Optimization Methods (CS1.404)
Spring 2025

Naresh Manwani

Machine Learning Lab, IIIT-H

February 13th, 2025

Descent Direction Methods

- We consider the unconstrained minimization problem as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x})$$

  where we assume that f is continuously differentiable over $\mathbb{R}^n$.
- In many cases, it might be very difficult to solve the equation $\nabla f(\mathbf{x}) = \mathbf{0}$ to find the stationary points.
- Even if it is possible to find the solutions of $\nabla f(\mathbf{x}) = \mathbf{0}$, if there are infinitely many solutions, finding the one corresponding to a local minima might be as difficult problem as original optimization problem.
- Due to these reasons, instead of finding the stationary points analytically, we consider adopting an iterative algorithm to find them.
- Iterative algorithms to find the stationary points are of the following form:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \mathbf{d}_k, \quad k = 0, 1, 2, \ldots,$$

  where $\mathbf{d}_k$ is the so-called direction t_k is the stepsize.

Descent Direction

Definition

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function over $\mathbb{R}^n$. A vector $\mathbf{d} \in \mathbb{R}^n$ $(\mathbf{d} \neq \mathbf{0})$ is said a **descent direction** of f at $\mathbf{x}$ if the directional derivative of f at $\mathbf{x}$ along the direction $\mathbf{d}$ is negative, i.e.,

$$\nabla f(\mathbf{x})^T \mathbf{d} < 0$$

Remark: Taking small enough steps along descent directions lead to a decrease of the function f.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Descent Property of Descent Directions

Lemma

Let f be a continuously differentiable function over an open set S of $\mathbb{R}^n$ and let $\mathbf{x} \in S$. Suppose that $\mathbf{d}$ is a descent direction of f at $\mathbf{x}$. Then there exist $\epsilon > 0$ such that

$$f(\mathbf{x} + \alpha \mathbf{d}) < f(\mathbf{x})$$

for any $\alpha \in (0, \epsilon]$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Schematic Descent Directions Method

Schematic Descent Directions Method

- **Initialization:** Pick $\mathbf{x}_0 \in \mathbb{R}^n$ arbitrarily
- **General Step:** For any $k = 0, 1, 2, \ldots$, set
  1. Pick a descent direction $\mathbf{d}_k$.
  2. Find a step size t_k satisfying $f(\mathbf{x}_k + t_k \mathbf{d}_k) < f(\mathbf{x}_k)$.
  3. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \mathbf{d}_k$.
  4. STOP if the stopping condition is satisfied and Output $\mathbf{x}_{k+1}$. Else go to Step (1).

Challanges:

1. How to choose the initial point $\mathbf{x}_0$?
2. How to choose the descent direction $\mathbf{d}_k$?
3. How to choose the stepsize t_k?
4. What should be the stopping condition?
5. Does the algorithm converge? If yes, then how fast does it converge? Does the convergence depend on $\mathbf{x}_0$?

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Stopping Condition

1. Stopping condition for a minimization problem is $\nabla f(\mathbf{x}_k) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}_k)$ is positive semi-definite.
2. A practical stopping condition is $\|\nabla f(\mathbf{x}_k)\| \leq \epsilon$.
3. Other stopping conditions

$$\|\nabla f(\mathbf{x}_k)\| < \epsilon(1 + |f(\mathbf{x}_k)|)$$

$$\frac{f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})}{|f(\mathbf{x}_k)|} \leq \epsilon$$

Finding Step Size t_k

- Step size t_k is chosen in such a way that $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$.
- The method of finding step size is called line search, since it a minimization of one dimensional function $g(t) = f(\mathbf{x}_k + t\mathbf{d}_k)$.
- Four popular choices for step size selection are as follows:
  - **Constant Step size:** $t_k = \eta$, $\forall k$. It is very simple approach, but it is unclear how to choose η. A large value of η might cause the algorithm to be nondecreasing and small η can cause very slow convergence.
  - **Diminishing Step Size:** $\alpha_k \to 0$, $\sum_{k=1}^{\infty} \alpha_k = \infty$. For example, $\alpha_k = \frac{1}{k}$.
    - Descent not guaranteed at each step; only later when becomes small.
    - $\sum_{k=1}^{\infty} \alpha_k = \infty$ imposed to guarantee progress does not become too slow.
    - Good theoretical guarantees, but unless the right sequence is chosen, can also be a slow method.

Finding Step Size t_k

- **Exact Line Search:** Here, t_k is the minimizer of f along the ray $\mathbf{x}_k + t\mathbf{d}_k$.

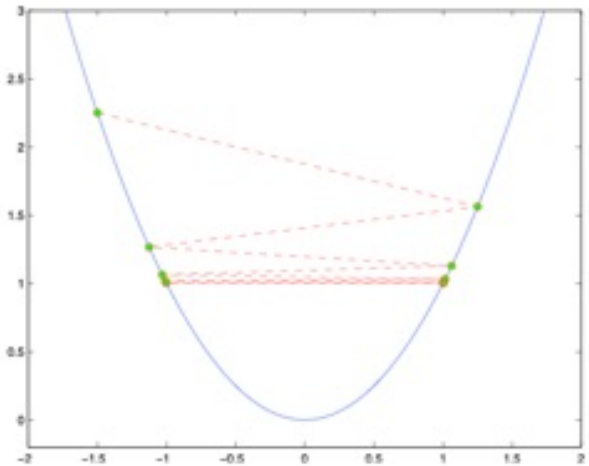
$$t_k = \arg\min_{t \geq 0} \; f(\mathbf{x}_k + t\mathbf{d}_k)$$

  It is an attractive approach, but it is not always possible to find the exact minimizer of $g(t) = f(\mathbf{x}_k + t\mathbf{d}_k)$.
- **Inexact Line Search:** This method iteratively finds t_k which minimizes f along the ray $\mathbf{x}_k + t\mathbf{d}_k$. It finds good enough step size which ensures sufficient decrease.

Example 1: How line search methods fail!

Large Step Sizes

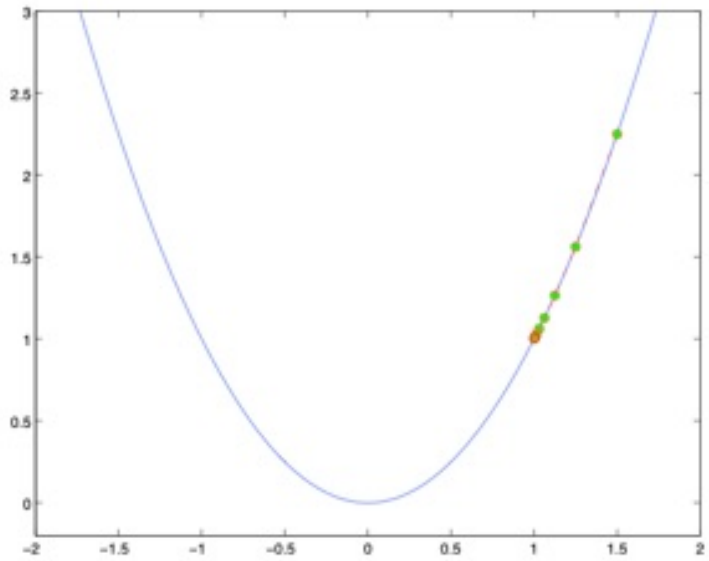
- The objective function $f(x) = x^2$. Global minimizer is $x^* = 0$ and optimal value of $f(x^*) = 0$.
- Iterates $x_{k+1} = x_k + \alpha_k d_k$ generated by the descent directions $d_k = (-1)^k$ and steps $\alpha_k = 2 + 3/2^k$ from $x_1 = 2$.
- $\{x\} = \{2, -3/2, 5/4, -9/8, \ldots\}$. As $k \to \infty$, x_k will oscillate between $+1$ and -1. Thus, the sequence x_k, $k = 1, 2, 3, \ldots$ does not converge.
- $\{f\} = \{4, 9/4, 25/16, 81/64, \ldots\}$. Thus, the function value decreases in each iteration. As $k \to \infty$, $f(x_k)$ will remain close to 1.
- **Key reason is a small decrease in function values relative to the step length.**



Example 2: How line search methods fail !

Small Step Sizes

- The objective function $f(x) = x^2$. Global minimizer is $x^* = 0$ and optimal value of $f(x^*) = 0$.
- Iterates $x_{k+1} = x_k + \alpha_k d_k$ generated by the descent directions $d_k = -1$, $\forall k$ and steps $\alpha_k = 1/2^k$ from $x_1 = 2$.
- $\{x\} = \{2, 3/2, 5/4, 9/8, \ldots\}$. As $k \to \infty$, x_k will converge to $+1$. But, $\lim_{k\to\infty} x_k \neq x^*$.
- $\{f\} = \{4, 9/4, 25/16, 81/64, \ldots\}$. Thus, the function value decreases in each iteration. As $k \to \infty$, $f(x_k)$ will remain close to 1.
- **Key reason is step sizes are too small compared to the initial rate of decrease of f.**



Sufficient Decrease Condition

Lemma

Let f be a continuously differentiable function over $\mathbb{R}^n$ and let $\mathbf{x} \in \mathbb{R}^n$. Suppose that $\mathbf{d} \in \mathbb{R}^n$ ($\mathbf{d} \neq \mathbf{0}$) is a descent direction of $\mathbf{d}$ at $\mathbf{x}$ and let $\alpha \in (0, 1)$. Then there exist $\epsilon > 0$ such that the inequality

$$f(\mathbf{x}) - f(\mathbf{x} + t\mathbf{d}) \geq -\alpha t \nabla f(\mathbf{x})^T \mathbf{d}$$

holds for all $t \in [0, \epsilon]$.

Armijo Line Search Method

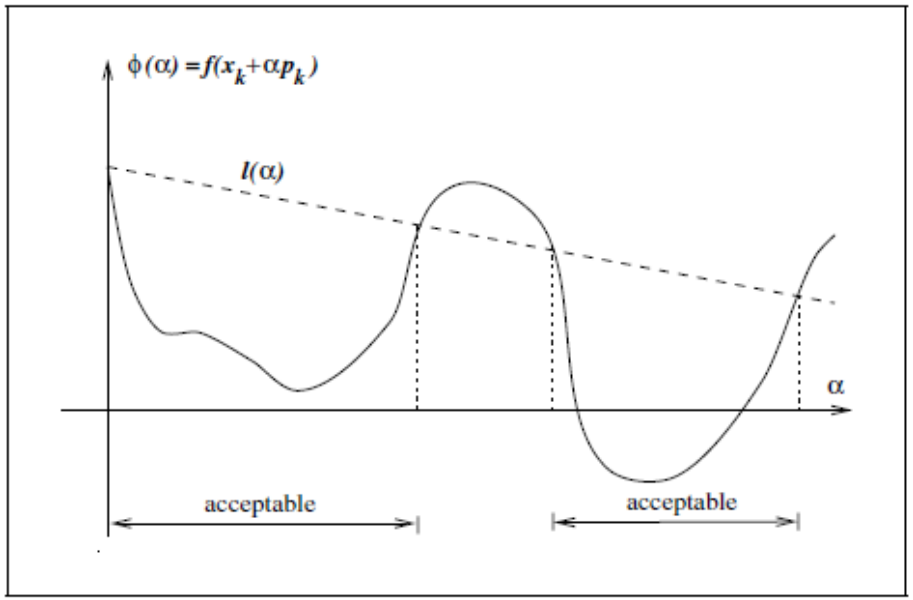
- Armijo inexact line search condition stipulates that α_k should, first of all, give sufficient decrease in the objective function f, as measured by the following inequality:

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$

  for some constant $c_1 \in (0, 1)$.
- Thus, the reduction in f should be proportional to both the step length α_k and the directional derivative $\nabla f(\mathbf{x}_k)\mathbf{d}_k$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Geometric Interpretation of Armijo Condition



- Consider $\phi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}_k)$ and $l(\alpha) = f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$.
- The function $l(\alpha)$ has negative slope $c_1 \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$, but because $c_1 \in (0, 1)$, it lies above the graph of ϕ for small positive values of α.
- The sufficient decrease condition states that α is acceptable only if $\phi(\alpha) \leq l(\alpha)$. In practice, c_1 is chosen to be quite small, say $c_1 = 10^{-4}$.

