Machine Learning
ZG565

BITS Pilani
Pilani Campus

Prof. Pankaj Agarwal

innovate achieve lead

BITS Pilani

Pilani Campus

Lecture No. – 3 | Linear Regression

Disclaimer and Acknowledgement

- The content for these slides has been obtained from books and various other source on the Internet
- We here by acknowledge all the contributors for their material and inputs.
- We have provided source information wherever necessary
- Students are requested to refer to the textbook w.r.t detailed content of the presentation deck shared over canvas
- We have reduced the slides from canvas and modified the content flow to suit the requirements of the course and for ease of class presentation

Source: "Probabilistic Machine Learning, An Introduction", Kevin P. Murphy, Slides of Prof.Sugata, Prof. Chetana from BITS Pilani, Prof. Raja vadhana from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

Course Plan

M1	Introduction
M2	Machine learning Workflow
M3	Linear Models for Regression
M4	Linear Models for Classification
M5	Decision Tree
M6	Instance Based Learning
M7	Support Vector Machine
M8	Bayesian Learning
M9	Ensemble Learning
M10	Unsupervised Learning
M11	Machine Learning Model Evaluation/Comparison

Agenda

- Introduction to Regression
- Linear Model for Regression
- Direct solution vs Iterative Method
- Gradient Descent
- Linear Basis Function
- Notion of Bias vs Variance

Regression Analysis

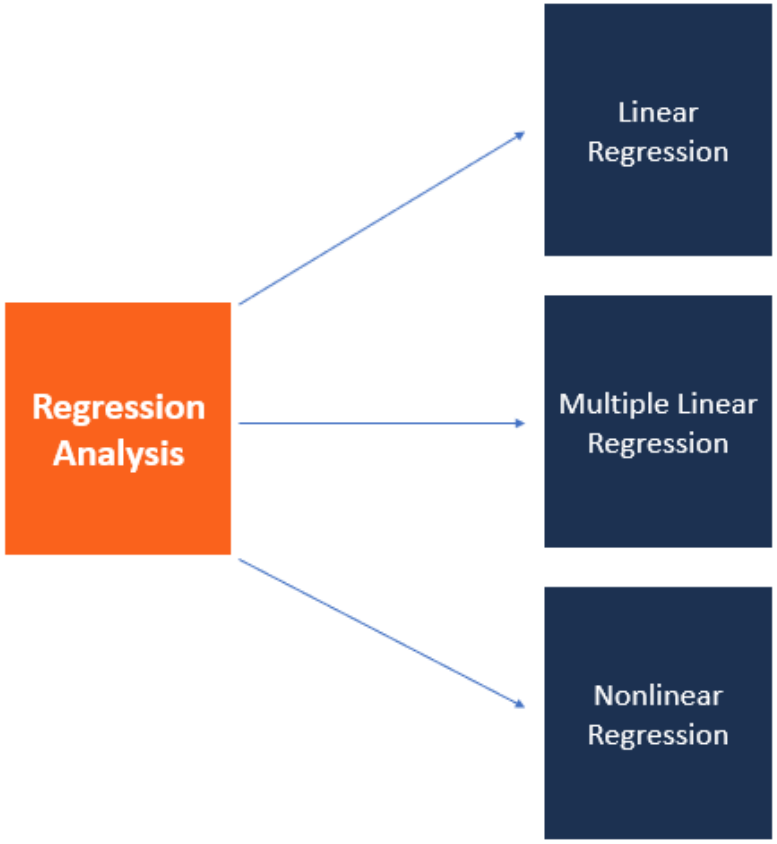
- Regression analysis is used to find trends in data.
- Typically, you need regression to answer whether and how some phenomenon influences the other or how several variables are related. For example, you can use it to determine *if* and *to what extent* the experience or gender impact salaries.
- For example, you might guess that there's a connection between how much you eat and how much you weigh; regression analysis can help you quantify that.
- For example, if you've been putting on weight over the last few years, it can predict how much you'll weigh in ten years time if you continue to put on weight at the same rate
- Essentially, regression is the **"best guess"** at using a set of data to make some kind of prediction.
- It's fitting a set of points to a graph.

Regression VS classification

- Regression and classification are categorized under the same umbrella of supervised machine learning.
- Both share the same concept of utilizing known datasets (referred to as training datasets) to make predictions.
- The main difference between them is that the output variable in regression is numerical (or continuous) while that for classification is categorical (or discrete).
- Examples of the common regression algorithms include **linear regression, Support Vector Regression (SVR), and regression trees.**
- Some algorithms, such as logistic regression, have the name "regression" in their names but they are not regression algorithms.

Regression techniques

- Regression is useful when you want **to forecast a response** using a new set of predictors
- Regression techniques **predict continuous responses**—for example, changes in temperature or fluctuations in electricity demand.
- Applications include forecasting stock prices, handwriting recognition, and acoustic signal processing.
- Common Regression techniques include **Linear, non linear, Gaussian Process Regression Model, SVM Regression , Generalized Linear Model, Regression Tree**



Regression

- Wish to learn a function f :: X → Y, where predicted output Y is real, given the n real training instances $\{<x^1,y^1>...<x^n,y^n>\}$.

- Examples include
  - predict weight from gender, height, age, …
  - Predict house price from locality, area, income, …
  - predict Google stock price today from Google, Yahoo, MSFT prices yesterday
  - predict each pixel intensity in robot's current camera image, from previous image and previous action

Inductive Learning Hypothesis : Interpretation

- Target Concept
- Discrete : f(x) ∈ {Yes, No, Maybe} Classification
- Continuous : f(x) ∈ [20-100] Regression
- Probability Estimation : f(x) ∈ [0-1]

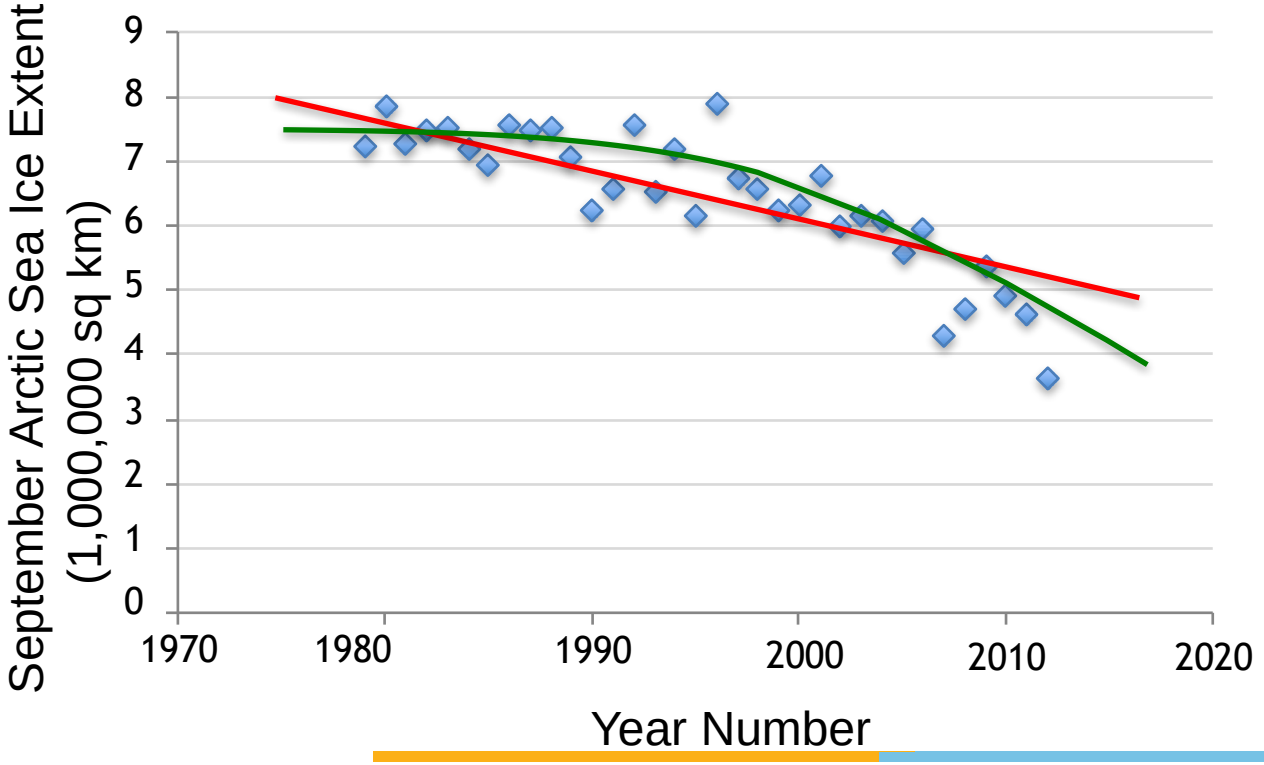
Sky	AirTemp	Altitude	Wind	Water	Forecast	Humidity
Sunny	Warm	Normal	Strong	Warm	Same	60
Sunny	Warm	High	Strong	Warm	Same	75
Rainy	Cold	High	Strong	Warm	Change	70
Sunny	Warm	High	Strong	Cool	Change	45

