Regression Analysis

Regression analysis is a powerful tool for uncovering the associations between variables observed in data, but cannot easily indicate causation. It is used in several contexts in business, finance, and economics. For instance, it is used to help investment managers value assets and understand the relationships between factors such as commodity prices and the stocks of businesses dealing in those commodities.

- A regression is a statistical technique that relates a dependent variable to one or more independent (explanatory) variables.
- A regression model is able to show whether changes observed in the dependent variable are associated with changes in one or more of the explanatory variables.
- It does this by essentially fitting a best-fit line and seeing how the data is dispersed around this line.
- Regression helps economists and financial analysts in things ranging from asset valuation to making predictions.
- In order for regression results to be properly interpreted, several assumptions about the data and the model itself must hold

Regression

Regression is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables)

$$Y_i = f(X_i, \beta) + e_i$$

Y_i = dependent variable
f = function
X_i = independent variable
β = unknown parameters
e_i = error terms

Coefficients are determined by the regression.

House Price= a + b×**Size**

Dependent Variable **Independent Variables**

HP = 30 + 10*(user size)

Coefficients are determined by the regression.

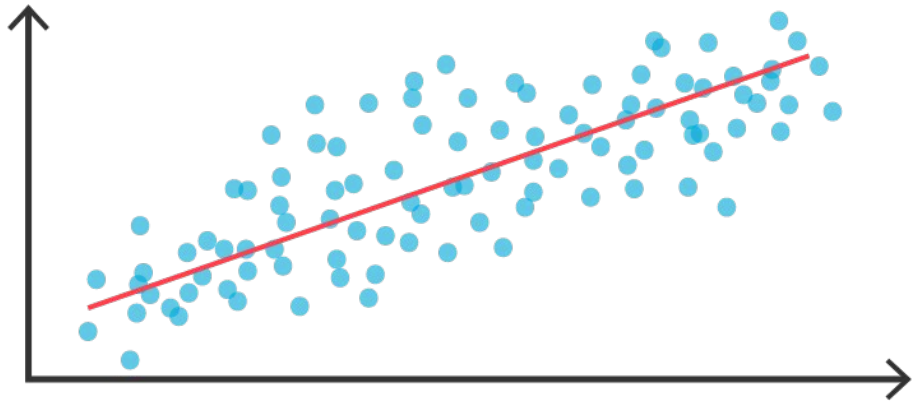
House Price= a + b×**Size** + c ×**No. Of room**

Dependent Variable **Independent Variables**

Regression types

Simple linear regression:

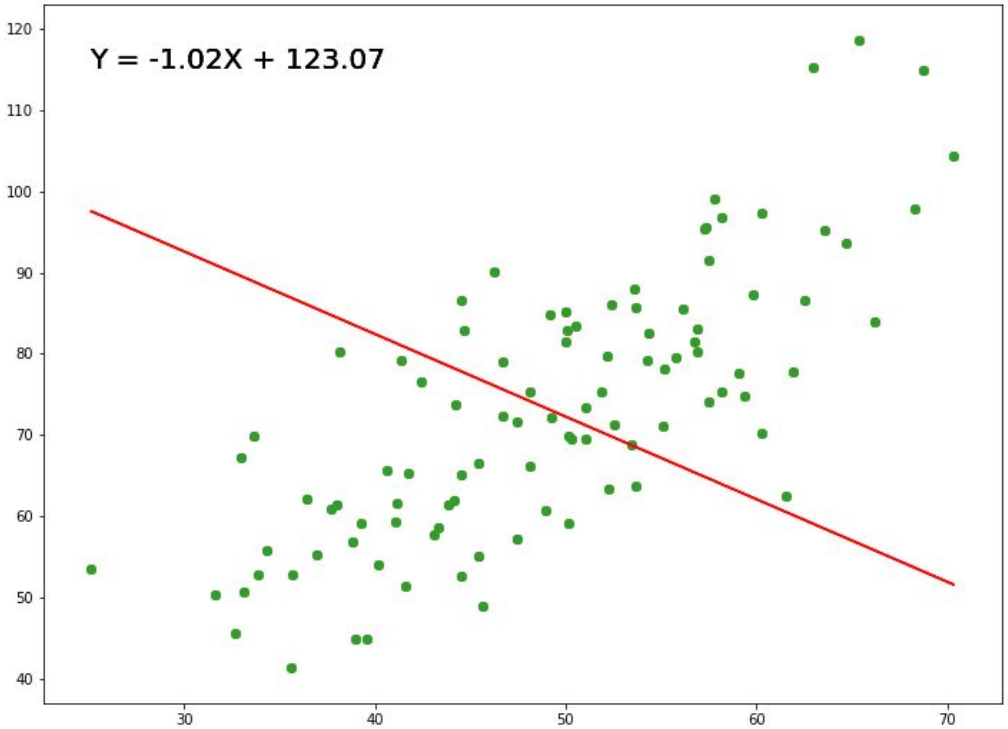
$$Y = a + bX + u$$



Multiple linear regression:

