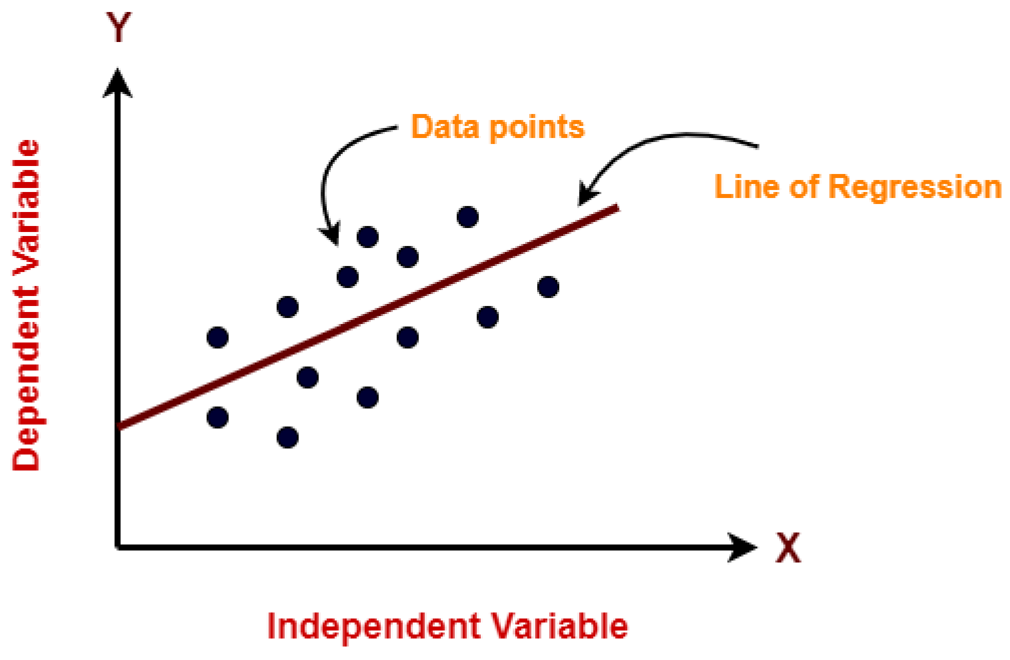
Linear Regression – Complete Notes

1. What is Linear Regression?

- Linear Regression is a **supervised machine learning algorithm** used for **regression tasks** (predicting continuous outcomes).
- The algorithm **learns from labeled data** (inputs XXX with known outputs YYY) to map inputs to output via a **linear function**.
- It assumes a **linear relationship** between input(s) and output: as input changes, output changes in a more-or-less proportional (linear) way.
- The form of that relationship is captured by a **straight line (in simple case)** or a hyperplane (in multiple dimensions).

Example:
You observe that as students study more hours, their scores rise. You can use their study hours (input) to build a linear model to predict future exam scores (output).

- Independent variable (input) = hours studied
- Dependent variable (output) = exam score



---

2. Why Linear Regression is Important

Linear Regression has many appealing properties and uses:

- **Simplicity & interpretability**: It's easy to understand, visualize, and explain.
- **Predictive power**: For tasks where the relationship is approximately linear, it gives good forecasts.
- **Foundation for other models**: Many advanced methods build on the linear model concept.
- **Efficiency**: Computations are cheaper (especially for small to medium dimensions).
- **Widely used in practice**: It's a go-to method in economics, business, health, social sciences — wherever modeling relationships is needed.
- **Insight into relationships**: The coefficients (slopes) tell how strongly each feature influences the target.

---

3. Best-Fit Line in Linear Regression

To make predictions, linear regression fits a line (or hyperplane) that best "fits" the data points.

Goal

- Find a line such that **predictions from the line are as close as possible** to the actual observed values.
- The closeness is measured via **errors (residuals)**, and the sum of errors (or sum of squared errors) is minimized.

Equation of the Best-Fit Line (Simple Case)

