Special Lectures

- Introduction to Continuous Optimization
- Application in Machine Learning

Fundamental problem of continuous optimization :

Find $\underline{x}_*$ such that $\underline{x}_*$ is the minimum of the cost function $f(\underline{x})$	OP

- Unconstrained problem
- $\underline{x} \in \mathbb{R}^n$
- $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is continuous and twice differentiable.

~~Constrained problems~~

Note :

- Parameter space for continuous optimization is $\mathbb{R}^n$
- In case of discrete optimization, the parameter space is $\mathbb{N}^n$ and the solution method is linear programming.

Why optimization ?

- "Operations Research" — optimizing industrial production, logistics, supply chains...
- Key technique in Machine Learning

Why study Mathematics behind Optimization ?

- Inbuilt optimization methods in R, Python, and Matlab.
- Important to understand error messages $\longrightarrow$

from these packages.

- Otherwise, risk of GIGO.

Fundamentals of Unconstrained Optimization:

Aim is to solve

$$\underline{x}_* = \arg\min_{\underline{x} \in \mathbb{R}^n} f(\underline{x})$$

← Unconstrained

Terminology:

- $\underline{x}_*$ is a global minimizer if

$$f(\underline{x}_*) \leq f(\underline{y}) \quad \forall \ \underline{y} \in \mathbb{R}^n.$$

- $\underline{x}_*$ is a local minimizer if there exists a neighbourhood $\mathcal{N}$ of $\underline{x}_*$ such that

$$f(\underline{x}_*) \leq f(\underline{y}) \quad \forall \ \underline{y} \in \mathcal{N}.$$

- $\underline{x}_*$ is a strict local minimizer if there exists a neighbourhood $\mathcal{N}$ of $\underline{x}_*$ such that

$$f(\underline{x}_*) < f(\underline{y}) \quad \forall \ \underline{y} \neq \underline{x}_* \in \mathcal{N}.$$

Necessary conditions for optimality:

If $\underline{x}_*$ solves the OP (i.e. is a local minimizer) then $\nabla f(\underline{x}_*) = 0$. (Analogous to First Derivative Test)

Furthermore, the Hessian matrix

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}\bigg|_{\underline{x}_*}$$

→

is positive-semi-definite (analogous to Second Derivative Test).

Necessary and Sufficient Conditions:

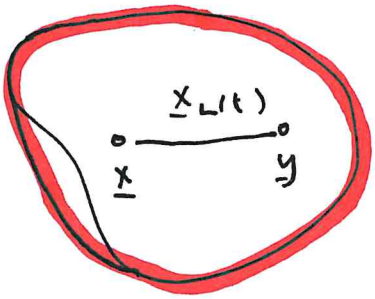
- $\nabla f(\underline{x}_*) = 0$
- H_{ij} is positive-definite:

$$\langle \xi, H\xi \rangle > 0 \quad \forall \ \xi \neq 0 \in \mathbb{R}^n$$

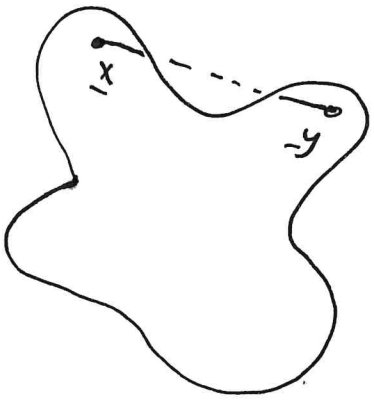
Convexity: S is a convex set if, for each pair $\underline{x}$ and $\underline{y}$ in S, the line segment

$$\underline{x}_L(t) = \underline{x}(1-t) + t\underline{y} \qquad t \in [0,1]$$

is contained entirely in S.



CONVEX



NOT CONVEX

A convex function: A function

$$f: (S \subset \mathbb{R}^n) \longrightarrow \mathbb{R}$$

is convex if:

- S is a convex set
- The following inequality holds for all $\underline{x}$ and $\underline{y}$ in S:

