Optimization Theory and Applications

Kun Zhu (zhukun@nuaa.edu.cn)

November 27, 2018

Introduction

- Conjugate direction methods are intermediate between steepest descent method and Newton's method
    - Efficiency
    - Computation complexity
- Conjugate direction methods have following properties
    - Solve quadratics of n variables in n steps
    - Usual implementations: need only gradient. No need to compute the Hessian
    - No matrix inversion and no storage of an $n \times n$ matrix are required
    - More complicated than steepest descent algorithm

Introduction

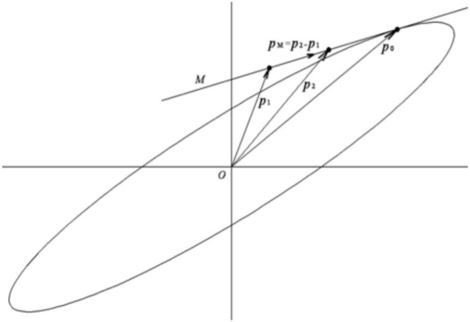
- Typically perform better than steepest descent, but not as well as Newton's method
- The crucial factor in the efficiency of an iterative search method is the direction of search at each iteration
- For certain functions, the best direction of search is in the ***Q-conjugate*** direction

Conjugate Vectors

- Given $Q \in \mathbb{R}^{n\times n}$, symmetric
- Two vectors $\mathbf{d}^{(1)}$ and $\mathbf{d}^{(2)}$ are Q-conjugate if $\mathbf{d}^{(1)T} Q \mathbf{d}^{(2)} = 0$
- ***Definition***: Let Q be a real symmetric $n \times n$ matrix. The directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(m)}$ are Q-conjugate if for all $i \neq j$, we have $\mathbf{d}^{(i)T} Q \mathbf{d}^{(j)} = 0$
- If $Q = \mathbf{I}$, conjugacy reduces to orthogonality

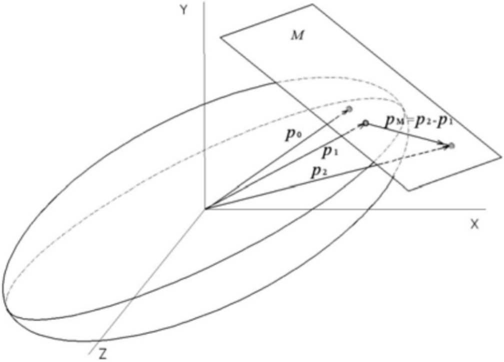
Conjugate Vectors

- Illustration of conjugate: For an ellipsoid
  $$[\mathbf{x}, \mathbf{y}]Q\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} = 1,$$
  We have
  $$\mathbf{p}_0^T Q(\mathbf{p}_2 - \mathbf{p}_1) = \mathbf{p}_0^T Q\mathbf{p}_M = 0$$



Conjugate Vectors

- Illustration of conjugate in $\mathbb{R}^3$



Conjugate Vectors

- ***Example***: Let

$$Q = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}$$

  Our goal is to construct a set of Q-conjugate vectors $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \mathbf{d}^{(2)}$

- Let

  $\mathbf{d}^{(0)} = [1, 0, 0]^T$, $\mathbf{d}^{(1)} = [d_1^{(1)}, d_2^{(1)}, d_3^{(1)}]$, $\mathbf{d}^{(2)} = [d_1^{(2)}, d_2^{(2)}, d_3^{(2)}]$

- We require that $\mathbf{d}^{(0)T} Q \mathbf{d}^{(1)} = 0$

Conjugate Vectors

- We have

$$\mathbf{d}^{(0)T}Q\mathbf{d}^{(1)} = [1,0,0]\begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}\begin{bmatrix} d_1^{(1)} \\ d_2^{(1)} \\ d_3^{(1)} \end{bmatrix} = 3d_1^{(1)} + d_3^{(1)}$$

