Optimization Techniques for Big Data Analysis

Chapter 5. Second Order Methods

Master of Science in Signal Theory and Communications
Dpto. de Señales, Sistemas y Radiocomunicaciones
E.T.S. Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

2023

1. Newton algorithm

2. Conjugate gradient method.

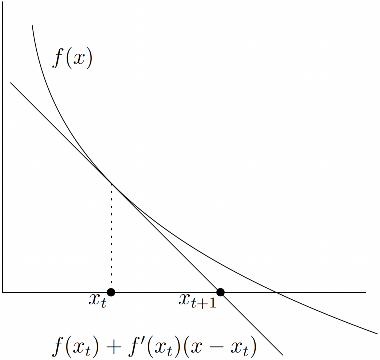
3. Quasi-Newton methods

Basics of Newton algorithm

It has been studied in the previous course *Fundamentals of Optimization* that Newton's method is much faster than gradient descent methods.

$$\begin{aligned} x_{k+1} &= x_k - \eta_k \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \\ &= x_k - \eta_k \triangle x_{New} \end{aligned}$$

due to the effect of the Hessian that makes the Newton step $\triangle x_{New}$ minimises the best (locally) quadratic approximation of $f(\cdot)$.



Basics of Newton algorithm

General update scheme:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \mathbf{G}(\mathbf{x}_t) \nabla f(\mathbf{x}_t)$$

where $\mathbf{G}(\mathbf{x}_t) \in \mathbb{R}^{d \times d}$ is some matrix:

Newton's method: $\mathbf{G}(\mathbf{x}_t) = \left(\nabla^2 f(\mathbf{x}_t)\right)^{-1} = \mathbf{H}^{-1}$

Gradient descent: $\mathbf{G}(\mathbf{x}_t) = \lambda \mathbf{I}$

Basics of Newton algorithm

General update scheme:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \mathbf{G}(\mathbf{x}_t) \nabla f(\mathbf{x}_t)$$

where $\mathbf{G}(\mathbf{x}_t) \in \mathbb{R}^{d \times d}$ is some matrix:

Newton's method: $\mathbf{G}(\mathbf{x}_t) = \left(\nabla^2 f(\mathbf{x}_t)\right)^{-1} = \mathbf{H}^{-1}$

Gradient descent: $\mathbf{G}(\mathbf{x}_t) = \lambda \mathbf{I}$

Newton's method: "adaptive gradient descent", adaptation is w.r.t. the local geometry of the function at $\mathbf{x}_t$.

Basics of Newton algorithm

General update scheme:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \mathbf{G}(\mathbf{x}_t) \nabla f(\mathbf{x}_t)$$

where $\mathbf{G}(\mathbf{x}_t) \in \mathbb{R}^{d \times d}$ is some matrix:

Newton's method: $\mathbf{G}(\mathbf{x}_t) = \left(\nabla^2 f(\mathbf{x}_t)\right)^{-1} = \mathbf{H}^{-1}$

Gradient descent: $\mathbf{G}(\mathbf{x}_t) = \lambda \mathbf{I}$

Newton's method: "adaptive gradient descent", adaptation is w.r.t. the local geometry of the function at $\mathbf{x}_t$.

Unfortunately, calculating $\mathbf{G}$ is unfeasible in most real cases.

Basics of Newton algorithm

General update scheme:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \mathbf{G}(\mathbf{x}_t) \nabla f(\mathbf{x}_t)$$

where $\mathbf{G}(\mathbf{x}_t) \in \mathbb{R}^{d \times d}$ is some matrix:

Newton's method: $\mathbf{G}(\mathbf{x}_t) = \left(\nabla^2 f(\mathbf{x}_t)\right)^{-1} = \mathbf{H}^{-1}$

Gradient descent: $\mathbf{G}(\mathbf{x}_t) = \lambda \mathbf{I}$

Newton's method: "adaptive gradient descent", adaptation is w.r.t. the local geometry of the function at $\mathbf{x}_t$.

Unfortunately, calculating $\mathbf{G}$ is unfeasible in most real cases. We are going to cover two sub-optimal approaches:

1. Conjugate Gradient methods: $\mathbf{H}$ is available, but its inverse is not.
2. Quasi-Newton methods: Approximate $\mathbf{G}$ iteratively using first order information (gradients).

Conjugate gradient method

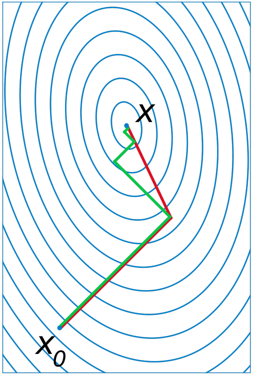
$$\text{Goal:} \quad \arg\min_{\mathbf{x}\in \mathrm{R}^d} \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

$$\text{Equivalently:} \quad \mathbf{A}\mathbf{x} = \mathbf{b}$$

Conjugate gradient method

$$\text{Goal:} \qquad \arg\min_{\mathbf{x}\in\mathrm{R}^d} \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

$$\text{Equivalently:} \qquad \mathbf{A}\mathbf{x} = \mathbf{b}$$

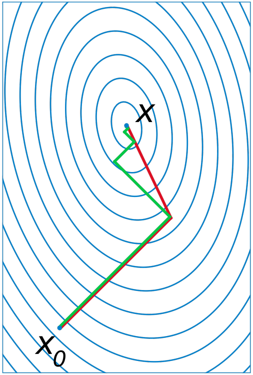


The conjugate gradient method is an iterative method for solving a linear system of equations, where $\mathbf{A}$ is symmetric and positive definite.