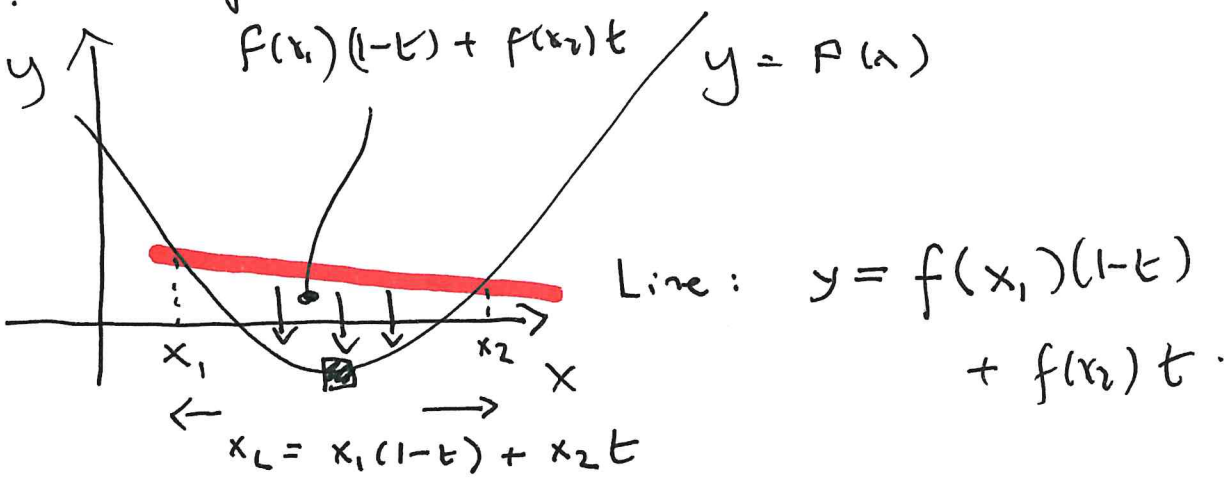
$$f(\underline{x}(1-t) + t\underline{y}) \leq (1-t)f(\underline{x}) + t f(\underline{y})$$

$$\forall \ t \in [0,1].$$

Fundamental Theorem of Convex Optimization:

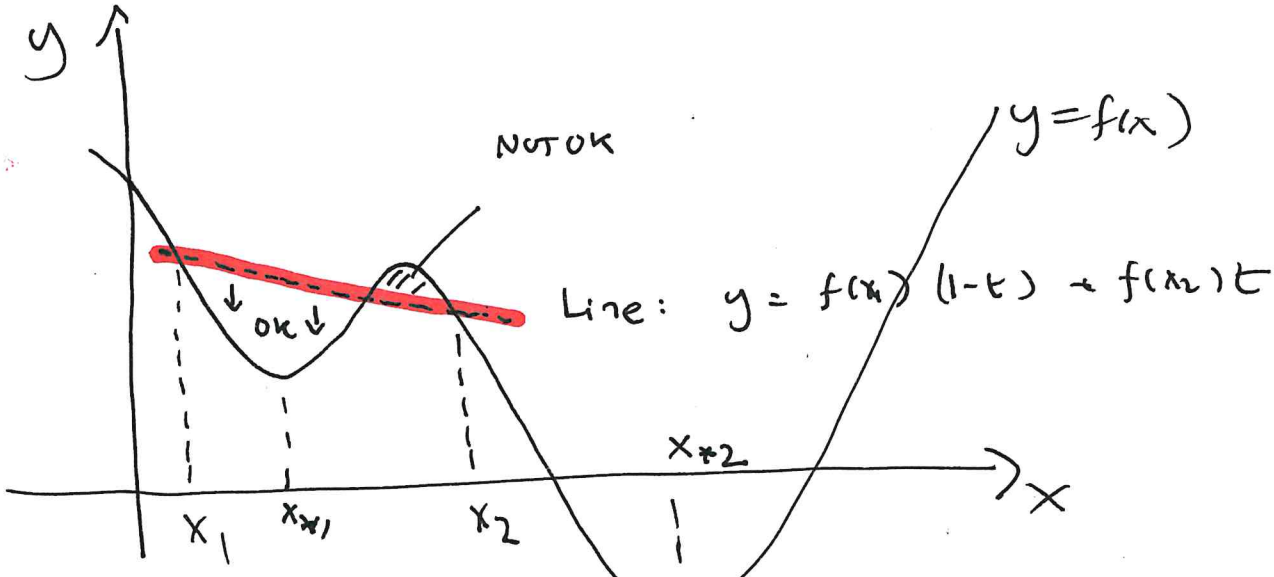
When f is a convex function, any local minimizer x_* is a global minimizer. If, in addition, f is differentiable, then any stationary point $\nabla f(x_*) = 0$ is a global minimizer of f.

IDEA:



Line: $y = f(x_1)(1-t) + f(x_2)t$.

UNIQUE GLOBAL MINIMIZER.



