CS427: Mathematics for Data Science *Instructor: Prabuchandran K. J.*

Lecture on 28 March 2023

Lecture 31

Scribes: B. Siddharth Prabhu (200010003), Devesh Kumar (200030017)

1 Recap

So far, we've taken a look at two linear search methods, viz., Gradient Descent method and Newton's Method. Below is an overview of different iterative methods for optimization:

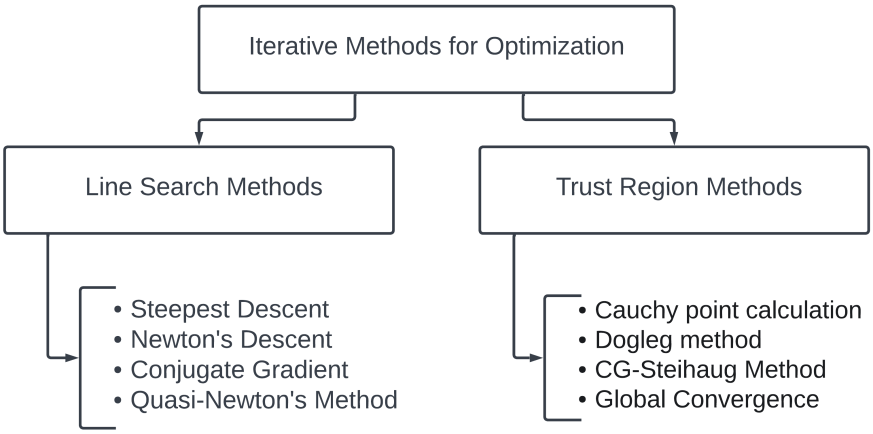


Figure 1: Iterative Algorithms for Optimization

In this Lecture, we shall cover the following topics:

- Line Search: Recap
- Exact and Inexact Line Search
- Characterization of Descent Direction
- Contour Planes
- Armijo-Goldstein Conditions for α_k
- Global Convergence Theorem: A Glimpse
- Problems Addressed for Line Search

(Note that the ordering of certain topics has been altered for cohesion and consistency purposes, in line with various reference materials.)

2 Line Search: Recap

Line search method is an iterative approach to find a local minimum of a function using its gradient. Each iteration of a line search-based algorithm computes a search direction d, and then finds an acceptable step size α, denoting how far to go in that direction. Each iteration is given by:

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

In most approaches, d is chosen in a direction of **descent**. This would require that $\nabla {f_k}^T d_k < 0$, where $f_k = f(x_k)$. Often, we have:

$$d_k = -B_k^{-1} \nabla f_k \tag{2}$$

B_k is chosen to be a symmetric and nonsingular matrix (ideally, positive definite).

Different values of B_k are used in different line search approaches, including:

Choice of B_k	Line Search Algorithm
I (Identity Matrix)	Steepest Descent Algorithm
$\nabla^2 f_k$ (Exact Hessian)	Newton's Method
$\approx \nabla^2 f_k$ (Approximate Hessian)	Quasi-Newton Method

Table 1: Usage of B_k

When d_k is defined as given in Equation (2), and B_k is positive definite, note that:

$$ {d_k}^T \nabla f_k = -{\nabla f_k}^T B_k^{-1} \nabla f_k < 0 \tag{3} $$

In this lecture, we will be exploring methods that help us in choosing values for α_k and d_k.

3 Exact and Inexact Line Search

3.1 Exact Line Search

Consider an iteration of a general line search algorithm, as depicted in Equation (1). Let us define $\Phi(\alpha) = f(x_k + \alpha d_k)$. Then, exact line search dictates that, to solve for α_k, we must compute:

$$ \alpha_k = \underset{\alpha}{argmin}\, \Phi(\alpha) \tag{4} $$

This approach is used when the cost of the minimization problem with one variable is low compared to the cost of computing the search direction itself. However, the algorithm is not appropriate for all practical applications. In the case of a unimodal f, minimization could be done using one of the search algorithms explained in the previous lectures, viz., Golden Section Search, Fibonacci Search.

3.2 Inexact Line Search

More practical strategies perform an inexact line search to identify a step length that achieves adequate reductions in f at minimal cost. In Section (6), we will discuss various termination conditions for line search algorithms. For now, consider a simple condition that requires a decrease in f with every iteration, given as $f(x_{k+1}) < f(x_k)$. This, by itself, is not a sufficient condition for convergence, as is visible in Figure (2), where the minimum value of f, $x^* = -1$. However, insufficient reduction in f at each step causes it to fail to converge to the minimizer of this convex function. In Section (6), we will enforce (and discuss) a *sufficiency* condition.

