Conjugate direction methods

Source: Ahmadi, ORF 363 slides.

Conjugate directions methods

- Yet another class of descent methods that are particularly clever and efficient: **the conjugate direction methods**
- They are used to minimize **quadratic** functions

Conjugate directions methods

- Yet another class of descent methods that are particularly clever and efficient: **the conjugate direction methods**
- They are used to minimize **quadratic** functions
- Idea: accelerate the convergence rate of gradient descent without inverting a matrix

Conjugate directions methods

- Yet another class of descent methods that are particularly clever and efficient: **the conjugate direction methods**
- They are used to minimize **quadratic** functions
- Idea: accelerate the convergence rate of gradient descent without inverting a matrix
- They are also used to solve large-scale linear systems defined by a positive definitematrix

Conjugate direction methods

We will try to minimize the quadratic function:

$$f(x) = \frac{1}{2}x^T Q x - b^T x \qquad Q \succ 0$$

Conjugate direction methods

We will try to minimize the quadratic function:

$$f(x) = \frac{1}{2}x^T Q x - b^T x \qquad Q \succ 0$$

- Like other descent methods, conjugate direction methods take the following iterative update:

$$x_{k+1} = x_k + \alpha_k d_k$$

- The direction is chosen using **conjugate directions** which will be defined shortly

Q-conjugate

Conjugate direction algorithm

Conjugate gradient algorithm

Q-conjugate

Definition. Let Q be an $n \times n$ real symmetric matrix. We say that a set of non-zero vectors $d_1, d_2, \ldots, d_m$ are Q-conjugate if

$$d_i^T Q d_j = 0 \qquad \text{for all } i \neq j$$

Q-conjugate

Definition. Let Q be an $n \times n$ real symmetric matrix. We say that a set of non-zero vectors $d_1, d_2, \ldots, d_m$ are Q-conjugate if

$$d_i^T Q d_j = 0 \qquad \text{for all } i \neq j$$

If Q is the identity matrix, then

$$d_i^T Q d_j = d_i^T d_j = d_i \cdot d_j = 0$$

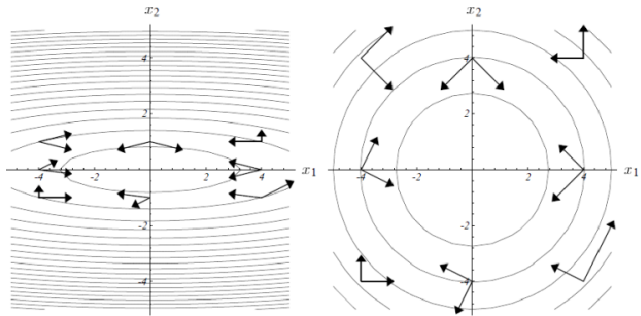
that is, d_i and d_j are orthogonal

Q-conjugate

Definition. Let Q be an $n \times n$ real symmetric matrix. We say that a set of non-zero vectors $d_1, d_2, \ldots, d_m$ are Q-conjugate if

$$d_i^T Q d_j = 0 \qquad \text{for all } i \neq j$$

For general Q:



Linear independence

Recall: vectors $d_0, d_1, \ldots, d_k$ are **linearly independent** if one of the vectors cannot be written as a linear combination of the others

Linear independence

Let Q be an $n \times n$ symmetric positive definite matrix. If the vectors $d_0, d_1, \ldots, d_k$ are Q-conjugate, then they are linearly independent

Q-conjugate

Conjugate direction algorithm

Conjugate gradient algorithm

The conjugate direction algorithm

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

Input: An $n \times n$ matrix Q, a vector $b \in \mathbb{R}^n$ and a set of n Q-conjugate directions $d_0, d_1, \ldots, d_{n-1}$
The conjugate direction algorithm: Pick an initial point $x_0 \in \mathbb{R}^n$