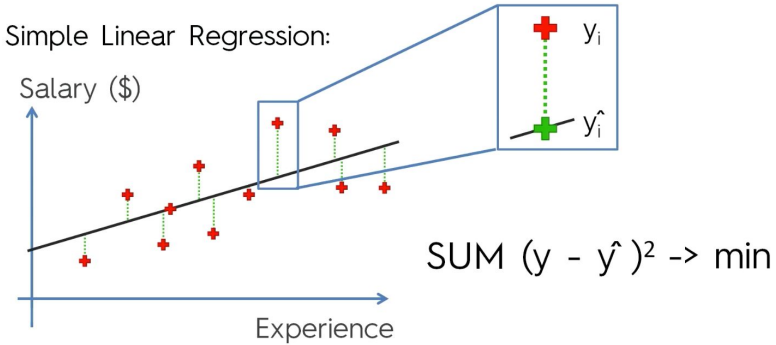
$$Y = a + b_1X_1 + b_2X_2 + b_3X_3 + ... + b_tX_t + u$$

Y = 1.3*X + 9
Y = 1.3*(30) + 9



Simple Linear Regression Idea (Ordinary Least Squares OLS)

- The idea of Simple Linear Regression is finding parameters Θ_0 and Θ_1 for which the error term is minimized(optimization problem).
- The model wi*ll minimize the sum of squared residuals* which is (cost/loss) function: we don't want our positive residuals to be cancelled by the negative ones, since they are equally penalizing for our model.
- This procedure is called Ordinary Least Squared error - OLS.



X1	Y1(target)	Y^
3	4	3.9

1. The Main Idea:

- We want to find the best-fit line $y = mx + b$ that minimizes the distance to all the data points.
- The goal is to determine the optimal values of m (slope) and b (intercept).

2. Defining the Error:

For each data point:

- The error is the difference between the actual value y_i and the predicted value $\hat{y}_i$ from the line.
- The error for one point: $\text{Error}_i = y_i - \hat{y}_i$.

3. Objective Function (Loss Function):

- To minimize the error, we square it to handle both positive and negative values and sum it up: $\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
- This is called the "Sum of Squared Errors"

4. Expanding the Equation:

Substitute $\hat{y}_i = mx_i + b$ into the equation: $\text{SSE} = \sum_{i=1}^{n} (y_i - (mx_i + b))^2$

6. Solving the System of Equations:

Solve these two equations simultaneously to find the optimal m and b.

Y = a0*x + a1

Slope m:

$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

Intercept b:

$$b = \bar{y} - m\bar{x}$$

Where:

- $\bar{x}$: Mean of x-values.
- $\bar{y}$: Mean of y-valu

7. Final Equation:

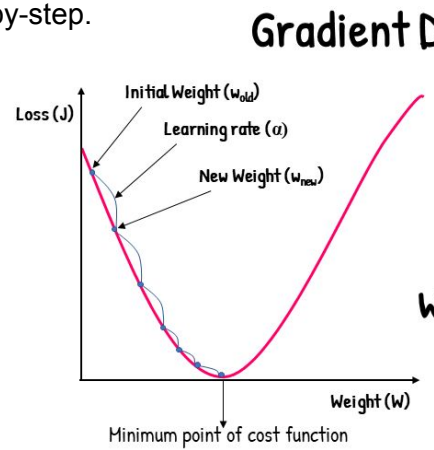
The final regression line is: $\hat{y} = mx + b$ Where m and b are computed to minimize the **Sum of Squared Errors (SSE)**.

$$y = bx + a$$

- x > القوة إلي بترمي فيها الكرة
- b > الميل يعني كل ما تزيد x شو بصير ب y
- a > من وين الخط هيبدأ (يعني من وين هيرمي الكرة اول مرة)
- y > القيمة الي بدك تتنبأ فيها يعني (المسافة عن السلة)