- Let $d_1^{(1)} = 1, d_2^{(1)} = 0, d_3^{(1)} = -3$, then

$$\mathbf{d}^{(1)} = [1,0,-3]^T$$

and

$$\mathbf{d}^{(0)T}Q\mathbf{d}^{(1)} = 0$$

Conjugate Vectors

- To find $\mathbf{d}^{(2)}$ which is Q-conjugate with $\mathbf{d}^{(0)}$ and $\mathbf{d}^{(1)}$, we require

$$\mathbf{d}^{(0)T}Q\mathbf{d}^{(2)} = 0 \quad \text{and} \quad \mathbf{d}^{(1)T}Q\mathbf{d}^{(2)} = 0$$

- We have

$$\mathbf{d}^{(0)T}Q\mathbf{d}^{(2)} = 3d_1^{(2)} + d_3^{(2)} = 0$$

$$\mathbf{d}^{(1)T}Q\mathbf{d}^{(2)} = -6d_2^{(2)} - 8d_3^{(2)} = 0$$

- Let $\mathbf{d}^{(2)} = [1, 4, -3]^T$, $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \mathbf{d}^{(2)}$ are mutually conjugate
- Note that there are many sets of vectors that are Q-conjugate

Conjugate Vectors

- ***Lemma***: Let Q be a symmetric positive definite $n \times n$ matrix. If the directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(k)} \in \mathbb{R}^n, k \leq n-1$, are nonzero and Q-conjugate, then they are linearly independent

- ***Proof***:
  - Suppose $\alpha_0, \ldots, \alpha_k$ satisfy
  $$\alpha_0 \mathbf{d}^{(0)} + \ldots + \alpha_k \mathbf{d}^{(k)} = 0$$
  - Want to show that $\alpha_0 = \cdots = \alpha_k = 0$
  - Premultiply equation by $\mathbf{d}^{(j)T} Q$ to get
  $$\alpha_j \mathbf{d}^{(j)T} Q \mathbf{d}^{(j)} = 0$$
  - Since $Q > 0$, we deduce that $\alpha_j = 0$

The Conjugate Direction Algorithm

- Consider the algorithm

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

  where, as usual

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

- Apply to quadratic:

  $$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q \mathbf{x} - \mathbf{x}^T \mathbf{b}$$

- Recall formula for α_k in this case:

  $$\alpha_k = -\frac{\mathbf{g}^{(k)T}\mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} Q \mathbf{d}^{(k)}}$$

- ***Basic Conjugate Direction Algorithm***: the directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n-1)}$ in the above algorithm are Q-conjugate

The Conjugate Direction Algorithm

- For quadratics, conjugate direction algorithm has the following nice property
- ***Theorem***: For any starting point $\mathbf{x}^{(0)}$ the basic conjugate direction algorithm converges to the unique $\mathbf{x}^*$ (that solves $Q\mathbf{x} = \mathbf{b}$) in n steps; that is $\mathbf{x}^{(n)} = \mathbf{x}^*$

The Conjugate Direction Algorithm

- ***Proof***:
  - Recall that the conjugate directions $\mathbf{d}^{(i)}, i = 0, \ldots, n-1$ are linearly independent
  - $\mathbf{x}^* - \mathbf{x}^{(0)} \in \mathbb{R}^n$ can be represented by a linear combination of $\mathbf{d}^{(i)}$ as

$$\mathbf{x}^* - \mathbf{x}^{(0)} = \beta_0 \mathbf{d}^{(0)} + \cdots + \beta_{n-1} \mathbf{d}^{(n-1)}$$

  - Multiply both sides by $\mathbf{d}^{(k)T} Q$, where $0 \leq k \leq n-1$. All terms on the right hand side will vanish by the Q-conjugate property, except the kth

$$\mathbf{d}^{(k)T} Q(\mathbf{x}^* - \mathbf{x}^{(0)}) = \beta_k \mathbf{d}^{(k)T} Q \mathbf{d}^{(k)}$$

  - Hence

$$\beta_k = \frac{\mathbf{d}^{(k)T} Q(\mathbf{x}^* - \mathbf{x}^{(0)})}{\mathbf{d}^{(k)T} Q \mathbf{d}^{(k)}}$$