Backtracking Line Search With Armijo

Backtracking

1. **Initialize:** $\alpha^{(0)} \in (0,1)$, $\tau \in (0,1)$, $l = 0$
2. Until $f(\mathbf{x}_k + \alpha^{(l)}\mathbf{d}_k) > f(\mathbf{x}_k) + c_1\alpha^{(l)}\nabla f(\mathbf{x}_k)^T\mathbf{d}_k$
   1. Set $\alpha^{(l+1)} = \tau\alpha^{(l)}$
   2. $l = l + 1$
3. $\alpha_k = \alpha^{(l)}$

In practice, the following choices are used

- $\tau \in (0.1, 0.5]$
- $c_1 \in [10^{-5}, 10^{-1}]$

Issue with Armijo's condition:

- It does not ensure that the step size is sufficiently large because Armijo's condition can be satisfied even with a very small step size.
- Backtracking partially addresses this by starting from large step sizes and checking Armijo's condition.
- But can we add some other condition to Armijo?

Armijo-Goldstein Line Search

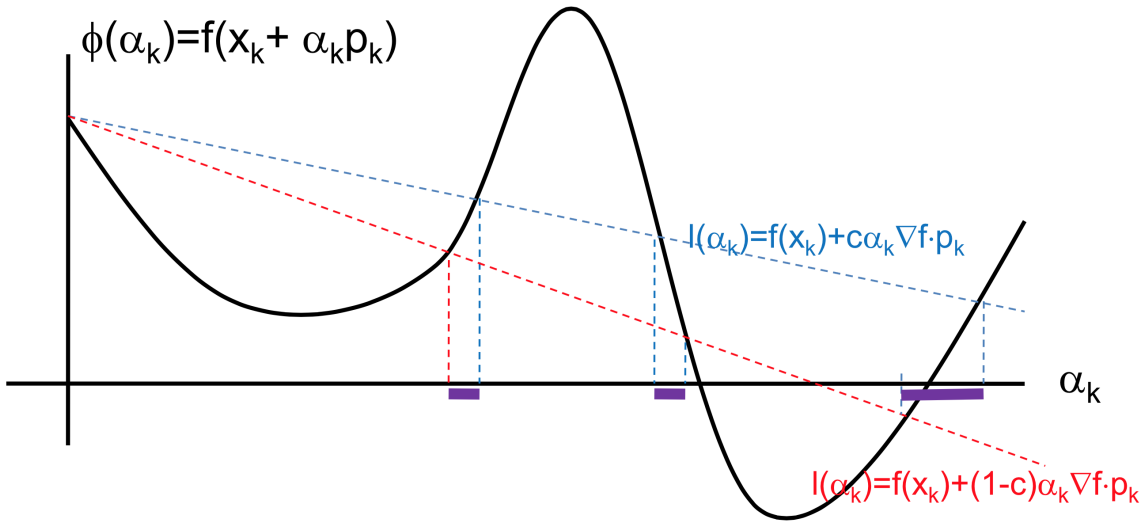
- Armijo-Goldstein inexact line search condition requires that α_k should be sufficiently large and it should give sufficient decrease in the objective function f as well.
- The condition is as follows.

  $$f(\mathbf{x}_k) + (1 - c_1)\alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k \leq f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$

  for some constant $c_1 \in (0, 1/2)$.
- The first inequality is introduced to control the step length from below.
- **Issue:** First inequality may exclude all minimizers of ϕ (see in figure). One can see that the Goldstein condition misses the first local minima.

Geometrical Interpretation of Goldstein Conditions

$$(1-c)\alpha_k \nabla f(x_k)^T p_k + f(x_k) \leq f(x_k + \alpha_k p_k) \leq c\alpha_k \nabla f(x_k)^T p_k + f(x_k)$$
$$(0 < c < 1/2)$$



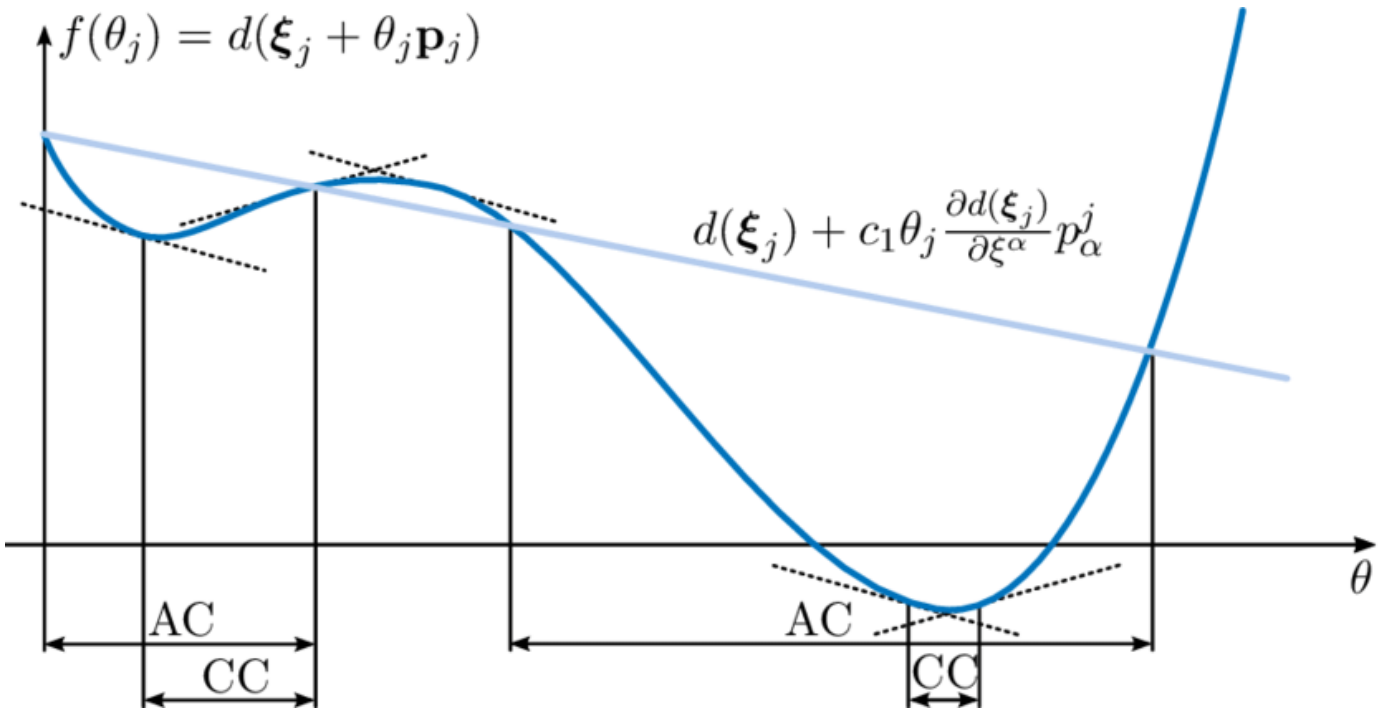
Armijo-Wolfe Condition

- Armijo-Wolfe condition is also used to rule out unacceptably short steps (called the curvature condition) and ensure sufficient decrease.
- The conditions are

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$
$$\nabla f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)^T \mathbf{d}_k \geq c_2 \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$

  for some constants $0 < c_2 < c_1 < 1$.
- LHS in the curvature condition is simply the derivative $\phi'(\alpha_k)$. So, the curvature condition ensures that the slope of ϕ at α_k is greater than c_2 times the initial slope $\phi'(0)$.
- If the slope $\phi'(\alpha)$ is strongly negative, we have an indication that we can reduce f significantly by moving further along the chosen direction. if $\phi'(\alpha_k)$ is only slightly negative or even positive, then we cannot expect more decrease in f in this direction, so it makes sense to terminate the line search.
- Thus, Wolf condition ensures sufficient rate of decrease of function value in the given direction.
- **Issue:** A step length may satisfy the Armijo-Wolfe conditions without being particularly close to a minimizer of ϕ.

Armijo-Wolfe Condition



Optimization Methods (CS1.404), Spring 2025

Naresh Manwani

Machine Learning Lab, IIIT-H

March 3rd, 2025

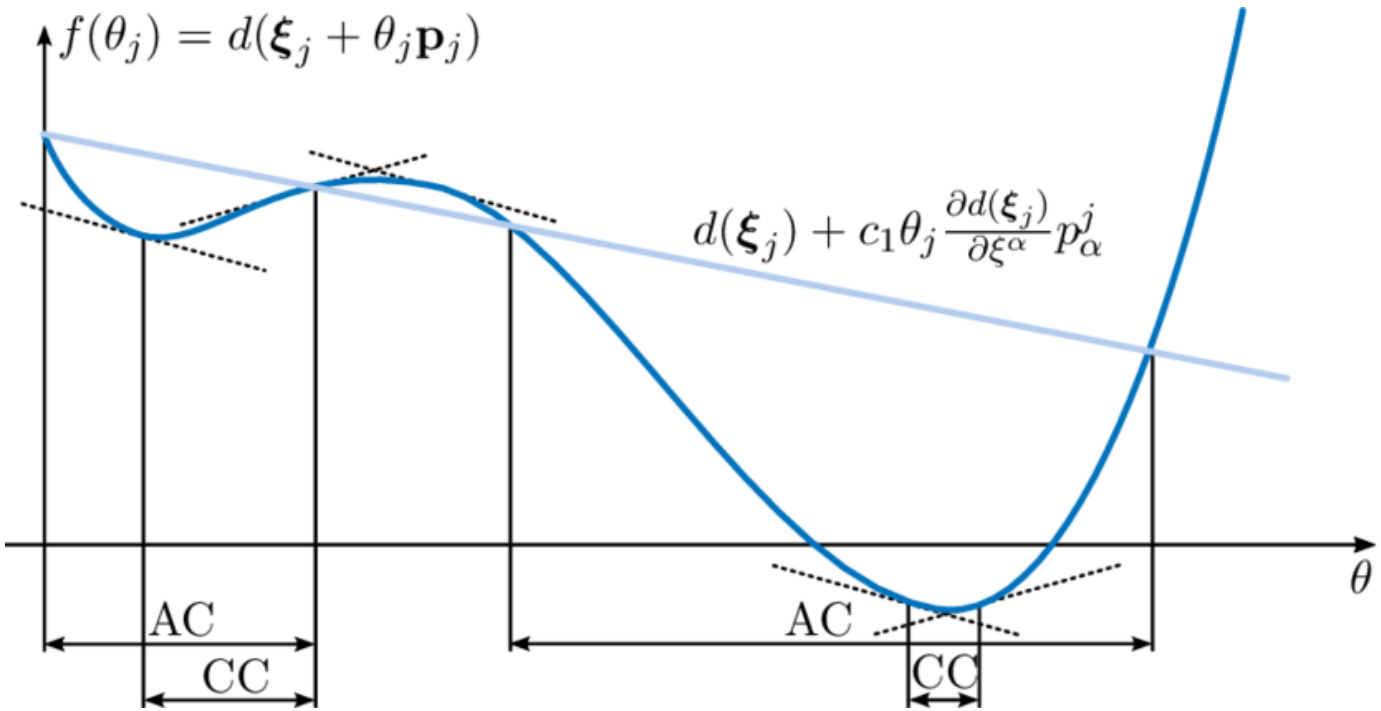
Armijo-Wolfe Condition

- Armijo-Wolfe condition is also used to rule out unacceptably short steps (called the curvature condition) and ensure sufficient decrease.
- The conditions are

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$
$$\nabla f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)^T \mathbf{d}_k \geq c_2 \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$

  for some constants $0 < c_2 < c_1 < 1$.
- LHS in the curvature condition is simply the derivative $\phi'(\alpha_k)$. So, the curvature condition ensures that the slope of ϕ at α_k is greater than c_2 times the initial slope $\phi'(0)$.
- If the slope $\phi'(\alpha)$ is strongly negative, we have an indication that we can reduce f significantly by moving further along the chosen direction. if $\phi'(\alpha_k)$ is only slightly negative or even positive, then we cannot expect more decrease in f in this direction, so it makes sense to terminate the line search.
- Thus, Wolf condition ensures sufficient rate of decrease of function value in the given direction.
- **Issue:** A step length may satisfy the Armijo-Wolfe conditions without being particularly close to a minimizer of ϕ.

Armijo-Wolfe Condition



Exact Line Search

Here, α_k is the minimizer of f along the ray $\mathbf{x}_k + \alpha \mathbf{d}_k$.

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

Example: Exact line search for quadratic function

- Let $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$, where A is an $n \times n$ symmetric positive definite matrix, $\mathbf{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$.
- Let $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{d} \in \mathbb{R}^n$ be a descent direction of f at $\mathbf{x}$.