Formal Representation: Interpretation

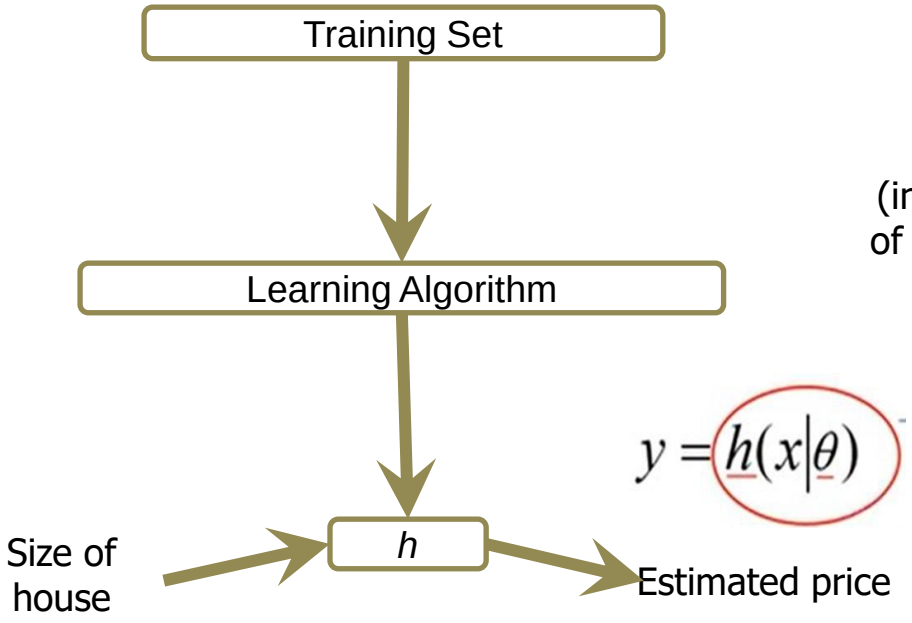
Supervised Learning: Regression

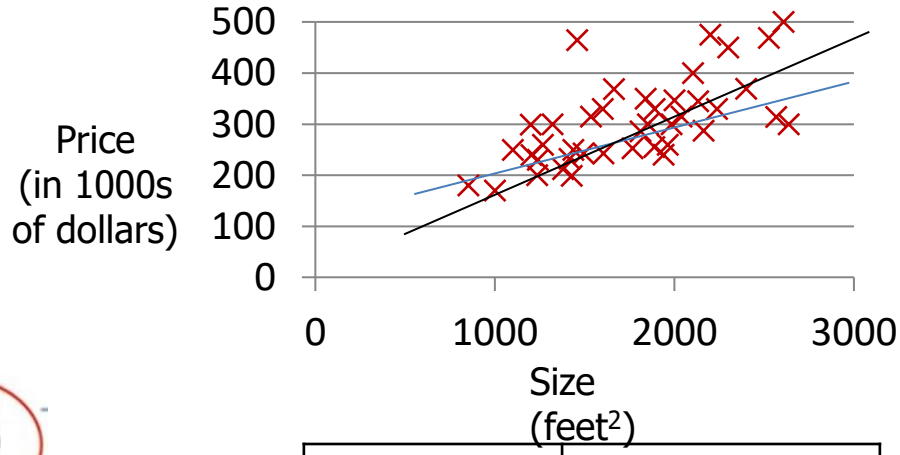
GOAL : Previously unseen records should be assigned a value as accurately as possible.

- Given (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)
- Learn a function $f(x)$ to predict y given x
  - y is real-valued



Machine Learning Process : Interpretation





Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's : Parameters

h(.) : Model

θ_i

How to choose 's ?

Types of Regression Models

(Education) x ⟶ y (Income)

multiple regression model

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

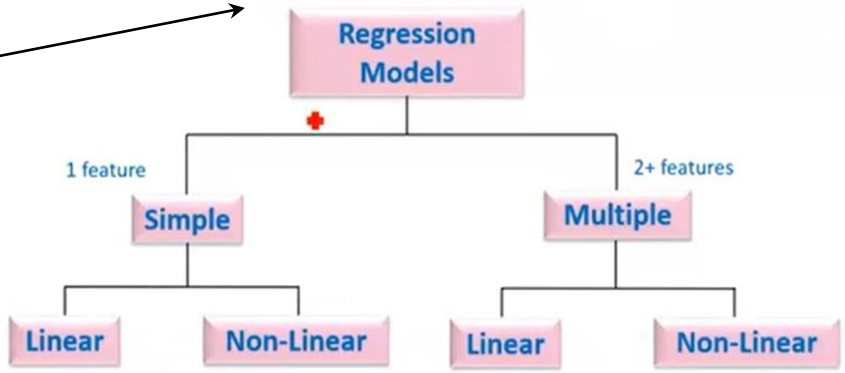
(Education) x_1

(Soft Skills) x_2

(Experience) x_3

(Age) x_4

y (Income)



- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$$

Assume $x_0 = 1$

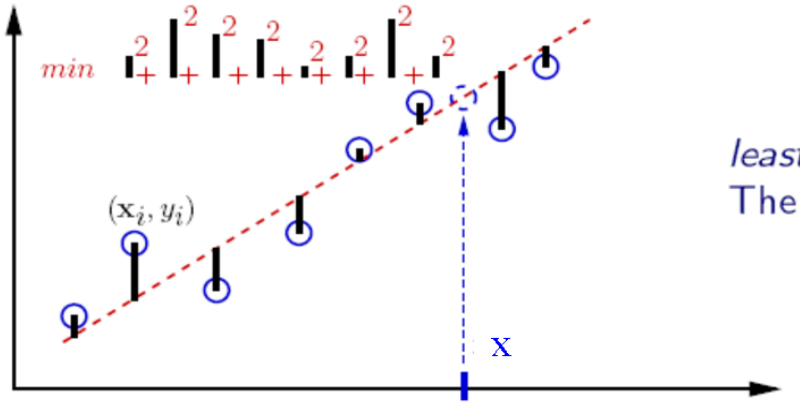
Linear Regression

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j = h_{\boldsymbol{\theta}}(x)$$

Assume $x_0 = 1$

- Fit model by minimizing sum of squared errors



least squares (LSQ)
The fitted line is used as a predictor

Figures are courtesy of Greg Shakhnarovich

Regression Analysis – Simple linear regression

- The simple linear model is expressed using the following equation: $\mathbf{Y = a + bX + \epsilon}$

Where:

- **Y** – Dependent variable
- **X** – Independent (explanatory) variable
- **a** – Intercept
- **b** – Slope
- **ϵ** – Residual (error)

