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.

Concept of Regression Tree

A Regression Tree is a type of Decision Tree used when the target (output) variable is continuous (not categorical).

It divides the data into smaller and smaller regions so that the value of the dependent variable (Y) within each region is as homogeneous as possible.

How It Works

1. Start with all the training data.
2. At each node, select the variable and the split point that minimizes the sum of squared errors (SSE) or variance within the resulting groups.
3. Repeat splitting recursively until a stopping condition is met (e.g., minimum number of samples, or no significant improvement).
4. The predicted value for each terminal node (leaf) is the mean of the target variable in that region.

Mathematical Criterion for Split

For a split based on variable X_j at split point s:

$$R_1(j,s) = \{X | X_j \leq s\}, \quad R_2(j,s) = \{X | X_j > s\}$$

The cost function is:

$$C(j,s) = \sum_{x_i \in R_1(j,s)} (y_i - \bar{y}_{R_1})^2 + \sum_{x_i \in R_2(j,s)} (y_i - \bar{y}_{R_2})^2$$

We choose (j^*, s^*) that **minimizes** $C(j,s)$.

Dataset

We want to predict the **Sales (Y)** based on the **Advertising Spend (X)** (in ₹ lakh):

Observation	X (Advertising)	Y (Sales)
1	2	4
2	4	6
3	6	8
4	8	10
5	10	11

We will build a **simple regression tree** using one variable X.

Step 1: Find Possible Splits

The possible split points are midpoints between X values:

$$s = 3, 5, 7, 9$$

Step 2: For each split, compute SSE

We'll calculate for each split:

$$SSE = \sum (y_i - \bar{y}_{left})^2 + \sum (y_i - \bar{y}_{right})^2$$

Split 1: s = 3

Left (X ≤ 3): Y = [4] → mean = 4

Right (X > 3): Y = [6, 8, 10, 11] → mean = 8.75

SSE =

Left: $(4-4)^2 = 0$

Right: $(6-8.75)^2 + (8-8.75)^2 + (10-8.75)^2 + (11-8.75)^2$

$= 7.56 + 0.56 + 1.56 + 5.06 =$ **14.74**

→ **Total SSE = 14.74**

Split 2: s = 5

Left ($X \leq 5$): Y = [4, 6] → mean = 5

Right ($X > 5$): Y = [8, 10, 11] → mean = 9.67

SSE =

Left: $(4-5)^2 + (6-5)^2 = 1 + 1 = 2$

Right: $(8-9.67)^2 + (10-9.67)^2 + (11-9.67)^2 = 2.78 + 0.11 + 1.78 = 4.67$

→ **Total SSE = 2 + 4.67 = 6.67**

Split 3: s = 7

Left ($X \leq 7$): Y = [4, 6, 8] → mean = 6

Right ($X > 7$): Y = [10, 11] → mean = 10.5

SSE =

Left: $(4-6)^2 + (6-6)^2 + (8-6)^2 = 4 + 0 + 4 = 8$

Right: $(10-10.5)^2 + (11-10.5)^2 = 0.25 + 0.25 = 0.5$

→ **Total SSE = 8 + 0.5 = 8.5**

Split 4: s = 9

Left ($X \leq 9$): Y = [4, 6, 8, 10] → mean = 7

Right ($X > 9$): Y = [11] → mean = 11

SSE =

Left: $(4-7)^2 + (6-7)^2 + (8-7)^2 + (10-7)^2 = 9 + 1 + 1 + 9 = 20$

Right: $(11-11)^2 = 0$

→ **Total SSE = 20**

Step 3: Choose the Best Split

Split	SSE	Decision
s = 3	14.74	—
s = 5	**6.67 (minimum)**	✅ Best
s = 7	8.5	—
s = 9	20	—

Hence, the **best split** is at **X = 5**.

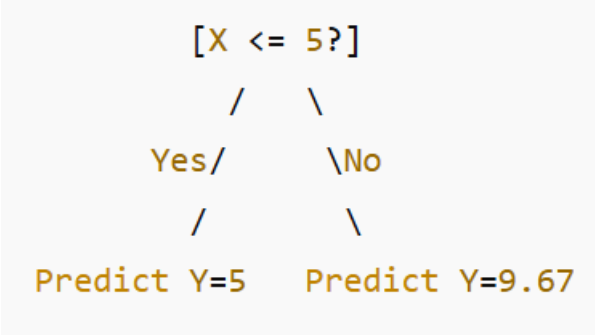
Step 4: Interpret the Tree

- **If X ≤ 5**: predict Y = 5
- **If X > 5**: predict Y = 9.67

Step 5: Predicted values

X	Y (Actual)	Predicted Y
2	4	5
4	6	5
6	8	9.67
8	10	9.67
10	11	9.67

Final Regression Tree



Mean Squared Error (MSE) in Regression Analysis

In regression analysis, the **Mean Squared Error (MSE)** is a commonly used metric to measure how well a regression model fits the data. It represents the **average of the squares of the errors** — that is, the average squared difference between the **actual (observed)** and the **predicted** values. It is used as a **loss function** in regression algorithms (e.g., linear regression, neural networks). During model training, the goal is often to **minimize the MSE**.

If the actual values are $y_1, y_2, \ldots, y_n$ and the corresponding predicted values from the regression model are $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$, then:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where:

- y_i = actual value
- $\hat{y}_i$ = predicted value
- n = number of data points

Interpretation

- **MSE = 0** means a perfect fit (predictions are exactly equal to actual values).
- A **smaller MSE** value indicates a better fit of the regression model to the data.
- A **larger MSE** value indicates poor predictive accuracy — the model's predictions deviate more from actual data.