For one independent variable xxx:

y=mx+b

Y = mx + c = (0.5)*5.5 + 32.76 =

y = m x + by=mx+b

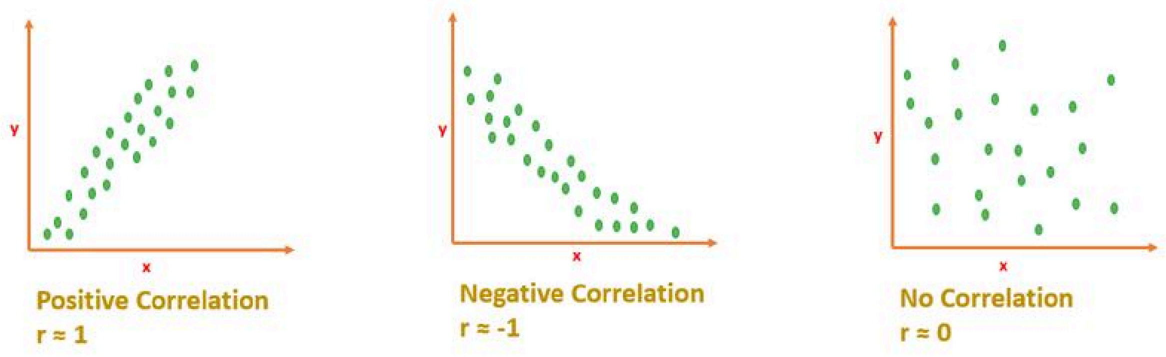
- yyy = predicted output

- xxx = input
- mmm = slope (gradient)
- bbb = intercept (value of yyy when x=0x = 0x=0)

In regression, we aim to choose mmm and bbb such that the line “fits” best.

Interpretation

- **Slope (m)**: how much yyy changes (on average) for a unit increase in xxx.
- **Intercept (b)**: the predicted value of yyy when x=0x = 0x=0 (though sometimes x=0x = 0x=0 may not make sense in context).



---

4. Minimizing the Error: Least Squares Method

To decide which mmm and bbb are “best,” we need a criterion. The most common one is the **Least Squares criterion**:

- Compute **residual** for each data point:
  residuali=yi−y^i\text{residual}_i = y_i - \hat{y}_iresiduali=yi−y^i
  (actual minus predicted)

- Form the **sum of squared residuals**:
  SSE=∑(yi−y^i)2SSE = \sum (y_i - \hat{y}_i)^2SSE=∑(yi−y^i)2
- We choose mmm and bbb to **minimize** that sum.

Why square residuals?

- To penalize large errors more heavily
- To make all errors non-negative (so negatives don't cancel positives)

Minimizing the SSE gives the **best-fit line** under assumptions of least squares.

---

5. Hypothesis Function

In machine learning terms, the model is called a **hypothesis**:

1. Simple Linear Regression

For one independent variable x:

h(x) = β0 + β1 * x

- β0 → Intercept (bias term)
- β1 → Coefficient (weight for the feature x)
- h(x) → Predicted value ŷ

---

2. Multiple Linear Regression

For multiple features x1, x2, ..., xk:

h(x1, x2, ..., xk) = β0 + β1 * x1 + β2 * x2 + ... + βk * xk

- β0 → Intercept (bias term)

- β1, ..., βk → Coefficients (weights for each feature)

- ŷ = h(x1, x2, ..., xk) → Predicted value

This hypothesis function is what our learning algorithm tunes (finds best coefficients) to approximate yyy.

---

6. Assumptions of Linear Regression

For the model to be statistically valid and reliable, linear regression relies on several assumptions:

1. **Linearity**
   - The relationship between inputs XXX and output YYY should be approximately linear.
2. **Independence of Errors**
   - The residuals (errors) should not be correlated with each other.
   - Particularly in time-series data, you must check for **autocorrelation**.
3. **Constant Variance (Homoscedasticity)**
   - The residuals should have roughly constant spread (variance) across values of XXX.
   - If the spread changes (fans out or narrows) — **heteroscedasticity** — it's problematic.
4. **Normality of Errors**
   - The residuals should be approximately normally distributed (bell curve).
   - Helps with inference (confidence intervals, hypothesis testing).
5. **No Multicollinearity**
   - In multiple regression: the independent variables should not be too strongly correlated with each other.

    - If two features are highly correlated, it's hard to separate their individual effects.

6. **No Autocorrelation (for time series)**
    - Errors should not show patterns (especially in sequential/time data).
7. **Additivity**
    - The effect of each feature on YYY is additive (unless you explicitly model interactions).

If assumptions are violated, the estimates may be biased, inefficient, or misleading.

---

7. Types of Linear Regression

- **Simple Linear Regression** (Univariate): one predictor xxx.
  y^=β0+β1x\hat{y} = \beta_0 + \beta_1 xy^=β0+β1x
- **Multiple Linear Regression** (Multivariate): multiple predictors x1,x2,…,xnx_1, x_2, \dots, x_nx1,x2,…,xn.
  y^=β0+β1x1+β2x2+⋯+βnxn\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_ny^=β0+β1x1+β2x2+⋯+βnxn

Use simple when only one feature matters; use multiple when you have more explanatory features.