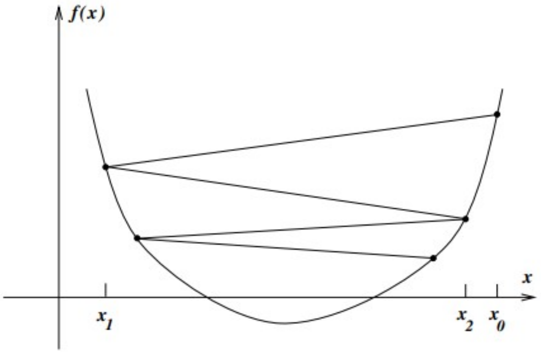


Figure 2: Insufficient Decrease in f

Hence, in brief, line search methods can be categorized into exact and inexact methods. The exact methods aim to find the exact minimizer at each iteration; while inexact methods compute step lengths to satisfy conditions such as Wolfe and Goldstein conditions.

4 Characterization of Descent Direction

It could be said that, for existence of a descent direction d from a given x, for a function $f : \mathbb{R}^n \to \mathbb{R}$, the following must be true:

$$\exists \ \overline{\alpha} \ \ : \ \ f(x + \alpha d) < f(x) \ \ \forall \, \alpha \in (0, \overline{\alpha}) \tag{5}$$

If no such $\overline{\alpha}$ exists for any d, then there is no descent direction from the given point. Also, if such a value does exist, then direction d is called the *descent direction*, and it can be proven that $\nabla f \cdot d < 0$, where $\cdot$ is used to denote dot product. (For vectors a and b, $a \cdot b = a^T b$.) The proof follows directly from the truncated Taylor series.

From the first order truncated Taylor series, we have the following equation:

$$f(x + \alpha d) = f(x) + \alpha \nabla f(\overline{x})^T d \tag{6}$$

where, $\overline{x} = x + td\,, \ \ t \in (0,1)\,, \ \ \alpha \geq 0$. Also, for a descent direction d, we have $f(x + \alpha d) < f(x)$, which means that $\nabla f(x)^T d < 0$. Due to continuity of the gradient, this implies the existence of $\overline{x}$ such that $\nabla f(\overline{x})^T d < 0$. This can be interpreted as follows:

$$\exists \ \delta > 0 \ \ : \|\nabla f(\overline{x}) - \nabla f(x)\| < \epsilon \ \ \forall \, \overline{x} \in B(x, \delta) \tag{7}$$

Without loss of generality, we can take d to be a unit vector, i.e., $\|d\| = 1$. This does not change anything about its direction. Using the Cauchy-Schwartz Inequality, we can effectively show that:

$$\begin{aligned} |(\nabla f(\overline{x}) - \nabla f(x))^T d| &\leq \|\nabla f(\overline{x}) - \nabla f(x)\| \|d\| \\ &= \|\nabla f(\overline{x}) - \nabla f(x)\| \\ &= \epsilon \ \ \text{(say)} \end{aligned} \tag{8}$$

Hence, $|(\nabla f(\overline{x}) - \nabla f(x))^T d| \leq \epsilon$, which implies:

$$\nabla f(x)^T d - \epsilon \leq \nabla f(\overline{x})^T d \leq \nabla f(x)^T d + \epsilon \tag{9}$$

Thus, we can choose $\overline{\alpha}$ in this direction to be equal to δ. Then, the next question that arises is, what about other directions, where $\nabla f^T d = 0$ or $\nabla f^T d > 0$?

- If $\nabla f^T d = 0$, then we may need to use the second order truncated Taylor series, and calculate the Hessian of f.
- If $\nabla f^T d > 0$, then it is not a descent direction.

Hence, we can characterize set of descent directions at a point x for a function $f : \mathbb{R}^n \to \mathbb{R}$ as a set D, such that:

$$D = \{d : \|d\| = 1 \wedge \nabla f^T d < 0\} \tag{10}$$

Example 4.1. *Consider the function* $f : \mathbb{R}^2 \to \mathbb{R}$, *given as* $f(x, y) = x^2 + y^2$. *Comment on the gradient of f at the point* $(1, 2)$.

