Non-Linear Programming (NLP)

Optimization problems where either the objective function or the constraints (or both) are non-linear fall into the category of Non-Linear Programming.

1. First-Order and Second-Order Conditions in NLP

Consider a general unconstrained optimization problem:

$$\min f(x), \quad x \in \mathbb{R}^n$$

where $f(x)$ is continuously differentiable.

1.1 First-Order Necessary Condition (FONC)

- At a **local minimum** x^*:

  $$\nabla f(x^*) = 0$$

  That is, the **gradient (vector of first derivatives)** vanishes.
- Geometric interpretation:
  At the minimum, the tangent plane is flat → no slope in any direction.
- Note: This is a **necessary** condition, not sufficient — because it also holds at local maxima and saddle points.

1.2 Second-Order Conditions (SOC)

- Let $H(x) = \nabla^2 f(x)$ denote the **Hessian matrix** (matrix of second derivatives).

At a candidate point x^* where $\nabla f(x^*) = 0$:

- If $H(x^*)$ is **positive definite**, then x^* is a **strict local minimum**.
- If $H(x^*)$ is **positive semi-definite**, then x^* may be a local minimum (not strict).
- If $H(x^*)$ is **negative definite**, then x^* is a **strict local maximum**.
- If $H(x^*)$ is **indefinite** (some eigenvalues positive, some negative), then x^* is a **saddle point**.

1.3 Constrained NLP

- With constraints:

  $$\min f(x), \quad g_i(x) \leq 0, \quad h_j(x) = 0$$

Karush-Kuhn-Tucker (KKT) conditions are the generalization of first-order conditions.

- KKT introduces **Lagrange multipliers** to handle constraints.
  (This belongs to later lectures, but good to keep in mind.)

2. Iterative Methods in NLP

- Analytical solutions (closed-form) are rare in NLP. Instead, we use **iterative methods**:

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

where:

- x_k: current iterate,
- d_k: search direction,
- α_k: step size (learning rate).

2.1 Gradient Descent (Steepest Descent)

- $d_k = -\nabla f(x_k)$.
- Moves in the direction of steepest decrease.
- Convergence can be **very slow** in ill-conditioned problems.

2.2 Newton's Method

- Uses second-order (Hessian) information:

$$d_k = -H(x_k)^{-1}\nabla f(x_k)$$

- Quadratic convergence near the optimum (very fast).
- Issues: computing Hessian is costly, may not be positive definite.

2.3 Quasi-Newton Methods

- Approximate Hessian instead of computing exactly.
- Example: **BFGS algorithm** (widely used in optimization libraries).

2.4 Issues with Iterative Methods

1. **Convergence to Local vs. Global Minimum**
   - Non-convex functions may trap algorithms in local minima.
2. **Choice of Initial Guess x_0**
   - Strongly affects performance and final result.
3. **Step Size (Learning Rate) Selection**
   - Too large → divergence.
   - Too small → very slow convergence.
4. **Ill-conditioning**
   - If level curves are elongated (like a narrow valley), gradient descent zig-zags and converges slowly.
   - Preconditioning or using Newton-type methods helps.

3. Line Search Methods

Line search methods focus on choosing **optimal step size** α_k in each iteration.

General iteration:

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

3.1 Stationarity of Limit Points in Steepest Descent

- If:
  1. The objective function $f(x)$ is continuously differentiable.
  2. Step sizes α_k are chosen by exact or inexact line search (satisfying descent conditions).

Then:

- Any **accumulation point** of the sequence $\{x_k\}$ generated by steepest descent is a **stationary point** (i.e., satisfies $\nabla f(x^*) = 0$).
- In practice: steepest descent may take many iterations to reach acceptable accuracy.

3.2 Successive Step-Size Reduction Algorithms

- Instead of fixing step size, start large and **reduce until progress is adequate**.
- Common rules:

(a) Backtracking Line Search

1. Choose initial $\alpha = 1$.
2. While:

$$f(x_k + \alpha d_k) > f(x_k) + c\alpha \nabla f(x_k)^T d_k$$

   (Armijo condition not satisfied), reduce α (e.g., $\alpha \leftarrow \beta\alpha$, with $\beta \in (0,1)$).
3. Accept the reduced α.

- Guarantees sufficient decrease in each step.

(b) Wolfe Conditions

- Stronger conditions to balance **sufficient decrease** and **curvature condition**.
- Ensures both progress and stability.