---

8. Cost / Loss Function for Linear Regression

The cost (or loss) function quantifies how far off the model's predictions are from the actual values.

- The common choice: **Mean Squared Error (MSE)**
  J(θ)=1n∑i=1n(y^i−yi)2J(\theta) = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2J(θ)=n1i=1∑n(y^i−yi)2
  where nnn is number of data points.

- Some use a version with 12n\frac{1}{2n}2n1 or 12\frac{1}{2}21 factors for convenience in derivative calculation.

Minimizing the cost gives us the best parameters.

---

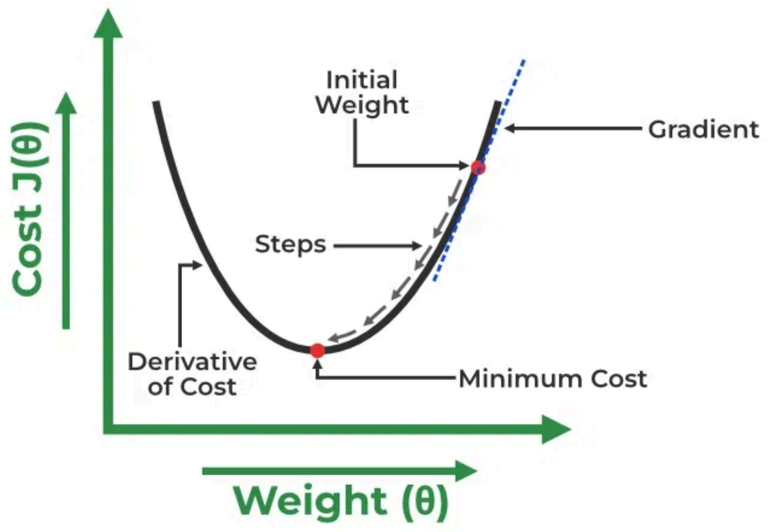
9. Gradient Descent for Linear Regression

Gradient Descent is an optimization algorithm to find the parameters that minimize the cost function.

How it works:

1. Initialize parameters (β0,β1,…\beta_0, \beta_1, \dotsβ0,β1,…) to some values (often random).
2. Compute predictions y^\hat{y}y^ using current parameters.
3. Compute the cost (MSE).
4. Compute partial derivatives (gradients) of cost w.r.t each parameter.
5. Update each parameter in the direction that reduces cost:

- α = learning rate (step size).
- Repeat steps 2–5 until convergence (cost no longer decreases meaningfully).



Gradient descent helps find the global minimum (for convex cost) for linear regression.

---

10. Evaluation Metrics for Linear Regression

How do we measure how good the model is? Several metrics are used:

1. Mean Squared Error (MSE)
Measures the average squared difference between actual and predicted values:

$MSE = (1 / n) \times \Sigma (y_i - \hat{y}_i)^2$

---

2. Mean Absolute Error (MAE)
Measures the average absolute difference between actual and predicted values:

$MAE = (1 / n) \times \Sigma |y_i - \hat{y}_i|$

---

3. Root Mean Squared Error (RMSE)
The square root of MSE, interpretable in the same units as the target variable:

$RMSE = \sqrt{MSE}$

---

4. Coefficient of Determination (R^2)
Indicates how much variance in the target is explained by the model:

$R^2 = 1 - (RSS / TSS)$

Where:

- $RSS = \Sigma (y_i - \hat{y}_i)^2 \rightarrow$ Residual Sum of Squares
- $TSS = \Sigma (y_i - \bar{y})^2 \rightarrow$ Total Sum of Squares
- R^2 ranges between 0 and 1; closer to 1 means better fit.

---

5. Adjusted R^2
Adjusts R^2 for the number of predictors to penalize adding irrelevant features:

Adjusted $R^2 = 1 - [(1 - R^2) \times (n - 1) / (n - k - 1)]$

Where:

- n = number of data points
- k = number of independent variables

- Helps prevent overestimation of model performance when adding unnecessary features.

These metrics help compare models and diagnose underfitting/overfitting.

---

11. 🧠 Regularization in Linear Regression

Regularization is a technique used to prevent overfitting in regression models by adding a penalty term to the cost function.
This penalty discourages the model from assigning excessively large weights to features.

When models overfit, they perform very well on training data but poorly on unseen (test) data.
Regularization introduces a bias–variance trade-off, improving generalization.

1. Lasso Regression (L1 Regularization)

Penalty Term:
$\lambda \times \Sigma |\theta_{□}|$

Lasso = abs(ypred - ytrue)

Modified Cost Function:
$J(\theta) = (1 / 2m) \times \Sigma (\hat{y}_i - y_i)^2 + \lambda \times \Sigma |\theta_{□}|$

Key Points:

- Encourages sparsity → Some coefficients become exactly zero.
- Performs feature selection by eliminating less important variables.
- Useful when you expect only a few features to have strong influence.