Gradient Descent in Linear Regression

Goal: Minimize the **cost/loss function** by adjusting model parameters step-by-step.



$$w_{new} = w_{old} - \alpha \frac{\delta J}{\delta w}$$

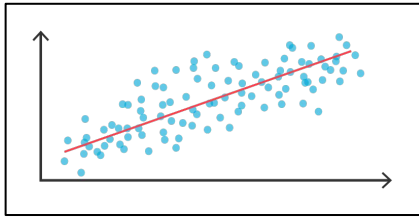
How it works:

1. **Initialize weights** with random values.
2. **Predict** outputs using the current weights.
3. **Calculate error** using a cost/loss function (e.g., MSE).
4. **Find gradient** → the slope of the error with respect to each weight.
5. **Update weights**: move in the opposite direction of the gradient **using the learning rate (α)**:
    - **Too big** → may overshoot the optimal point.
    - **Too small** → very slow convergence.
    - **Example:** walking in a valley at night → big steps might skip the bottom, small steps reach it slowly.
6. **Repeat** until the error stops decreasing or a set number of iterations is reached.

بدنا نعرف سعر الكيلو الي يخلي الزبون يشتري مني بناء على درجة الحرارة

Sales (Y target) > temp (X feature) (simple linear Regression)

بيحاول يرسم خط مستقيم بيقطع اكبر عدد من الداتا (line fit data) (يربط بين درجة الحرارة والعدد الكيلوات المباعة)



Y = X* slope + intercept

Y = X* α + β

(α,β = weight = Coefficients

Parameters إلي الخورازمية هتجيبهم

)

1. OLS – Ordinary Least Squares (بتجيب الأوزان مباشرة بمعادلات جبرية)
2. Gradient Descent (بتجيب الأوزان بالتدريج)

بدنا نعرف سعر الكيلو الي يخلي الزبون يشتري مني بناء على درجة الحرارة

Sales (Y target) > temp (X feature) (simple linear Regression)

بيحاول يرسم خط مستقيم بيقطع اكبر عدد من الداتا (line fit data) (يربط بين درجة الحرارة والعدد

Y = X* slope + intercept

Y = X* α + β

sales = temp* α + β

(α,β = weight = Coefficients

Parameters إلي الخورازمية هتجيبهم

)

OLS (لما يكون عدد ال rows قليل) (بطيئة) (دقتها عالية) (حساسة للقيم الشاذة)

Error = Y - Ypred

sum(y - ypred)^2 >>>>> cost function (min Error)

sum(y - (X* α + β))^2

Derivative w.r.t α to get α value

Derivative w.r.t β to get β value

- صار عندي معادلتين لكلا من β,α بتحسبهم مباشرة من كل قيم x ,y الحقيقين الي موجودين باعمدة الداتا عندي (استخدمنا كل قيم x,y يعني كل rows in csv تبعي)
- فبالتالي هيك عملنا minimize for cost function يعني قللنا الفرق بين القيم المتوقعة والقيم الحقيقية

1. OLS – Ordinary Least Squares (بتجيب الأوزان مباشرة بمعادلات جبرية)

بدنا نعرف سعر الكيلو الي يخلي الزبون يشتري مني بناء على درجة الحرارة

Sales (Y target) > temp (X feature) (simple linear Regression)

بيحاول يرسم خط مستقيم بيقطع اكبر عدد من الداتا (line fit data) (يربط بين درجة الحرارة والعدد الكيلوات المباعة)

5. Iterative Updates:

At each iteration:

1. Compute $\hat{y}_i = mx_i + b$.
2. Calculate the gradients:
   - $\frac{\partial J}{\partial m}$
   - $\frac{\partial J}{\partial b}$
3. Update m and b:

$$m = m - \alpha \frac{\partial J}{\partial m}$$

$$b = b - \alpha \frac{\partial J}{\partial b}$$

Gradient Descent (لما يكون عدد ال rows كبير) (اسرع) (دقتها اقل مقارنة ب ols
لانها ما بتشوف كل ال rows) (حساسة للقيم الشاذة)

