Nonlinear Conjugate Gradient Methods

Leo Shen

Purdue University

Table of Contents

Linear CG I: Motivation

- Solve the linear system $Ax = b$ where $A \in \mathbb{R}^{n \times n}$ is an SPD matrix. Equivalent to minimizing the quadratic function $\phi(x) = \frac{1}{2}x^T Ax - bx$ because $\nabla\phi(x) = Ax - b$.

Linear CG I: Motivation

- Solve the linear system $Ax = b$ where $A \in \mathbb{R}^{n \times n}$ is an SPD matrix. Equivalent to minimizing the quadratic function $\phi(x) = \frac{1}{2}x^T Ax - bx$ because $\nabla\phi(x) = Ax - b$.
- Convergence rate: CG converges linearly for $x_j \to x_*$, quadratically for $\phi(x_j) \to \phi(x_*)$, where rate depends on κ

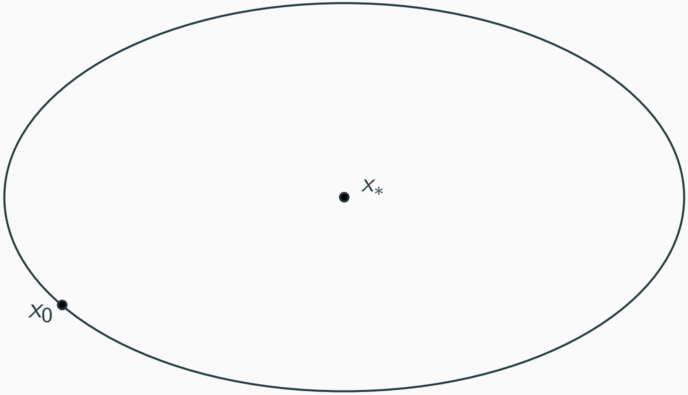
Linear CG I: Motivation

- Solve the linear system $Ax = b$ where $A \in \mathbb{R}^{n\times n}$ is an SPD matrix. Equivalent to minimizing the quadratic function $\phi(x) = \frac{1}{2}x^T Ax - bx$ because $\nabla\phi(x) = Ax - b$.
- Convergence rate: CG converges linearly for $x_j \to x_*$, quadratically for $\phi(x_j) \to \phi(x_*)$, where rate depends on κ
- Termination in $|\{\lambda_1, ..., \lambda_n\}| \leq n$ iterations with exact arithmetic.

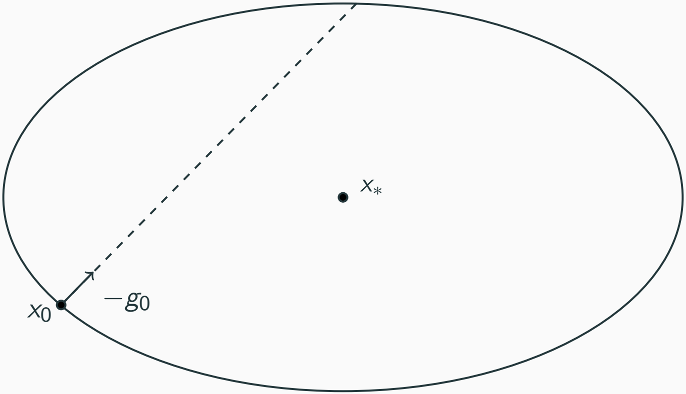
Linear CG I: Motivation

- Solve the linear system $Ax = b$ where $A \in \mathbb{R}^{n \times n}$ is an SPD matrix. Equivalent to minimizing the quadratic function $\phi(x) = \frac{1}{2}x^T Ax - bx$ because $\nabla\phi(x) = Ax - b$.
- Convergence rate: CG converges linearly for $x_j \to x_*$, quadratically for $\phi(x_j) \to \phi(x_*)$, where rate depends on κ
- Termination in $|\{\lambda_1, ..., \lambda_n\}| \leq n$ iterations with exact arithmetic.
- Notation: let $g_j = \nabla\phi(x_j)$, and $f_j = f(x_j)$ for general f