Remarks on Line Search

- Backtracking is simple and widely used.
- Exact line search (finding optimal α\alphaα analytically) is rare in practice because it may require solving another optimization problem.
- In machine learning and data analytics, **fixed learning rate with occasional reduction** is common (a practical variant of step-size reduction).

First-order condition: Gradient must vanish at local optima.

Second-order condition: Hessian determines nature of stationary point (min/max/saddle).

Iterative methods: Gradient descent, Newton's, Quasi-Newton are main techniques.

Issues: Step-size selection, local minima, ill-conditioning, sensitivity to starting point.

Line search: Essential for efficient convergence; backtracking and Wolfe conditions widely used.

Steepest descent: Limit points are stationary, but convergence may be slow.

First and Second Order Conditions

Concept Recap

- At local optimum, gradient must vanish: $\nabla f(x^*) = 0$.
- Nature of point determined by Hessian matrix.

Problem 1:

Find the stationary points of

$$f(x) = x^2 - 4x + 5$$

and determine their nature.

Solution:

- Gradient:
  $\frac{df}{dx} = 2x - 4$.
- Set = 0: $2x - 4 = 0 \Rightarrow x = 2$.
- Hessian: $f''(x) = 2 > 0$.
- **Conclusion:** Minimum at $x = 2$, value = $f(2) = 1$.

Problem 2:

For the function

$$f(x, y) = x^2 + y^2 - 2x - 4y + 5$$

find the minimum point.

Solution:

- Gradient:
  $\nabla f(x, y) = (2x - 2,\ 2y - 4)$.
- Stationary point: $2x - 2 = 0 \Rightarrow x = 1$, $2y - 4 = 0 \Rightarrow y = 2$.
- Hessian:
  $\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ (positive definite).
- **Conclusion:** Minimum at $(1, 2)$, value = $f(1, 2) = 0$.

Problem 3:

Check the nature of stationary point for:

$$f(x, y) = x^2 - y^2$$

Solution:

- Gradient: $(2x, -2y)$.
- Stationary point: $(0, 0)$.
- Hessian: $\begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$.
- Eigenvalues: $+2, -2 \rightarrow$ indefinite.
- **Conclusion:** $(0, 0)$ is a **saddle point**.

Remarks:

First-order → stationarity.

Second-order → min/max/saddle classification.

Iterative Methods in NLP

Concept

- Use iterative updates:

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

- Gradient descent: $d_k = -\nabla f(x_k)$.
- Newton's method: $d_k = -H^{-1}(x_k)\nabla f(x_k)$.

Problem 1: Gradient Descent in 1D

Minimize

$$f(x) = (x - 3)^2$$

using gradient descent, starting from $x_0 = 0$, with $\alpha = 0.1$.

Solution:

- Gradient: $\nabla f(x) = 2(x - 3)$.
- Iteration: $x_{k+1} = x_k - 0.1(2(x_k - 3))$.
- Step 1: $x_1 = 0 - 0.1(-6) = 0.6$.
- Step 2: $x_2 = 0.6 - 0.1(-4.8) = 1.08$.
- Step 3: $x_3 = 1.08 - 0.1(-3.84) = 1.464$.
- Converges towards $x = 3$.

Problem 2: Newton's Method in 1D

Minimize

$$f(x) = x^2 + 4x + 4$$

(start from $x_0 = 2$).

Solution:

- Gradient: $f'(x) = 2x + 4$.
- Hessian: $f''(x) = 2$.
- Newton update: $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$.
- Step 1: $x_1 = 2 - (8/2) = -2$.
- Gradient at -2 is 0 → convergence.
- **Minimum at $x = -2$.**

Problem 3: Gradient Descent in 2D

Minimize

$$f(x, y) = (x - 1)^2 + (y - 2)^2$$

starting from $(0, 0)$, $\alpha = 0.1$.

Solution:

- Gradient: $\nabla f(x, y) = (2(x - 1), 2(y - 2))$.
- Start: $(0, 0)$. Gradient = $(-2, -4)$.
- Update: $(x, y) = (0, 0) - 0.1(-2, -4) = (0.2, 0.4)$.
- Next step: Gradient = $(-1.6, -3.2)$.
- Update: $(0.36, 0.72)$.
- Converges towards $(1, 2)$.

Remarks:

- Gradient descent = slow but general.
- Newton = fast but needs Hessian.