Error = Y - Ypred

sum(y - ypred)^2 >>>>> cost function (min Error)

sum(y - (X* m + b))^2

1) بفترض قيم عشوائية ل b,a

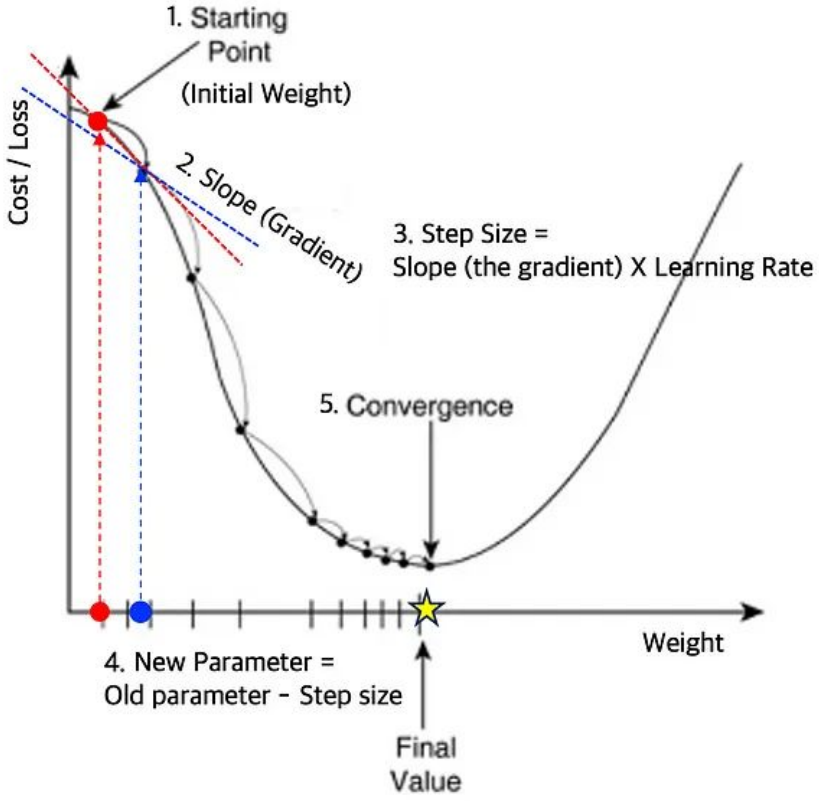
Derivative w.r.t m to get m value

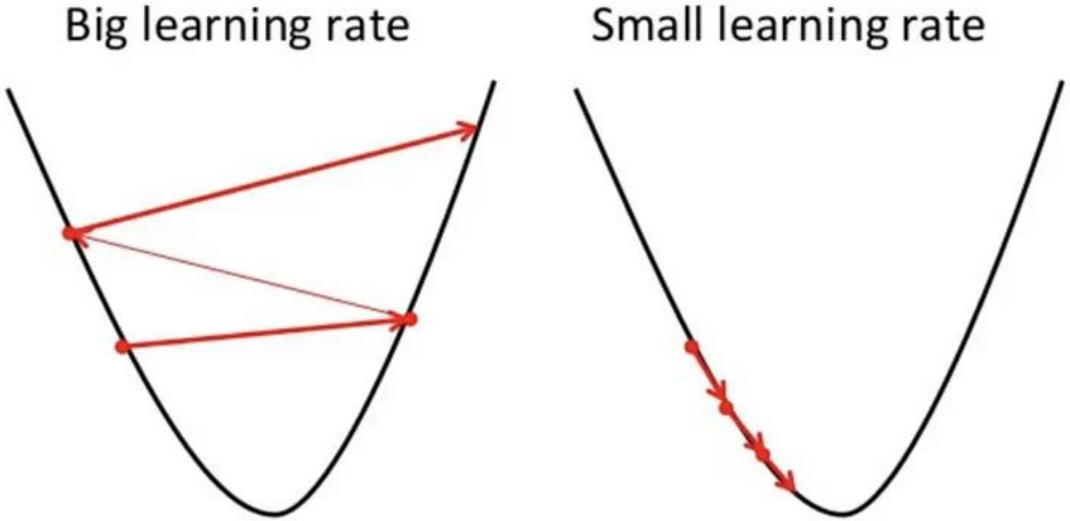
Derivative w.r.t b to get m value

2) α is a learning rate (step size)