Conjugate gradient method

Goal: $\underset{\mathbf{x}\in \mathrm{R}^d}{\arg\min} \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$

Equivalently: $\mathbf{A}\mathbf{x} = \mathbf{b}$



The conjugate gradient method is an iterative method for solving a linear system of equations, where $\mathbf{A}$ is symmetric and positive definite.

Definition: A set of non zero vectors $\{\mathbf{p}_0, \mathbf{p}_1, \cdots \mathbf{p}_{d-1}\}$ is called **A**-orthogonal (conjugate) if:

$$\mathbf{p}_i^T\mathbf{A}\mathbf{p}_j = 0 \qquad \forall i \neq j$$

Intuition behind the conjugate gradient method

Let's consider the updating rule:

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

Intuition behind the conjugate gradient method

Let's consider the updating rule:

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

We can decompose the optimum solution as follows [1]:

$$\begin{aligned} \mathbf{x}_{QP} &= \mathbf{x}_0 + (\mathbf{x}_{QP} - \mathbf{x}_0) \\ &= \mathbf{x}_0 + \textstyle\sum_{j=0}^{d-1} \alpha_j \mathbf{p}_j \end{aligned}$$

Intuition behind the conjugate gradient method

Let's consider the updating rule:

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

We can decompose the optimum solution as follows [1]:

$$\begin{aligned} \mathbf{x}_{QP} &= \mathbf{x}_0 + (\mathbf{x}_{QP} - \mathbf{x}_0) \\ &= \mathbf{x}_0 + \textstyle\sum_{j=0}^{d-1} \alpha_j \mathbf{p}_j \end{aligned}$$

If $\{\mathbf{p}_j\}$ are orthogonal:

$$\mathbf{p}_k^T \mathbf{x}_{QP} = \mathbf{p}_k^T \mathbf{x}_0 + \alpha_k \mathbf{p}_k^T \mathbf{p}_k$$

$$\alpha_k = \frac{\mathbf{p}_k^T (\mathbf{x}_{QP} - \mathbf{x}_0)}{\mathbf{p}_k^T \mathbf{p}_k} = \frac{\mathbf{p}_k^T \mathbf{x}_r}{\mathbf{p}_k^T \mathbf{p}_k}$$

Intuition behind the conjugate gradient method

Let's consider the updating rule:

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

We can decompose the optimum solution as follows [1]:

$$\begin{aligned} \mathbf{x}_{QP} &= \mathbf{x}_0 + (\mathbf{x}_{QP} - \mathbf{x}_0) \\ &= \mathbf{x}_0 + \textstyle\sum_{j=0}^{d-1} \alpha_j \mathbf{p}_j \end{aligned}$$

If $\{\mathbf{p}_j\}$ are orthogonal:

$$\mathbf{p}_k^T \mathbf{x}_{QP} = \mathbf{p}_k^T \mathbf{x}_0 + \alpha_k \mathbf{p}_k^T \mathbf{p}_k$$

$$\alpha_k = \frac{\mathbf{p}_k^T (\mathbf{x}_{QP} - \mathbf{x}_0)}{\mathbf{p}_k^T \mathbf{p}_k} = \frac{\mathbf{p}_k^T \mathbf{x}_r}{\mathbf{p}_k^T \mathbf{p}_k}$$

Intuition behind the conjugate gradient method

Since we know $\mathbf{A}$,

$$\begin{aligned} \mathbf{A}\mathbf{x}_{QP} &= \mathbf{A}\mathbf{x}_0 + \mathbf{A}(\mathbf{x}_{QP} - \mathbf{x}_0) \\ &= \mathbf{A}\mathbf{x}_0 + \textstyle\sum_{j=0}^{d-1} \alpha_j \mathbf{A}\mathbf{p}_j \end{aligned}$$

Intuition behind the conjugate gradient method

Since we know $\mathbf{A}$,

$$\begin{aligned}\mathbf{A}\mathbf{x}_{QP} &= \mathbf{A}\mathbf{x}_0 + \mathbf{A}(\mathbf{x}_{QP} - \mathbf{x}_0) \\ &= \mathbf{A}\mathbf{x}_0 + \sum_{j=0}^{d-1} \alpha_j \mathbf{A}\mathbf{p}_j\end{aligned}$$

Premultiplying by $\mathbf{p}_k^T$, we could get:

$$\alpha_k = \frac{\mathbf{p}_k^T(\mathbf{b} - \mathbf{A}\mathbf{x})}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k} = \frac{-\mathbf{g}_0^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k}$$

only if $\{\mathbf{p}_j\}$ are $\mathbf{A}$-orthogonal.