Simple Linear Regression

If you know something about X and this knowledge helps you to predict about y

In this equation: y=aX+b

Y – Dependent Variable

a – Slope

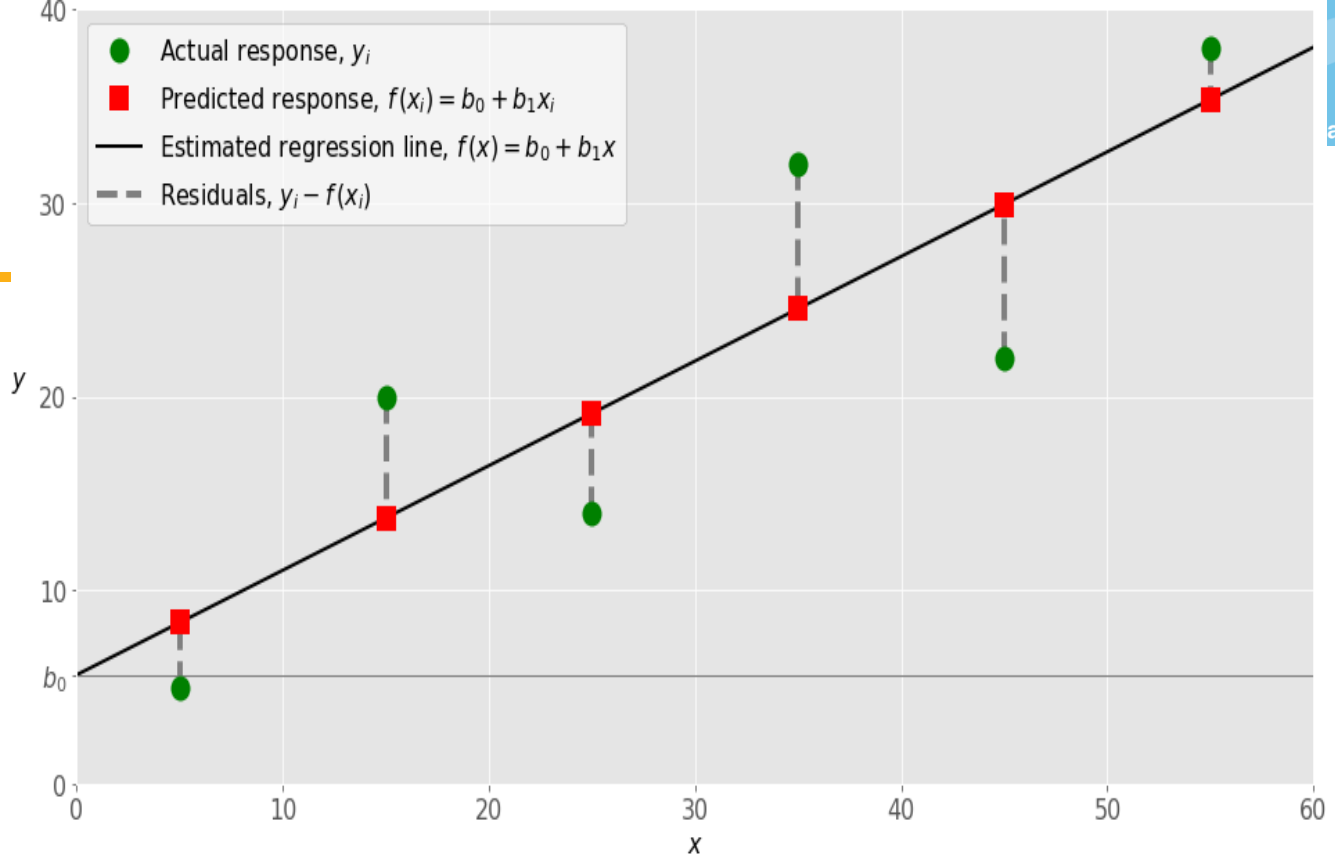
X – Independent variable

b – Intercept

These coefficients **a** and **b** are derived based on **minimizing the sum of squared difference of distance between data points and regression line.**

Look at the example. Here we have identified the best fit line having linear equation **y=0.2811x+13.9**. Now using this equation, we can find the weight, knowing the height of a person.

Slope of 2 means every change of 2 units in X will yield 2 units change in Y



The estimated regression function (black line) has the equation $f(x) = b_0 + b_1x$. Your goal is to calculate the optimal values of the predicted weights b_0 and b_1 that minimize SSR and determine the estimated regression function. The value of b_0, also called the intercept, shows the point where the estimated regression line crosses the y axis. It is the value of the estimated response $f(x)$ for x = 0. The value of b_1 determines the slope of the estimated regression line.

Least Squares Approach

Let's consider a simple linear regression model with a dependent variable y and an independent variable x. The linear regression equation is given by:

$$y = mx + b$$

where:

- y is the dependent variable,
- x is the independent variable,
- m is the slope of the line,
- b is the y-intercept.

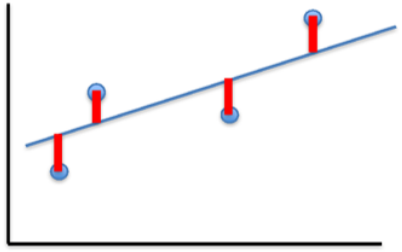
Given a set of data points (x_i, y_i), the objective is to find the values of m and b that minimize the sum of squared differences between the observed y values and the predicted y values based on the linear equation. Mathematically, the sum of squared differences, often denoted as $J(m, b)$, is expressed as:

$$J(m, b) = \sum_{i=1}^{n} (y_i - (mx_i + b))^2$$

Least Squares Linear Regression

Objective: minimize the sum of the squared differences between the observed and predicted values.

it seeks to find the line that minimizes the sum of the squared residuals (the vertical distances between each observed data point and the corresponding point on the regression line).



The goal is to minimize this cost function J with respect to the parameters m and b. The least squares approach finds the values of m and b that make the partial derivatives of J with respect to m and b equal to zero:

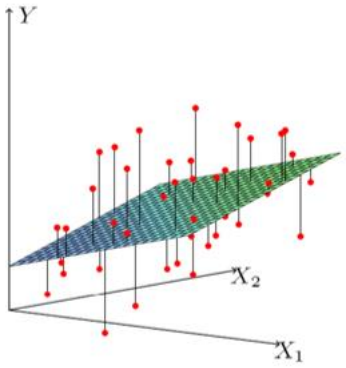
$$\frac{\partial J}{\partial m} = 0$$

$$\frac{\partial J}{\partial b} = 0$$

Solving these equations simultaneously provides the optimal values for m and b. The closed-form solutions for m and b in the least squares approach are given by:

$$m = \frac{n(\sum_{i-1}^{n} x_i y_i) - (\sum_{i-1}^{n} x_i)(\sum_{i-1}^{n} y_i)}{n(\sum_{i-1}^{n} x_i^2) - (\sum_{i-1}^{n} x_i)^2}$$

$$b = \frac{\sum_{i-1}^{n} y_i - m(\sum_{i-1}^{n} x_i)}{n}$$



These formulas provide the slope (m) and y-intercept (b) of the best-fitting line that minimizes the sum of squared differences between the observed and predicted values, making it a widely used technique in linear regression analysis.

Simple Linear Regression Example

	Student	x_i	y_i	$(x_i - x)$	$(y_i - y)$	$(x_i - x)^2$	$(y_i - y)^2$	$(x_i - x)(y_i - y)$
	1	95	85	17	8	289	64	136
	2	85	95	7	18	49	324	126
	3	80	70	2	-7	4	49	-14
	4	70	65	-8	-12	64	144	96
	5	60	70	-18	-7	324	49	126
Sum		390	385			730	630	470
Mean		78	77					

The regression equation is a linear equation of the form: $\hat{y} = b_0 + b_1 x$

To conduct a regression analysis, we need to solve for b_0 and b_1. Computations are shown below.

$$b_1 = \Sigma [(x_i - \bar{x})(y_i - \bar{y})] / \Sigma [(x_i - \bar{x})^2]$$

$$b_1 = 470/730 = 0.644$$

$$b_0 = \bar{y} - b_1 * \bar{x}$$

$$b_0 = 77 - (0.644)(78) = 26.768$$

Therefore, the regression equation is: $\hat{y} = 26.768 + 0.644x$.