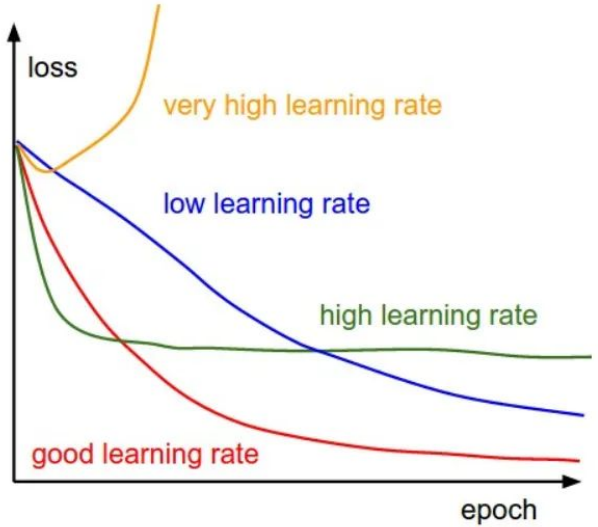
- صار عندي معادلتين لكلا من b,m بتحسبهم مباشرة من كل قيم x ,y الحقيقين الي موجودين باعمدة الداتا عندي (استخدمنا كل قيم x,y يعني كل rows in csv تبعي)
- فبالتالي هيك عملنا minimize for cost function يعني قللنا الفرق بين القيم المتوقعة والقيم الحقيقية

Gradient Descent



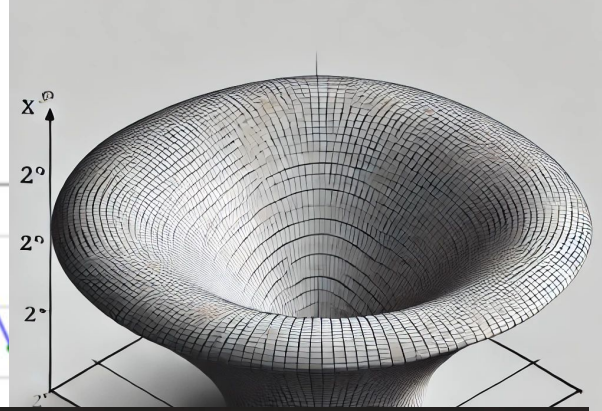


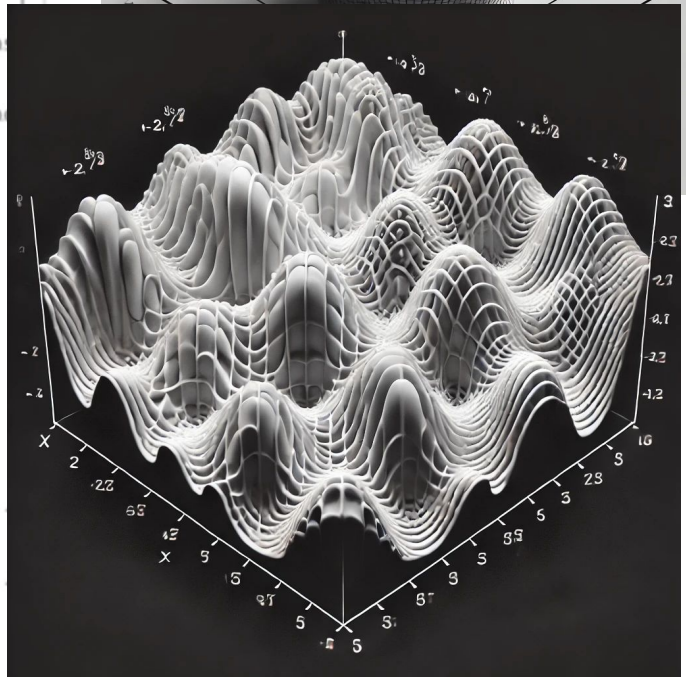
Gradient Descent



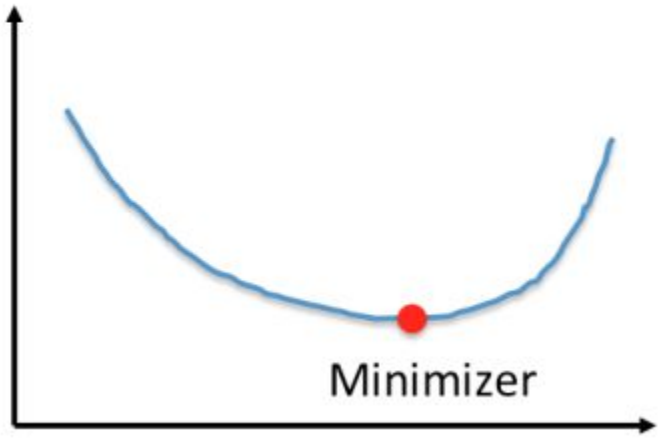
Linear Regression Idea (Gradient Descent)

- Another optimization algorithm used in linear Regression is Gradient Descent.
- It is important, easy to implement, and used in machine learning and deep learning to minimize a cost/loss function .
- Gradient descent algorithms works when error function is :
  - Differentiable: function has a derivative for each point.
  - Convex:
    - For a univariate function, the line segment connecting two function's points lays on or above its curve (it does not cross it).
    - Calculate the second derivative if its value is always bigger than 0 then function is convex.

