcal{C},$$

where f is convex and h is concave.

$$g_k(\mathbf{x}) = f(\mathbf{x}) + \langle \partial h(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + h(\mathbf{x}_k),$$

convex concave procedure (CCCP)

where ∂h is a super-gradient of h.



low-rankness regularizer

$$\min_{\mathbf{X}} \sum_{i=1}^{\min(m,n)} h(\sigma_i(\mathbf{X})) + f(\mathbf{X}), \text{ where } h \text{ is concave on } \mathbb{R}_+.$$

$$h(\sigma_i) \leq h(\sigma_i^k) + w_i^k(\sigma_i - \sigma_i^k), \quad w_i^k \in \partial h(\sigma_i).$$

Majorization Minimization

- How to choose the majorant function?

$$\min_{\mathbf{x}} \; f(\mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C}, \quad \text{where } f(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{Y}} h(\mathbf{x}, \mathbf{y}).$$

Variational surrogate: $g_k(\mathbf{x}) = h(\mathbf{x}, \mathbf{y}_k^*), \quad \text{where } \mathbf{y}_k^* = \operatorname{argmin}_{\mathbf{y} \in \mathcal{Y}} h(\mathbf{x}_k, \mathbf{y}).$

Schatten-p norm: $\|\mathbf{X}\|_{S_p} = \left(\sum_i \sigma_i^p(\mathbf{X}) \right)^{1/p}$, low-rankness regularizer.

Theorem 1. *With compatiable dimensions and* $\frac{1}{p} = \sum_{i=1}^{I} \frac{1}{p_i}$:

$$\frac{1}{p} \|\mathbf{X}\|_{S_p}^p = \min_{\mathbf{X} = \Pi_{i=1}^I \mathbf{X}_i} \sum_{i=1}^{I} \frac{1}{p_i} \|\mathbf{X}_i\|_{S_{p_i}}^{p_i}.$$

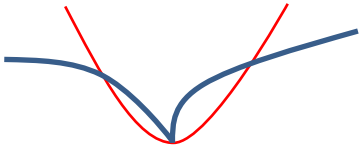
If $0 < p < 1$, we can still choose $p_i \geq 1$.

Majorization Minimization

- How to choose the majorant function?

$$\min_{\mathbf{x}} \; f(\mathbf{x}) + h(\mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C},$$

where f is convex and h is non-convex.



$$g_k(\mathbf{x}) = f(\mathbf{x}) + \mathbf{x}^T \mathbf{H}(\mathbf{x}_k)\mathbf{x},$$

where $\mathbf{H}(\mathbf{x}_k)$ satisfies: $\mathbf{H}(\mathbf{x}_k) \succeq \mathbf{0}$, $\mathbf{x}_k^T \mathbf{H}(\mathbf{x}_k)\mathbf{x}_k = h(\mathbf{x}_k)$ and $\mathbf{x}^T \mathbf{H}(\mathbf{x}_k)\mathbf{x} \geq h(\mathbf{x}), \forall \|\mathbf{x}\| \geq \varepsilon_k$.

l_p-norm, sparsity regularizer

$$h(\mathbf{x}) = \|\mathbf{x}\|_p^p \Longrightarrow \mathbf{x}^T \mathbf{H}(\mathbf{x}_k)\mathbf{x} = \mathbf{x}^T \operatorname{Diag}(|x_{k,i}|^{p-2})\mathbf{x}.$$

$$h(\mathbf{X}) = \operatorname{tr}\left((\mathbf{X}\mathbf{X}^T)^{p/2}\right) \Longrightarrow \langle \mathbf{X}, \mathbf{H}(\mathbf{X}_k)\mathbf{X} \rangle = \operatorname{tr}\left((\mathbf{X}_k\mathbf{X}_k^T)^{p/2-1}\mathbf{X}\mathbf{X}^T\right).$$

Schatten-*p* norm, low-rankness regularizer

Iteratively Reweighted Least Squares