The Conjugate Direction Algorithm

- ***Proof***: (Cont.)
  - According to the algorithm, $\mathbf{x}^{(k)}$ can be obtained by
    $$\begin{aligned}\mathbf{x}^{(k)} &= \mathbf{x}^{(k-1)} + \alpha_{k-1}\mathbf{d}^{(k-1)} \\ &= \mathbf{x}^{(0)} + \alpha_0\mathbf{d}^{(0)} + \cdots + \alpha_{k-1}\mathbf{d}^{(k-1)}\end{aligned}$$
  - Therefore
    $$\mathbf{x}^{(k)} - \mathbf{x}^{(0)} = \alpha_0\mathbf{d}^{(0)} + \cdots + \alpha_{k-1}\mathbf{d}^{(k-1)}$$
  - Write
    $$\mathbf{x}^* - \mathbf{x}^{(0)} = \left(\mathbf{x}^* - \mathbf{x}^{(k)}\right) + \left(\mathbf{x}^{(k)} - \mathbf{x}^{(0)}\right)$$
  - Multiply both sides by $\mathbf{d}^{(k)T}Q$, we obtain
    $$\begin{aligned}\mathbf{d}^{(k)T}Q\left(\mathbf{x}^* - \mathbf{x}^{(0)}\right) &= \mathbf{d}^{(k)T}Q\left(\mathbf{x}^* - \mathbf{x}^{(k)}\right) + \mathbf{d}^{(k)T}Q\left(\mathbf{x}^{(k)} - \mathbf{x}^{(0)}\right) \\ &= \mathbf{d}^{(k)T}Q\left(\mathbf{x}^* - \mathbf{x}^{(k)}\right) + \mathbf{d}^{(k)T}Q\left(\alpha_0\mathbf{d}^{(0)} + \cdots + \alpha_{k-1}\mathbf{d}^{(k-1)}\right) \\ &= \mathbf{d}^{(k)T}Q\left(\mathbf{x}^* - \mathbf{x}^{(k)}\right)\end{aligned}$$

The Conjugate Direction Algorithm

- ***Proof***: (Cont.)
  - Recall that $\mathbf{g}^{(k)} = Q\mathbf{x}^{(k)} - \mathbf{b}$ and $Q\mathbf{x}^* = \mathbf{b}$. Thus

$$\begin{aligned}
\mathbf{d}^{(k)T}Q(\mathbf{x}^* - \mathbf{x}^{(0)}) &= \mathbf{d}^{(k)T}Q(\mathbf{x}^* - \mathbf{x}^{(k)}) \\
&= \mathbf{d}^{(k)T}Q\mathbf{x}^* - \mathbf{d}^{(k)T}Q\mathbf{x}^{(k)} \\
&= -\mathbf{d}^{(k)T}(Q\mathbf{x}^{(k)} - \mathbf{b}) \\
&= -\mathbf{d}^{(k)T}\mathbf{g}^{(k)}
\end{aligned}$$

  - Accordingly

$$\begin{aligned}
\beta_k &= \frac{\mathbf{d}^{(k)T}Q(\mathbf{x}^* - \mathbf{x}^{(0)})}{\mathbf{d}^{(k)T}Q\mathbf{d}^{(k)}} \\
&= -\frac{\mathbf{d}^{(k)T}\mathbf{g}^{(k)}}{\mathbf{d}^{(k)T}Q\mathbf{d}^{(k)}} \\
&= \alpha_k
\end{aligned}$$

The Conjugate Direction Algorithm

- ***Example***: Find the minimizer of

$$f(x_1, x_2) = \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \mathbf{x} - \mathbf{x}^T \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

  using the conjugate direction method with $\mathbf{x}^{(0)} = [0, 0]^T$, and Q-conjugate directions $\mathbf{d}^{(0)} = [1, 0]^T$ and $\mathbf{d}^{(1)} = [-\frac{3}{8}, \frac{3}{4}]^T$