NOT A CONVEX FUNCTION.

- Local min @ x_{*1}
- Global min @ x_{*2}

The Model Problem:

$$f(\underline{x}) = c + \langle \underline{a}, \underline{x} \rangle + \frac{1}{2} \langle \underline{x}, B\underline{x} \rangle$$

- c is constant
- $\underline{a}$ is a constant vector
- B is a positive-definite matrix

$$\nabla f = \underline{a} + B\underline{x}$$

$$\nabla f = 0 \Rightarrow \underline{x} = -B^{-1}\underline{a}$$

$$\boxed{\underline{x}_* = -B^{-1}\underline{a}}$$

Part 2 — Numerical Optimization

Two main methods for unconstrained optimization:

- Linesearch
- Trust Region

We look briefly at Linesearch. This is an iterative method, which aims to get closer and closer to the solution with successive guesses.

If we label the guesses as

$$\underline{x}_0, \underline{x}_1, \dots, \underline{x}_k, \underline{x}_{k+1} \dots$$

we have:

$$\underline{x}_{k+1} = \underline{x}_k + \underline{s}_k$$

Here, the vector $\underline{s}_k$ can be decomposed as:

$$\underline{s}_k = \alpha_k \underline{p}_k$$

α_k: STEPSIZE

$\underline{p}_k$: SEARCH DIRECTION

Once an appropriate search direction has been selected, α_k can be chosen as:

$$\alpha_k = \arg \min_{\alpha > 0} f(\underline{x}_k + \alpha \underline{p}_k)$$

As this is a 1D optimization problem, it is easier to solve than the original OP. Thus, we have reduced the original n-dimensional OP to a sequence of successive steps:

- Finding the search direction
- Solving a 1D OP.

Finding the search direction: Three main methods:

- Steepest Descent (SD)
- Newton / Quasi-Newton
- Trust Region.

SD: Choose $\underline{p}$ to decrease cost function as much as possible. We have:

$$\underline{p} \cdot \nabla f(\underline{x}_k) = \text{Directional derivative in direction } \underline{p}$$

Maximum negative change when $\underline{p} \propto -\nabla f(x_k)$.

Hence:

$$\underline{p}_k = \frac{-\nabla f(x_k)}{\| \nabla f(x_k) \|_2}$$

Newton: Idea is to approximate $f(\underline{x}_k + \underline{s}_k)$ as a quadratic function. Key departure from SD:

$\underline{p}_k$ contains info about the stepsize and the search direction. $\longrightarrow$

We have:

$$\underline{x}_{k+1} = \underline{x}_k + p_k .$$

$$f(\underline{x}_{k+1}) = f(\underline{x}_k + p_k)$$

$$\simeq \underbrace{f(\underline{x}_k) + p \cdot \nabla f(\underline{x}_k) + \tfrac{1}{2} p_i p_j \frac{\partial^2 f}{\partial x_i \partial x_j}\Big|_{\underline{x}_k}}_{= m_k(p)}$$

Minimize $m_k(p)$ — Quadratic model problem.

$m_k(p)$ is minimized when

$$p = -B^{-1}\underline{a}$$

where $\underline{a} = \nabla f(\underline{x}_k)$, $B_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}\Big|_{\underline{x}_k}$

Notation:

$$\underline{x}_{k+1} = \underline{x}_k + p_k^N \leftarrow \text{for Newton}$$

$$p_k^N = -B^{-1}\underline{a}$$

p_k^N contains info about stepsize and search direction.

Convergence: For $\underline{x}_0$ ← STARTING VALUE / INITIAL GUESS sufficiently close to $\underline{x}_*$, linesearch (SD) has the property that

$$\|\underline{x}_{k+1} - \underline{x}_*\|_2 \leq C\|\underline{x}_k - \underline{x}_*\|_2$$

Constant C depends on properties of $\partial^2 f/\partial x_i \partial x_j\,|_{\underline{x}_*}$, if C is close to 1 then convergence can be v. slow.

In contrast, Newton has the property that:

$$\|\underline{x}_{k+1} - \underline{x}_*\|_2 \leq \tilde{C} \|\underline{x}_k - \underline{x}_*\|_2^2$$

- Quadratic convergence
- Newton is much faster than Linesearch but a matrix inversion needs to be performed at each iteration ($O(n^3)$).

Secant Method: Approximation of B^{-1} at each iteration. One particular approx. is the BFGS algorithm — in widespread use ($O(n^2)$).

Part 3 — Application to Machine Learning