Multiple Linear Regression – Complete Notes

◆ Introduction

Multiple Linear Regression (MLR) is an extension of Simple Linear Regression. It models the relationship between a dependent variable (Y) and two or more independent variables (X_1, X_2, ..., X_n).

It assumes a linear relationship between the predictors and the response variable.

◆ What is Multiple Linear Regression?

- A supervised learning regression technique.

- Target (output) variable Y is continuous.

- Used when we want to predict Y based on multiple features.

- For example:

  - Predicting house price based on area, location, number of bedrooms, age, etc.

  - Predicting salary based on education level, years of experience, skill set, etc.

◆ Mathematical Model (Equation)

$Y = b_0 + b_1X_1 + b_2X_2 + \cdots + b_nX_n + \varepsilon$

Where:

Y : Dependent variable (response)

$X_1 ... X_n$: Independent variables (features)

b_0 : Intercept (constant term)

$b_1 ... b_n$: Coefficients for each predictor

ε : Error term (residuals)

Each b_i represents the effect of one unit change in X_i, holding other variables constant.

◆ Objective

Estimate the best-fitting coefficients b_0, b_1, ..., b_n to minimize the prediction error.

Use Least Squares Method to minimize:

$MSE = (1/n) \times \sum(Y_i - \hat{Y}_i)^2$

◆ Assumptions of Multiple Linear Regression

- Linearity – Relationship between predictors and response is linear.

- Independence – Observations are independent of each other.

- Homoscedasticity – Constant variance of residuals (errors).

- No multicollinearity – Predictors are not highly correlated with each other.

- Normality – Residuals are normally distributed.

- **Matrix Representation**

$Y = X\beta + \varepsilon$

Y: $n \times 1$ vector of target values

X: $n \times (p+1)$ matrix with a column of 1s for intercept

β: $(p+1) \times 1$ vector of coefficients

ε: error vector

- **Solving for Coefficients – Closed-Form Solution**

$\hat{\beta} = (X^T X)^{-1} X^T Y$

- **Python Implementation**

```
# Step 1: Import Libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

# Step 2: Dataset

X = np.array([[1, 2], [2, 3], [4, 5], [3, 6], [5, 8]])

Y = np.array([10, 12, 20, 18, 26])

# Step 3: Train Model

model = LinearRegression()

model.fit(X, Y)

print("Intercept:", model.intercept_)

print("Coefficients:", model.coef_)

# Step 4: Predictions

Y_pred = model.predict(X)
```

```
print("Predictions:", Y_pred)
```

🔹 Interpreting Coefficients

b_1: change in Y for one unit increase in X_1, holding X_2 constant.

b_2: change in Y for one unit increase in X_2, holding X_1 constant.

🔹 Model Evaluation Metrics

- MSE = $(1/n) \times \sum(y_i - \hat{y}_i)^2$ → Mean squared error
- RMSE = $\sqrt{MSE}$ → Root mean squared error
- MAE = $(1/n) \times \sum|y_i - \hat{y}_i|$ → Mean absolute error
- R^2 = 1 - (SSR/SST) → Goodness of fit

🔹 Pros and Cons

Advantages:

- Easy to interpret.
- Efficient for small to medium-sized datasets.
- Works well when assumptions are met.

Limitations:

- Sensitive to multicollinearity.
- Can underperform on nonlinear data.
- Assumes linear relationships.

🔹 When to Use MLR

- You have multiple features affecting the target.
- The relationship is expected to be linear.
- You need a baseline model.

🔹 Visual Example

MLR with 2 features forms a plane in 3D; in higher dimensions, it becomes a hyperplane.

✅ Summary

- Model: $Y = b_0 + b_1X_1 + b_2X_2 + ... + b_nX_n + \varepsilon$
- Goal: Minimize MSE
- Technique: Least Squares (OLS)
- Assumptions: Linearity, Independence, Homoscedasticity, No Multicollinearity, Normality
- Evaluation: MSE, RMSE, MAE, R^2

- Python: sklearn.linear_model.LinearRegression