Then

$$\arg\min_{t \geq 0} \; f(\mathbf{x} + t\mathbf{d}) = -\frac{\nabla f(\mathbf{x})^T \mathbf{d}}{\mathbf{d}^T A \mathbf{d}}$$

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Steepest Gradient Descent

- In the gradient method, the descent direction is chosen as the negative of the gradient at the current point: $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$. For such $\mathbf{d}_k$, we see that $\nabla f(\mathbf{x}_k)^T \mathbf{d}_k = -\|\mathbf{d}_k\|^2 < 0$.
- This is also called the steepest gradient descent direction.

Lemma: Optimality of the Steepest Gradient Descent Diirection

Let f be a continuously differentiable function, and let $\mathbf{x} \in \mathbb{R}^n$ be a non-stationary point (i.e., $\nabla f(\mathbf{x}) \neq \mathbf{0}$). Then, the optimal solution of

$$\min_{\mathbf{d} \in \mathbb{R}^n} \nabla f(\mathbf{x})^T \mathbf{d}$$
$$\text{s.t. } \|\mathbf{d}\| = 1$$

is $\mathbf{d} = -\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Steepest Gradient Descent Algorithm

- **Input:** $\epsilon > 0$ - tolerance parameter
- **Initialization:** Pick $\mathbf{x}_0 \in \mathbb{R}^n$ arbitrarily.
- **General Step:** For any $k = 0, 1, 2, \ldots$ execute the following steps
  1. Fix $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$
  2. Pick stepsize t_k by a line search on the function

$$g(t) = f(\mathbf{x}_k + t\mathbf{d}_k)$$

  3. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k\mathbf{d}_k$
  4. If $\|\nabla f(\mathbf{x}_k)\| \leq \epsilon$, then stop and $\mathbf{x}_{k+1}$ is the output.

Example 1: Gradient Descent with Exact Line Search on Quadratic Function

- Consider function $f(x, y) = x^2 + 2y^2$, whose optimal solution is $(0, 0)$ with optimal value 0.
- Let $(x_0, y_0) = (2, 1)$, $\epsilon = 10^{-5}$.
- The Gradient descent approach stops in 13 iterations and finds a solution that is pretty close to the optimal value. $(x^*, y^*) = (0.1254 * 10^{-5}, -0627 * 10^{-5})$.

```
iter_number =   1 norm_grad = 1.885618 fun_val = 0.666667
iter_number =   2 norm_grad = 0.628539 fun_val = 0.074074
iter_number =   3 norm_grad = 0.209513 fun_val = 0.008230
iter_number =   4 norm_grad = 0.069838 fun_val = 0.000914
iter_number =   5 norm_grad = 0.023279 fun_val = 0.000102
iter_number =   6 norm_grad = 0.007760 fun_val = 0.000011
iter_number =   7 norm_grad = 0.002587 fun_val = 0.000001
iter_number =   8 norm_grad = 0.000862 fun_val = 0.000000
iter_number =   9 norm_grad = 0.000287 fun_val = 0.000000
iter_number =  10 norm_grad = 0.000096 fun_val = 0.000000
iter_number =  11 norm_grad = 0.000032 fun_val = 0.000000
iter_number =  12 norm_grad = 0.000011 fun_val = 0.000000
iter_number =  13 norm_grad = 0.000004 fun_val = 0.000000
```

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Example 1: Gradient Descent with Constant Step Size on Quadratic Function

- Consider function $f(x, y) = x^2 + 2y^2$, whose optimal solution is $(0, 0)$ with optimal value 0.
- Let $(x_0, y_0) = (2, 1)$, $\epsilon = 10^{-5}$, $t_k = 0.1$.
- The Gradient descent approach stops in 58 iterations.
- The stepsize was too small, which caused slow convergence.

```
iter_number =   1 norm_grad = 4.000000 fun_val = 3.280000
iter_number =   2 norm_grad = 2.937210 fun_val = 1.897600
iter_number =   3 norm_grad = 2.222791 fun_val = 1.141888
      :                :                :
iter_number =  56 norm_grad = 0.000015 fun_val = 0.000000
iter_number =  57 norm_grad = 0.000012 fun_val = 0.000000
iter_number =  58 norm_grad = 0.000010 fun_val = 0.000000
```

Example 1: Gradient Descent with Backtracking Line Search on Quadratic Function

- Consider function $f(x, y) = x^2 + 2y^2$, whose optimal solution is $(0, 0)$ with optimal value 0.
- Let $(x_0, y_0) = (2, 1)$, $\epsilon = 10^{-5}$, $\tau = 0.5$, $s = 2$, $c_1 = 0.25$.
- The Gradient descent approach stops in 2 iterations and outputs the exact optimal solution.
- **For this example, inexact line search performs better than exact line search.**

```
iter_number =   1 norm_grad = 2.000000 fun_val = 1.000000
iter_number =   2 norm_grad = 0.000000 fun_val = 0.000000
```

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Example 2: Gradient Descent with Backtracking Line Search on Quadratic Function

- Consider function $f(x, y) = x^2 + \frac{1}{100}y^2$, whose optimal solution is $(0, 0)$ with optimal value 0.
- Let $(x_0, y_0) = (\frac{1}{100}, 1)$, $\epsilon = 10^{-5}$, $\tau = 0.5$, $s = 2$, $c_1 = 0.25$.
- The Gradient descent approach stops in 201 iterations.

```
iter_number =   1 norm_grad = 0.028003 fun_val = 0.009704
iter_number =   2 norm_grad = 0.027730 fun_val = 0.009324
iter_number =   3 norm_grad = 0.027465 fun_val = 0.008958
      :                   :                  :
iter_number = 201 norm_grad = 0.000010 fun_val = 0.000000
```

Convergence of Steepest Gradient Descent

- For different quadratic functions, we observe that the convergence time varies for gradient descent.
- Can we find a measure that can predict how many iterations are needed for the convergence of the Gradient method?
- This measure would quantify, in some sense, the hardness of the problem.
- One such measure that can partially answer the above question is **condition number**.

Convergence of Steepest Gradient Descent with Exact Line Search for Quadratic Function

- Let $f(\mathbf{x}) = \mathbf{x}^T A\mathbf{x}$, where A is a symmetric positive definite matrix.
- For Steepest descent, $\mathbf{d}_k = -\nabla f(\mathbf{x}_k) = -2A\mathbf{x}_k$.
- Exact line search will result in $t_k = \arg\min_{t\geq 0} \; f(\mathbf{x}_k + t\mathbf{d}_k) = \frac{\mathbf{d}_k^T\mathbf{d}_k}{2\mathbf{d}_k^T A\mathbf{d}_k}$. Using this, we get

$$
\begin{aligned}
f(\mathbf{x}_k + t_k\mathbf{d}_k) &= f(\mathbf{x}_k) + t_k^2\mathbf{d}_k^T A\mathbf{d}_k + 2t_k\mathbf{d}_k^T A\mathbf{x}_k \\
&= \mathbf{x}_k^T A\mathbf{x}_k + \frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{4\mathbf{d}_k^T A\mathbf{d}_k} + \frac{\mathbf{d}_k^T\mathbf{d}_k}{2\mathbf{d}_k^T A\mathbf{d}_k}\mathbf{d}_k^T(-\mathbf{d}_k) \\
&= \mathbf{x}_k^T A\mathbf{x}_k - \frac{1}{4}\frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{\mathbf{d}_k^T A\mathbf{d}_k} = \mathbf{x}_k^T A\mathbf{x}_k\left(1 - \frac{1}{4}\frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{(\mathbf{d}_k^T A\mathbf{d}_k)(\mathbf{x}_k^T A\mathbf{x}_k)}\right) \\
&= \mathbf{x}_k^T A\mathbf{x}_k\left(1 - \frac{1}{4}\frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{(\mathbf{d}_k^T A\mathbf{d}_k)(\mathbf{x}_k^T AA^{-1}A\mathbf{x}_k)}\right) \\
&= \left(1 - \frac{(\mathbf{d}_k^T\mathbf{d}_k)^2}{(\mathbf{d}_k^T A\mathbf{d}_k)(\mathbf{d}_k^T A^{-1}\mathbf{d}_k)}\right) f(\mathbf{x}_k)
\end{aligned}
$$

Convergence of Steepest Gradient Descent with Exact Line Search for Quadratic Function

Kantorovich Inequality

Let A be a positive definite $n \times n$ matrix. Then for any $\mathbf{x} \in \mathbb{R}^n$ ($\mathbf{x} \neq \mathbf{0}$), the inequality

$$\frac{(\mathbf{x}^T\mathbf{x})^2}{(\mathbf{x}^TA\mathbf{x})(\mathbf{x}^TA^{-1}\mathbf{x})} \geq \frac{4\lambda_{max}(A)\lambda_{min}(A)}{(\lambda_{max}(A)+\lambda_{min}(A))^2}$$

holds.

Lemma

Let $\{\mathbf{x}_k\}_{k\geq 0}$ be the sequence generated by the gradient descent method with exact line search for finding the minimizer of $f(\mathbf{x}) = \mathbf{x}^TA\mathbf{x}$. Then, for any $k = 0, 1, 2, \ldots$

$$f(\mathbf{x}_{k+1}) \leq \left(\frac{M-m}{M+m}\right)^2 f(\mathbf{x}_k)$$

where $M = \lambda_{max}(A)$ and $m = \lambda_{min}(A)$.

Condition Number and Convergence

Condition Number

Let A be an $n \times n$ positive definite matrix. Then the **condition number** of A is defined as

$$\chi(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

- For quadratic functions with large condition numbers, the gradient method might require a large number of iterations to converge.
- Matrices with large condition numbers are called **ill conditioned**.
- Matrices with small condition numbers are called **well conditioned**.
- In the case of non-quadratic functions, the rate of convergence of $\mathbf{x}_k$ to a given stationary point $\mathbf{x}^*$ depends on the condition number of $\nabla^2 f(\mathbf{x}^*)$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Example: Rosenbrock Function

- The Rosenbrock function is the following function

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

- The optimal solution is $(1, 1)$ with the optimal value 0.
- The Rosenbrock function is extremely ill-conditioned at the optimal solution.

$$\nabla f(\mathbf{x}) = \begin{pmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{pmatrix}$$

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} -400x_2 + 1200x_1^2 + 2 & -400x_1 \\ -400x_1 & 200 \end{pmatrix}$$

- $(1, 1)$ is unique stationary point.
- $\nabla^2 f(1, 1) = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix}$
- Condition number of $\nabla^2 f(1, 1)$ is 2.508×10^3

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Example: Steepest Descent with Backtracking on Rosenbrock Function

- Starting point $\mathbf{x}_0 = [2, 5]^T$. The run required 6890 iterations. So, ill-conditioning of $\nabla^2 f(1, 1)$ has significant impact.

```
iter_number =    1 norm_grad = 118.254478 fun_val = 3.221022
iter_number =    2 norm_grad = 0.723051 fun_val = 1.496586
      :                :                :
iter_number = 6889 norm_grad = 0.000019 fun_val = 0.000000
iter_number = 6890 norm_grad = 0.000009 fun_val = 0.000000
```

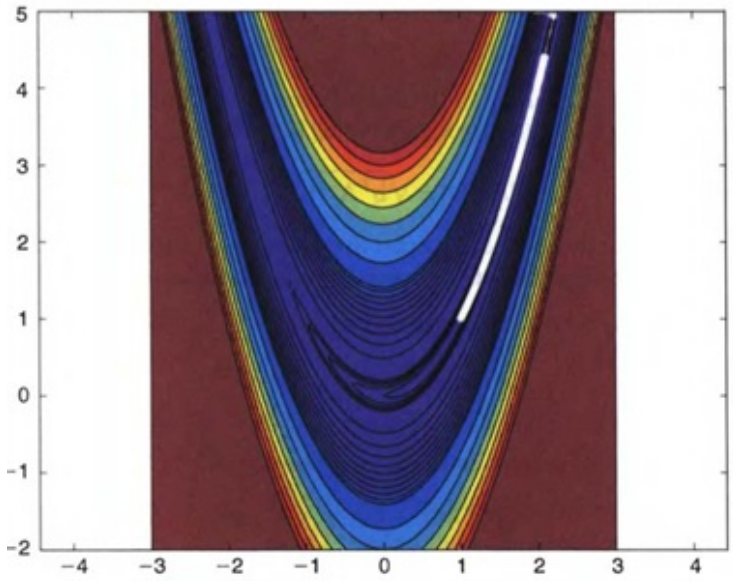


Figure: Banana-shaped contour lines of the Rosenbrock function surrounding the unique stationary point $(1, 1)$. Along with it thousands of iterations of steepest descent.

Convergence Analysis of Gradient Descent

L-Smooth Functions

An L-smooth function is continuously differentiable and that its gradient ∇f is Lipschitz continuous over $\mathbb{R}^n$, meaning that there exists $L > 0$ for which

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \quad \text{for any } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

The class of functions with Lipschitz gradient with constant L are denoted by $\mathcal{C}_L^1$.