Solution. We have gradient of f defined as $\nabla f : \mathbb{R}^2 \to \mathbb{R}^2$, where $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]^T$. Hence,

$$\nabla f(x, y) = [2x, 2y]^T$$

Thus, $\nabla f(1, 2) = [2, 4]^T$. This is the maximum rate of change of f from the point $(1, 2)$. □

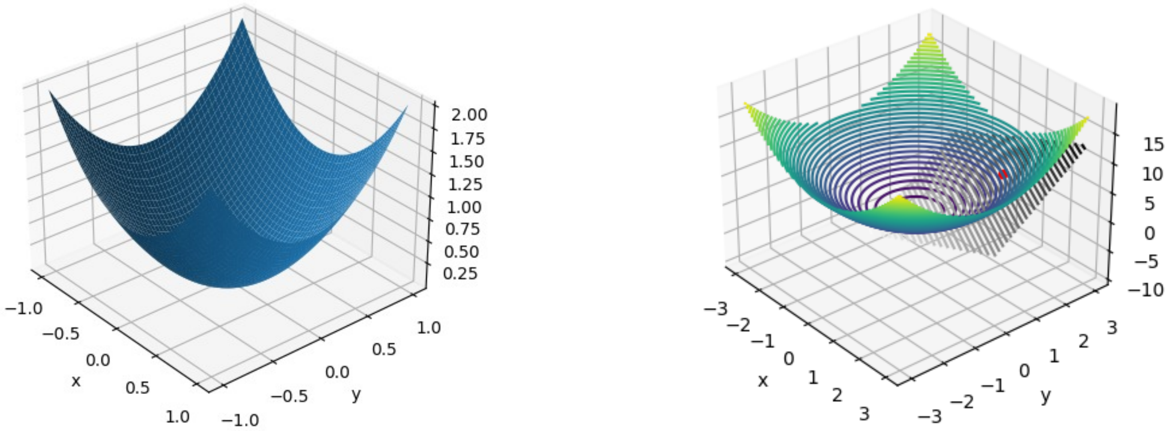
Let us also consider the tangent plane at the point of the surface mentioned in the example presented, such that $z = f(x, y)$. A first order approximation would for such a plane (of the form $ax + by + cz + d = 0$) would be given as,

$$z = z_0 + \left(\frac{\partial f}{\partial x}\right)(x - x_0) + \left(\frac{\partial f}{\partial y}\right)(y - y_0) \tag{11}$$

We know that $(1, 2)$ lies on this plane. So, $x_0 = 1$ and $y_0 = 2$. (Hence, $z_0 = 1^2 + 2^2 = 5$). Substituting all known values, we have:

$$\begin{aligned} z &= 5 + 2(x - 1) + 4(y - 2) \\ \Rightarrow z &= 2x + 4y - 5 \end{aligned} \tag{12}$$

Visually, the function and the tangent plane to the point $(1, 2)$ are as shown in Figure (3).



(a) Function plot in 3-D

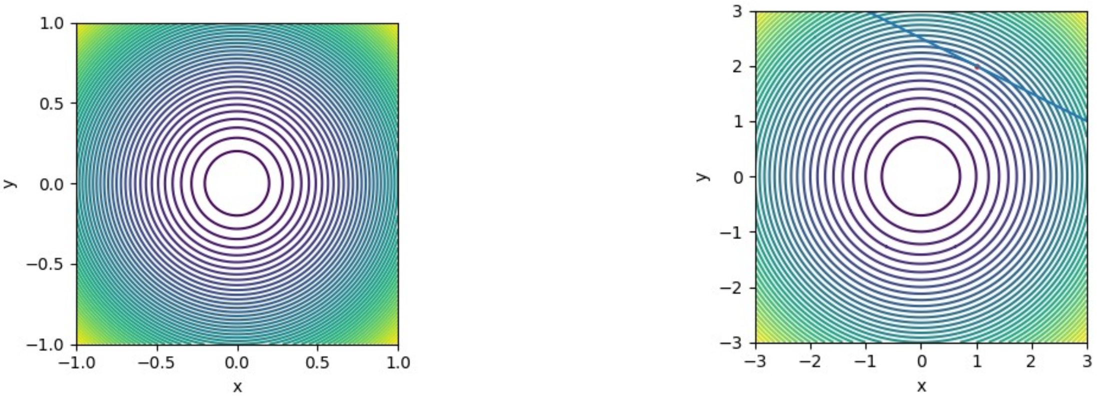
(b) Function along with tangent plane

Figure 3: Plot of the function $f(x, y) = x^2 + y^2$

For the above function, a 3-D view captures the graph well, but visualizing the set of points that all have the same function value isn't trivial. So, we need a 2-D representation to see this. This should be reasonable, since the function itself lives in $\mathbb{R}^2$.