Example

- Consider the following set of points: {(-2 , -1) , (1 , 1) , (3 , 2)}
  a) Determine the value of slope & intercept and build the regression equation line formula for the given data points.
  b) Plot the given points and the regression line in the same rectangular system of axes.

x	y	x y	x^2
-2	-1	2	4
1	1	1	1
3	2	6	9
$\Sigma x = 2$	$\Sigma y = 2$	$\Sigma xy = 9$	$\Sigma x^2 = 14$

- We now use the above formula to calculate a and b as follows

  $a = (n\Sigma x\, y - \Sigma x \Sigma y) / (n\Sigma x^2 - (\Sigma x)^2)$

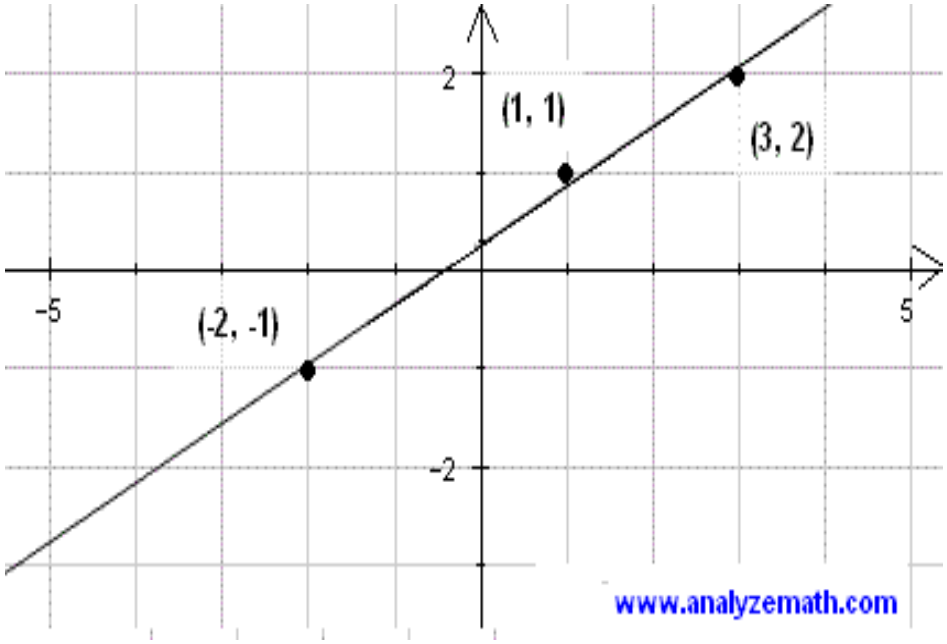
  $= (3*9 - 2*2) / (3*14 - 2^2) = 23/38 = 0.60$

  $b = (1/n)(\Sigma y - a\, \Sigma x) = (1/3)(2 - (23/38)*2)$

  $= 5/19 = 0.26$

b) We now graph the regression line given by $y = a\, x + b$

$y = 0.60 + 0.26x$



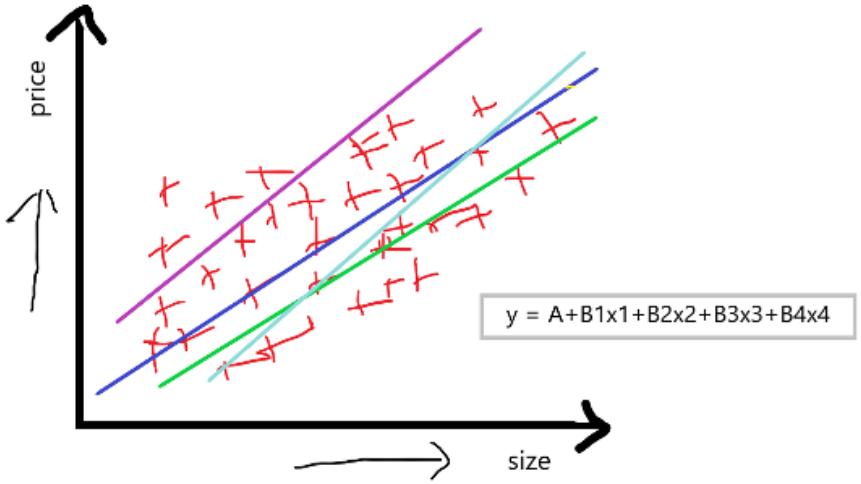
Regression Analysis – Multiple linear regression

If we have one dependent feature and multiple independent features then basically call it a multiple linear regression

Multiple Linear Regression (MLR) is basically indicating that we will have many features Such as f1, f2, f3, f4, and our output feature f5. If we take the same example as above we discussed, suppose:

- f1 is the size of the house,
- f2 is bad rooms in the house,
- f3 is the locality of the house,
- f4 is the condition of the house, and
- f5 is our output feature, which is the price of the house.

- equation: y = A+B1x1+B2x2+B3x3+B4x4



our aim in using the multiple linear regression is that we have to compute A, which is an intercept. The key parameters B1, B2, B3, and B4 are the slopes or coefficients concerning this independent feature. This basically indicates that if we increase the value of x1 by 1 unit, then B1 will tell you how much it will affect the price of the house. The others B2, B3, and B4, also work similarly.

Regression Analysis – Linear model assumptions

Linear regression analysis is based on six fundamental assumptions:

The dependent and independent variables show a linear relationship between the slope and the intercept.

- change in the dependent variable is proportional to the change in the independent variable(s).
- If you are analyzing the relationship between hours of study (x) and exam scores (y), linearity would mean that an increase in study hours is associated with a consistent increase (or decrease) in exam scores.

The independent variable is not random.

- independent variable is not influenced by any other variable and is not random. It is treated as fixed and not subject to random variations.
- **Example**: In a study predicting income (y) based on years of education (x), assuming education is not influenced by other random factors during the study period.

The value of the residual (error) is zero: Zero Mean of Residuals

- The residuals (errors) are the differences between the observed and predicted values. The assumption is that the average of these residuals is zero.
- Example: If the predicted values tend to be too high for some observations and too low for others, the residuals should balance out such that their overall average is zero.

Regression Analysis – Linear model assumptions

Linear regression analysis is based on six fundamental assumptions:

Homoscedasticity (Constant Variance of Residuals):

- The variability of the residuals should be constant across all levels of the independent variable(s). In other words, the spread of the residuals should be roughly the same for all values of x.
- Example: If you are predicting house prices (y) based on the size of the house (x), homoscedasticity would mean that the variability in the errors is constant for all sizes of houses.

No Autocorrelation of Residuals:

- There should be no systematic pattern in the residuals. The value of the residuals for one observation should not be correlated with the value of the residuals for another observation.
- Example: If you are predicting stock prices (y) based on historical data (x), no autocorrelation means that the error in predicting today's stock price is not systematically related to the error in predicting yesterday's stock price.

Normality of Residuals:

- The assumption is that the residuals follow a normal distribution. This is not a critical assumption for large sample sizes due to the Central Limit Theorem, but it can be important for smaller samples.
- **Example:** If the residuals are normally distributed, it implies that most of the errors are small, and extreme errors are rare.

Central Limit Theorem : It states that, regardless of the distribution of the population, the distribution of the sum (or average) of a large number of independent, identically distributed random variables approaches a normal (Gaussian) distribution.

Learning the Model Parameters

Closed Form Solution Approach

Gradient Descent Approach

How to choose θ_i's ?

Closed Form Solution Approach

- The closed-form solution for learning the model parameters in linear regression involves finding an analytical expression that directly computes the optimal values for the parameters without the need for an iterative optimization algorithm.
- This closed-form solution is obtained by setting the gradient of the cost function with respect to the parameters to zero and solving the resulting system of equations.
- For simple linear regression (with one independent variable), the closed-form solution involves the following equations:

Simple Linear Regression:

For a simple linear regression with one independent variable, the linear equation is $y = \theta_0 + \theta_1 x$. The mean squared error (MSE) cost function is given by:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

where $h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$.

The closed-form solutions for the optimal parameters θ_0 and θ_1 are obtained by setting the partial derivatives of the cost function with respect to θ_0 and θ_1 to zero and solving the resulting system of equations.

Closed Form Solution Approach

Step 1: Define the Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$

Step 2: Partial Derivatives:

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})$$
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \cdot x^{(i)}$$

Step 3: Set Partial Derivatives to Zero:

$$\sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) = 0$$
$$\sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \cdot x^{(i)} = 0$$

Step 4: Solve for Parameters:

Solving the above equations simultaneously provides the closed-form solutions for θ_0 and θ_1:

$$\theta_1 = \frac{n(\sum_{i-1}^{n} x^{(i)} y^{(i)}) - (\sum_{i-1}^{n} x^{(i)})(\sum_{i-1}^{n} y^{(i)})}{n(\sum_{i-1}^{n} (x^{(i)})^2) - (\sum_{i-1}^{n} x^{(i)})^2}$$

$$\theta_0 = \frac{\sum_{i-1}^{n} y^{(i)} - \theta_1 (\sum_{i-1}^{n} x^{(i)})}{n}$$

Objective in Multiple Linear Regression:

In multiple linear regression, we have n independent variables: $x_1, x_2, \ldots, x_n$. The linear regression equation is given by:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n + \varepsilon$$

- y is the dependent variable,
- θ_0 is the intercept,
- $\theta_1, \theta_2, \ldots, \theta_n$ are the coefficients,
- $x_1, x_2, \ldots, x_n$ are the independent variables,
- ε is the error term.