Intuition behind the conjugate gradient method

Since we know $\mathbf{A}$,

$$\begin{aligned}\mathbf{A}\mathbf{x}_{QP} &= \mathbf{A}\mathbf{x}_0 + \mathbf{A}(\mathbf{x}_{QP} - \mathbf{x}_0) \\ &= \mathbf{A}\mathbf{x}_0 + \sum_{j=0}^{d-1} \alpha_j \mathbf{A}\mathbf{p}_j\end{aligned}$$

Premultiplying by $\mathbf{p}_k^T$, we could get:

$$\alpha_k = \frac{\mathbf{p}_k^T(\mathbf{b} - \mathbf{A}\mathbf{x})}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k} = \frac{-\mathbf{g}_0^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k}$$

only if $\{\mathbf{p}_j\}$ are $\mathbf{A}$-orthogonal. So, we don't need to know the optimum to estimate α_k.

Intuition behind the conjugate gradient method

Since we know $\mathbf{A}$,

$$\begin{aligned} \mathbf{A}\mathbf{x}_{QP} &= \mathbf{A}\mathbf{x}_0 + \mathbf{A}(\mathbf{x}_{QP} - \mathbf{x}_0) \\ &= \mathbf{A}\mathbf{x}_0 + \textstyle\sum_{j=0}^{d-1} \alpha_j \mathbf{A}\mathbf{p}_j \end{aligned}$$

Premultiplying by $\mathbf{p}_k^T$, we could get:

$$\alpha_k = \frac{\mathbf{p}_k^T(\mathbf{b} - \mathbf{A}\mathbf{x})}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k} = \frac{-\mathbf{g}_0^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k}$$

only if $\{\mathbf{p}_j\}$ are $\mathbf{A}$-orthogonal. So, we don't need to know the optimum to estimate α_k.

But, what about $\mathbf{p}_k$?

Intuition behind the conjugate gradient method

$$
\begin{aligned}
\mathbf{p}_0 &= \mathbf{g}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b} \\
\mathbf{x}_1 &= \mathbf{x}_0 + \alpha_0 \mathbf{p}_0
\end{aligned}
$$

Intuition behind the conjugate gradient method

$$
\begin{aligned}
\mathbf{p}_0 &= \mathbf{g}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b} \\
\mathbf{x}_1 &= \mathbf{x}_0 + \alpha_0 \mathbf{p}_0
\end{aligned}
$$

By assuring the gradient at $\mathbf{x}_1$ is orthogonal to $\mathbf{p}_0$,

$$
\mathbf{g}_1^T \mathbf{p}_0 = (\mathbf{A}\mathbf{x}_1 - \mathbf{b})^T \mathbf{p}_0 = 0
$$

and after some replacements,

$$
\alpha_0 = \frac{-\mathbf{g}_0^T \mathbf{p}_0}{\mathbf{p}_0^T \mathbf{A} \mathbf{p}_0}
$$

Intuition behind the conjugate gradient method

$$\begin{aligned} \mathbf{p}_0 &= \mathbf{g}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b} \\ \mathbf{x}_1 &= \mathbf{x}_0 + \alpha_0 \mathbf{p}_0 \end{aligned}$$

By assuring the gradient at $\mathbf{x}_1$ is orthogonal to $\mathbf{p}_0$,

$$\mathbf{g}_1^T \mathbf{p}_0 = (\mathbf{A}\mathbf{x}_1 - \mathbf{b})^T \mathbf{p}_0 = 0$$

and after some replacements,

$$\alpha_0 = \frac{-\mathbf{g}_0^T \mathbf{p}_0}{\mathbf{p}_0^T \mathbf{A} \mathbf{p}_0}$$

Now that we can estimate $\mathbf{x}_1$,

$$\begin{aligned} \mathbf{g}_1 &= \mathbf{A}\mathbf{x}_1 - \mathbf{b} \\ \mathbf{x}_2 &= \mathbf{x}_1 + \alpha_1 \mathbf{p}_1 \end{aligned}$$

Intuition behind the conjugate gradient method

The composite direction is:

$$\mathbf{p}_1 = \mathbf{g}_1 + \beta_1 \mathbf{p}_0$$

How do we define β_1?