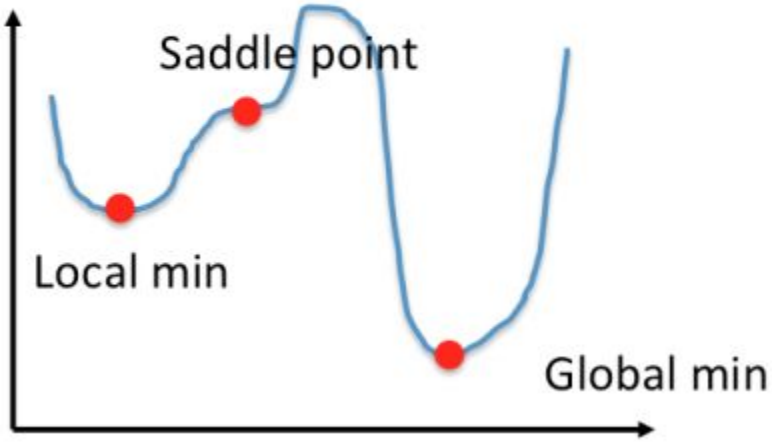




Convex



Non-Convex



Simple Linear Regression (Minimize cost using Gradient Descent)

1. The Main Idea:

- We aim to fit a line $y = mx + b$ that minimizes the error between predicted values $\hat{y}_i$ and actual values y_i.
- Instead of directly solving for m and b, we iteratively find them by minimizing the **cost function** using Gradient Descent.

---

2. Define the Cost Function:

The cost function for linear regression is the Mean Squared Error (MSE):

$$J(m, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where:

- y_i: Actual values.
- $\hat{y}_i = mx_i + b$: Predicted values.
- n: Total number of data points.

3. The Gradient Descent Algorithm:

Gradient Descent updates m and b iteratively to minimize $J(m, b)$.

Update Rules:

$$m = m - \alpha \frac{\partial J}{\partial m}$$

$$b = b - \alpha \frac{\partial J}{\partial b}$$

Where:

- α: Learning rate (controls step size).
- $\frac{\partial J}{\partial m}$: Partial derivative of J with respect to m.
- $\frac{\partial J}{\partial b}$: Partial derivative of J with respect to b.

4. Compute the Partial Derivatives:

Using the cost function $J(m, b)$, the gradients are:

Gradient w.r.t. m:

$$\frac{\partial J}{\partial m} = -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - \hat{y}_i)$$

Gradient w.r.t. b:

$$\frac{\partial J}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)$$

5. Iterative Updates:

At each iteration:

1. Compute $\hat{y}_i = mx_i + b$.
2. Calculate the gradients:
   - $\frac{\partial J}{\partial m}$
   - $\frac{\partial J}{\partial b}$
3. Update m and b:

$$m = m - \alpha\frac{\partial J}{\partial m}$$

$$b = b - \alpha\frac{\partial J}{\partial b}$$

6. Convergence:

- Repeat the updates until $J(m, b)$ converges to a minimum (or the changes become very small).
- This gives the optimal values of m and b that minimize the cost.

---

7. Final Result:

After convergence, the best-fit line is:

$$\hat{y} = mx + b$$

Where m and b are the values that minimize the cost function.

Practical: Linear Regression

- To apply linear regression on a data set we use LinearRegression API offered by sklearn.linear_model .
- Score help us evaluate how our model did (discussed later).
- Sklearn is:
  - Simple and efficient tools for predictive data analysis.
  - Accessible to everybody, and reusable in various contexts.
  - Built on NumPy, SciPy, and matplotlib.
  - Open source, commercially usable.

```python
from sklearn.linear_model import LinearRegression

# make model object
model = LinearRegression()

# train
model.fit(x_train, y_train)

# test
model.predict(x_test)

# calculate R2 score on training data
model.score(x_train, y_train)

# calculate R2 score on testing data
model.score(x_test, y_test)

# get model parameters
model.coef_
model.intercept_
```