Matrix Representation:

We can represent the independent variables as a matrix X and the coefficients as a vector θ:

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \ldots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \ldots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \ldots & x_n^{(m)} \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

The linear regression equation can then be represented in matrix form as:

$$Y = X\theta + \varepsilon$$

Y is the vector of observed values.

Cost Function:

The cost function for multiple linear regression using the mean squared error (MSE) is:

$$J(\theta) = \frac{1}{2m}(Y - X\theta)^T(Y - X\theta)$$

Closed-Form Solution:

To find the closed-form solution, we need to minimize the cost function by taking its derivative with respect to θ and setting it to zero:

$$\frac{\partial J(\theta)}{\partial \theta} = 0$$

Solving this equation gives the optimal values for θ. The closed-form solution is expressed as:

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

note :

- while the closed-form solution is elegant and computationally efficient, it may not always be applicable or efficient for very large datasets, and numerical instability can occur if the matrix *X'X* is not invertible.
- In such cases, iterative optimization algorithms like gradient descent are commonly used. However, for many standard linear regression problems, the closed-form solution is both accurate and efficient.

Closed Form Solution Approach

Suppose we have the following data:

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 5 \\ 1 & 5 & 6 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \qquad Y = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

1. Calculate $X^T X$:

$$X^T X = \begin{bmatrix} 4 & 14 & 18 \\ 14 & 54 & 68 \\ 18 & 68 & 86 \end{bmatrix}$$

1. Calculate $X^T Y$:

$$X^T Y = \begin{bmatrix} 10 \\ 38 \\ 48 \end{bmatrix}$$

1. Calculate $(X^T X)^{-1}$:

$$(X^T X)^{-1} = \begin{bmatrix} 1.457 & -0.514 & 0.343 \\ -0.514 & 0.229 & -0.171 \\ 0.343 & -0.171 & 0.114 \end{bmatrix}$$

1. Finally, calculate θ:

$$\theta = (X^T X)^{-1} X^T Y = \begin{bmatrix} 0.5 \\ 0.6 \\ 0.7 \end{bmatrix}$$

So, the closed-form solution gives us the optimal values for θ_0, θ_1, and θ_2 as 0.5, 0.6, and 0.7, respectively. The linear regression equation for this example is:

$$y = 0.5 + 0.6x_1 + 0.7x_2$$

These values minimize the mean squared error, providing the best-fitting plane through the given data points in a least squares sense.

Why Derivative of Cost function is desired?

- The derivative of a function is a key concept in calculus that measures how the function changes with respect to its input.

- The derivative of a function is crucial when minimizing cost functions because it provides information about the rate of change of the function with respect to its parameters.

- In the context of machine learning and optimization, this information is used to find the minimum of a cost function efficiently

Cost Function:

Consider a cost function $J(\theta)$ that depends on parameters θ (e.g., coefficients in linear regression). The goal is to find the values of θ that minimize the cost function, representing the best-fit model.

Derivative (Gradient): $\frac{\partial J}{\partial \theta}$

The derivative of J with respect to θ, denoted as $\frac{\partial J}{\partial \theta}$ or $J'(\theta)$, represents the rate of change of the cost function with respect to changes in θ. In other words, it tells us how much the cost would increase or decrease if we make a small change in θ.

Why Derivative of Cost function is desired?

Gradient Descent:

Gradient descent is an iterative optimization algorithm used to minimize the cost function. The update rule for each iteration is:

$$\theta := \theta - \alpha \frac{\partial J}{\partial \theta}$$

- α is the learning rate, determining the size of the steps taken,
- $\frac{\partial J}{\partial \theta}$ is the derivative of the cost function.

Notion of Maxima and Minima of a function

To find maxima and minima, one typically looks for critical points, where the derivative is zero or undefined. These critical points may correspond to extrema, but further analysis is needed to determine whether they are local or global.

1. **Critical Points:**
   - A point $x = c$ is a critical point if $f'(c) = 0$ or $f'(c)$ is undefined.
2. **Second Derivative Test:**
   - For a critical point c, if $f''(c) > 0$, then c is a local minimum.
   - If $f''(c) < 0$, then c is a local maximum.
   - If $f''(c) = 0$, the test is inconclusive.

Consider the function $f(x) = x^2 - 4x + 4$.

Calculating Derivatives:

1. **First Derivative:**
   $f'(x) = 4x - 8$
2. **Critical Points:**
   Solve $4x - 8 = 0$ for x:
   $x = 2$
3. **Second Derivative:**
   $f''(x) = 4$

Analysis of Critical Point:

1. **First Derivative Test:**
   - For $x < 2$, $f'(x) < 0$, indicating a decreasing slope.
   - For $x > 2$, $f'(x) > 0$, indicating an increasing slope.
   - Therefore, $x = 2$ is a local minimum.
2. **Second Derivative Test:**
   - Since $f''(2) > 0$, the critical point $x = 2$ is indeed a local minimum.

Global Minimum:

The function $f(x) = x^2 - 4x + 4$ is a quadratic with a minimum at $x = 2$. Since the parabola opens upwards, this is also the global minimum.

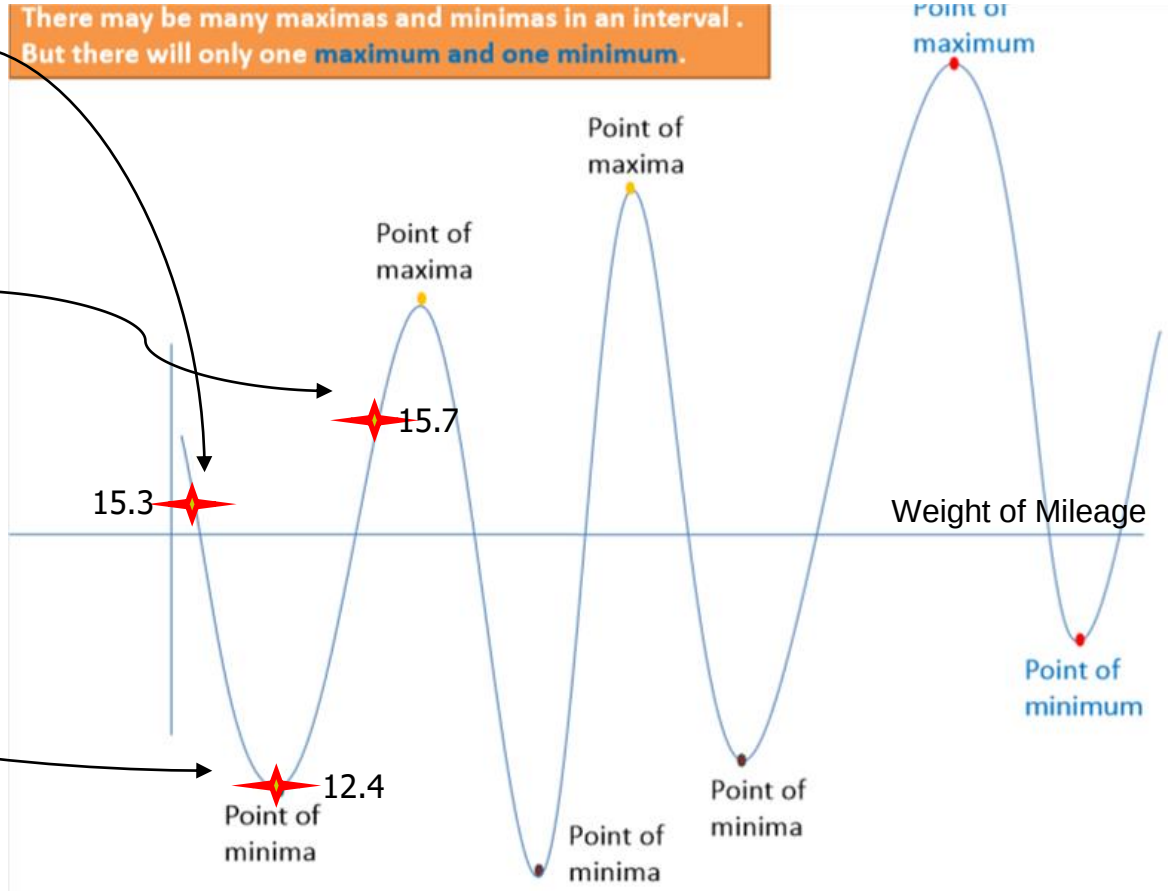
Notion of Maxima and Minima of a function

In Machine Learning

CarPrice = 1 + 0.05Mileage

CarPrice = 1 + 0.25Mileage

CarPrice = 1 + 0.75Mileage



Mileage (in kmpl)	*Car Price (in cr)*
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51

Sum of absolute Error / Training Error

Notion of Maxima and Minima of a function

Local Minimum:

- A point $x = c$ is a local minimum of a function $f(x)$ if there exists a neighborhood around c such that $f(c)$ is less than or equal to $f(x)$ for all x in that neighborhood.
- Mathematically, $f(c) \leq f(x)$ for x in a neighborhood of c.