Intuition behind the conjugate gradient method

The composite direction is:

$$\mathbf{p}_1 = \mathbf{g}_1 + \beta_1 \mathbf{p}_0$$

How do we define β_1?

$$\mathbf{p}_1^T \mathbf{A} \mathbf{p}_0 = \mathbf{g}_1^T \mathbf{A} \mathbf{p}_0 + \beta_1 \mathbf{p}_0^T \mathbf{A} \mathbf{p}_0$$

$$\beta_1 = \frac{-\mathbf{g}_1^T \mathbf{A} \mathbf{p}_0}{\mathbf{p}_0^T \mathbf{A} \mathbf{p}_0}$$

Intuition behind the conjugate gradient method

The composite direction is:

$$\mathbf{p}_1 = \mathbf{g}_1 + \beta_1 \mathbf{p}_0$$

How do we define β_1?

$$\mathbf{p}_1^T \mathbf{A} \mathbf{p}_0 = \mathbf{g}_1^T \mathbf{A} \mathbf{p}_0 + \beta_1 \mathbf{p}_0^T \mathbf{A} \mathbf{p}_0$$

$$\beta_1 = \frac{-\mathbf{g}_1^T \mathbf{A} \mathbf{p}_0}{\mathbf{p}_0^T \mathbf{A} \mathbf{p}_0}$$

and using a similar previous analysis, we can carry out the required update:

$$\alpha_1 = \frac{-\mathbf{g}_1^T \mathbf{p}_1}{\mathbf{p}_1^T \mathbf{A} \mathbf{p}_1}$$

Conjugate gradient method. Algorithm description

Giving a starting point $\mathbf{x}_0$, making $\mathbf{p}_0 = \mathbf{g}_0$ and $\beta_0 = 0$, the algorithm is represented by:

1. Initialize: $k = 0$
2. While $\mathbf{g}_k \neq 0$
3. $\qquad \alpha_k = \frac{-\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$
4. $\qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
5. $\qquad \mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1}) = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}$
6. $\qquad \beta_{k+1} = \frac{-\mathbf{g}_{k+1}^T \mathbf{A} \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$
7. $\qquad \mathbf{p}_{k+1} = \mathbf{g}_{k+1} + \beta_{k+1} \mathbf{p}_k$
8. $\qquad k = k + 1$

It can be shown that this algorithm converges to the solution $\mathbf{x}_{QP}$ in at most d steps.

Example 5.1

Solve the following system of equation $\mathbf{Ax} = \mathbf{b}$ where

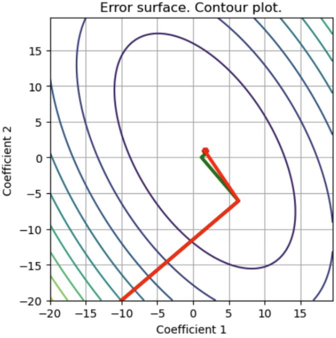
$$\mathbf{A} = \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \text{ starting from } \mathbf{x}_0 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

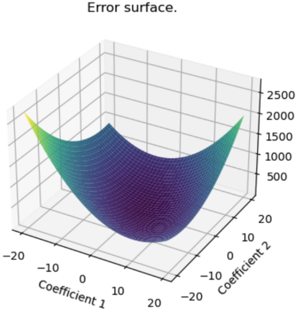
As this is a quadratic problem, we just need two iterations to solve the problem. Implement the algorithm and verify it.

Solution: $[0.09090909, 0.63636364]$

Example 5.2

Have a look at example 5.2 on the repository; it corresponds to applying the Conjugate gradient for solving ridge regression.





Quasi-Newton methods. Basic idea

$$\begin{aligned}
\text{GD:}\quad & \mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{I} \nabla f(\mathbf{x}_k) \\
\text{Newton:}\quad & \mathbf{x}_{k+1} = \mathbf{x}_k - \eta (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k) \\
\text{Quasi-N:}\quad & \mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{G}_k \nabla f(\mathbf{x}_k)
\end{aligned}$$

Quasi-Newton methods hope for:

1. $\mathbf{G}_k$ is more useful than $\mathbf{I}$
2. $\mathbf{G}_k$ is less expensive to compute than the inverse of the Hessian.

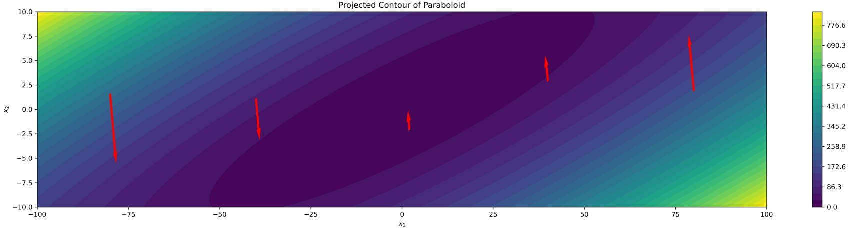
Quasi-Newton methods. Basic idea

$$\text{GD:} \quad \mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{I} \nabla f(\mathbf{x}_k)$$
$$\text{Newton:} \quad \mathbf{x}_{k+1} = \mathbf{x}_k - \eta (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$
$$\text{Quasi-N:} \quad \mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{G}_k \nabla f(\mathbf{x}_k)$$

Quasi-Newton methods hope for:

1. $\mathbf{G}_k$ is more useful than $\mathbf{I}$
2. $\mathbf{G}_k$ is less expensive to compute than the inverse of the Hessian.

A possible naïve approach:



Quasi-Newton methods. Naïve approach

If elongation is almost aligned with the coordinate, we could get an acceptable solution by rescaling.

$$\mathbf{G} = \left(\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix} \right)^{-1}$$

Quasi-Newton methods. Naïve approach

If elongation is almost aligned with the coordinate, we could get an acceptable solution by rescaling.

$$\mathbf{G} = \left(\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix} \right)^{-1}$$

What if elongation is not well aligned to axes?