5 Contour Planes

A contour of a function is a surface or curve along which the function has a constant value, so that the surface or curve joins points of equal value. For a function in in $\mathbb{R}^2$, it is a plane section of the three-dimensional graph of the function f parallel to the xy-plane. The contours for the function mentioned above, along with the tangent plane, now a line, are shown in Figure (4).



(a) Function Contours in 2-D

(b) Contours along with tangent line

Figure 4: Contours of the function $f(x, y) = x^2 + y^2$

Theorem 5.1. *For a function $f : \mathbb{R}^n \to \mathbb{R}$, let S be the contour plane such that $f(x) = k$, where k is a constant, and $x = (x_1, x_2, ..., x_n)$. Let P be a point that lies on the contour S. Then, $\nabla f(x)$ is orthogonal to S at P. (In other words, the gradient of a function at a point is always perpendicular to the contour passing through that point.)*

Proof. In class, we considered a proof for the case where $n = 2$. Here, let us generalize this for higher dimensions, considering a point $P = (p_1, p_2, ..., p_n)$ on a contour plane S, where $f(x) = k$, with $k \in \mathbb{R}$. This implies that $f(p_1, p_2, ..., p_n) = k$. We will prove that the gradient of f at P, denoted as $\nabla f|_P$ is perpendicular to the surface, and hence any curve that lies on the surface and passes through P.

Consider any curve on the surface, given as $\gamma(t) = (x_1(t), x_2(t), ..., x_n(t))^T$. Here, $\gamma(t_P) = P$, such that $P = (p_1, p_2, ..., p_n)^T$. Since the curve is on the given contour, we have: $g(t) = f(\gamma(t)) = k$. Differentiating with respect to t, we have:

$$\frac{dg}{dt} = 0$$

Using chain rule, we have:

$$\Rightarrow \left.\frac{\partial f}{\partial x_1}\right|_P \left.\frac{dx_1}{dt}\right|_{t_P} + \left.\frac{\partial f}{\partial x_2}\right|_P \left.\frac{dx_2}{dt}\right|_{t_P} + ... + \left.\frac{\partial f}{\partial x_n}\right|_P \left.\frac{dx_n}{dt}\right|_{t_P} = 0$$

In vector form, using dot product,

$$\Rightarrow \left(\left.\frac{\partial f}{\partial x_1}\right|_P, \left.\frac{\partial f}{\partial x_2}\right|_P, ..., \left.\frac{\partial f}{\partial x_n}\right|_P \right)^T \cdot \left(\left.\frac{dx_1}{dt}\right|_{t_P}, \left.\frac{dx_2}{dt}\right|_{t_P}, ..., \left.\frac{dx_n}{dt}\right|_{t_P} \right)^T = 0$$

$$\Rightarrow \left.\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right)^T\right|_P \cdot \left.\left(\frac{dx_1}{dt}, \frac{dx_2}{dt}, ..., \frac{dx_n}{dt} \right)^T\right|_{t_P} = 0$$

$$\Rightarrow \nabla f(P)^T \cdot \gamma'(t_P)^T = 0$$

Hence, $\nabla f(P)^T \gamma'(t_P) = 0$. Since the dot product is zero, we have shown that the gradient is perpendicular to the tangent to any curve that lies on the contour S.

Thus, we have proven that $\nabla f(x)$ is orthogonal to any contour S of f (where $f(x) = k$), at any point P on S. □

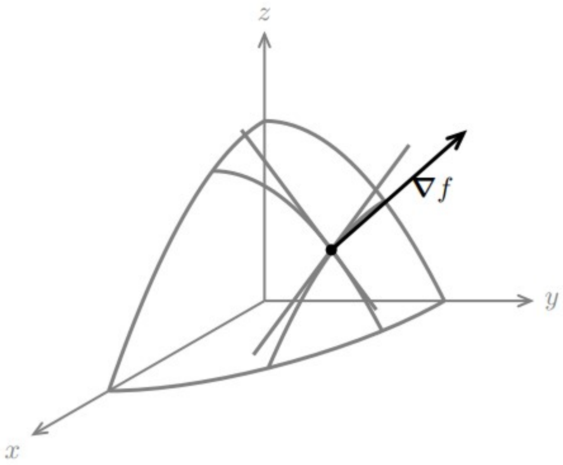


Figure 5: $\nabla f \perp S$

5.1 Example of Contour Planes: $f(x, y) = x + y$

For this function, we have, $\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^T = (1, 1)^T$. Hence, the value of the gradient has the same value for all $(x, y) \in \mathbb{R}^2$. The contours obtained can be observed to form a **ramp**.