Local Minimum:

- $x = 2$ is a local minimum because it's a critical point where $f'(x) = 0$ and $f''(x) > 0$.
- The function decreases to the left of $x = 2$ and increases to the right, indicating a valley in the graph.

Global Minimum:

- A point $x = c$ is a global minimum of a function $f(x)$ if $f(c)$ is less than or equal to $f(x)$ for all x in the entire domain of $f(x)$.
- Mathematically, $f(c) \leq f(x)$ for all x in the domain of $f(x)$.

Global Minimum:

- Since the parabola opens upwards, the local minimum at $x = 2$ is also the global minimum.
- The global minimum value is $f(2) = 2 \times 2^2 - 8 \times 2 + 8 = 0$.

Notion of Maxima and Minima of a function

Local Maximum:

- A point $x = c$ is a local maximum of a function $f(x)$ if there exists a neighborhood around c such that $f(c)$ is greater than or equal to $f(x)$ for all x in that neighborhood.
- Mathematically, $f(c) \geq f(x)$ for x in a neighborhood of c.

Global Maximum:

- A point $x = c$ is a global maximum of a function $f(x)$ if $f(c)$ is greater than or equal to $f(x)$ for all x in the entire domain of $f(x)$.
- Mathematically, $f(c) \geq f(x)$ for all x in the domain of $f(x)$.

Gradient Descent Approach

What is Gradient Descent or Steepest Descent?

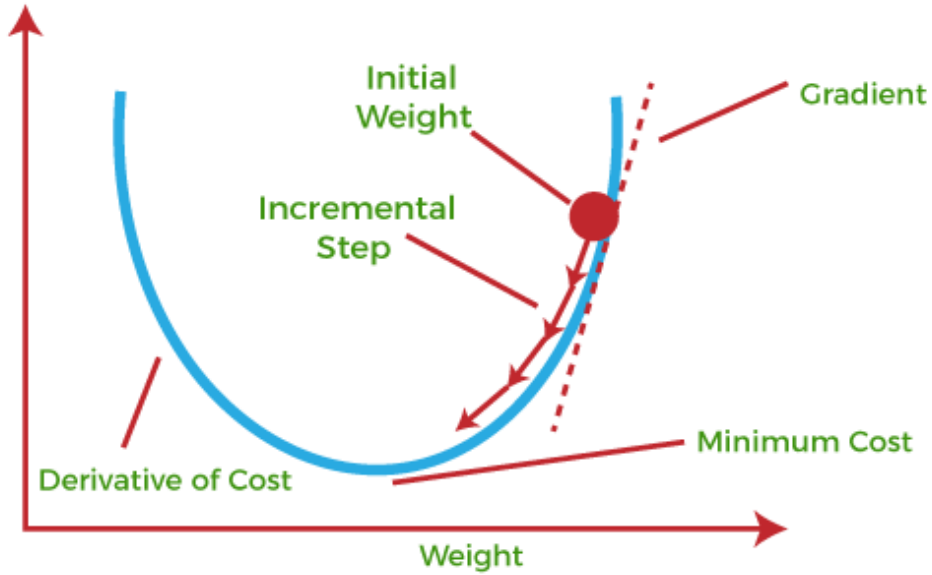
- Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results.
- ***It helps in finding the local minimum of a function.***
- Further, gradient descent is also used to train Neural Networks.
- The main objective of gradient descent is to minimize the function using iteration of parameter updates

The best way to define the local minimum or local maximum of a function using gradient descent is as follows:

- If we move towards a negative gradient or away from the gradient of the function at the current point, it will give the **local minimum** of that function.
- Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the **local maximum** of that function.

Gradient Descent Cost Function

The main objective of using a gradient descent algorithm is to minimize the cost function using iteration.



To achieve the goal, it performs two steps iteratively:

- Calculates the first-order derivative of the function to compute the **gradient or slope** of that function.
- Move away from the direction of the gradient, which means slope increased from the current point by alpha times, where Alpha is defined as Learning Rate. It is a tuning parameter in the optimization process which helps to decide the length of the steps.

Gradient Descent Cost Function

1. **Cost Function:**
   - The cost function, denoted as $J(\theta)$, measures the error or difference between the predicted values of a model (hypothesis) and the actual values.
2. **Minimization Objective:**
   - The objective is to find the values of parameters (θ) that minimize the cost function, leading to a model that best fits the data.
3. **Gradient Descent:**
   - Gradient descent iteratively adjusts the parameters in the direction opposite to the gradient of the cost function. This process continues until the algorithm converges to the minimum.

Mathematics:

1. **Cost Function:**
   - For a linear regression problem, the cost function is often represented as the mean squared error:
     $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$
     where $h_\theta(x^{(i)})$ is the predicted value, $y^{(i)}$ is the actual value, and m is the number of training examples.

Gradient Descent Cost Function

2. **Gradient Descent Update Rule:**
   - The parameters are updated using the gradient of the cost function with respect to each parameter (θ_j):

     $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

     where α is the learning rate.
3. **Partial Derivatives:**
   - The partial derivatives represent the rate of change of the cost with respect to each parameter. For the mean squared error, the partial derivatives are calculated as:

     $\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Example:

Consider a simple linear regression problem with one parameter (θ_1) and a cost function defined as the mean squared error:

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

where $h_\theta(x^{(i)}) = \theta_1 \cdot x^{(i)}$ is the predicted value.

Gradient Descent Cost Function

1. **Gradient Descent Update Rule:**

   $\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$

2. **Iterative Process:**
   - The algorithm iteratively updates θ_1 based on the gradient until convergence.

Suppose we have the following dataset:

$(x^{(1)}, y^{(1)}) = (1, 3)$
$(x^{(2)}, y^{(2)}) = (2, 5)$
$(x^{(3)}, y^{(3)}) = (3, 7)$

Gradient Descent Iterations:

1. **Initialize Parameters:**
   - Set an initial value for the parameter θ_1: $\theta_1 = 0$.
2. **Compute Predictions:**
   - Compute the predicted values using the current parameter:

   $h_\theta(x^{(i)}) = \theta_1 \cdot x^{(i)}$
   $h_\theta(x^{(1)}) = 0$
   $h_\theta(x^{(2)}) = 0$
   $h_\theta(x^{(3)}) = 0$

Gradient Descent Cost Function

3. **Compute Cost:**
   - Calculate the cost using the predictions and actual values:

$$J(\theta_1) = \frac{1}{6}\sum_{i=1}^{3}(h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{6}(3^2 + 5^2 + 7^2) = \frac{83}{6}$$

4. **Compute Gradients:**
   - Compute the partial derivative of the cost with respect to the parameter θ_1:

$$\frac{\partial}{\partial\theta_1}J(\theta_1) = \frac{1}{3}\sum_{i=1}^{3}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$
$$\frac{\partial}{\partial\theta_1}J(\theta_1) = \frac{1}{3}\sum_{i=1}^{3}(0 - y^{(i)}) \cdot x^{(i)}$$
$$\frac{\partial}{\partial\theta_1}J(\theta_1) = \frac{1}{3}\sum_{i=1}^{3} -y^{(i)} \cdot x^{(i)}$$
$$\frac{\partial}{\partial\theta_1}J(\theta_1) = -\frac{1}{3}\sum_{i=1}^{3} y^{(i)} \cdot x^{(i)}$$

5. **Update Parameters:**
   - Update the parameter using the gradient descent update rule:

$$\theta_1 := \theta_1 - \alpha\frac{\partial}{\partial\theta_1}J(\theta_1)$$
$$\theta_1 := \theta_1 + \alpha\frac{1}{3}\sum_{i=1}^{3} y^{(i)} \cdot x^{(i)}$$

where α is the learning rate.