Examples:

- **Linear Functions:** Given $\mathbf{a} \in \mathbb{R}^n$, the function $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x}$ is in $\mathcal{C}_0^1$.
- **Quadratic Functions:** Let A be an $n \times n$ symmetric matrix, $\mathbf{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Then,

$$\|f(\mathbf{x}) - f(\mathbf{y})\| = 2\|A(\mathbf{x} - \mathbf{y})\| \leq 2\|A\|.\|\mathbf{x} - \mathbf{y}\|$$

Thus, the Lipschitz constant of ∇f is $2\|A\|$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Interpretation of L-Smoothness

- The gradient of a function measures how the function changes when we move in a particular direction from a point.
- If the gradient were to change arbitrarily quickly, the old gradient does not give us much information at all, even if we take a small step.
- In contrast, smoothness assures us that the gradient cannot change too quickly. Therefore, we have an assurance that the gradient information is informative within a region around where it is taken. The implication is that we can decrease the function's value by moving in the direction opposite of the gradient.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Properties of L-Smooth Function

Theorem 4.20 (Chapter 4: Introduction to Nonlinear Optimization by Amir Beck)

Let f be a twice continuously differentiable function over $\mathbb{R}^n$. Then the following two claims are equivalent.

- $f \in \mathcal{C}_L^1(\mathbb{R}^n)$
- $\|\nabla^2 f(\mathbf{x})\| \leq L$ for any $\mathbf{x} \in \mathbb{R}^n$.

See the proof in the book.

Example: Let $f : \mathbb{R} \to \mathbb{R}$ be given by $f(x) = \sqrt{1+x^2}$. Then,

$$0 \leq f''(x) = \frac{1}{(1+x^2)^{3/2}} \leq 1$$

for any $x \in \mathbb{R}$. Thus, $f \in \mathcal{C}_1^1$.

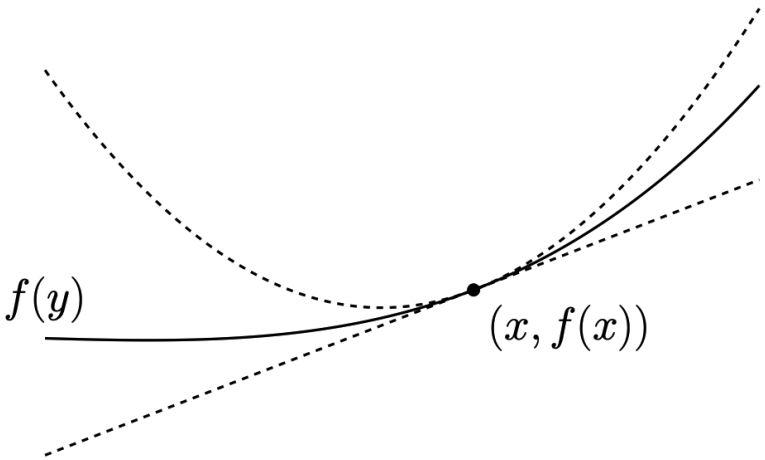
Descent Property of L-Smooth Functions

Lemma 4.22 (Chapter 4: Introduction to Nonlinear Optimization by Amir Beck)

Let $f \in C_L^1(\mathbb{R}^n)$ for some $L > 0$. Then, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, it holds that

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2$$

See the proof in the book.



Comments:

1. This result shows that an L-smooth function can be bounded above by a quadratic function over the entire space.
2. This result is very useful in the convergence proofs of gradient-based methods.

Descent Property of Steepest Descent for L-Smooth Functions

Lemma (Sufficient Decrease of the Gradient Method)

Suppose that $f \in \mathcal{C}_L^1(\mathbb{R}^n)$. Let $\{\mathbf{x}_k\}_{k\geq 0}$ be the sequence generated by the gradient method for solving $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ with one of the following stepsize strategies:

- constant stepsize $\bar{t} \in (0, \frac{2}{L})$
- exact line search
- backtracking procedure with parameters $s \in \mathbb{R}_{++}$, $\alpha \in (0,1)$, $\beta \in (0,1)$.

Then for any $\mathbf{x} \in \mathbb{R}^n$ and $t > 0$

$$f(\mathbf{x}) - f(\mathbf{x} - t\nabla f(\mathbf{x})) \geq M\|\nabla f(\mathbf{x})\|^2$$

where

$$M = \begin{cases} \bar{t}\left(1 - \frac{\bar{t}L}{2}\right), & \text{constant stepsize} \\ \frac{1}{2L}, & \text{exact line search} \\ \alpha \min\left\{s, \frac{2(1-\alpha)\beta}{L}\right\}, & \text{backtracking} \end{cases}$$

- Above result shows that at each iteration the decrease in the function value is at least a constant times the squared norm of the gradient.

Convergence of the Steepest Descent for L-Smooth Functions

Lemma (Sufficient Decrease of the Gradient Method)

Suppose that $f \in \mathbb{C}_L^{1,1}(\mathbb{R}^n)$. Let $\{\mathbf{x}_k\}_{k\geq 0}$ be the sequence generated by the gradient method for solving $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ with one of the following stepsize strategies:

- constant stepsize $\bar{t} \in (0, \frac{2}{L})$
- exact line search
- backtracking procedure with parameters $s \in \mathbb{R}_{++}$, $\alpha \in (0,1)$, $\beta \in (0,1)$.

Assume that f is bounded below over $\mathbb{R}^n$, that is, there exists $m \in \mathbb{R}$ such that $f(\mathbf{x}) > m$ for all $\mathbf{x} \in \mathbb{R}^n$. Then we have the following:

1. The sequence $\{f(\mathbf{x}_k)\}_{k\geq 0}$ is non-increasing. In addition, for any $k \geq 0$, $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$ unless $\nabla f(\mathbf{x}_k) = \mathbf{0}$.
2. $\nabla f(\mathbf{x}_k) \to \mathbf{0}$ as $k \to \infty$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Optimization Methods (CS1.404), Spring 2025

Naresh Manwani

Machine Learning Lab, IIIT-H

March 5th, 2025

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Diagonal Scaling to Improve Condition Number

- Consider the problem $\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} - \mathbf{c}^T\mathbf{x}$, where $\mathbf{H}$ is a symmetric positive definite matrix.
- Condition number of Hessian matrix controls the convergence rate of steepest descent.
- Faster convergence if Hessian is closer to a scalar multiple of the identity matrix.
- Can we transform the problem into another space in which the condition number of the Hessian becomes Identity?
- Let $\mathbf{H} = \mathbf{L}\mathbf{L}^T$ be the Cholesky decomposition of H.
- Define $\mathbf{y} = \mathbf{L}^T\mathbf{x}$.
- Consider the transformed function $h(\mathbf{y}) = f(\mathbf{L}^{-T}\mathbf{y})$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Diagonal Scaling to Improve Condition Number

$$h(\mathbf{y}) = f(\mathbf{L}^{-T}\mathbf{y}) = \frac{1}{2}\mathbf{y}^T\mathbf{L}^{-1}\mathbf{H}\mathbf{L}^{-T}\mathbf{y} - \mathbf{c}^T(\mathbf{L}^{-T}\mathbf{y})$$
$$= \frac{1}{2}\mathbf{y}^T\mathbf{L}^{-1}\mathbf{L}\mathbf{L}^T\mathbf{L}^{-T}\mathbf{y} - \mathbf{c}^T(\mathbf{L}^{-T}\mathbf{y})$$
$$= \frac{1}{2}\mathbf{y}^T\mathbf{y} - \mathbf{c}^T(\mathbf{L}^{-T}\mathbf{y})$$

- The hessian of $h(\mathbf{y})$ is identity matrix.
- Let us apply steepest descent on $\mathbf{y}$ space.

$$\mathbf{y}^{k+1} = \mathbf{y}^k - \nabla h(\mathbf{y}^k) = \mathbf{y}^k - \mathbf{L}^{-1}\nabla f(\mathbf{L}^{-T}\mathbf{y}^k)$$

- Applying transformation $\mathbf{L}^{-T}$ on both sides

$$\mathbf{L}^{-T}\mathbf{y}^{k+1} = \mathbf{L}^{-T}\mathbf{y}^k - \mathbf{L}^{-T}\mathbf{L}^{-1}\nabla f(\mathbf{L}^{-T}\mathbf{y}^k)$$
$$\Rightarrow \quad \mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{H}^{-T}\nabla f(\mathbf{x}^k) = \mathbf{x}^k - \mathbf{H}^{-1}\nabla f(\mathbf{x}^k)$$

- This method is called **Newton Method**.

Newton Method

- Consider $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$, where $f \in \mathcal{C}^2(\mathbb{R}^n)$.
- Newton's method used second-order information to determine the descent direction.
- At each iteration, it uses a second-order Taylor series approximation of f at $\mathbf{x}_k$ and finds the minimum of it to get $\mathbf{x}_{k+1}$.

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \left\{ f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x}-\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^T\nabla^2 f(\mathbf{x}_k)(\mathbf{x}-\mathbf{x}_k) \right\}$$

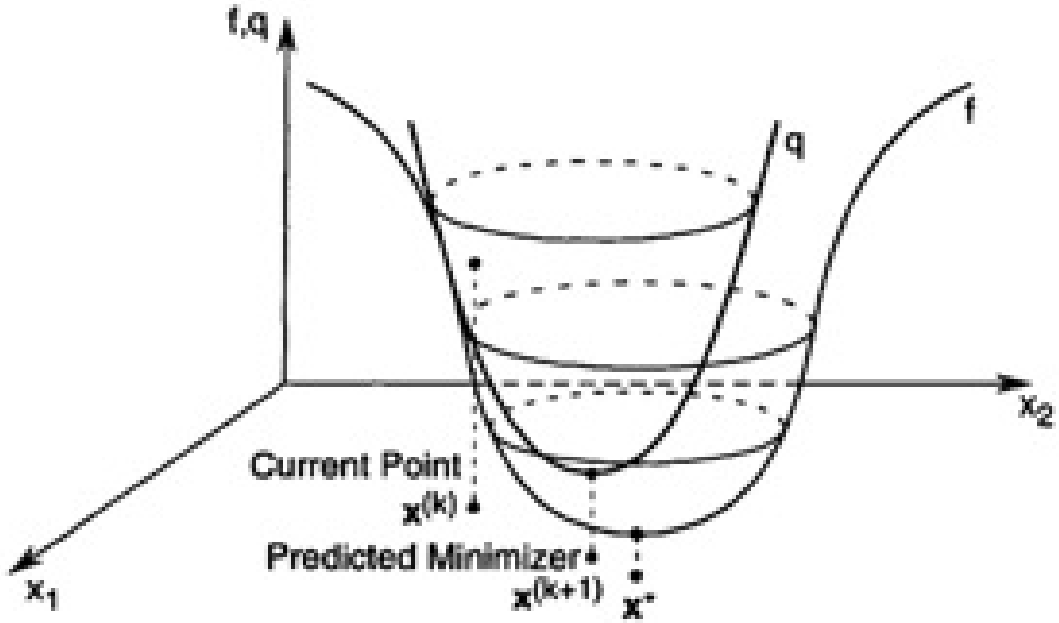
- The above formula is well defined only if we further assume that $\nabla^2 f(\mathbf{x}_k)$ is positive definite. Under this assumption, the unique minimizer is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k)$$

- **Newton Direction:** $\mathbf{d}_N = -(\nabla^2 f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k)$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Geometry of Newton Method



Pure Newton Method

- **Input:** $\epsilon > 0$ -tolerance parameter
- **Initialization:** Pick $\mathbf{x}_0 \in \mathbb{R}^n$ arbitrarily
- **General Step :** For any $k = 0, 1, 2, \ldots$ execute the following steps:
  1. Compute the Newton's direction, which is the solution to the linear system: $\nabla^2 f(\mathbf{x}_k)\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$.
  2. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$
  3. If $\|\nabla f(\mathbf{x}_{k+1})\| \leq \epsilon$, then STOP and output $\mathbf{x}_{k+1}$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Convergence of Newton Method for Quadratic Functions

- Newton method requires that $\nabla^2 f(\mathbf{x})$ is positive definite for every $\mathbf{x}$ (strict convexity).
- Consider quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} - \mathbf{c}^T\mathbf{x}$ such that matrix $\mathbf{H}$ is real symmetric and positive definite matrix.
- We know that the unique global minimizer of f is $\mathbf{x}^* = \mathbf{H}^{-1}\mathbf{c}$.
- We see that $\nabla f(\mathbf{x}) = \mathbf{H}\mathbf{x} - \mathbf{c}$ and $\nabla^2 f(\mathbf{x}) = \mathbf{H}$.
- Applying Newton method on this function for $\mathbf{x}_0$ as initial point, we see that

$$\mathbf{x}_1 = \mathbf{x}_0 - \nabla^2 f(\mathbf{x}_0)^{-1}\nabla f(\mathbf{x}_0) = \mathbf{x}_0 - \mathbf{H}^{-1}(\mathbf{H}\mathbf{x}_0 - \mathbf{c}) = \mathbf{H}^{-1}\mathbf{c}$$

- Thus, using Newton's method, we reach to the global minima of a quadratic and strictly convex function in one step.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Convergence of Newton Method for General Functions

- Newton method requires that $\nabla^2 f(\mathbf{x})$ is positive definite for every $\mathbf{x}$ (strict convexity).
- Which implies a unique optimal solution $\mathbf{x}^*$ exists.
- However, this is not enough to guarantee convergence.
- Consider the following example.

