Introduction to Continuous Optimization (45 min). I

Continuous optimization:

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

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

Constrained problem:

Find $\underline{x}_*$ such that $\underline{x}_*$ is the minimum of $f(\underline{x})$, and subject to:

$$\begin{cases} c_i(\underline{x}) = 0, & i \in \mathcal{E} \quad \text{EQUALITY CONSTRAINTS} \\ c_i(\underline{x}) \geq 0, & i \in \mathcal{I} \quad \text{INEQUALITY CONSTRAINTS} \end{cases}$$

Continuous optimization:

- Parameter space is $\mathbb{R}^n$, $f: \mathbb{R}^n \to \mathbb{R}$.
- In case of discrete optimization, the parameter space is $\mathbb{N}^n$ or $\{0,1\}^n$, and the solution method here is to do LINEAR PROGRAMMING.

FDS Lectures on Optimization

Example:

FDS
Lectures on
Optimization

Solve

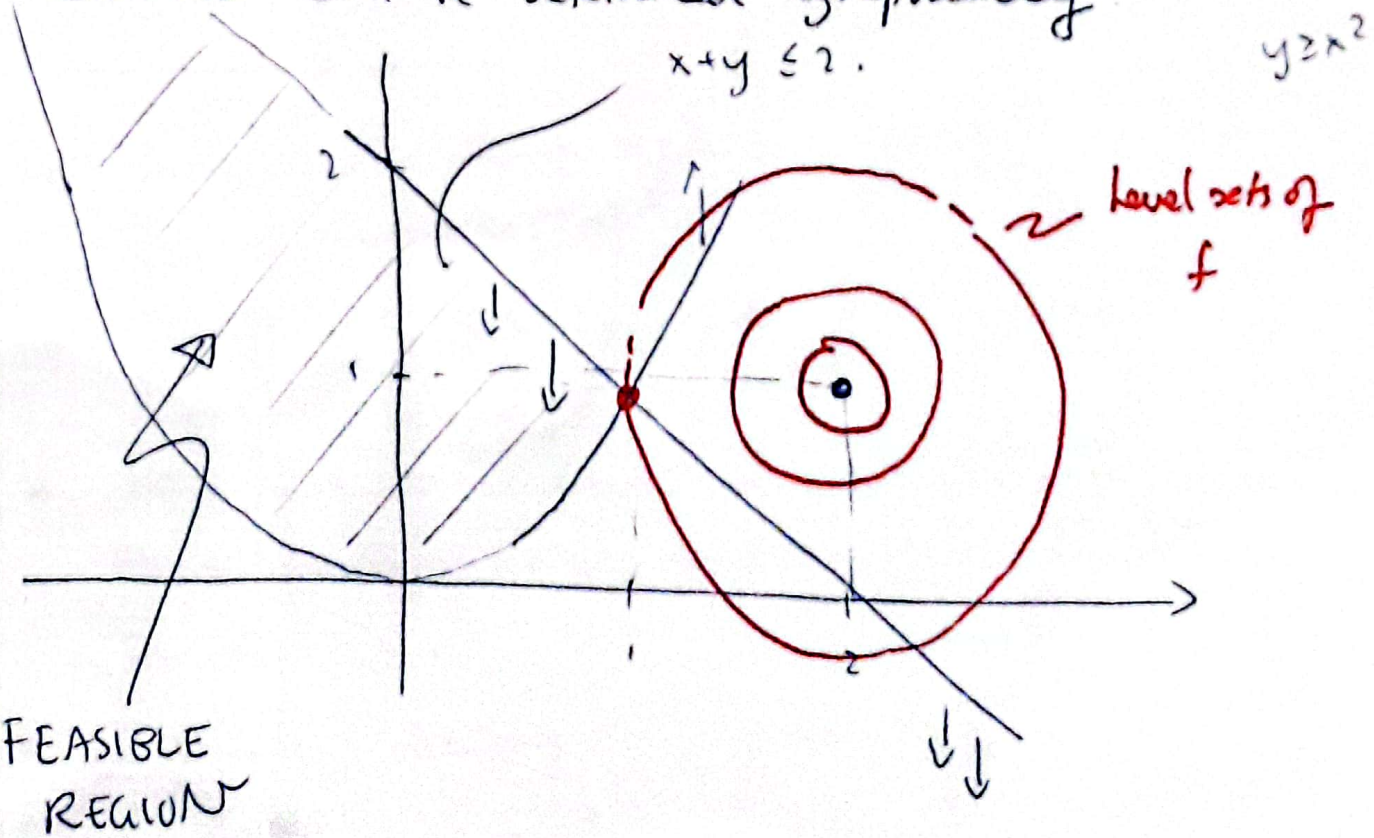
$$\min\left[(x-2)^2+(y-1)^2\right]$$

subject to:
$$\begin{cases} x^2-y \le 0 \\ x+y \le 2 \end{cases}$$

- Objective function: $f(x,y) = (x-2)^2+(y-1)^2$
- Parameter space is $\mathbb{R}^2$, $\underline{x} = (x,y)$
- Two inequality constraints:

$$c(\underline{x}) = \begin{pmatrix} c_1(\underline{x}) \\ c_2(\underline{x}) \end{pmatrix} = \begin{pmatrix} -x^2+y \\ -x-y+2 \end{pmatrix}$$

Solution can be determined graphically:



By inspection, the solution ("MINIMIZER")

- Is in the feasible region
- Is on a level set $f = \text{const}$.