Gradient Descent Cost Function

Gradient Descent Iteration 2:

Assuming a learning rate (α) of 0.1, we'll update the parameter θ_1 using the gradient descent update rule:

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

Previous Values:

- $\theta_1 = 0$
- $J(\theta_1) = \frac{83}{6}$
- $\frac{\partial}{\partial \theta_1} J(\theta_1) = -\frac{1}{3} \sum_{i=1}^{3} y^{(i)} \cdot x^{(i)}$

Update Rule:

$$\theta_1 := \theta_1 + 0.1 \cdot \frac{1}{3} \sum_{i=1}^{3} y^{(i)} \cdot x^{(i)}$$

Updated Values:

$$\theta_1 := 0 + 0.1 \cdot \frac{1}{3} \cdot (-3 \cdot 1 - 5 \cdot 2 - 7 \cdot 3)$$
$$\theta_1 := 0 + 0.1 \cdot \frac{1}{3} \cdot (-3 - 10 - 21)$$
$$\theta_1 := 0 + 0.1 \cdot \frac{1}{3} \cdot (-34)$$
$$\theta_1 := 0 - 0.1 \cdot \frac{34}{3}$$
$$\theta_1 \approx -1.13$$

Compute Predictions:

Now, compute the predictions with the updated parameter:

$$h_\theta(x^{(i)}) = \theta_1 \cdot x^{(i)}$$

$$h_\theta(x^{(1)}) \approx -1.13 \cdot 1$$
$$h_\theta(x^{(2)}) \approx -1.13 \cdot 2$$
$$h_\theta(x^{(3)}) \approx -1.13 \cdot 3$$

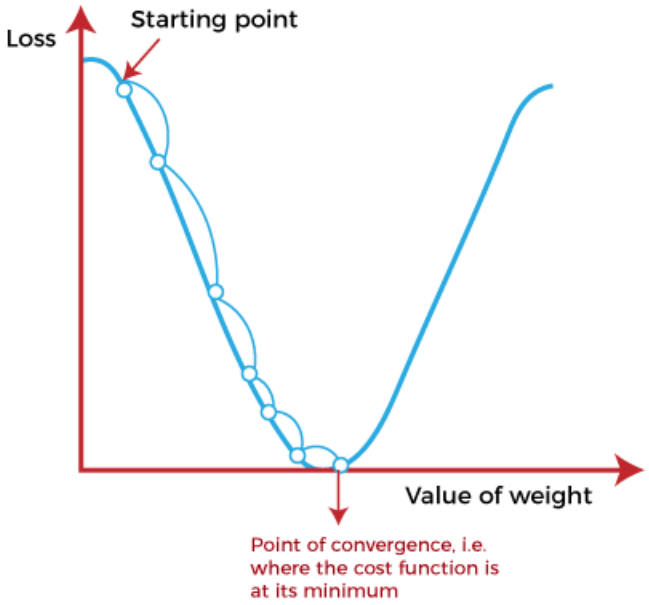
How Gradient Descent works?

The equation for simple linear regression is given as:

$$Y=mX+c$$

Where 'm' represents the slope of the line, and 'c' represents the intercepts on the y-axis.

- The starting point(shown in above fig.) is used to evaluate the performance as it is considered just as an arbitrary point.
- At this starting point, we will derive the first derivative or slope and then use a tangent line to calculate the steepness of this slope. Further, this slope will inform the updates to the parameters (weights and bias).
- The slope becomes steeper at the starting point or arbitrary point, but whenever new parameters are generated, then steepness gradually reduces, and at the lowest point, it approaches the lowest point, which is called **a point of convergence.**



Gradient Descent Cost Function

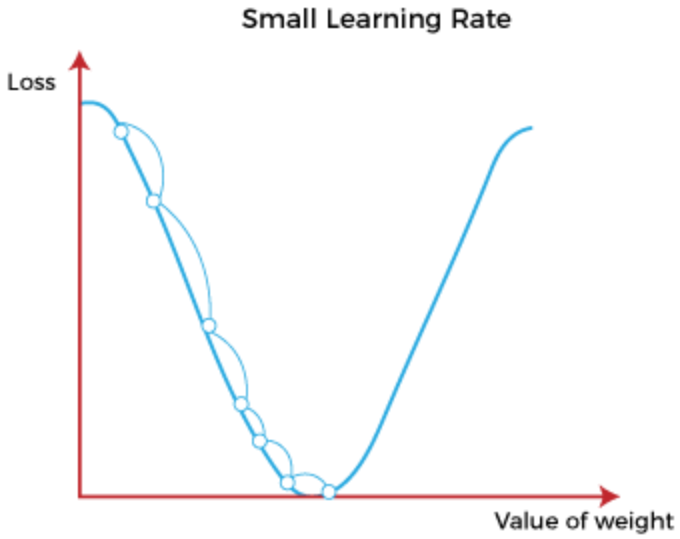
innovate achieve lead

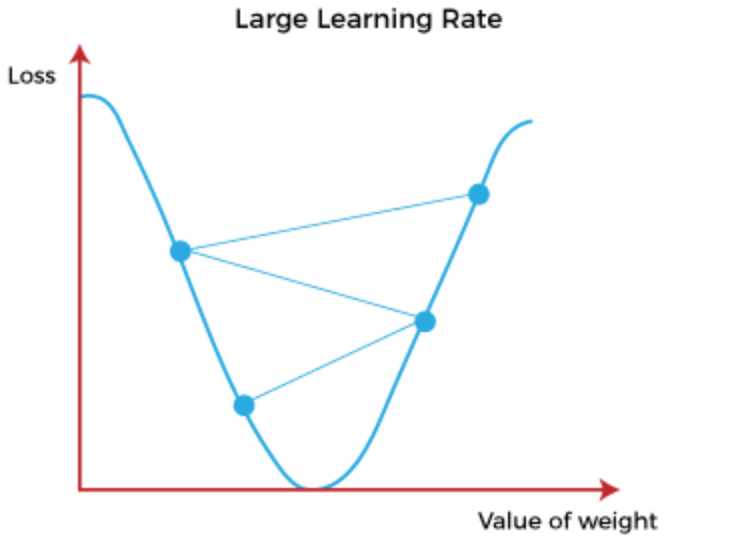
•Direction & Learning Rate

These two factors are used to determine the partial derivative calculation of future iteration and allow it to the point of convergence or local minimum or global minimum.

Learning Rate:

It is defined as the step size taken to reach the minimum or lowest point. This is typically a small value that is evaluated and updated based on the behavior of the cost function. If the learning rate is high, it results in larger steps but also leads to risks of overshooting the minimum. At the same time, a low learning rate shows the small step sizes, which compromises overall efficiency but gives the advantage of more precision.





How Gradient Descent works?

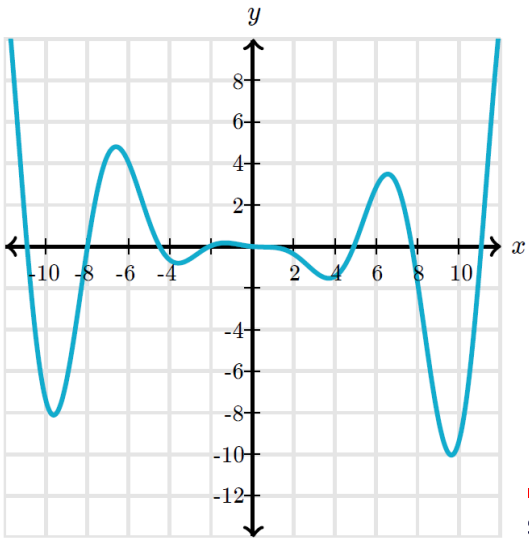
To *minimize* the function, we can instead follow the negative of the gradient, and thus go in the direction of steepest descent. This is gradient descent. Formally, if we start at a point x_0 and move a positive distance α in the direction of the negative gradient, then our new and improved x_1 will look like this:

$$x_1 = x_0 - \alpha \nabla f(x_0)$$

Starting from an initial guess x_0, we keep improving little by little until we find a local minimum. This process may take thousands of iterations, so we typically implement gradient descent with a computer.