Example

- Consider the function $f(x) = \sqrt{1+x^2}$. The minimizer of f is $x = 0$.
- $f'(x) = \frac{x}{\sqrt{1+x^2}}$, $f''(x) = \frac{1}{(1+x^2)^{3/2}}$.
- Therefore, the Pure Newton method update equations are

$$x_{k+1} = x_k - (1+x_k^2)^{3/2} \frac{x_k}{\sqrt{1+x_k^2}} = x_k - x_k(1+x_k^2) = -x_k^3$$

- Newton method converges to $x^* = 0$ when $|x_0| < 1$. For $|x_0| > 1$, it diverges.

Quadratic Local Convergence of Newton's Method

Theorem

Let f be a twice continuously differentiable function defined over $\mathbb{R}^n$. Assume that

- There exists $m > 0$ for which $\nabla^2 f(\mathbf{x}) \succeq m\mathbf{I}$ for any $\mathbf{x} \in \mathbb{R}^n$,
- There exists $L > 0$ for which $\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Let $\{\mathbf{x}_k\}_{k \geq 0}$ be the sequence generated by Newton's Method, and let $\mathbf{x}^*$ be the unique minimizer of f over $\mathbb{R}^n$. Then for any $k = 0, 1, 2, \ldots$ the inequality

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq \frac{L}{2m}\|\mathbf{x}_k - \mathbf{x}^*\|^2$$

holds. In addition, if $\|\mathbf{x}^* - \mathbf{x}_0\| \leq \frac{m}{L}$, then

$$\|\mathbf{x}^* - \mathbf{x}_k\| \leq \frac{2m}{L}\left(\frac{1}{2}\right)^{2^k}, \ k = 0, 1, 2, \ldots$$

- Thus, near the optimal solution, the error $e_k = \|\mathbf{x}^* - \mathbf{x}_k\|$ satisfies the inequality $e_{k+1} \leq Me_k^2$ for some positive $M > 0$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Example 2: $\nabla f(\mathbf{x}) \succeq m\mathbf{I}$ not satisfied

- Consider the problem $\min_{x_1,x_2} \sqrt{1+x_1^2} + \sqrt{1+x_2^2}$. The optimal solution is $(0,0)$.
- Hessian of the function is $\nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{1}{(1+x_1^2)^{3/2}} & 0 \\ 0 & \frac{1}{(1+x_2^2)^{3/2}} \end{pmatrix} \succeq \mathbf{0}$.
- Even though the Hessian is positive definite, there does not exist an $m > 0$ for which $\nabla^2 f(\mathbf{x}) \succeq m\mathbf{I}$. As $x_1, x_2 \to \infty$, $\nabla^2 f(\mathbf{x})$ becomes a zero matrix.
- Basic assumption for convergence is not satisfied.
- This is reflected in implementation also.
- Newton's method with initial vector $\mathbf{x}_0 = (1,1)$ and tolerance parameter $\epsilon = 10^{-8}$ we obtain convergence after 37 iterations.
- Newton's method with initial vector $\mathbf{x}_0 = (10,10)$ diverges.

```
iter=  1 f(x)=2.8284271247
iter=  2 f(x)=2.8284271247
   :                :
iter= 30 f(x)=2.8105247315
iter= 31 f(x)=2.7757389625
iter= 32 f(x)=2.6791717153
iter= 33 f(x)=2.4507092918
iter= 34 f(x)=2.1223796622
iter= 35 f(x)=2.0020052756
iter= 36 f(x)=2.0000000081
iter= 37 f(x)=2.0000000000
```

(a) Starting point (1,1). Not much progress in 30 iterations. Converges in 37 iterations.

```
iter=  1 f(x)=2000.0009999997
iter=  2 f(x)=1999999999.9999990000
iter=  3 f(x)=19999999999999973000000000.0000000
iter=  4 f(x)=199999999999999230000000000000000000....
iter=  5 f(x)=       Inf
```

(b) Starting point (10,10). Newton's method diverges.

Issues with the Newton Method

- Requires computing inverse of hessian in each iteration. It can be computationally intensive if the number of variables is large.
- No guarantee that $\mathbf{d}_N = -\nabla^2 f(\mathbf{x})^{-1} \nabla f(\mathbf{x})$ is descent direction as the algorithm does not check if the hessian is positive definite.
- Problem happens when hessian is singular in some iteration.
- No guarantee that the function value decreases in each iteration (as no line search is used).
- Convergence is sensitive to the initial point.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Newton method with Backtracking Line Search

Damped Newton Method

- **Input:** $\alpha, \beta \in (0,1)$ - parameters for the backtracking procedure. $\epsilon > 0$ -tolerance parameter
- **Initialization:** Pick $\mathbf{x}_0 \in \mathbb{R}^n$ arbitrarily
- **General Step :** For any $k = 0, 1, 2, \ldots$ execute the following steps:
  1. Compute the Newton's direction, which is the solution to the linear system: $\nabla^2 f(\mathbf{x}_k)\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$.
  2. Set $t_k = 1$. While,
  $$f(\mathbf{x}_k) - f(\mathbf{x}_k + t_k \mathbf{d}_k) < -\alpha t_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$
  Set $t_k = \beta t_k$.
  3. $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$
  4. If $\|\nabla f(\mathbf{x}_{k+1})\| \leq \epsilon$, then STOP and output $\mathbf{x}_{k+1}$.

One can also use other step-size selection methods.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Newton Method with Backtracking on Example 2

- Consider the problem $\min_{x_1,x_2} \sqrt{1+x_1^2} + \sqrt{1+x_2^2}$. The optimal solution is $(0,0)$.
- Take initial point (10,10).
- Using backtracking line search with $\alpha = \beta = 0.5$ and $\epsilon = 10^{-8}$ Newton method converges in 17 iterations.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Levenberg Marquardt Algorithm

- If the hessian matrix $\nabla^2 f(\mathbf{x}_k)^{-1}$ is not positive definite, the Newton direction $\mathbf{d}_N = -\nabla^2 f(\mathbf{x})^{-1} \nabla f(\mathbf{x})$ may not remain a descent direction.
- This issue can be resolved by updating the Newton update in the following way.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I})^{-1} \nabla f(\mathbf{x}_k)$$

  where $\mu_k \geq 0$.
- The idea is as follows.
  - Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $\nabla^2 f(\mathbf{x}_k)$ and $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the corresponding eigenvectors.
  - If $\nabla^2 f(\mathbf{x}_k)$ is not positive definite, then some of the eigenvalues of it are negative.
  - Matrix $\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I}$ has eigenvalues $\lambda_1 + \mu_k, \ldots, \lambda_n + \mu_k$ with $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the corresponding eigenvectors.
  - if μ_k is chosen sufficiently large, all eigenvalues of $\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I}$ can become positive.
  - In that case $-(\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I})^{-1} \nabla f(\mathbf{x}_k)$ becomes a descent direction.

Choosing μ_k

Choosing μ_k

1. Start with some μ_k (a small value)
2. Do the Cholesky factorization of $\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I}$.
3. If Unsuccessful, increase the value of μ_k and go to step 2,

- If μ_k is very large, then this method becomes same as steepest descent.
- If μ_k is very small, then this method becomes same as Newton method.

Levenberg Marquardt Algorithm

- **Input:** Tolerance parameter $\epsilon > 0$, lower bound on minimum eigenvalue $\delta > 0$
- **Initialization:** Pick $\mathbf{x}_0 \in \mathbb{R}^n$ arbitrarily. Set $k = 0$.
- While $(\|\nabla f(\mathbf{x}_k)\| > \epsilon)$
  1. Find the smallest $\mu_k \geq 0$ such that the smallest eigenvalue of $\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I}$ is greater than δ.
  2. Set $\mathbf{d}_k = -(\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I})^{-1} \nabla f(\mathbf{x}_k)$
  3. Find $\alpha_k > 0$ using backtracking
  4. Update $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
  5. $k = k + 1$
- **Output:** $\mathbf{x}^* = \mathbf{x}_k$ as stationary point of $f(\mathbf{x})$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Need for Cholesky Factorization

- In Levenberg Marquardt algorithm, it is required to validate the positive definiteness of the matrix $\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I}$.
- Another issue is to solve the equation $\nabla^2 f(\mathbf{x}_k)\mathbf{d} = -\nabla f(\mathbf{x}_k)$ in general for Newton method.
- These two issues are resolved using Cholesky factorization.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Solving $\mathbf{Ax} = \mathbf{b}$ using Cholesky Factorization

- Let $\mathbf{A}$ be $n \times n$ positive definite matrix. Cholesky factorization of $\mathbf{A}$ has the form $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a lower triangular $n \times n$ matrix whose diagonal is positive
- Given the Cholesky factorization, equation $\mathbf{Ax} = \mathbf{b}$ can be solved in following two steps.
  - Find the solution $\mathbf{u}$ of $\mathbf{Lu} = \mathbf{b}$
  - Find the solution $\mathbf{x}$ of $\mathbf{L}^T\mathbf{x} = \mathbf{u}$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Optimization Methods (CS1.404), Spring 2025

Naresh Manwani

Machine Learning Lab, IIIT-H

March 13th, 2025

Need for Cholesky Factorization

- In Levenberg Marquardt algorithm, it is required to validate the positive definiteness of the matrix $\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I}$.
- Another issue is to solve the equation $\nabla^2 f(\mathbf{x}_k)\mathbf{d} = -\nabla f(\mathbf{x}_k)$ in general for Newton method.
- These two issues are resolved using Cholesky factorization.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Solving $\mathbf{Ax} = \mathbf{b}$ using Cholesky Factorization

- Let $\mathbf{A}$ be $n \times n$ positive definite matrix. Cholesky factorization of $\mathbf{A}$ has the form $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a lower triangular $n \times n$ matrix whose diagonal is positive
- Given the Cholesky factorization, equation $\mathbf{Ax} = \mathbf{b}$ can be solved in following two steps.
  - Find the solution $\mathbf{u}$ of $\mathbf{Lu} = \mathbf{b}$
  - Find the solution $\mathbf{x}$ of $\mathbf{L}^T\mathbf{x} = \mathbf{u}$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Cholesky Factorization Algorithm

- The computation of Cholesky factorization is done using a simple recursive approach.
- Consider the following block matrix partitioning of the matrices A and L.

$$\mathbf{A} = \begin{pmatrix} A_{11} & \mathbf{A}_{21} \\ \mathbf{A}_{12} & \mathbf{A}_{22} \end{pmatrix} \qquad \mathbf{L} = \begin{pmatrix} L_{11} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{pmatrix}$$

  where $A_{11} \in \mathbb{R}, \mathbf{A}_{21} \in \mathbb{R}^{(n-1)\times 1}, \mathbf{A}_{22} \in \mathbb{R}^{(n-1)\times(n-1)}, L_{11} \in \mathbb{R}$, $\mathbf{L}_{21} \in \mathbb{R}^{(n-1)\times 1}, \mathbf{L}_{22} \in \mathbb{R}^{(n-1)\times(n-1)}$.
- Since $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, we have

$$\begin{pmatrix} A_{11} & \mathbf{A}_{21} \\ \mathbf{A}_{12} & \mathbf{A}_{22} \end{pmatrix} = \begin{pmatrix} L_{11}^2 & L_{11}\mathbf{L}_{21}^T \\ L_{11}\mathbf{L}_{21} & \mathbf{L}_{21}\mathbf{L}_{21}^T + \mathbf{L}_{22}\mathbf{L}_{22}^T \end{pmatrix}$$

Cholesky Factorization Algorithm: Continue

- Therefore, in particular $L_{11} = \sqrt{A_{11}}$, $\mathbf{L}_{21} = \frac{1}{\sqrt{A_{11}}}\mathbf{A}_{12}^T$.
- $\mathbf{L}_{22}\mathbf{L}_{22}^T = \mathbf{A}_{22} - \mathbf{L}_{21}\mathbf{L}_{21}^T = \mathbf{A}_{22} - \frac{1}{A_{11}}\mathbf{A}_{12}^T\mathbf{A}_{12}$.
- We are left with the task of Cholesky factorization of $(n-1) \times (n-1)$ matrix $\mathbf{A}_{22} - \frac{1}{A_{11}}\mathbf{A}_{12}^T\mathbf{A}_{12}$.
- We keep following the above procedure and we can get the complete Cholesky factorization.
- The algorithm for Cholesky factorization will find a solution only if all the diagonal elements l_{ii} that are computed during the process are positive, so that computing their square root is possible.
- The positiveness of these elements is equivalent to the property that the matrix to be factored is positive definite.
- Therefore, the Cholesky factorization process can be viewed as a criteria for positive definiteness.