Quasi-Newton methods. Naïve approach

If elongation is almost aligned with the coordinate, we could get an acceptable solution by rescaling.

$$\mathbf{G} = \left(\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix} \right)^{-1}$$

What if elongation is not well aligned to axes?
Example:

$$\mathbf{H} = \begin{bmatrix} 1 & 0.99 \\ 0.99 & 1 \end{bmatrix}$$

The approximation would equal the identity, and we get just **GD**.

Quasi-Newton methods. Basic formulation

Goal: Approximate $\mathbf{H}$ without requiring expensive computation

Key idea: Use curvature information along the generated trajectory to build the approximation recursively.

Quasi-Newton methods. Basic formulation

Goal: Approximate $\mathbf{H}$ without requiring expensive computation

Key idea: Use curvature information along the generated trajectory to build the approximation recursively.

The formulation starts from a quadratic approximation [2]:

$$\tilde{f}(\mathbf{x}_k + \Delta_{\mathbf{x}}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \Delta_{\mathbf{x}} + \frac{1}{2} \Delta_{\mathbf{x}}^T \mathbf{B}_k \Delta_{\mathbf{x}}$$

Quasi-Newton methods. Basic formulation

Goal: Approximate $\mathbf{H}$ without requiring expensive computation

Key idea: Use curvature information along the generated trajectory to build the approximation recursively.

The formulation starts from a quadratic approximation [2]:

$$\tilde{f}(\mathbf{x}_k + \Delta_{\mathbf{x}}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \Delta_{\mathbf{x}} + \frac{1}{2} \Delta_{\mathbf{x}}^T \mathbf{B}_k \Delta_{\mathbf{x}}$$

The minimizer $\Delta_{\mathbf{x}_k}$ of this convex quadratic model is

$$\Delta_{\mathbf{x}_k} = -\mathbf{B}_k^{-1} \nabla f(\mathbf{x}_k)$$

Quasi-Newton methods. Basic formulation

Goal: Approximate $\mathbf{H}$ without requiring expensive computation

Key idea: Use curvature information along the generated trajectory to build the approximation recursively.

The formulation starts from a quadratic approximation [2]:

$$\tilde{f}(\mathbf{x}_k + \Delta_{\mathbf{x}}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \Delta_{\mathbf{x}} + \frac{1}{2}\Delta_{\mathbf{x}}^T \mathbf{B}_k \Delta_{\mathbf{x}}$$

The minimizer $\Delta_{\mathbf{x}_k}$ of this convex quadratic model is

$$\Delta_{\mathbf{x}_k} = -\mathbf{B}_k^{-1} \nabla f(\mathbf{x}_k)$$

So, **QN**: $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \mathbf{B}_k^{-1} \nabla f(\mathbf{x}_k) = \mathbf{x}_k + \eta_k \Delta_{\mathbf{x}_k}$

Update $\mathbf{B}_k$ iteratively.

Quasi-Newton methods. Basic formulation

The point is choosing a feasible $\mathbf{B}_{k+1}$.

- We would like $\mathbf{B}_k^{-1}\nabla f(\mathbf{x}_k)$ to be easy to compute.
- We require $\tilde{f}(\mathbf{x}_{k+1}+\Delta_{\mathbf{x}})$ matches the gradient of $f(\cdot)$ at the last two iterations (it is matching curvature at two points).

Quasi-Newton methods. Basic formulation

The point is choosing a feasible $\mathbf{B}_{k+1}$.

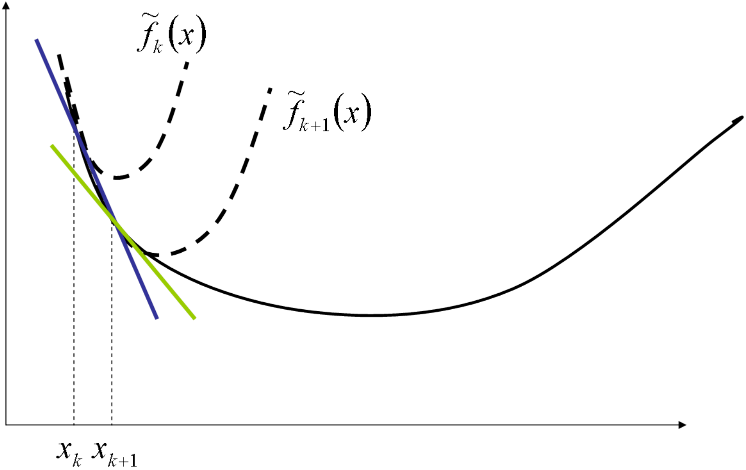
- We would like $\mathbf{B}_k^{-1}\nabla f(\mathbf{x}_k)$ to be easy to compute.
- We require $\tilde{f}(\mathbf{x}_{k+1}+\Delta_{\mathbf{x}})$ matches the gradient of $f(\cdot)$ at the last two iterations (it is matching curvature at two points).