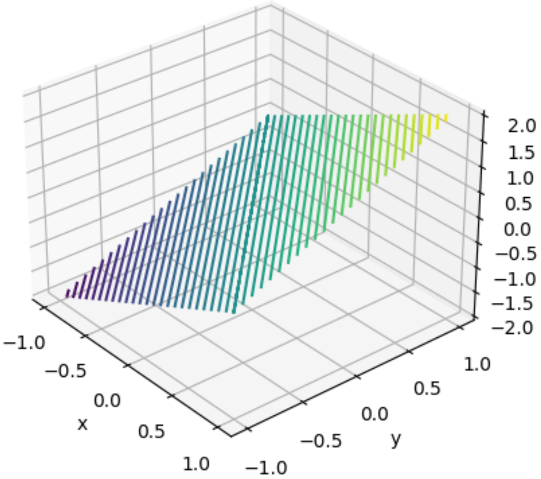


Figure 6: Contours of $f(x, y) = x + y$

5.2 Example of Contour Planes: $f(x, y, z) = x^2 + y^2 + z^2$

For this function, we have, $\nabla f(x, y, z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)^T = (2x, 2y, 2z)^T$. Hence, the value of the gradient has the same value for all $(x, y, z) \in \mathbb{R}^3$. The contours obtained can be observed to form **spherical shells**.

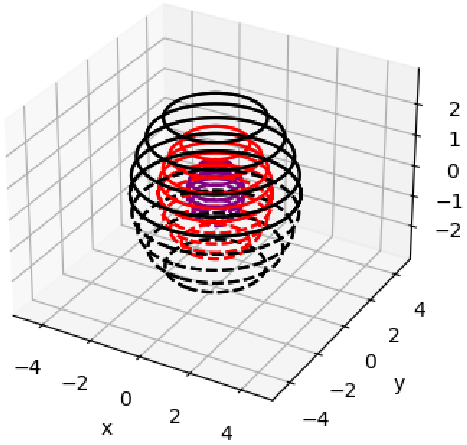


Figure 7: Contours of $f(x, y, z) = x^2 + y^2 + z^2$

6 Armijo-Goldstein Conditions for α_k

So far, we know that $\Phi(\alpha_k) = f(x_k + \alpha_k d_k)$. The sufficiency condition, also called the *Armijo* condition, is given by:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k {\nabla f_k}^T d_k \quad = l(\alpha), \text{(say)} \tag{13}$$

Here, $c_1 \in (0, 1)$. Also, in terms of Φ,

$$\Phi(\alpha_k) \leq \Phi(0) + c_1 \alpha_k \Phi'(0) \tag{14}$$

Also, the *Goldstein* condition, which is used to prevent smaller values of α_k, is given by:

$$f(x_k + \alpha_k d_k) \geq f(x_k) + c_2 \alpha_k \nabla {f_k}^T d_k \ , \ \ 0 < c_1 < c_2 < 1 \tag{15}$$

Together, we can write the above inequalities in a form commonly referred to as the *Armijo-Goldstein inequality*, given below. (Note that $0 < c < 1$, and recall that $f_k = f(x_k)$.)

$$f(x_k) + (1-c)\alpha_k \nabla {f_k}^T d_k \leq f(x_k + \alpha_k d_k) \leq f(x_k) + c\alpha_k \nabla {f_k}^T d_k \tag{16}$$

The Armijo condition from Equation (13) is illustrated in Figure (8), where p_k is used to denote our d_k. This condition states that α is acceptable only if $\Phi(\alpha) \leq l(\alpha)$. In other words, the reduction in f should be proportional to both the step length α_k and the directional derivative $\nabla {f_k}^T d_k$. The function $l(\alpha)$ has negative slope, but lies above Φ for small positive α. In practice, c_1 is chosen to be $\approx 10^{-4}$.