Coordinate Descent Method

- Consider the problem $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$, where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function.
- Coordinate descent method works as follows.
  1. For every coordinate variable x_i, $i \in \{1, \ldots, n\}$, minimize $f(\mathbf{x})$ with respect to x_i, keeping other variables x_j, $j \neq i$ constant.
  2. Repeat the above process in step 1 until some stopping condition is satisfied.

Coordinate Descent Method

Algorithm

- **Input:** $\epsilon > 0$ (tolerance parameter)
- **Initialize:** $\mathbf{x}_1$, $k = 1$
- **Step 1:** Set $\mathbf{d}_k = \mathbf{e}_k$, where $\mathbf{e}_k$ is k^{th} basis vector of standard basis of $\mathbb{R}^n$
- **Step 2:** Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$, where $\alpha_k = \arg\min_{\alpha \in \mathbb{R}} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$.
- **Step 3:**
  - If $(\|\nabla f(\mathbf{x}_k)\| \leq \epsilon)$
    - then output $\mathbf{x}^* = \mathbf{x}_k$
  - Else if $(k = n)$
    - Set $\mathbf{x}_1 = \mathbf{x}_{k+1}$ and repeat from Step 2.
  - Else,
    - Set $k = k + 1$ and repeat from Step 2.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Coordinate Descent Method on Quadratic Functions

- **For convex quadratic functions of n variables, above algorithm converges in n-steps.**
  - Example 1: $\min_{\mathbf{x}\in\mathbb{R}^2}\ 4x_1^2 + x_2^2$ (spherical contours). Take $\mathbf{x}_0 = (-1, -1)$. Coordinate descent method finds minimizer in two steps.
  - Example 2: $\min_{\mathbf{x}\in\mathbb{R}^2}\ 4x_1^2 + x_2^2 - 2x_1x_2$ (elliptical contours) . Take $\mathbf{x}_0 = (-1, -1)$. Coordinate descent method does not converge in two steps.
- In other words, when the objective function is separable in terms of variables (hessian is diagonal), then coordinate descent method will find $\mathbf{x}^*$ in n-steps if there are n-variables.
- When objective function is not separable in variables, then Hessian is not diagonal. Coordinate descent method will not find minimizer in n-steps.
- Can we choose $\mathbf{d}_1, \ldots, \mathbf{d}_n$ in such a way that it converges in n-steps?

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Conjugate Directions

Definition

Let Q be a real symmetric $n \times n$ matrix. The directions $\mathbf{d}_0, \ldots, \mathbf{d}_{n-1}$ are Q-conjugate if, for all $i \neq j$, we have $\mathbf{d}_i^T Q \mathbf{d}_j = 0$.

Lemma

Let Q be a symmetric positive definite $n \times n$ matrix. Let directions $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_k \in \mathbb{R}^n$ $(k \leq n-1)$ are Q-conjugate, then they are linearly independent.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Conjugate Directions: Example 1

- Let $f(x_1, x_2) = 4x_1^2 + x_2^2 - 2x_1x_2$
- Hessian $H = \begin{pmatrix} 8 & -2 \\ -2 & 2 \end{pmatrix}$
- Let $\mathbf{d}_0 = (1, 0)^T$
- Then the conjugate direction $\mathbf{d}_1 = (a, b)^T$ would satisfy $\mathbf{d}_0^T H \mathbf{d}_1 = 0$.
- This results in relation $8a - 2b = 0$. Thus, we can take $\mathbf{d}_1 = (1, 4)^T$.

Conjugate Directions: Example 2

- Let $Q = \begin{pmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{pmatrix}$
- Let $\mathbf{d}_0 = (1, 0, 0)^T$.
- Now, we want to find $\mathbf{d}_1 = (a, b, c)^T$ which is Q-conjugate to $\mathbf{d}_0$. We require $\mathbf{d}_0^T H \mathbf{d}_1 = 0$. Which results in relation $3a + c = 0$. So, we can choose $\mathbf{d}_1 = (1, 0, -3)^T$.
- Now, we want to find $\mathbf{d}_2 = (e, f, g)^T$ which is Q-conjugate to $\mathbf{d}_0$ and $\mathbf{d}_1$. So, we get the conditions $3e + g = 0$ and $-6f - 8g = 0$. We can choose $\mathbf{d}_2 = (1, 4, -3)^T$.

Choosing Conjugate Directions

- A systematic procedure for finding Q-conjugate directions can be developed using the idea of Gram-Schmidt algorithm of transforming a given basis of $\mathbb{R}^n$ into an orthogonal basis of $\mathbb{R}^n$.
- For a symmetric matrix matrix H, orthogonal eigenvectors of H itself are H-conjugate.
  - Let $\mathbf{v}_1$ and $\mathbf{v}_2$ are mutually orthonormal eigenvectors of H corresponding to eigenvalues λ_1 and λ_2.
  - Then $\mathbf{v}_1^T H \mathbf{v}_2 = \lambda_2 \mathbf{v}_1^T \mathbf{v}_2 = 0$.

Conjugate Direction Method

- Consider minimization problem $\min_{\mathbf{x}\in\mathbb{R}^n} \ \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x}$, where H is symmetric positive definite matrix.
- Let $\mathbf{x}_0$ be the initial parameters.
- Let $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{n-1} \in \mathbb{R}^n$ be H-conjugate directions.
- As we know that these conjugate directions are linearly independent. We can write any $\mathbf{x} - \mathbf{x}_0 \in \mathbb{R}^n$ as a linear combination of these conjugate directions. Thus,

$$\mathbf{x} - \mathbf{x}_0 = \sum_{i=0}^{n-1} \alpha_i \mathbf{d}_i$$

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Conjugate Direction Method - Continue

- Given $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{n-1} \in \mathbb{R}^n$ and $\mathbf{x}_0 \in \mathbb{R}^n$, the above minimization problem becomes

$$\begin{aligned}
\phi(\boldsymbol{\alpha}) &= \frac{1}{2}\left(\mathbf{x}_0 + \sum_{i=0}^{n-1}\alpha_i\mathbf{d}_i\right)^T H\left(\mathbf{x}_0 + \sum_{i=0}^{n-1}\alpha_i\mathbf{d}_i\right) + \mathbf{c}^T\left(\mathbf{x}_0 + \sum_{i=0}^{n-1}\alpha_i\mathbf{d}_i\right) \\
&= \frac{1}{2}\mathbf{x}_0^T H\mathbf{x}_0 + \frac{1}{2}\left(\sum_{i=0}^{n-1}\alpha_i\mathbf{d}_i\right)^T H\left(\sum_{i=0}^{n-1}\alpha_i\mathbf{d}_i\right) + \left(\sum_{i=0}^{n-1}\alpha_i\mathbf{d}_i\right) H\mathbf{x}_0 \\
&\quad + \mathbf{c}^T\left(\mathbf{x}_0 + \sum_{i=0}^{n-1}\alpha_i\mathbf{d}_i\right) \\
&= \frac{1}{2}\mathbf{x}_0^T H\mathbf{x}_0 + \frac{1}{2}\sum_{i=0}^{n-1}\alpha_i^2\mathbf{d}_i^T H\mathbf{d}_i + \sum_{i=0}^{n-1}\alpha_i\mathbf{d}_i^T H\mathbf{x}_0 + \mathbf{c}^T\left(\mathbf{x}_0 + \sum_{i=0}^{n-1}\alpha_i\mathbf{d}_i\right) \\
&= \sum_{i=0}^{n-1}\left(\frac{1}{2}(\mathbf{x}_0 + \alpha_i\mathbf{d}_i)^T H(\mathbf{x}_0 + \alpha_i\mathbf{d}_i) + \mathbf{c}^T(\mathbf{x}_0 + \alpha_i\mathbf{d}_i)\right) \\
&\quad - (n-1)\left(\frac{1}{2}\mathbf{x}_0^T H\mathbf{x}_0 + \mathbf{c}^T\mathbf{x}_0\right)
\end{aligned}$$

Conjugate Direction Method - Continue

- Ignoring the constant term, we define a new function

$$\psi(\boldsymbol{\alpha}) = \sum_{i=0}^{n-1} \left(\frac{1}{2}(\mathbf{x}_0 + \alpha_i \mathbf{d}_i)^T H(\mathbf{x}_0 + \alpha_i \mathbf{d}_i) + \mathbf{c}^T(\mathbf{x}_0 + \alpha_i \mathbf{d}_i) \right)$$

- ψ is separable in terms of $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$, which are our new optimization variables.
- Minimizing ψ with respect to α_i, we get

$$\alpha_i^* = -\frac{\mathbf{d}_i^T(H\mathbf{x}_0 + \mathbf{c})}{\mathbf{d}_i^T H \mathbf{d}_i}$$

- $\mathbf{x}^* = \mathbf{x}_0 + \sum_{i=0}^{n-1} \alpha_i^* \mathbf{d}_i$

Basic Conjugate Direction Algorithm

Given starting point $\mathbf{x}_0$ and H conjugate directions $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{n-1} \in \mathbb{R}^n$, the Conjugate Direction Algorithm works as follows:

- For $(k = 0, 1, \ldots, n-1)$
  - $\nabla f(\mathbf{x}_k) = H\mathbf{x}_k + \mathbf{c}$
  - $\alpha_k = -\frac{\nabla f(\mathbf{x}_k)^T \mathbf{d}_k}{\mathbf{d}_k^T H \mathbf{d}_k}$
  - $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$

Theorem

Consider minimization problem $\min_{\mathbf{x} \in \mathbb{R}^n} \ \frac{1}{2}\mathbf{x}^T H \mathbf{x} + \mathbf{c}^T \mathbf{x}$, where H is symmetric positive definite matrix. For any starting point $\mathbf{x}_0$ and H conjugate directions $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{n-1} \in \mathbb{R}^n$, the **Basic Conjugate Direction Algorithm** converges to the unique $\mathbf{x}^*$ (that solves $H\mathbf{x}^* + \mathbf{c} = \mathbf{0}$) in n-steps; that is $\mathbf{x}_n = \mathbf{x}^*$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Optimization Methods (CS1.404), Spring 2025

Naresh Manwani

Machine Learning Lab, IIIT-H

March 28, 2025

Conjugate Gradient Algorithm for Non-Quadratic Problems

- To minimize a non-quadratic function, we first find a quadratic approximation at $\mathbf{x}_k$ using Taylor series and minimize it using conjugate descent to find $\mathbf{x}_{k+1}$.
- We replace H by Hessian at that iteration.
- The conjugate descent algorithm requires computation of Hessian at each iteration which makes it computationally expensive.
- An efficient implementation of conjugate descent eliminates the evaluation of Hessian at each step.
- Note that in conjugate descent algorithm, Hessian appears in the expression of α_k and β_k.
- Because $\alpha_k = \arg\min_{\alpha \geq 0} \ f(\mathbf{x}_k + \alpha \mathbf{d}_k)$, closed form formula for α_k can be replaced by numerical line search method.
- To eliminate Hessian from the formula of β_k, there are three possible ways.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Hestenes-Stiefel Formula