Let's represent $\tilde{f}(\mathbf{x}_{k+1}+\Delta_{\mathbf{x}}) = \tilde{f}_{k+1}(\Delta_{\mathbf{x}})$, the last condition implies:

$$\begin{aligned} \nabla \tilde{f}_{k+1}\Big|_{\Delta_{\mathbf{x}}=0} &= \nabla f\left(\mathbf{x}_{k+1}\right) \\ \nabla \tilde{f}_{k+1}\Big|_{\Delta_{\mathbf{x}}=-\eta_k \Delta_{\mathbf{x}_k}} &= \nabla f\left(\mathbf{x}_{k}\right) \end{aligned}$$

These conditions mean that $\mathbf{B}_k$ locally approximates the Hessian.

Quasi-Newton methods. Basic formulation



Quasi-Newton methods. Basic formulation

Let's check what the conditions imply,

$$\tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = f(\mathbf{x}_{k+1}) + \nabla f(\mathbf{x}_{k+1})^T \Delta_{\mathbf{x}} + \frac{1}{2}\Delta_{\mathbf{x}}^T \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

Quasi-Newton methods. Basic formulation

Let's check what the conditions imply,

$$\tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = f(\mathbf{x}_{k+1}) + \nabla f(\mathbf{x}_{k+1})^T \Delta_{\mathbf{x}} + \frac{1}{2}\Delta_{\mathbf{x}}^T \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

By differentiating with respect to $\Delta_{\mathbf{x}}$,

$$\nabla \tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = \nabla f(\mathbf{x}_{k+1}) + \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

Quasi-Newton methods. Basic formulation

Let's check what the conditions imply,

$$\tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = f(\mathbf{x}_{k+1}) + \nabla f(\mathbf{x}_{k+1})^T \Delta_{\mathbf{x}} + \frac{1}{2}\Delta_{\mathbf{x}}^T \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

By differentiating with respect to $\Delta_{\mathbf{x}}$,

$$\nabla \tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = \nabla f(\mathbf{x}_{k+1}) + \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

Therefore,

(i) $\nabla \tilde{f}_{k+1}(0) = \nabla f(\mathbf{x}_{k+1})$ ✓

Quasi-Newton methods. Basic formulation

Let's check what the conditions imply,

$$\tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = f(\mathbf{x}_{k+1}) + \nabla f(\mathbf{x}_{k+1})^T \Delta_{\mathbf{x}} + \frac{1}{2}\Delta_{\mathbf{x}}^T \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

By differentiating with respect to $\Delta_{\mathbf{x}}$,

$$\nabla \tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = \nabla f(\mathbf{x}_{k+1}) + \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

Therefore,

(i) $\nabla \tilde{f}_{k+1}(0) = \nabla f(\mathbf{x}_{k+1})$ ✓

(ii) $\nabla \tilde{f}_{k+1}(-\eta_k \Delta_{\mathbf{x}_k}) = \nabla f(\mathbf{x}_{k+1}) - \eta_k \mathbf{B}_{k+1} \Delta_{\mathbf{x}_k}$

Quasi-Newton methods. Basic formulation

Let's check what the conditions imply,

$$\tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = f(\mathbf{x}_{k+1}) + \nabla f(\mathbf{x}_{k+1})^T \Delta_{\mathbf{x}} + \frac{1}{2} \Delta_{\mathbf{x}}^T \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

By differentiating with respect to $\Delta_{\mathbf{x}}$,

$$\nabla \tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = \nabla f(\mathbf{x}_{k+1}) + \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

Therefore,

(*i*) $\nabla \tilde{f}_{k+1}(0) = \nabla f(\mathbf{x}_{k+1})$ ✓

(*ii*) $\nabla \tilde{f}_{k+1}(-\eta_k \Delta_{\mathbf{x}_k}) = \nabla f(\mathbf{x}_{k+1}) - \eta_k \mathbf{B}_{k+1} \Delta_{\mathbf{x}_k} = \nabla f(\mathbf{x}_k)$

Quasi-Newton methods. Basic formulation

Let's check what the conditions imply,

$$\tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = f(\mathbf{x}_{k+1}) + \nabla f(\mathbf{x}_{k+1})^T \Delta_{\mathbf{x}} + \frac{1}{2}\Delta_{\mathbf{x}}^T \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

By differentiating with respect to $\Delta_{\mathbf{x}}$,

$$\nabla \tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = \nabla f(\mathbf{x}_{k+1}) + \mathbf{B}_{k+1}\Delta_{\mathbf{x}}$$

Therefore,

(i) $\nabla \tilde{f}_{k+1}(0) = \nabla f(\mathbf{x}_{k+1})$ ✓

(ii) $\nabla \tilde{f}_{k+1}(-\eta_k \Delta_{\mathbf{x}_k}) = \nabla f(\mathbf{x}_{k+1}) - \eta_k \mathbf{B}_{k+1}\Delta_{\mathbf{x}_k} = \nabla f\left(\mathbf{x}_k\right)$