The level sets of f are circles. So x_* is on a circle of minimal radius that just barely lives inside the feasible region:

$$x_* = (1,1).$$

Check: On the bdy of the feasible region we have $y = x^2$ and $y = 2 - x$, hence $2 - x = x^2$, hence $x^2 + x - 2 = 0$. Hence, $(x,y) = (1,1)$. ∎

Why Optimization:

- Nonlinear least squares, fitting a model to a data set.
- Inbuilt optimization methods in R, Python, and Matlab.
- Important to understand error messages from these packages (e.g. condition number of Jacobian).
- Otherwise, there is a risk of GIGO.

Fundamentals of Unconstrained Optimization:

Aim is to solve:

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

Some 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 (is a local minimizer), then $\nabla f(\underline{x}_*) = 0$. (Analogous to the first Derivative Test).

Furthermore, the Hessian matrix

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

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

Necessary and Sufficient Conditions:

- $\nabla f(x_n) = 0$.
- H_{ij} is positive-definite:

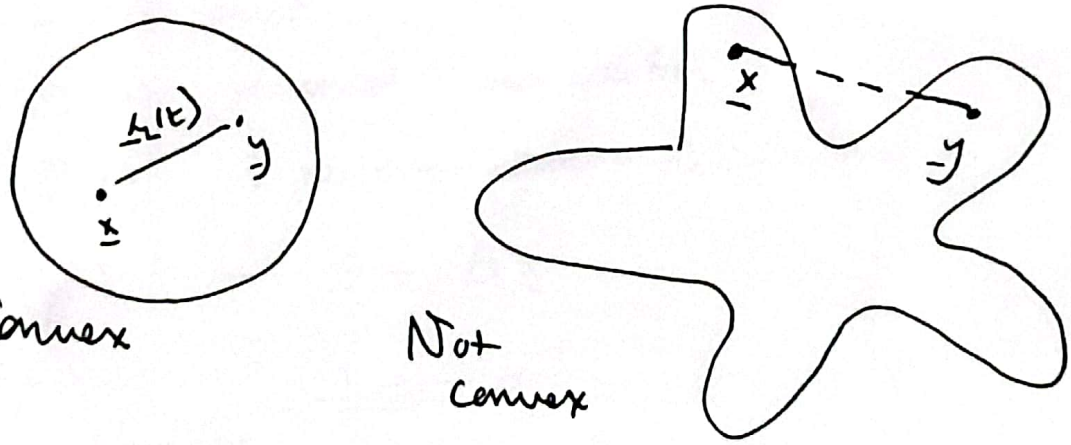
$$\langle \xi, H\xi \rangle > 0 \quad \forall \xi \neq 0 \in \mathbb{R}^n.$$

Convexity: A convex set.

S is a convex set if, for each pair x and y in S, the line segment

$$x_L(t) = x(1-t) + yt, \quad t \in [0,1]$$

is contained entirely in S.



A convex function: A function

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

is a convex function if:

- S is a convex set
- The following inequality holds for all x and y in S:

$$f(x(1-t) + yt) \leq (1-t) f(x) + t f(y) \quad \forall t \in [0,1].$$

Fundamental Theorem of Convex Optimization:

When f is a convex function, any local minimizer x^* is a global minimizer of f. If, in addition, f is differentiable, Then any stationary point x^* is a global minimizer of f.

Proof: ACM 40990.

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 const.
- $\underline{a}$ is a const. vector
- B is a positive-definite matrix.

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

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

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

Numerical Methods for Optimization : VII

- 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, $\underline{s}_k$ can be ~~decomposed~~ written 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 methods :

- Steepest Descent (SD)
- Newton
- Secant

Look at SD. Directional derivative of f along $\underline{p}$ is $\underline{p} \cdot \nabla f$. At $\underline{x}_k$ this is $\underline{p} \cdot \nabla f(\underline{x}_k)$.

If we choose $\underline{p}$ such that f decreases as fast as possible, we would require

$$\underline{p} \propto -\nabla f(\underline{x}_k).$$

To make $\underline{p}$ a unit vector (α is the stepsize) we take:

$$\underline{p} = -\frac{\nabla f(\underline{x}_k)}{\| \nabla f(\underline{x}_k) \|}.$$

Algorithm :

Choose $\underline{x}_0$ (sufficiently close to $\underline{x}_*$)

for $k = 0, 1, 2, \ldots$ **do**

  Compute $\underline{p}_k$

  Choose the stepsize α_k

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

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

end for

CamScanner

Newton Method :

Idea is to approximate $f(x_k + s_k)$ as a quadratic. Key departure : p_k includes info about both the stepsize and the search direction, such that :

$$x_{k+1} = x_k + p_k .$$

Quadratic approx :

$$f(x_k + p) \simeq \underbrace{f(x_k) + p \cdot \nabla f(x_k) + \frac{1}{2} \sum_i \sum_j p_i p_j \frac{\partial^2 f}{\partial x_i \partial x_j}\Big|_{x_k}}_{m_k(p)}$$

Model problem again!

Solution is

$$p_k = \arg\min_{p \in \mathbb{R}^n} m_k(p)$$

is : $p_k = -B^{-1} a$, where

$$B_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}\Big|_{x_k} , \quad a = \nabla f(x_k) .$$

Notation :

$$p_k^N = -B^{-1} a , \text{ where}$$

the superscript N indicates that p_k^N contains info about the stepsize and the search direction.

Convergence: For $\underline{x}_0$ sufficiently close to $\underline{x}_*$, linesearch has the property that:

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

- C depends on properties of $\partial^2 f / \partial x_i \partial x_j |_{\underline{x}_*}$ — can be very slow.

In contrast, Newton has the property that:

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

This is 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 approximation is the BFGS algorithm — in widespread