- Recall that $\beta_k = \frac{\mathbf{g}_{k+1}^T H \mathbf{d}_k}{\mathbf{d}_k^T H \mathbf{d}_k}$.
- Here, we replace $H\mathbf{d}_k$ by the term $\frac{\mathbf{g}_{k+1} - \mathbf{g}_k}{\alpha_k}$.
  $\left(\frac{\mathbf{g}_{k+1} - \mathbf{g}_k}{\alpha_k} = \frac{H\mathbf{x}_{k+1} + \mathbf{c} - H\mathbf{x}_k - \mathbf{c}}{\alpha_k} = \frac{H(\mathbf{x}_{k+1} - \mathbf{x}_k)}{\alpha_k} = H\mathbf{d}_k\right)$
- Using this in the β_k formula, we get $\beta_k = \frac{\mathbf{g}_{k+1}^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{d}_k^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}$.
- **For quadratic functions, $\beta_k = \frac{\mathbf{g}_{k+1}^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{d}_k^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}$ is same as $\beta_k = \frac{\mathbf{g}_{k+1}^T H \mathbf{d}_k}{\mathbf{d}_k^T H \mathbf{d}_k}$.**

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Hestenes-Stiefel Approach

Initialize: The starting point $\mathbf{x}_0$ and the tolerance parameter $\epsilon > 0$, Set $k = 0$
Assign $\mathbf{d}_0 = -\mathbf{g}_0$
while $\|\mathbf{g}_k\| > \epsilon$ **do**
  $\alpha_k = \arg\min_{\alpha>0} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
  Compute $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$
  if $(k < n-1)$ **then**
    $\beta_k = \frac{\mathbf{g}_{k+1}^T(\mathbf{g}_{k+1}-\mathbf{g}_k)}{\mathbf{d}_k^T(\mathbf{g}_{k+1}-\mathbf{g}_k)}$
    $\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k$
    $k = k+1$
  else
    $\mathbf{x}_0 = \mathbf{x}_{k+1}$
    $\mathbf{d}_0 = -\mathbf{g}_{k+1}$
    $k = 0$
  end if
end while
Output: $\mathbf{x}^* = \mathbf{x}_k$, a stationary point of f.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Polak-Ribiere Formula

- Starting from Hestenes-Stiefel formula, we multiply out the denominator to get $\beta_k = \frac{\mathbf{g}_{k+1}^T(\mathbf{g}_{k+1}-\mathbf{g}_k)}{\mathbf{d}_k^T\mathbf{g}_{k+1}-\mathbf{d}_k^T\mathbf{g}_k}$.
- But, we know that $\mathbf{d}_k^T\mathbf{g}_{k+1} = 0$.
- Also, since $\mathbf{d}_k = -\mathbf{g}_k + \beta_{k-1}\mathbf{d}_{k=1}$, we get

$$\mathbf{g}_k^T\mathbf{d}_k = -\mathbf{g}_k^T\mathbf{g}_k + \beta_{k-1}\mathbf{g}_k^T\mathbf{d}_{k-1} = -\mathbf{g}_k^T\mathbf{g}_k$$

- Thus, we get $\beta_k = \frac{\mathbf{g}_{k+1}^T(\mathbf{g}_{k+1}-\mathbf{g}_k)}{\mathbf{g}_k^T\mathbf{g}_k}$.
- This expression for β_k is called Polak-Ribiere Formula.
- **For quadratic functions, $\beta_k = \frac{\mathbf{g}_{k+1}^T(\mathbf{g}_{k+1}-\mathbf{g}_k)}{\mathbf{g}_k^T\mathbf{g}_k}$ is same as $\beta_k = \frac{\mathbf{g}_{k+1}^T H\mathbf{d}_k}{\mathbf{d}_k^T H\mathbf{d}_k}$.**

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Polak-Ribiere Approach

Initialize: The starting point $\mathbf{x}_0$ and the tolerance parameter $\epsilon > 0$, Set $k = 0$
Assign $\mathbf{d}_0 = -\mathbf{g}_0$
while $\|\mathbf{g}_k\| > \epsilon$ **do**
  $\alpha_k = \arg\min_{\alpha>0} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
  Compute $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$
  if $(k < n-1)$ **then**
    $\beta_k = \frac{\mathbf{g}_{k+1}^T(\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{g}_k^T \mathbf{g}_k}$
    $\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k$
    $k = k + 1$
  else
    $\mathbf{x}_0 = \mathbf{x}_{k+1}$
    $\mathbf{d}_0 = -\mathbf{g}_{k+1}$
    $k = 0$
  end if
end while
Output: $\mathbf{x}^* = \mathbf{x}_k$, a stationary point of f.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Fletcher Reeves Formula

- Starting with the Polak-Ribiere Formula, we get $\beta_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1} - \mathbf{g}_{k+1}^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{g}_k}$.
- We know that $\mathbf{d}_k = -\mathbf{g}_k + \beta_k \mathbf{d}_{k-1}$. Thus, $\mathbf{g}_{k+1}^T \mathbf{d}_k = -\mathbf{g}_{k+1}^T \mathbf{g}_k + \beta_k \mathbf{g}_{k+1}^T \mathbf{d}_{k-1}$.
- But, we know that $\mathbf{g}_{k+1}^T \mathbf{d}_k = \mathbf{g}_{k+1}^T \mathbf{d}_{k-1} = 0$.
- Thus, $\mathbf{g}_{k+1}^T \mathbf{g}_k = 0$.
- This leads to $\beta_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}$.
- This is called Fletcher Reeves formula.
- **For quadratic functions, $\beta_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}$ is same as $\beta_k = \frac{\mathbf{g}_{k+1}^T H \mathbf{d}_k}{\mathbf{d}_k^T H \mathbf{d}_k}$.**

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Fletcher Reeves Approach

Initialize: The starting point $\mathbf{x}_0$ and the tolerance parameter $\epsilon > 0$, Set $k = 0$
Assign $\mathbf{d}_0 = -\mathbf{g}_0$
while $\|\mathbf{g}_k\| > \epsilon$ **do**
  $\alpha_k = \arg\min_{\alpha>0} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
  Compute $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$
  if $(k < n-1)$ **then**
    $\beta_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}$
    $\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k$
    $k = k + 1$
  else
    $\mathbf{x}_0 = \mathbf{x}_{k+1}$
    $\mathbf{d}_0 = -\mathbf{g}_{k+1}$
    $k = 0$
  end if
end while
Output: $\mathbf{x}^* = \mathbf{x}_k$, a stationary point of f.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Summary: Conjugate Gradient Methods

- Conjugate direction methods can be regarded as being between the method of steepest descent (first-order method that uses gradient) and Newton's method (second-order method that uses Hessian as well).
  - Steepest descent is slow.
  - Newton method is fast, but we need to calculate the inverse of the Hessian matrix.
  - **Conjugate gradient uses gradient only and faster than steepest descent.**
- Conjugate gradient method attempts to accelerate gradient descent by building in momentum.
  - Recall $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
  - Using $\mathbf{d}_k = -\mathbf{g}_k + \beta_{k-1}\mathbf{d}_{k-1}$, we get

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k \mathbf{d}_k \\ &= \mathbf{x}_k - \alpha_k \mathbf{g}_k + \alpha_k \beta_{k-1}\mathbf{d}_{k-1}\end{aligned}$$

  - Using $\mathbf{d}_{k-1} = \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\alpha_{k-1}}$, we get

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k + \frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}}(\mathbf{x}_k - \mathbf{x}_{k-1})$$

momentum term

Quasi Newton Methods

- **Newton Method:** Given a function $f \in \mathcal{C}^2(\mathbb{R}^n)$, Newton method finds the descent direction by solving $H_k \mathbf{d}_k = -\mathbf{g}_k$, where $H_k = \nabla^2 f(\mathbf{x}_k)$ and $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$.
- **Quasi Newton Method:** Given a function $f \in \mathcal{C}^1(\mathbb{R}^n)$, quasi-Newton method finds descent direction as $\mathbf{d}_k = -B_k \mathbf{g}_k$, where B_k is a positive definite matrix.
  - B_k^{-1} is either H_k or its approximation.
  - $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k^{QN} = \mathbf{x}_k - \alpha_k B_k \mathbf{g}_k$
  - Given $\mathbf{x}_k$, $\mathbf{x}_{k+1}$, $\mathbf{g}_k$, $\mathbf{g}_{k+1}$ and B_k, how to get symmetric positive definite B_{k+1}?
  - Are there any conditions that B_{k+1} needs to satisfy?

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Quasi-Newton Method

- We find quadratic approximation of f at $\mathbf{x}_{k+1}$ using B_{K+1} as follows.
  $f_{k+1}(\mathbf{x}) = f(\mathbf{x}_{k+1}) + \mathbf{g}_{k+1}^T(\mathbf{x} - \mathbf{x}_{k+1}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_{k+1})^T B_{k+1}^{-1}(\mathbf{x} - \mathbf{x}_{k+1})$
- We require that $\nabla f_{k+1}(\mathbf{x}_k) = \mathbf{g}_k$ and $\nabla f_{k+1}(\mathbf{x}_{k+1}) = \mathbf{g}_{k+1}$.
- Therefore, using the first condition, we require
  $\nabla f_{k+1}(\mathbf{x}_k) = \mathbf{g}_k = \mathbf{g}_{k+1} + B_{k+1}^{-1}(\mathbf{x}_k - \mathbf{x}_{k-1})$.
- Letting $\gamma_k = \mathbf{g}_{k+1} - \mathbf{g}_k$ and $\boldsymbol{\delta}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, we get $B_{k+1}\gamma_k = \boldsymbol{\delta}_k$.
- This condition is also called **Quasi-Newton condition**.
- B_{k+1} should be positive definite. Thus, $\gamma_k^T B_{k+1}\gamma_k = \gamma_k^T \boldsymbol{\delta}_k > 0$.
  - From Wolfe line search condition

$$\mathbf{g}_{k+1}^T \mathbf{d}_k \geq c_2 \mathbf{g}_k^T \mathbf{d}_k, \quad \text{where } c_2 \in (0,1)$$

$$\Rightarrow (\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{d}_k \geq (c_2 - 1)\mathbf{g}_k^T \mathbf{d}_k$$

We know that $c_2 - 1 < 0$ and $\mathbf{g}_k^T \mathbf{d}_k = -\mathbf{g}_k^T B_k \mathbf{g}_k < 0$ as B_k is positive definite matrix. Thus, we get

$$(\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{d}_k \geq (c_2 - 1)\mathbf{g}_k^T \mathbf{d}_k > 0 \Rightarrow \gamma_k^T \mathbf{d}_k > 0$$

$$\Rightarrow \gamma_k^T \boldsymbol{\delta}_k > 0, \quad \text{using } \mathbf{d}_k = \frac{1}{\alpha_k}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \frac{1}{\alpha_k}\boldsymbol{\delta}_k$$

  - When Wolfe condition is satisfied in a line search, $\exists B_{k+1}$ which satisfies Quasi-Newton condition.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Symmetric Rank One Correction

- Here, we want to update B_k to B_{k+1} by adding a rank one matrix $a_k \mathbf{z}_k \mathbf{z}_k^T$, where $a_k \in \mathbb{R}(a \neq 0)$ and $\mathbf{z}_k \in \mathbb{R}^n(\mathbf{z} \neq \mathbf{0})$. Thus,

$$B_{k+1} = B_k + a_k \mathbf{z}_k \mathbf{z}_k^T$$

- Now, we choose a_k and $\mathbf{z}_k$ such that B_{k+1} satisfies Quasi-Newton condition. Thus, we want

$$\begin{aligned} & B_{k+1}\gamma_k = \boldsymbol{\delta}_k \\ \Rightarrow & (B_k + a_k \mathbf{z}_k \mathbf{z}_k^T)\gamma_k = \boldsymbol{\delta}_k \\ \Rightarrow & a_k \mathbf{z}_k \mathbf{z}_k^T \gamma_k = \boldsymbol{\delta}_k - B_k \gamma_k \end{aligned}$$

- Let $\mathbf{z}_k = \boldsymbol{\delta}_k - B_k \gamma_k$. Therefore, $a_k \mathbf{z}_k^T \gamma_k = 1$.
- That gives $\alpha_k = \frac{1}{(\boldsymbol{\delta}_k - B_k \gamma_k)^T \gamma_k}$.