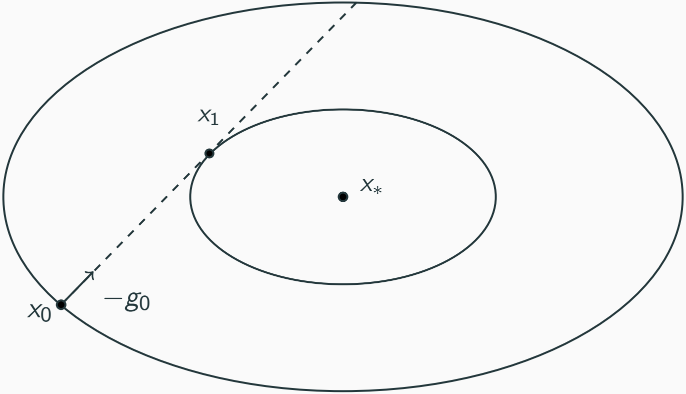
Linear CG II: Algorithm Sketch



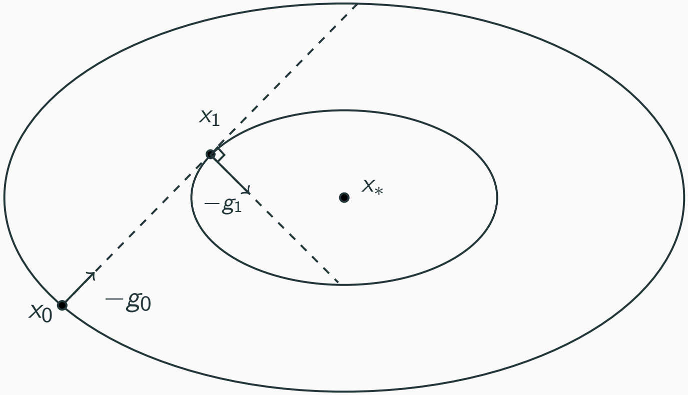
Linear CG II: Algorithm Sketch



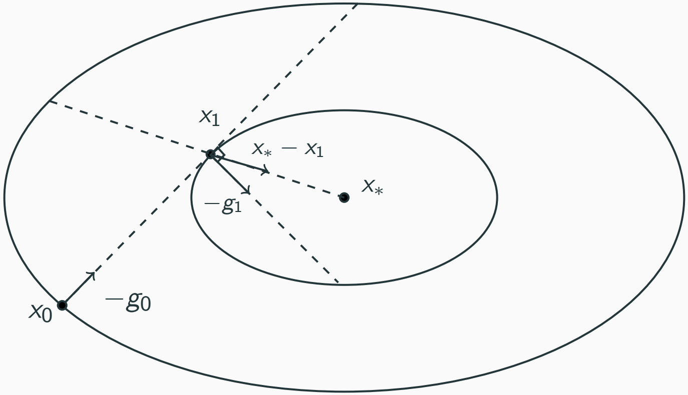
Linear CG II: Algorithm Sketch



Linear CG II: Algorithm Sketch

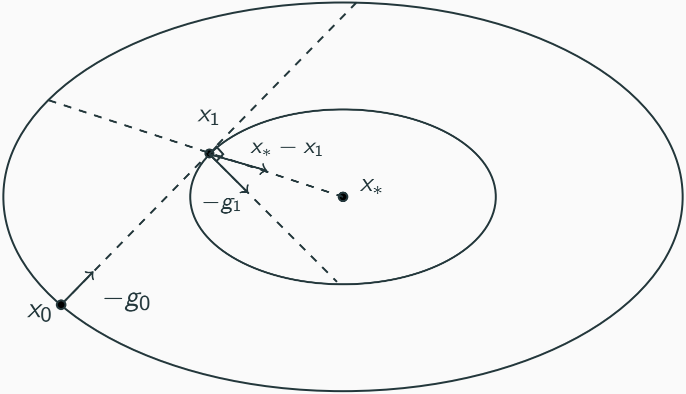


Linear CG II: Algorithm Sketch



- g_0 and $x_* - x_1$ are A-conjugate:
  $g_0^T A(x_* - x_1) = \langle g_0, b - Ax_1 \rangle = -g_0 \cdot g_1 = 0.$

Linear CG II: Algorithm Sketch



- g_0 and $x_* - x_1$ are A-conjugate:
  $g_0^T A(x_* - x_1) = \langle g_0, b - Ax_1 \rangle = -g_0 \cdot g_1 = 0.$
- Goal: find a vector d_1 that is A-conjugate to g_0, then $x_* - x_1 = \gamma d_1$ where γ can be found with line search

Linear CG III: Algorithm Pseudocode

Notice that g_0 and g_1 span $\mathbb{R}^2$, so $d_1 = g_1 + \beta g_0$. Linear CG finds a new conjugate search direction that is conjugate to *all* previous directions: $d_k^T d_j = 0$ for $k > j$