- We have $\mathbf{g}^{(0)} = -\mathbf{b} = [1, -1]^T$

- And hence

$$\alpha_0 = -\frac{\mathbf{g}^{(0)T}\mathbf{d}^{(0)}}{\mathbf{d}^{(0)T}Q\mathbf{d}^{(0)}} = -\frac{[1, -1]\begin{bmatrix} 1 \\ 0 \end{bmatrix}}{[1, 0]\begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}} = -\frac{1}{4}$$

- Thus

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \frac{1}{4}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{1}{4} \\ 0 \end{bmatrix}$$

The Conjugate Direction Algorithm

- To find $\mathbf{x}^{(2)}$ we compute

  $$\mathbf{g}^{(1)} = Q\mathbf{x}^{(1)} - \mathbf{b} = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -1/4 \\ 0 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -3/2 \end{bmatrix}$$

  and

  $$\alpha_1 = -\frac{\mathbf{g}^{(1)T}\mathbf{d}^{(1)}}{\mathbf{d}^{(1)T}Q\mathbf{d}^{(1)}} = -\frac{[0, -3/2]\begin{bmatrix} -3/8 \\ 3/4 \end{bmatrix}}{[1,0]\begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} -3/8 \\ 3/4 \end{bmatrix}} = 2$$

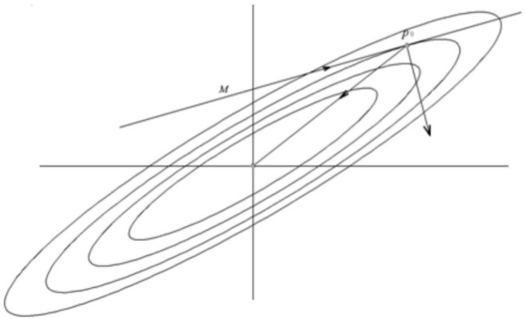
- Therefore

  $$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)} = [-1/4, 0]^T - 2[-3/8, 3/4]^T = [-1, 3/2]^T$$

- Because f is a quadratic function in two variables, $\mathbf{x}^{(2)} = \mathbf{x}^*$

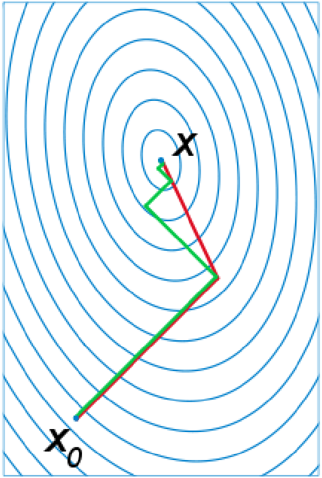
The Conjugate Direction Algorithm

- For a quadratic function of n variables, the conjugate direction method reaches the solution after n steps
- A comparison between conjugate direction method and gradient descent method



The Conjugate Direction Algorithm

- Another example



The Conjugate Gradient Algorithm

- Recall conjugate direction algorithm:
  $$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$$
  where
  $$\alpha_k = \arg\min_\alpha f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$$
  $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots$ are Q-conjugate

- Conjugate direction method is efficient but requires provision of a set of Q-conjugation directions

- How do we generate the directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots$
  - For each k, we generate $\mathbf{d}^{(k+1)}$ based on current and past data. For example, $\mathbf{d}^{(k)}, \mathbf{g}^{(k)}$, and $\mathbf{g}^{(k+1)}$
  - Two methods for generating successive directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots$
    - Conjugate gradient method
    - Quasi Newton method

The Conjugate Gradient Algorithm

- The conjugate gradient algorithm computes the directions as the algorithm progresses
- At each stage, the direction is calculated as a linear combination of previous direction and the current gradient
- All generated directions are mutually Q-conjugate

The Conjugate Gradient Algorithm

- We consider the quadratic function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{x}^T\mathbf{b}, \mathbf{x} \in \mathbb{R}$$

- The first search direction from $\mathbf{x}^{(0)}$: the direction of steepest descent, i.e.,

$$\mathbf{d}^{(0)} = -\mathbf{g}^{(0)}$$

- Thus

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0\mathbf{d}^{(0)}$$

where

$$\alpha_0 = \arg\min_{\alpha\geq 0} f(\mathbf{x}^{(0)} + \alpha\mathbf{d}^{(0)}) = -\frac{\mathbf{g}^{(0)T}\mathbf{d}^{(0)}}{\mathbf{d}^{(0)T}Q\mathbf{d}^{(0)}}$$