Thus, using $\mathbf{x}_k$, $\mathbf{x}_{k+1}$, $\mathbf{g}_{k+1}$ and $\mathbf{g}_k$, we get

$$B_{k+1}^{SR1} = B_k + \frac{(\boldsymbol{\delta}_k - B_k \gamma_k)(\boldsymbol{\delta}_k - B_k \gamma_k)^T)}{(\boldsymbol{\delta}_k - B_k \gamma_k)^T \gamma_k}$$

Quasi-Newton Method (Rank One Correction)

1: **Initialize:** The starting point $\mathbf{x}_0$, Symmetric positive definite matrix B_0 and the tolerance parameter $\epsilon > 0$, Set $k = 0$
2: **while** $\|\mathbf{g}_k\| > \epsilon$ **do**
3: $\mathbf{d}_k = -B_k \mathbf{g}_k$
4: Find α_k along $\mathbf{d}_k$ such that

- $f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) < f(\mathbf{x}_k)$
- α_k satisfies Armijo-Wolfe condition

5: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
6: Find B_{k+1} as

$$B_{k+1} = B_k + \frac{(\boldsymbol{\delta}_k - B_k \boldsymbol{\gamma}_k)(\boldsymbol{\delta}_k - B_k \boldsymbol{\gamma}_k)^T)}{(\boldsymbol{\delta}_k - B_k \boldsymbol{\gamma}_k)^T \boldsymbol{\gamma}_k}$$

7: $k = k + 1$
8: **end while**
9: **Output:** $\mathbf{x}^* = \mathbf{x}_k$, a stationary point of f.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Example: Rank One Correction

- Consider the problem min $f(x, y) = 4x^2 + y^2 - 2xy$
- For this problem, $\mathbf{x}^* = [0 \;\; 0]^T$, $H = \begin{bmatrix} 8 & -2 \\ -2 & 2 \end{bmatrix}$, $H^{-1} = \begin{bmatrix} 0.1667 & 0.1667 \\ 0.1667 & 0.6667 \end{bmatrix}$
- We run **rank one correction** approach with $\mathbf{x}_0 = [-2 \;\; -2]^T$ and B_0 as identity matrix.
- We see that the algorithm converges in 3 steps. Below are the updates in each step.

k	x_k	y_k	B_k	$\|\mathbf{g}_k\|$
0	-2	-2	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	12.0
1	0	-2	$\begin{bmatrix} 0.1833 & 0.2333 \\ 0.2333 & 0.9333 \end{bmatrix}$	5.65
2	0.1538	0.1536	$\begin{bmatrix} 0.1667 & 0.1667 \\ 0.1667 & 0.6667 \end{bmatrix}$	0.92
3	0	0	H^{-1}	0

Quasi-Newton Algorithm Applied on Quadratic Functions

- Consider the problem $\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x}$, where H is a symmetric positive definite matrix.
- To solve this problem using rank one correction method, at every iteration k
  - B_{k+1} is symmetric positive definite.
  - B_{k+1} is obtained from $\mathbf{x}_k$, $\mathbf{x}_{k+1}$, $\mathbf{g}_{k+1}$ and $\mathbf{g}_k$.
  - B_{k+1} satisfies Quasi-Newton condition, $B_{k+1}\gamma_k = \delta_k$
- Note that $\mathbf{g}_{k+1} - \mathbf{g}_k = H\mathbf{x}_{k+1} + \mathbf{c} - H\mathbf{x}_k - \mathbf{c} = H(\mathbf{x}_{k+1} - \mathbf{x}_k)$. Which means, $\gamma_k = H\delta_k$.

Lemma: Hereditary Property

SR1 correction approach applied to quadratic function with positive definite Hessian H, we have

$$B_{k+1}\gamma_i = \delta_i,\ 0 \leq i \leq k.$$

When f is quadratic, the hereditary property is satisfied by SR1 regardless of how the line search is performed.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Convergence of SR1 Applied on Quadratic Functions

Theorem1: For Quadratic Functions

Consider SR1 quasi-Newton algorithm applied to a quadratic function with positive definite Hessian H. Then, for any starting point $\mathbf{x}_0$ and any symmetric starting matrix B_0, the sequence of iterates $\mathbf{x}_k$ generated by SR1 converges to the minimizer in n-steps, provided $(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k \neq 0, \forall k$. Moreover, if n-steps are performed and $\boldsymbol{\delta}_0, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_{n-1}$ are linearly independent, then $B_n = H^{-1}$.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Convergence of SR1 Applied on General Functions

- For general nonlinear functions, the SR1 update continues to generate good Hessian approximations under certain conditions.

Theorem

Suppose that f is twice continuously differentiable, and that its Hessian is bounded and Lipschitz continuous in a neighborhood of a point x^*. Let $\mathbf{x}_k,\ k = 0, 1, \ldots$ be any sequence of iterates such that $\mathbf{x}_k \to \mathbf{x}^*$ for some $\mathbf{x}^* \in \mathbb{R}^n$. Suppose in addition that the $|(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k| \geq r\|\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k\|.\|\boldsymbol{\gamma}_k\|$ holds for all k, for some $r \in (0, 1)$, and that $\boldsymbol{\delta}_0, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_{n-1}$ are linearly independent. Then, the matrices B_k generated by the SR1 updating formula satisfy

$$\lim_{k\to\infty} \|B_k - \nabla^2 f(\mathbf{x}^*)\| = 0$$

SR1: Some Remarks

- SR1 is a simple and elegant way to use the information gathered during two consecutive iterations to update B_k.
- B_{k+1} is positive definite if $(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k >$ which can not be guaranteed for every k.
- Numerical difficulties happen if $(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k$ is close to 0.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Issues with SR1

- The main drawback of SR1 updating is that the $(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k$ can vanish.
- Even when the objective function is a convex quadratic, there may be steps on which no symmetric rank-1 update satisfies the Quasi-Newton condition.
- By reasoning in terms of B_k, we see that there are three cases:
  - If $(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k \neq 0$, then there is a unique rank-one updating formula which satisfies Quasi-Newton condition.
  - If $\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k = \mathbf{0}$, then the only updating formula satisfying Quasi-Newton condition is $B_{k+1} = B_k$.
  - If $\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k \neq \mathbf{0}$ and $(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k = 0$, then there is no symmetric rank-one updating formula satisfying the Quasi-Newton Condition.

Strategy to Handle the Case $\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k \neq \mathbf{0}$ and $(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k = 0$

- It has been observed in practice that SR1 performs well simply by skipping the update if the denominator is small.
- More specifically, the SR1 update is applied only if $(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k \geq r\|\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k\|\,\|\boldsymbol{\gamma}_k\|$ where $r \in (0,1)$ is a small number, say $r = 10^{-8}$.
- If this condition is not satisfied, then we update it as $B_{k+1} = B_k$.
- The condition $(\boldsymbol{\delta}_k - B_k\boldsymbol{\gamma}_k)^T\boldsymbol{\gamma}_k = 0$ occurs infrequently since it requires certain vectors to be aligned in a specific way. When it does occur, skipping the update appears to have no negative effects on the iteration.

Optimization Methods (CS1.404), Spring 2024
Lecture 17

Naresh Manwani

Machine Learning Lab, IIIT-H

March 14th, 2024

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Rank Two Correction Quasi newton Method

- Given that B_k is symmetric and positive definite matrix, let

$$B_{k+1} = B_k + \alpha \mathbf{u}\mathbf{u}^T + \beta \mathbf{v}\mathbf{v}^T$$

- B_{k+1} is required to satisfy Quasi-Newton condition. Thus,

$$\alpha \mathbf{u}^T \gamma_k \mathbf{u} + \beta \mathbf{v}^T \gamma_k \mathbf{v} = \boldsymbol{\delta}_k - B_k \gamma_k$$

- Letting $\alpha \mathbf{u}^T \gamma_k = \beta \mathbf{v}^T \gamma_k = 1$, we get $\mathbf{u} + \mathbf{v} = \boldsymbol{\delta}_k - B_k \gamma_k$. Taking $\mathbf{u} = \boldsymbol{\delta}_k$ and $\mathbf{v} = -B_k \gamma_k$, we get

$$\alpha^{-1} = \mathbf{u}^T \gamma_k = \boldsymbol{\delta}_k^T \gamma_k$$
$$\beta^{-1} = \mathbf{v}^T \gamma_k = -\gamma_k^T B_k \gamma_k$$

- Therefore, we get the following update for B_{k+1}:

$$B_{k+1} = B_k + \frac{\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\delta}_k^T \gamma_k} - \frac{B_k \gamma_k \gamma_k^T B_k}{\gamma_k^T B_k \gamma_k}$$

- This update is called DFP named after Davidson, Fletcher and Powell.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

Rank Two Correction Quasi newton Method

Theorem

Given that B_k is symmetric and positive definite, B_{k+1} generated by DFP is symmetric and positive definite.

DFP Quasi-Newton Algorithm

1: **Initialize:** The starting point $\mathbf{x}_0$, Symmetric positive definite matrix B_0 and the tolerance parameter $\epsilon > 0$, Set $k = 0$
2: **while** $\|\mathbf{g}_k\| > \epsilon$ **do**
3: $\mathbf{d}_k = -B_k \mathbf{g}_k$
4: Find α_k along $\mathbf{d}_k$ such that

- $f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) < f(\mathbf{x}_k)$
- α_k satisfies Armijo-Wolfe condition

5: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
6: Find B_{k+1} as

$$B_{k+1} = B_k + \frac{\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\delta}_k^T \boldsymbol{\gamma}_k} - \frac{B_k \boldsymbol{\gamma}_k \boldsymbol{\gamma}_k^T B_k}{\boldsymbol{\gamma}_k^T B_k \boldsymbol{\gamma}_k}$$

7: $k = k + 1$
8: **end while**
9: **Output:** $\mathbf{x}^* = \mathbf{x}_k$, a stationary point of f.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

BFGS Algorithm

- Quasi Newton condition requires $B_{k+1}\gamma_k = \delta_k$ to hold for all k.
- Assume that we want to approximate the Hessian H_{k+1} rather than its inverse. Let $G_{k+1} = B_{k+1}^{-1}$ which approximates Hessian H_{k+1}.
- Then, Quasi-Newton condition would result into $G_{k+1}\delta_k = \gamma_k$.
- Rank two update of G_{k+1} will have the form

$$G_{k+1} = G_k + \alpha \mathbf{u}\mathbf{u}^T + \beta \mathbf{v}\mathbf{v}^T$$

- G_{k+1} is required to satisfy Quasi-Newton condition. Thus,

$$\alpha \mathbf{u}^T \delta_k \mathbf{u} + \beta \mathbf{v}^T \delta_k \mathbf{v} = \gamma_k - G_k \delta_k$$

- Letting $\alpha \mathbf{u}^T \delta_k = \beta \mathbf{v}^T \delta_k = 1$, we get $\mathbf{u} + \mathbf{v} = \gamma_k - G_k \delta_k$. Taking $\mathbf{u} = \gamma_k$ and $\mathbf{v} = -G_k \delta_k$, we get

$$\alpha^{-1} = \mathbf{u}^T \delta_k = \gamma_k^T \delta_k$$
$$\beta^{-1} = \mathbf{v}^T \delta_k = -\delta_k^T G_k \delta_k$$

- Therefore, we get the following update for B_{k+1}:

$$G_{k+1} = G_k + \frac{\gamma_k \gamma_k^T}{\gamma_k^T \delta_k} - \frac{G_k \delta_k \delta_k^T G_k}{\delta_k^T G_k \delta_k}$$

BFGS Algorithm

- Next step is to find B_{k+1} as G_{k+1}^{-1}.
- We use Sherman-Morrison Formula to find G_{k+1}^{-1}.
  $(A + \mathbf{u}\mathbf{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1+\mathbf{v}^T A^{-1}\mathbf{u}}$
- Applying this formula twice to G_{k+1}, we get

$$B_{k+1}^{BFGS} = B_k^{BFGS} + \left(1 + \frac{\gamma_k^T B_k^{BFGS} \gamma_k}{\delta_k^T \gamma_k}\right) \frac{\delta_k \delta_k^T}{\delta_k^T \gamma_k} - \left(\frac{\delta_k \gamma_k^T B_k^{BFGS} + B_k^{BFGS} \gamma_k \delta_k^T}{\delta_k^T \gamma_k}\right)$$

Broyden Family

- $B_{k+1}(\phi) = \phi B_{k+1}^{BFGS} + (1-\phi) B_{k+1}^{DFP}$, where $\phi \in [0,1]$

Broyden Familty Quasi-Newton Algorithm

1: **Initialize:** The starting point $\mathbf{x}_0$, Symmetric positive definite matrix B_0 and the tolerance parameter $\epsilon > 0$, Set $k = 0$
2: **while** $\|\mathbf{g}_k\| > \epsilon$ **do**
3: $\mathbf{d}_k = -B_k \mathbf{g}_k$
4: Find α_k along $\mathbf{d}_k$ such that

- $f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) < f(\mathbf{x}_k)$
- α_k satisfies Armijo-Wolfe condition

5: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
6: Find B_{k+1} as

$$B_{k+1}(\phi) = \phi B_{k+1}^{BFGS} + (1 - \phi) B_{k+1}^{DFP}$$

where $\phi \in [0, 1]$.
7: $k = k + 1$
8: **end while**
9: **Output:** $\mathbf{x}^* = \mathbf{x}_k$, a stationary point of f.