Observation (i)	Actual Value y_i	Predicted Value $\hat{y}_i$	Error $y_i - \hat{y}_i$	Squared Error $(y_i - \hat{y}_i)^2$
1	10	12	-2	4
2	8	9	-1	1
3	12	11	1	1
4	14	13	1	1

$$\text{MSE} = \frac{4+1+1+1}{4} = \frac{7}{4} = 1.75$$

Root Mean Square Error (RMSE)	$\sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2}$	Square root of MSE; same units as the response variable.
R^2 (Coefficient of Determination)	$1 - \frac{\text{SSR}}{\text{SST}}$	Measures proportion of variance explained by the model.

Advantages of MSE

- Easy to compute and widely used.
- Useful in optimization algorithms (like gradient descent in machine learning).
- Penalizes large errors heavily due to squaring, which helps discourage outliers.

Limitations of MSE

- Sensitive to outliers because errors are squared.
- The value is not in the same units as the original data (since errors are squared).
- May not be intuitive for interpretation compared to MAE.

Usage in Model Evaluation

In regression analysis and machine learning:

- MSE helps compare different models — the one with the lowest MSE is usually preferred.
- It is commonly used as the loss function in training models like **Linear Regression**, **Ridge Regression**, and **Neural Networks**.

Numerical on multiple regression analysis

Problem

A company wants to predict **Sales (Y)** based on two independent variables:

- X_1 = Advertising expenditure (in ₹ lakhs)
- X_2 = Number of salespeople

The following data is collected:

Observation	X_1	X_2	Y (Sales)
1	1	2	2
2	2	1	3
3	3	4	6
4	4	3	7
5	5	5	11

We need to find the regression equation:

$$Y = b_0 + b_1 X_1 + b_2 X_2$$

STEP 1: Compute the required means

$$\bar{X}_1 = \frac{1+2+3+4+5}{5} = 3$$

$$\bar{X}_2 = \frac{2+1+4+3+5}{5} = 3$$

$$\bar{Y} = \frac{2+3+6+7+11}{5} = 5.8$$

STEP 2: Compute the deviations

Obs	X_1	X_2	Y	$(X_1-\bar{X}_1)$	$(X_2-\bar{X}_2)$	$(Y-\bar{Y})$
1	1	2	2	-2	-1	-3.8
2	2	1	3	-1	-2	-2.8
3	3	4	6	0	1	0.2
4	4	3	7	1	0	1.2
5	5	5	11	2	2	5.2

STEP 3: Compute the necessary sums

Quantity	Formula	Calculation	Value
S_{x1x1}	$\Sigma(X_1-\bar{X}_1)^2$	$(-2)^2 + (-1)^2 + 0^2 + 1^2 + 2^2$	10
S_{x2x2}	$\Sigma(X_2-\bar{X}_2)^2$	$(-1)^2 + (-2)^2 + 1^2 + 0^2 + 2^2$	10
S_{x1x2}	$\Sigma(X_1-\bar{X}_1)(X_2-\bar{X}_2)$	(-2)(-1) + (-1)(-2) + 0(1) + 1(0) + 2(2)	8
S_{x1y}	$\Sigma(X_1-\bar{X}_1)(Y-\bar{Y})$	(-2)(-3.8)+(-1)(-2.8)+0(0.2)+1(1.2)+2(5.2)	21.0
S_{x2y}	$\Sigma(X_2-\bar{X}_2)(Y-\bar{Y})$	(-1)(-3.8)+(-2)(-2.8)+1(0.2)+0(1.2)+2(5.2)	17.0

STEP 4: Formulas for b_1 and b_2

For two independent variables:

$$b_1 = \frac{S_{x2x2}S_{x1y} - S_{x1x2}S_{x2y}}{S_{x1x1}S_{x2x2} - S_{x1x2}^2}$$

$$b_2 = \frac{S_{x1x1}S_{x2y} - S_{x1x2}S_{x1y}}{S_{x1x1}S_{x2x2} - S_{x1x2}^2}$$

Compute the denominator:

$$D = S_{x1x1}S_{x2x2} - S_{x1x2}^2 = (10)(10) - (8)^2 = 100 - 64 = 36$$

Now compute numerators

$$\text{Numerator for } b_1 = (10)(21.0) - (8)(17.0) = 210 - 136 = 74$$

$$\text{Numerator for } b_2 = (10)(17.0) - (8)(21.0) = 170 - 168 = 2$$

Hence $b_1 = 74/36 = 2.056$, $b_2 = 2/36 = 0.056$

STEP 5: Compute the intercept b_0

$$b_0 = \bar{Y} - b_1\bar{X}_1 - b_2\bar{X}_2$$

Value of b_0 is = 5.8 – (2.056 *3) – (0.056 * 3) = = 5.8 – 6.336 = - 0.536

Final Regression Equation is Y = - 0.536 + 2.056 X_1 + 0.056 X_2

Interpretation

- When both X_1 and X_2 = 0, predicted sales = **– 0.536** (the intercept).
- For every unit increase in **advertising expenditure (X_1)**, sales increase by 2.056 **units** (holding X_2 constant).
- For every additional **salesperson (X_2)**, sales increase by **0.667 units** (holding X_1 constant).