The conjugate direction algorithm

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

Input: An $n \times n$ matrix Q, a vector $b \in \mathbb{R}^n$ and a set of n Q-conjugate directions $d_0, d_1, \ldots, d_{n-1}$
The conjugate direction algorithm: Pick an initial point $x_0 \in \mathbb{R}^n$

For $k = 0$ to $n - 1$:

1. Let $g_k = \nabla f(x_k) = Qx_k - b$

The conjugate direction algorithm

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

Input: An $n \times n$ matrix Q, a vector $b \in \mathbb{R}^n$ and a set of n Q-conjugate directions $d_0, d_1, \ldots, d_{n-1}$
The conjugate direction algorithm: Pick an initial point $x_0 \in \mathbb{R}^n$

For $k = 0$ to $n - 1$:

1. Let $g_k = \nabla f(x_k) = Q x_k - b$
2. Let $\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$

The conjugate direction algorithm

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

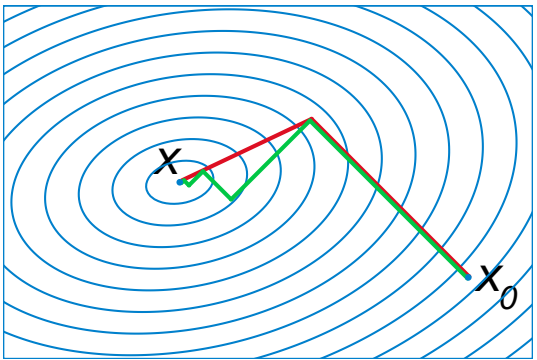
Input: An $n \times n$ matrix Q, a vector $b \in \mathbb{R}^n$ and a set of n Q-conjugate directions $d_0, d_1, \ldots, d_{n-1}$
The conjugate direction algorithm: Pick an initial point $x_0 \in \mathbb{R}^n$

For $k = 0$ to $n - 1$:

1. Let $g_k = \nabla f(x_k) = Q x_k - b$
2. Let $\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$
3. Let $x_{k+1} = x_k + \alpha_k d_k$

Convergence

The algorithm above converges to the unique minimum x^* of $f(x) = \frac{1}{2}x^T Q x - b^T x$ in n steps; i.e., $x_n = x^*$



red: conjugate direction. green: steepest descent.

Convergence

The algorithm above converges to the unique minimum x^* of $f(x) = \frac{1}{2}x^T Q x - b^T x$ in n steps; i.e., $x_n = x^*$

Finding conjugate directions

The algorithm assumed that we already have conjugate directions $d_0, d_1, \ldots, d_k$

But, given Q, how can we find these directions?

Finding conjugate directions

The algorithm assumed that we already have conjugate directions $d_0, d_1, \ldots, d_k$

But, given Q, how can we find these directions?

The Gram-Schmidt procedure: Let $v_0, \ldots, v_{n-1}$ be a set of linearly independent vectors (you can take the standard Euclidean basis). Then, the vectors $d_0 \ldots, d_{n-1}$ constructed as follows are Q-conjugate

$$d_0 = v_0$$

$$d_{i+1} = v_{i+1} - \sum_{i=0}^{n} \frac{v_{i+1}^T Q d_m}{d_m^T Q d_m} d_m$$

Q-conjugate

Conjugate direction algorithm

Conjugate gradient algorithm

- There is a much more efficient way than the Gram-Schmidt method
- This method can generate conjugate directions on the fly in each iteration

- There is a much more efficient way than the Gram-Schmidt method
- This method can generate conjugate directions on the fly in each iteration
- This is the **conjugate gradient** (CG) algorithm