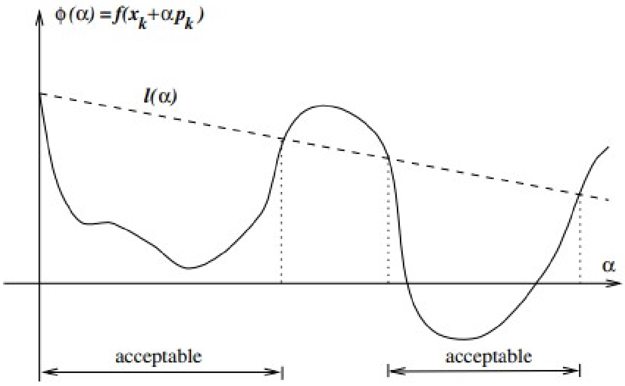


Figure 8: Sufficient Decrease (Armijo) condition

However, the Armijo inequality itself is not sufficient to ensure that the algorithm makes reasonable progress, since we observe that it is satisfied for all sufficiently small values of α. So, the Goldstein condition is used to control the step length from below, as seen in Figure (9), where p_k is used to denote our d_k.

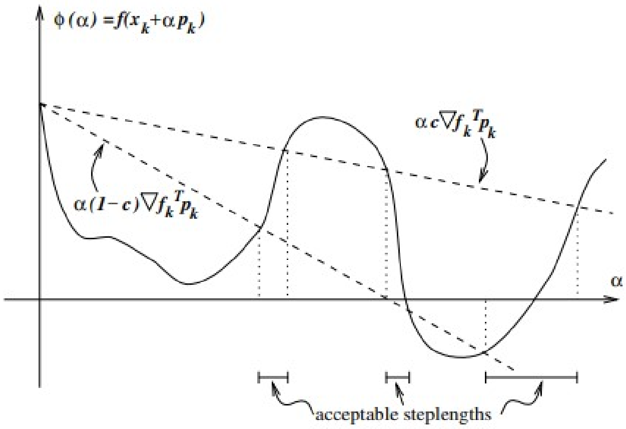


Figure 9: Armijo-Goldstein conditions

An issue that may arise with this approach is that the first inequality in (16) may exclude all minimizers of Φ. This approach is often used in Newton-type methods but is not well suited for Quasi-Newton methods that maintain a positive definite Hessian approximation.

In the next lecture, we will look at the Wolfe conditions, which are more applicable than the above-described conditions for quasi-Newton methods.

7 Global Convergence Theorem: A Glimpse

We use the term *globally convergent* to refer to algorithms for which the property in Equation (17) is satisfied. (The theorem itself wasn't proven in class, but information relating to a proof is in the links provided.)

$$\lim_{k \to \infty} \|\nabla f_k\| = 0 \tag{17}$$

Theorem 7.1. *If Steepest Descent Algorithm is used in conjunction with the Armijo-Goldstein conditions, then $x_k \to x^*$. (i.e., we have Global Convergence.)*

Proof. Ways to look at this, including some proofs, are available in the following links:

- Slides 8 and 9 of Lecture 4 of this course.
- This lecture handout.

□

8 Problems Addressed for Line Search

We have seen that line search methods consist of two steps: finding a descent direction d, and a step size α. Issues that may occur when choosing step size α are as follows:

- Too small α: No progress, premature convergence
- Too large α: May lead to divergence, or function value may not decrease substantially.

The conditions and methods described in this lecture aim to mitigate these problems for unconstrained optimization problems. In the next lecture, we will explore the Wolfe condition, and Backtracking Line Search.

9 References

1. Karunakaran, Dhanoop. "Line search and Trust region methods: two optimisation strategies." *Medium*, 8 October 2020, Link
2. Cao, Lihe et al. "Line search methods." *Cornell University Computational Optimization Open Textbook*, 16 December 2021, Link
3. Nocedal, Jorge et al. "Numerical optimization." (2006) New York, NY: Springer. ISBN: 978-0-387-30303-1
4. Lecture Slides by Prof. Iman Shames for the course 'Introduction to Optimization' taught in 2019 (ELEN90026) at the University of Melbourne. Link
5. Lecture Slides by Prof. Markus Grasmair for the course 'Optimization I' taught in Spring 2019 (TMA4180) at the Norwegian University of Science and Technology. Link
6. Lecture Slides by Prof. Solmaz S. Kia for the course 'Optimization Methods' taught in Spring 2019 (MAE206) at the University of California, Irvine. Link
7. Lecture Handouts by Prof. Amir Ali Ahmadi for the course 'Computing and Optimization' taught in Fall 2021 (ORF 363/COS 323) at Princeton University, NJ. Link

10 Related Code

Some of the plots in this lecture have been coded by us. Refer to this repo on GitHub!