The Conjugate Gradient Algorithm

- Next, we search from $\mathbf{d}^{(1)}$ which is Q-conjugate to $\mathbf{d}^{(0)}$
- We choose $\mathbf{d}^{(1)}$ as a linear combination of $\mathbf{g}^{(1)}$ and $\mathbf{d}^{(0)}$

$$\mathbf{d}^{(1)} = -\mathbf{g}^{(1)} + \beta_0 \mathbf{d}^{(0)}$$

- In general

$$\mathbf{d}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k \mathbf{d}^{(k)},\ k = 0, 1, 2, \ldots$$

- The coefficients β_k are chosen to make that $\mathbf{d}^{(k+1)}$ is Q-conjugate to $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(k)}$

The Conjugate Gradient Algorithm

- This can be accomplished by choosing

$$\beta_k = \frac{\mathbf{g}^{(k+1)T} Q \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} Q \mathbf{d}^{(k)}}$$

- Derivation:
  - We need $\mathbf{d}^{(k)T} Q \mathbf{d}^{(k+1)} = 0$
  - And $\mathbf{d}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k \mathbf{d}^{(k)}$
  - Hence

$$0 = \mathbf{d}^{(k)T} Q \mathbf{d}^{(k+1)} = -\mathbf{d}^{(k)T} Q \mathbf{g}^{(k+1)} + \beta_k \mathbf{d}^{(k)T} Q \mathbf{d}^{(k)}$$

  - We obtain

$$\beta_k = \frac{\mathbf{g}^{(k+1)T} Q \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} Q \mathbf{d}^{(k)}}$$

The Conjugate Gradient Algorithm

- The conjugate gradient algorithm
  - Step 1: set $k := 0$; select the initial point $\mathbf{x}^{(0)}$
  - Step 2: calculate $\mathbf{g}^{(0)} = \nabla f(\mathbf{x}^{(0)})$. If $\mathbf{g}^{(0)} = 0$ stop; else, set $\mathbf{d}^{(0)} = -\mathbf{g}^{(0)}$
  - Step 3: calculate $\alpha_k = -\frac{\mathbf{g}^{(k)T}\mathbf{d}^{(k)}}{\mathbf{d}^{(k)T}Q\mathbf{d}^{(k)}}$
  - Step 4: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
  - Step 5: $\mathbf{g}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)})$. If $\mathbf{g}^{(k+1)} = 0$, stop
  - Step 6: calculate $\beta_k = \frac{\mathbf{g}^{(k+1)T}Q\mathbf{d}^{(k)}}{\mathbf{d}^{(k)T}Q\mathbf{d}^{(k)}}$
  - Step 7: $\mathbf{d}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k \mathbf{d}^{(k)}$
  - Step 8: Set $k := k + 1$; go to step 3
- α_k controls the step size, β_k controls the direction

The Conjugate Gradient Algorithm

- ***Proposition***: In the conjugate gradient algorithm, the directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n-1)}$ are Q-conjugate
- ***Proof***:
  - Main idea: use induction
  - First show that $\mathbf{d}^{(0)T} Q \mathbf{d}^{(1)} = 0$
  - Then assume $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(k)}, k < n-1$, are Q-conjugate, that is

    $$\mathbf{d}^{(k)T} Q \mathbf{d}^{(j)} = 0, j = 0, \ldots, k-1$$

    and we need to show that

    $$\mathbf{d}^{(k+1)T} Q \mathbf{d}^{(j)} = 0, j = 0, \ldots, k$$

The Conjugate Gradient Algorithm

- ***Example***: Consider the quadratic function

  $$f(x_1, x_2, x_3) = \frac{3}{2}x_1^2 + 2x_2^2 + \frac{3}{2}x_3^2 + x_1x_3 + 2x_2x_3 - 3x_1 - x_3$$