Some observation

Let be Q be an $n \times n$ symmetric positive definite matrix, $f(x) = \frac{1}{2}x^T Q x - b^T x$, $\{d_0, d_1, \ldots, d_{n-1}\}$ a set of Q-conjugate directions, and $x_1, \ldots, x_n$ the sequence of points generated by the conjugate direction algorithm. Let $g_k = \nabla f(x_k) = Qx_k - b$. Then, for any k,

$$g_{k+1}^T d_i = 0 \qquad \text{for } i = 0, 1, \ldots, k$$

Some observation

Let be Q be an $n \times n$ symmetric positive definite matrix, $f(x) = \frac{1}{2}x^T Qx - b^T x$, $\{d_0, d_1, \ldots, d_{n-1}\}$ a set of Q-conjugate directions, and $x_1, \ldots, x_n$ the sequence of points generated by the conjugate direction algorithm. Let $g_k = \nabla f(x_k) = Qx_k - b$. Then, for any k,

$$g_{k+1}^T d_i = 0 \qquad \text{for } i = 0, 1, \ldots, k$$

Consequence, if you want to find a new conjugate direction at step k, just take the **current gradient**!

The conjugate gradient algorithm

The update rule for the directions is simple: In each iteration, the new conjugate direction is a linear combination of the current gradient and the previous conjugate direction.

The conjugate gradient algorithm

1. Set $k = 0$; Select an initial point $x_0 \in \mathbb{R}^n$
2. $g_0 = \nabla f(x_k)$. If $g_0 = 0$, stop; else, set $d_0 = -g_0$

The conjugate gradient algorithm

1. Set $k = 0$; Select an initial point $x_0 \in \mathbb{R}^n$
2. $g_0 = \nabla f(x_k)$. If $g_0 = 0$, stop; else, set $d_0 = -g_0$
3. $\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$
4. $x_{k+1} = x_k + \alpha_k d_k$

The conjugate gradient algorithm

1. Set $k = 0$; Select an initial point $x_0 \in \mathbb{R}^n$
2. $g_0 = \nabla f(x_k)$. If $g_0 = 0$, stop; else, set $d_0 = -g_0$
3. $\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$
4. $x_{k+1} = x_k + \alpha_k d_k$
5. $g_{k+1} = \nabla f(x_{k+1})$. If $g_{k+1} = 0$, stop

The conjugate gradient algorithm

1. Set $k = 0$; Select an initial point $x_0 \in \mathbb{R}^n$
2. $g_0 = \nabla f(x_k)$. If $g_0 = 0$, stop; else, set $d_0 = -g_0$
3. $\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$
4. $x_{k+1} = x_k + \alpha_k d_k$
5. $g_{k+1} = \nabla f(x_{k+1})$. If $g_{k+1} = 0$, stop
6. $\beta_{k+1} = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}$
7. $d_{k+1} = -g_{k+1} + \beta_{k+1} d_k$
8. Set $k \to k + 1$. Go back to step 3

Solving linear systems

- We can use the conjugate gradient method to solve a linear system $Ax = b$
- This method is often the method of choice for large-scale (sparse) linear systems

Solving linear systems

Suppose we are solving $Ax = b$ where A is symmetric and positive definite

Solution: Define the quadratic function

$$f(x) = x^T Ax - b^T x$$

Use the conjugate gradient algorithm to find its global minimum

$$x^* = A^{-1}b$$

Solving linear systems

Is the assumption $A \succ 0$ too restrictive?

- Suppose we want to solve $Ax = b$, where A is invertible but neither positive definite nor symmetric

Solving linear systems

Is the assumption $A \succ 0$ too restrictive?

- Suppose we want to solve $Ax = b$, where A is invertible but neither positive definite nor symmetric

- Solution: multiply both sides of equation by A^T:

$$A^T Ax = A^T b$$

Solving linear systems

Is the assumption $A \succ 0$ too restrictive?

- Suppose we want to solve $Ax = b$, where A is invertible but neither positive definite nor symmetric
- Solution: multiply both sides of equation by A^T:

$$A^T Ax = A^T b$$

- $A^T A$ is symmetric. It is also positive definite if A is invertible (why?).