2. Ridge Regression (L2 Regularization)

Penalty Term:
$\lambda \times \Sigma\, \theta_j^2$

Modified Cost Function:
$J(\theta) = (1 / 2m) \times \Sigma\, (\hat{y}_i - y_i)^2 + \lambda \times \Sigma\, \theta_j^2$

Key Points:

- Penalizes large coefficients by shrinking them toward zero (but not exactly zero).
- Works well when all features contribute somewhat to the output.
- Reduces model variance but adds slight bias.

3. Elastic Net Regression (Combination of L1 and L2)

Penalty Term:
$\alpha\lambda \times \Sigma\, |\theta_j| + (1/2)(1 - \alpha)\lambda \times \Sigma\, \theta_j^2$

Modified Cost Function:
$J(\theta) = (1 / 2m) \times \Sigma\, (\hat{y}_i - y_i)^2 + \alpha\lambda \times \Sigma\, |\theta_j| + (1/2)(1 - \alpha)\lambda \times \Sigma\, \theta_j^2$

Key Points:

- Combines both Lasso and Ridge penalties.
- λ (Lambda): Controls overall regularization strength.
- α (Alpha): Controls the mix between L1 and L2 regularization:
  - $\alpha = 1 \rightarrow$ Pure Lasso
  - $\alpha = 0 \rightarrow$ Pure Ridge
  - $0 < \alpha < 1 \rightarrow$ Mix of both

Elastic Net works well when there are multiple correlated features, where Lasso alone might arbitrarily drop one.

12. Python Implementation Example (Sketch)

Here's a high-level sketch (based on reference) of how linear regression can be implemented in Python from scratch:

1. **Import libraries**
   `pandas`, `numpy`, `matplotlib`
2. **Load dataset**, split into training & testing sets.
3. **Define a class** `LinearRegression` with methods:
   - `forward_propagation`: compute predictions y^=mx+c\hat{y} = m x + cy^=mx+c
   - `cost_function`: compute MSE
   - `backward_propagation`: compute derivatives of cost wrt parameters
   - `update_parameters`: apply gradient descent updates
   - `train`: loop over many iterations, update parameters gradually, record loss
4. **Train** the model on the training data, plot the regression line and how it evolves.
5. **Use parameters** to predict on new/test data.

This implementation helps you deeply understand how linear regression mechanics work (beyond library calls).

---

13. Applications of Linear Regression

Linear regression is used across many fields. Some examples:

- **Finance & Economics**: Predicting stock prices, company performance metrics.
- **Real Estate**: Estimating property prices based on features (size, location, number of rooms).

- **Marketing / Business**: Forecasting sales given ad spend, seasonal factors, promotions.
- **Healthcare**: Predicting patient outcomes (e.g., length of hospital stay) from clinical variables.
- **Social Sciences / Psychology**: Modeling influence of factors on outcomes, e.g., income as a function of education, experience.

It helps both in predictive modeling and in **inferential analysis** (understanding which features matter and how).

---

14. Advantages & Disadvantages

Advantages

- Simple to understand, implement & interpret
- Coefficients give meaningful insights
- Efficient to train, especially for moderate-size data
- Serves as a good baseline for more complex models
- Works well if the true relationship is close to linear

Disadvantages / Limitations

- Assumes linearity — fails for non-linear relationships
- Sensitive to outliers — a few extreme points can skew the line
- Requires statistical assumptions (normality, independence, constant variance)
- Multicollinearity among predictors causes instability
- Can't capture complex patterns or interactions unless explicitly modeled
- Overfitting / underfitting risk if model is too flexible or too simple

---

Summary & Flow

1. **Problem**: Identify target YYY (continuous) and input features XXX.
2. **Hypothesis**: assume model y^=β0+∑βixi\hat{y} = \beta_0 + \sum \beta_i x_iy^=β0+∑βixi.
3. **Cost function**: choose MSE to quantify error.
4. **Optimization**: use gradient descent (or direct methods) to find parameters that minimize cost.
5. **Check assumptions**: linearity, independence, homoscedasticity, normality, multicollinearity.
6. **Evaluate model**: use MSE, MAE, RMSE, R2R^2R2, adjusted R2R^2R2.
7. **Regularize** if overfitting threatens: ridge, lasso, elastic net.
8. **Deploy / Use model**: make predictions for new data; interpret coefficients to derive insights.