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

Quasi-Newton methods. Basic formulation

Let's check what the conditions imply,

$$\tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = f(\mathbf{x}_{k+1}) + \nabla f(\mathbf{x}_{k+1})^T \Delta_{\mathbf{x}} + \frac{1}{2}\Delta_{\mathbf{x}}^T \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

By differentiating with respect to $\Delta_{\mathbf{x}}$,

$$\nabla \tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = \nabla f(\mathbf{x}_{k+1}) + \mathbf{B}_{k+1}\Delta_{\mathbf{x}}$$

Therefore,

(i) $\nabla \tilde{f}_{k+1}(0) = \nabla f(\mathbf{x}_{k+1})$ ✓

(ii) $\nabla \tilde{f}_{k+1}(-\eta_k \Delta_{\mathbf{x}_k}) = \nabla f(\mathbf{x}_{k+1}) - \eta_k \mathbf{B}_{k+1} \Delta_{\mathbf{x}_k} = \nabla f(\mathbf{x}_k)$

$$\mathbf{B}_{k+1}\underbrace{(\mathbf{x}_{k+1} - \mathbf{x}_k)}_{\mathbf{s}_k} = \underbrace{\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)}_{\mathbf{y}_k}$$

Quasi-Newton methods. Basic formulation

Let's check what the conditions imply,

$$\tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = f(\mathbf{x}_{k+1}) + \nabla f(\mathbf{x}_{k+1})^T \Delta_{\mathbf{x}} + \frac{1}{2}\Delta_{\mathbf{x}}^T \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

By differentiating with respect to $\Delta_{\mathbf{x}}$,

$$\nabla \tilde{f}_{k+1}(\Delta_{\mathbf{x}}) = \nabla f(\mathbf{x}_{k+1}) + \mathbf{B}_{k+1} \Delta_{\mathbf{x}}$$

Therefore,

(i) $\nabla \tilde{f}_{k+1}(0) = \nabla f(\mathbf{x}_{k+1})$ ✓

(ii) $\nabla \tilde{f}_{k+1}(-\eta_k \Delta_{\mathbf{x}_k}) = \nabla f(\mathbf{x}_{k+1}) - \eta_k \mathbf{B}_{k+1} \Delta_{\mathbf{x}_k} = \nabla f(\mathbf{x}_k)$

$$\mathbf{B}_{k+1}\mathbf{s}_k = \mathbf{y}_k \;\rightarrow\; \text{Secant equation}$$

Quasi-Newton methods. Basic formulation

For $d > 1$, the secant equation is undetermined.

Quasi-Newton methods. Basic formulation

For $d > 1$, the secant equation is undetermined. So, **QN** algorithms use:

$$\begin{aligned} \mathbf{B}_{k+1} &= \arg\min \|\mathbf{B} - \mathbf{B}_k\| \\ \text{s.t.} \quad & \mathbf{B} = \mathbf{B}^T \\ & \mathbf{B}\mathbf{s}_k = \mathbf{y}_k \end{aligned}$$

Quasi-Newton methods. Basic formulation

For $d > 1$, the secant equation is undetermined. So, **QN** algorithms use:

$$\begin{aligned} \mathbf{B}_{k+1} &= \arg\min \|\mathbf{B} - \mathbf{B}_k\| \\ \text{s.t.} \quad & \mathbf{B} = \mathbf{B}^T \\ & \mathbf{B}\mathbf{s}_k = \mathbf{y}_k \end{aligned}$$

Each choice of the norm $\|\cdot\|$ gives different $\mathbf{B}_{k+1}$ and defines a different **QN** method. The most widely used algorithms uses the **Weigthed Frobenious norm (WFN)**:

$$\|\mathbf{A}\|_W^2 = \|\mathbf{W}^{1/2}\mathbf{A}\mathbf{W}^{1/2}\|_F \tag{1}$$

where $\mathbf{W} = \int_0^1 \nabla^2 f(\mathbf{x}_k + \tau\eta_k\Delta_{\mathbf{x}_k})d\tau$

Quasi-Newton methods. Basic formulation

The previous choice of $\mathbf{W}$ makes Eq. (1) non-dimensional. Its solution gives rise to the method called **Davidon–Fletcher–Powell (DFP)**.

$$\mathbf{B}_{k+1} = (\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T)\mathbf{B}_k(\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T) + \rho_k \mathbf{y}_k \mathbf{y}_k^T, \quad \rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}$$

Quasi-Newton methods. Basic formulation

The previous choice of $\mathbf{W}$ makes Eq. (1) non-dimensional. Its solution gives rise to the method called **Davidon–Fletcher–Powell (DFP)**.

$$\mathbf{B}_{k+1} = (\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T)\mathbf{B}_k(\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T) + \rho_k \mathbf{y}_k \mathbf{y}_k^T, \quad \rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}$$

Since the $\mathbf{x}_{k+1}$ updated rule requires $\mathbf{G}_k = \mathbf{B}_k^{-1}$, the DFP algorithm uses:

$$\mathbf{G}_{k+1} = \mathbf{G}_k - \frac{\mathbf{G}_k \mathbf{y}_k \mathbf{y}_k^T \mathbf{G}_k}{\mathbf{y}_k^T \mathbf{G}_k \mathbf{y_k}} + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}$$

Quasi-Newton methods. Basic formulation

The DFP method was soon superseded by the **Broyden–Fletcher–Goldfarb-Shanno (BFGS)** method, which avoids the need to invert the Hessian calculation and formulates the algorithm to approximate $\mathbf{G}_k$ directly.