Algorithm 1 Linear Conjugate Gradient Method

pick arbitrary $x_0 \in \mathbb{R}^n$, set $d_0 = Ax_0 - b = g_0$
while $g_j \neq 0$ **do**
  set $\alpha_j = \frac{g_j^T g_j}{d_j^T A d_j}$ (minimization along search direction)
  $x_{j+1} = x_j + \alpha_j d_j$
  $\beta_j = \frac{g_{j+1}^T g_{j+1}}{g_j^T g_j}$
  $d_{j+1} = -g_{j+1} + \beta_j d_j$
end while

Nonlinear CG I: Motivation

- General differentiable function f, not necessarily quadratic

Nonlinear CG I: Motivation

- General differentiable function f, not necessarily quadratic
- Algorithm: $x_{j+1} = x_j + \alpha_j d_j$ where $d_{j+1} = -g_{j+1} + \beta_j d_j$ where β_j is some scalar

Nonlinear CG I: Motivation

- General differentiable function f, not necessarily quadratic
- Algorithm: $x_{j+1} = x_j + \alpha_j d_j$ where $d_{j+1} = -g_{j+1} + \beta_j d_j$ where β_j is some scalar
- Two issues: finding the minimizer α_j and the correct β_j to give 'conjugacy'

Nonlinear CG II: Nonlinear Line Search & Conjugacy

- Line Search: Find α_j that minimizes $f(x_j + \alpha_j d_j)$: Typically sufficient to use inexact line search satisfying Wolfe Conditions:

$$
\begin{aligned}
f(x_j + \alpha_j d_j) &\le f_j + \delta \alpha_j g_j^T d_j \\
g(x_j + \alpha_j d_j)^T d_j &> \sigma g_j^T d_j
\end{aligned}
$$

Nonlinear CG II: Nonlinear Line Search & Conjugacy

- Line Search: Find α_j that minimizes $f(x_j + \alpha_j d_j)$: Typically sufficient to use inexact line search satisfying Wolfe Conditions:

$$f(x_j + \alpha_j d_j) \leq f_j + \delta \alpha_j g_j^T d_j$$
$$g(x_j + \alpha_j d_j)^T d_j > \sigma g_j^T d_j$$

- Conjugacy: weakened to conjugacy for quadratic f, otherwise that d_{j+1} is a descent direction, $d_{j+1}^T g_{j+1} < 0$ or $d_{j+1}^T g_{j+1} < -c\|g_{j+1}\|^2$

Nonlinear CG III: Dai-Yuan Algorithim

- If f is quadratic, then

$$\beta_j = \frac{\|g_{j+1}\|^2}{\|g_j\|^2} = \frac{g_{j+1}^T(g_{j+1} - g_j)}{\|g_j\|^2} = \frac{g_{j+1}^T(g_{j+1} - g_j)}{d_j^T(g_{j+1} - g_j)}$$

Nonlinear CG III: Dai-Yuan Algorithim

- If f is quadratic, then

$$\beta_j = \frac{\|g_{j+1}\|^2}{\|g_j\|^2} = \frac{g_{j+1}^T(g_{j+1} - g_j)}{\|g_j\|^2} = \frac{g_{j+1}^T(g_{j+1} - g_j)}{d_j^T(g_{j+1} - g_j)}$$

- If not, different choice of β_j gives different algorithm. Dai-Yuan:

$$\beta_j = \frac{\|g_{j+1}\|^2}{d_j^T(g_{j+1} - g_j)}$$

Nonlinear CG III: Dai-Yuan Algorithim

- If f is quadratic, then

$$\beta_j = \frac{\|g_{j+1}\|^2}{\|g_j\|^2} = \frac{g_{j+1}^T(g_{j+1} - g_j)}{\|g_j\|^2} = \frac{g_{j+1}^T(g_{j+1} - g_j)}{d_j^T(g_{j+1} - g_j)}$$

- If not, different choice of β_j gives different algorithm. Dai-Yuan:

$$\beta_j = \frac{\|g_{j+1}\|^2}{d_j^T(g_{j+1} - g_j)}$$

- Each d_j is a search direction by induction, since $d_j^T(g_{j+1} - g_j) \geq (\sigma - 1)d_j^T g_j > 0$ by the wolfe condition so

$$g_{j+1}^T d_{j+1} = \frac{\|g_{j+1}\|^2}{d_j^T(g_{j+1} - g_j)} d_j^T g_j \leq \frac{\|g_{j+1}\|^2}{\sigma - 1} < 0.$$

Nonlinear CG IV: Dai-Yuan Convergence Proof

Theorem
Suppose ∇f is L-Lipschitz and f bounded below. Let $\{x_j\}_j$ be the sequence generated by Dai-Yuan, then $x_J = x_*$ for some $J < \infty$ or $\liminf_{j\to\infty} \|g_j\| = 0$.