  We find the minimizer using the conjugate gradient algorithm, using the starting point $\mathbf{x}^{(0)} = [0, 0, 0]^T$

- We can represent f as

  $$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{x}^T\mathbf{b}$$

  where

  $$Q = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix}$$

The Conjugate Gradient Algorithm

- We have

$$\begin{aligned} g(\mathbf{x}) &= \nabla f(\mathbf{x}) = Q\mathbf{x} - \mathbf{b} \\ &= [3x_1 + x_3 - 3, 4x_2 + 2x_3, x_1 + 2x_2 + 3x_3 - 1]^T \end{aligned}$$

- Hence

$$\begin{aligned} g^{(0)} &= [-3, 0, -1]^T \\ d^{(0)} &= -g^{(0)} \\ \alpha_0 &= -\tfrac{g^{(0)T} d^{(0)}}{d^{(0)T} Q d^{(0)}} = \tfrac{10}{36} = 0.2778 \end{aligned}$$

and

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = [0.8333, 0, 0.2778]^T$$

The Conjugate Gradient Algorithm

- The next stage yields

$$g^{(1)} = \nabla f(\mathbf{x}^{(1)}) = [-0.2222, 0.5556, 0.6667]^T$$

$$\beta_0 = \frac{\mathbf{g}^{(1)T} Q \mathbf{d}^{(0)}}{\mathbf{d}^{(0)T} Q \mathbf{d}^{(0)}} = 0.08025$$

We can now compute

$$\mathbf{d}^{(1)} = -\mathbf{g}^{(1)} + \beta_0 \mathbf{d}^{(0)} = [0.4630, -0.5556, -0.5864]^T$$

Hence

$$\alpha_1 = -\frac{\mathbf{g}^{(1)T} \mathbf{d}^{(1)}}{\mathbf{d}^{(1)} Q \mathbf{d}^{(1)}} = 0.2187$$

and

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)} = [0.9346, -0.1215, 0.1495]^T$$

The Conjugate Gradient Algorithm

- To perform the third iteration, we compute

$$\mathbf{d}^{(2)} = \nabla f(\mathbf{x}^{(2)}) = [-0.04673, -0.1869, 0.1402]^T$$

$$\beta_1 = \frac{\mathbf{g}^{(2)T} Q \mathbf{d}^{(1)}}{\mathbf{d}^{(1)} Q \mathbf{d}^{(1)}} = 0.07075$$

$$\mathbf{d}^{(2)} = -\mathbf{g}^{(2)} + \beta_1 \mathbf{d}^{(1)} = [0.07948, 0.1476, -0.1817]^T$$

- Hence

$$\alpha_2 = -\frac{\mathbf{g}^{(2)} \mathbf{d}^{(2)}}{\mathbf{d}^{(2)T} Q \mathbf{d}^{(2)}} = 0.8231$$

and

$$\mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \alpha_2 \mathbf{d}^{(2)} = [1, 0, 0]^T$$

- Note that $g^{(3)} = \nabla f(\mathbf{x}^{(3)}) = 0$. Since f is quadratic with three variables, $\mathbf{x}^* = \mathbf{x}^{(3)}$

The Conjugate Gradient Algorithm for Nonquadratic Problems

- The conjugate gradient algorithm is an implementation of the conjugate direction method
- It can be extended to general nonlinear function
- Interpreting quadratic $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{x}^T\mathbf{b}$ as a second-order Taylor series approximation of objective function
- If the expansion point is close to the minimizer, may have good approximation

The Conjugate Gradient Algorithm for Nonquadratic Problems

- For quadratic function f, the Hessian is constant
- For a general nonlinear function, the Hessian needs to be re-evaluated at each iteartion
- Is there any method to eliminate the Hessian evaluation for an efficient conjugate gradient algorithm implementation?