Quasi-Newton methods. Basic formulation

The DFP method was soon superseded by the **Broyden–Fletcher–Goldfarb-Shanno (BFGS)** method, which avoids the need to invert the Hessian calculation and formulates the algorithm to approximate $\mathbf{G}_k$ directly.

$$\begin{aligned} \mathbf{G}_{k+1} &= \arg\min \|\mathbf{G} - \mathbf{G}_k\| \\ \text{s.t.} \quad & \mathbf{G} = \mathbf{G}^T \\ & \mathbf{G}\mathbf{y}_k = \mathbf{s}_k \end{aligned}$$

Using the WFN, the solution is

$$\mathbf{G}_{k+1} = (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T)\mathbf{G}_k(\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T$$

BFGS algorithm description

For certain tolerance ϵ, given a starting point $\mathbf{x}_0$ and making $\mathbf{G}_0 = \mathbf{I}$

1. Initialize: $k = 0$
2. While $\|\nabla f\|_2 > \epsilon$
3. $\quad \Delta_{\mathbf{x}_k} = -\mathbf{G}_k \nabla f(\mathbf{x}_k)$
4. $\quad \mathbf{x}_{k+1} = \mathbf{x}_k + \eta_k \Delta_{\mathbf{x}_k}$. Use line search to get η_k
5. $\quad \mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$
6. $\quad \mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$
7. $\quad \rho_k = 1/\mathbf{y}_k^T \mathbf{s}_k$
8. $\quad \mathbf{G}_{k+1} = (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T)\mathbf{G}_k(\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T$
9. $\quad k = k + 1$

Limited memory BFGS

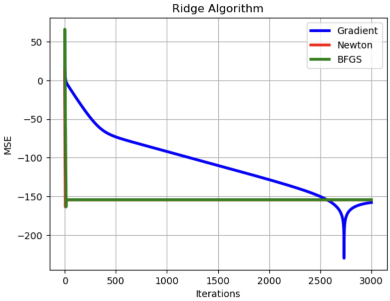
When solving large-scale problems whose Hessian matrices cannot be computed at a reasonable cost or are not sparse, limited memory QN methods maintain simple and compact approximations of the Hessian matrices by saving a few vectors of dimension d instead of the whole matrix.

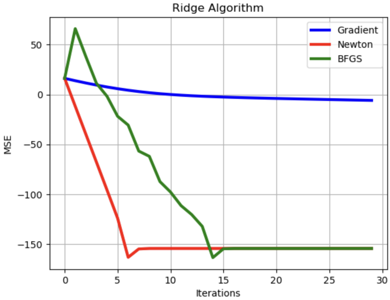
In simple terms, it approximates the product $\mathbf{G}_k \nabla f(\mathbf{x}_k)$ by a sequence of multiplications and summations of m previous $\{\mathbf{s}_i, \mathbf{y}_i\}$; $i = k - m, ..., k - 1$ vectors.

A detailed explanation of it is out of this course's scope.

Case study 5.1

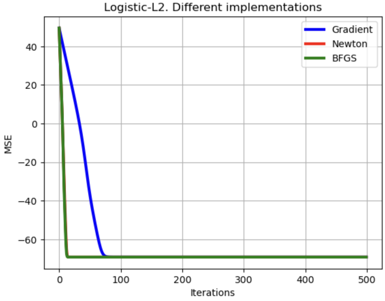
The goal of this case study is to check the performance of algorithms BFGS compared with standard Gradient and Newton solutions for the Ridge problem.

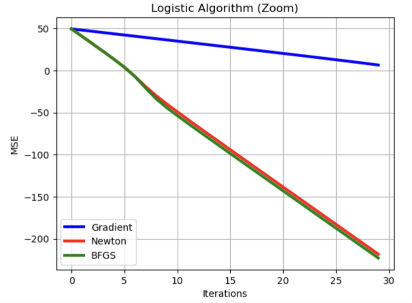




Case study 5.2

The objective now is to check the performance of BFGS compared with standard Gradient and Newton solutions for the Logistic-L2 problem.





It can be noticed that quasi-Newton is as competitive as Newton itself but with much less computational burden.

Acknowledgments

I would like to acknowledge several sources I have used to create slides

- Andrew Reader's course at King's College London. `https://www.youtube.com/@AndrewJReader`
- Constantine Caramanis' course at University of Texas `https://www.youtube.com/@constantine.caramanis`

Questions?

References

[1] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[2] Jorge Nocedal and J. Wright Stephen. *Numerical optimization.* Spinger, 2006.

Thank You

Julián D. Arias-Londoño
julian.arias@upm.es