Nonlinear CG IV: Dai-Yuan Convergence Proof

Theorem
Suppose ∇f is L-Lipschitz and f bounded below. Let $\{x_j\}_j$ be the sequence generated by Dai-Yuan, then $x_J = x_*$ for some $J < \infty$ or $\liminf_{j\to\infty} \|g_j\| = 0$.

- Proof: Start with $d_{j+1} + g_{j+1} = \beta_j d_j$, square both sides and divide by $(g_{j+1}^T d_{j+1})^2$ to get

$$\begin{aligned}
\frac{\|d_{j+1}\|^2}{(g_{j+1}^T d_{j+1})^2} &= \frac{\beta_j^2 \|d_j\|^2}{(g_{j+1}^T d_{j+1})^2} - \frac{2}{g_{j+1}^T d_{j+1}} - \frac{\|g_{j+1}\|^2}{(g_{j+1}^T d_{j+1})^2} \\
&\leq \frac{\|d_j\|^2}{(g_j^T d_j)^2} + \frac{1}{\|g_{j+1}\|^2}
\end{aligned}$$

Nonlinear CG IV: Dai-Yuan Convergence Proof

- Then

$$\frac{\|d_j\|^2}{(g_j^T d_j)^2} = \sum_{k=0}^{j} \frac{\|d_{k+1}\|^2}{(g_{k+1}^T d_{k+1})^2} - \frac{\|d_k\|^2}{(g_k^T d_k)^2} \leq \sum_{k=1}^{j} \frac{1}{\|g_k\|^2}.$$

Nonlinear CG IV: Dai-Yuan Convergence Proof

- Then

$$\frac{\|d_j\|^2}{(g_j^T d_j)^2} = \sum_{k=0}^{j} \frac{\|d_{k+1}\|^2}{(g_{k+1}^T d_{k+1})^2} - \frac{\|d_k\|^2}{(g_k^T d_k)^2} \leq \sum_{k=1}^{j} \frac{1}{\|g_k\|^2}.$$

- Suppose by contradiction, $\|g_j\| \geq c > 0$ for all j. Then

$$\sum_{k=1}^{j} \frac{1}{\|g_j\|^2} \leq \frac{j}{c^2}, \quad \sum_{j=1}^{\infty} \frac{c^2}{j} \leq \sum_{j=1}^{\infty} \frac{(g_j^T d_j)^2}{\|d_j\|^2}$$

Nonlinear CG IV: Dai-Yuan Convergence Proof

- Then

$$\frac{\|d_j\|^2}{(g_j^T d_j)^2} = \sum_{k=0}^{j} \frac{\|d_{k+1}\|^2}{(g_{k+1}^T d_{k+1})^2} - \frac{\|d_k\|^2}{(g_k^T d_k)^2} \leq \sum_{k=1}^{j} \frac{1}{\|g_k\|^2}.$$

- Suppose by contradiction, $\|g_j\| \geq c > 0$ for all j. Then

$$\sum_{k=1}^{j} \frac{1}{\|g_j\|^2} \leq \frac{j}{c^2}, \quad \sum_{j=1}^{\infty} \frac{c^2}{j} \leq \sum_{j=1}^{\infty} \frac{(g_j^T d_j)^2}{\|d_j\|^2}$$

Zoutendijk Condition
for any method $x_{j+1} = x_j + \alpha_j d_j, d_{j+1} = -g_{j+1} + \beta_j d_j$ using Wolfe line search conditions, if ∇f is L-Lipschitz and f bounded below,

$$\sum_{j=1}^{\infty} \frac{(g_j^T d_j)^2}{\|d_j\|^2} = \sum_{j=1}^{\infty} \cos^2 \theta_j \|g_j\|^2 < \infty$$

More Examples of Nonlinear CG

Characterizing Nonlinear CG

- reduces to Linear CG for quadratics, first-order method, $O(n)$

More Examples of Nonlinear CG

Characterizing Nonlinear CG

- reduces to Linear CG for quadratics, first-order method, $O(n)$

Extensions of Nonlinear CG

- More formulas for β_j, not always derived from Linear CG
- Ex: Stronger descent condition, guarantees strong convergence without Lipschitz requirement (preprint)
- Hybrid method: pick different β_j based on some conditions
- Combine with accelerated gradient descent