The Conjugate Gradient Algorithm for Nonquadratic Problems

- We can observe that Q appears only in the computation of the scalars α_k and β_k
- For α_k, we have

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

- Accordingly, the closed-form formula for α_k in the algorithm can be replaced by a numerical line search procedure

The Conjugate Gradient Algorithm for Nonquadratic Problems

- Easy way to compute β_k
  - Recall that
    $$\beta_k = \frac{\mathbf{g}^{(k+1)T} Q \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} Q \mathbf{d}^{(k)}}$$
  - The above formula not immediately useful because it involves Q
  - Elimination of Q from the formula is also possible
  - There exists modifications of the conjugate gradient algorithms that depend only on the function and gradient values at each iteration

The Conjugate Gradient Algorithm for Nonquadratic Problems

- Useful formulas for β_k
- Hestenes-Stiefel formula:

$$\beta_k = \frac{\mathbf{g}^{(k+1)T}[\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}{\mathbf{d}^{(k)T}[\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}$$

- Polak-Ribiere formula:

$$\beta_k = \frac{\mathbf{g}^{(k+1)T}[\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}{\mathbf{g}^{(k+1)}\mathbf{g}^{(k)}}$$

- Fletcher-Reeves formula:

$$\beta_k = \frac{\mathbf{g}^{(k+1)T}\mathbf{g}^{(k+1)}}{\mathbf{g}^{(k)T}\mathbf{g}^{(k)}}$$

Hestenes-Stiefel Formula

- Recall that in conjugate gradient algorithm

$$\beta_k = \frac{\mathbf{g}^{(k+1)T} Q \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} Q \mathbf{d}^{(k)}}$$

- Main idea: Replacing the term $Q\mathbf{d}^{(k)}$ by the term $(\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)})/\alpha_k$, which are equal in the quadratic case

- Accordingly, we have the Hestenes-Stiefel formula

$$\beta_k = \frac{\mathbf{g}^{(k+1)T}[\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}{\mathbf{d}^{(k)T}[\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}$$

Hestenes-Stiefel Formula

- ***Proof***:
  - Recall $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
  - Premultiplying both sides by Q and subtracting $\mathbf{b}$ from both sides

    $$Q\mathbf{x}^{(k+1)} - \mathbf{b} = Q\mathbf{x}^{(k)} - \mathbf{b} + \alpha_k Q\mathbf{d}^{(k)}$$
  - Recall $\mathbf{g}^{(k)} = Q\mathbf{x}^{(k)} - \mathbf{b}$
  - We get

    $$\mathbf{g}^{(k+1)} = \mathbf{g}^{(k)} + \alpha_k Q\mathbf{d}^{(k)}$$
  - Accordingly

    $$Q\mathbf{d}^{(k)} = \frac{\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}}{\alpha_k}$$

Polak-Ribiere and Formulas

- Starting from the Hestenes-Stiefel formula, we can get the Polak-Ribiere formula

$$\beta_k = \frac{\mathbf{g}^{(k+1)T}[\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}{\mathbf{g}^{(k)T}\mathbf{g}^{(k)}}$$

- Starting with the Polak-Ribiere formula, we can get the Fletcher-Reeves formula

$$\beta_k = \frac{\mathbf{g}^{(k+1)T}\mathbf{g}^{(k+1)}}{\mathbf{g}^{(k)T}\mathbf{g}^{(k)}}$$

The Conjugate Gradient Algorithm for Nonquadratic Problems

- The above three formulas all lead to conjugate direction algorithms (i.e., the resulting directions are Q-conjugate when applied to a quadratic with Hessian Q)
- The conjugate gradient algorithm using the above formulas for β_k can be applied to any function f
- If f is quadratic, all the three formulas are equivalent
- If f is not a quadratic, the algorithm will not usually reach the solution in n steps, because the search direction may not be Q-conjugate after several iterations
- One possible solution is to reset the direction as the minus gradient, and then iterate

The Conjugate Gradient Algorithm for Nonquadratic Problems

- For nonlinear objective, the accuracy of line search is important
- For general f, the formulas have different performance. Performance highly depends on f
- If using in-exact line search, Hestenes-Stiefel formula is recommended
- Modifications are possible. For example, Powell's formula (modification of Polak-Ribiere)

$$\beta_k = \max\left[0, \frac{\mathbf{g}^{(k+1)T}[\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}{\mathbf{g}^{(k)T}\mathbf{g}^{(k)}}\right]$$