Example 1

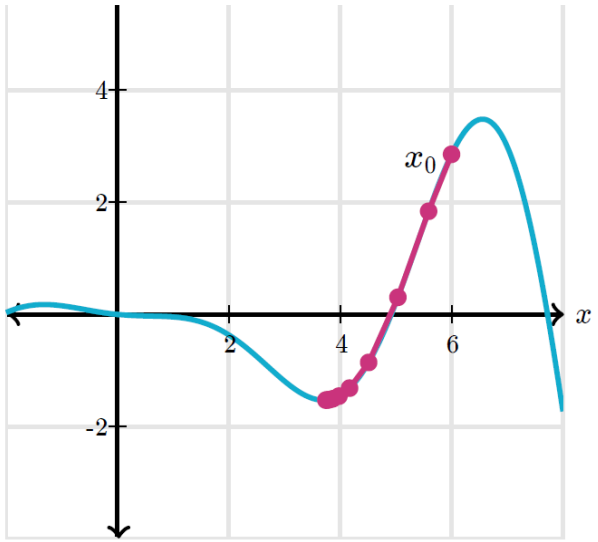
Consider the function $f(x) = \dfrac{x^2 \cos(x) - x}{10}$.

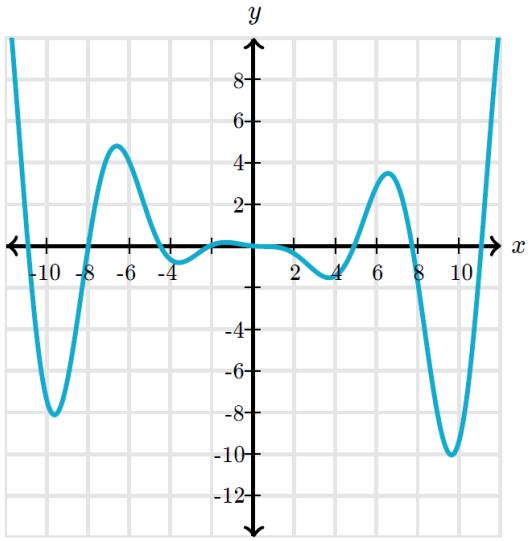


How Gradient Descent works?

As we can see from the graph, this function has many local minima. Gradient descent will find different ones depending on our initial guess and our step size.

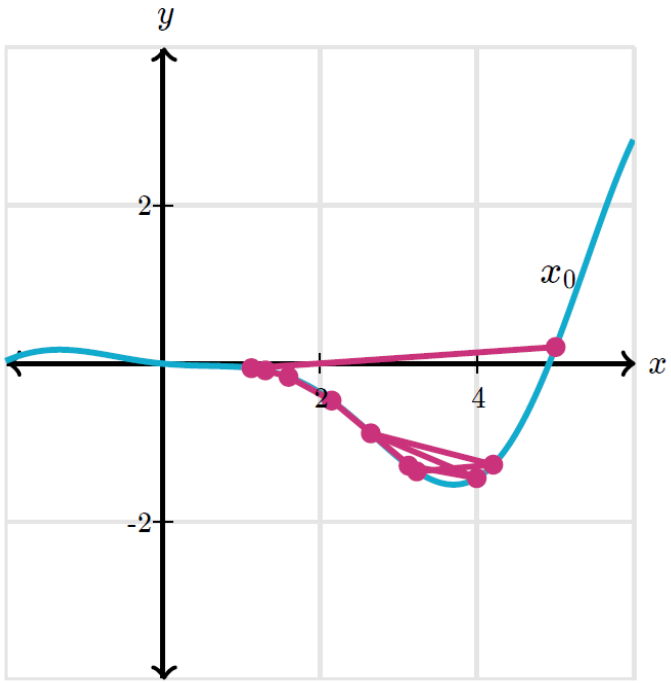
If we choose $x_0 = 6$ and $\alpha = 0.2$, for example, gradient descent moves as shown in the graph below. The first point is x_0, and lines connect each point to the next in the sequence. After only 10 steps, we have converged to the minimum near $x = 4$.





How Gradient Descent works?

If we use the same x_0 but $\alpha = 1.5$, it seems as if the step size is too large for gradient descent to converge on the minimum. We'll return to this when we discuss the limitations of the algorithm.

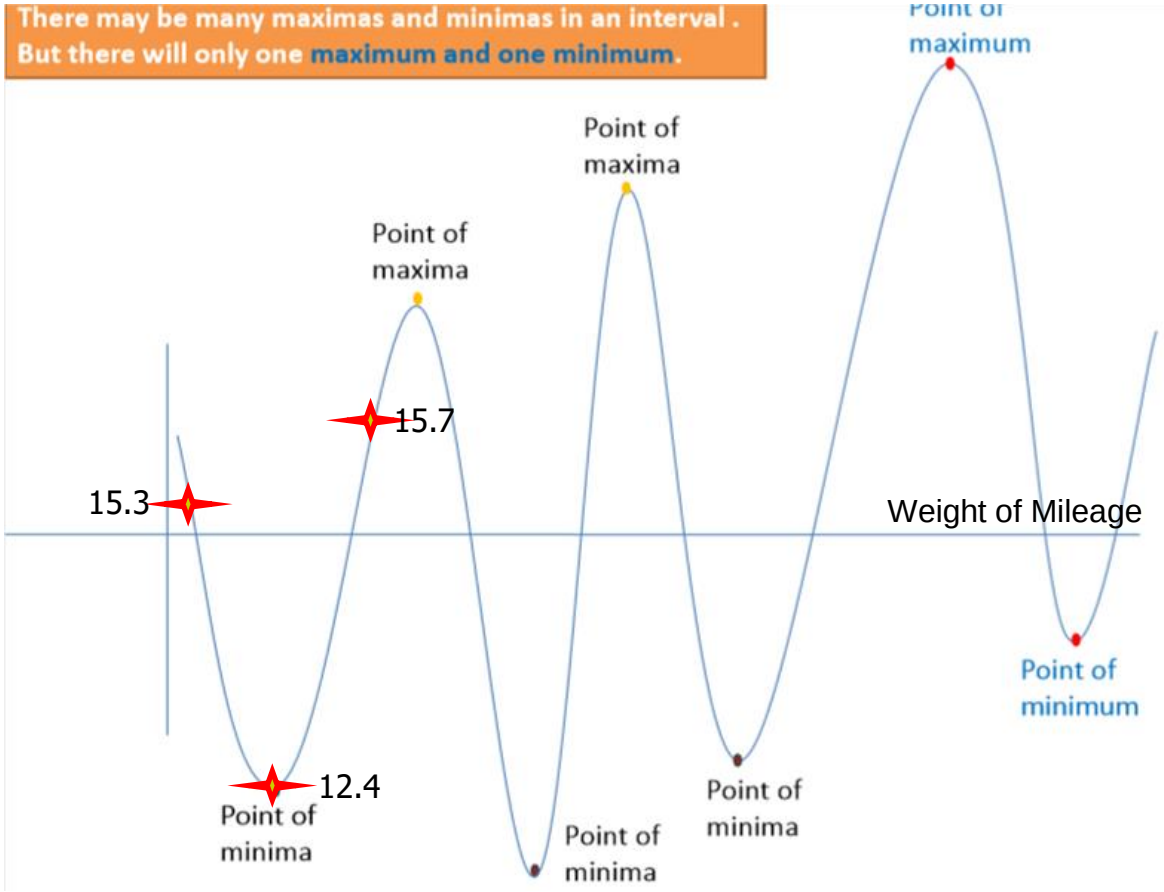


Notion of Gradient Descent

CarPrice = 1 + 0.05Mileage

CarPrice = 1 + 0.25Mileage

CarPrice = 1 + 0.75Mileage



Mileage (in kmpl)	*Car Price (in cr)*
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51

Sum of absolute Error / Training Error

Polynomial Regression

Polynomial Regression

- **Polynomial Regression** is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modeled as an *nth* degree polynomial.
- In other words, instead of fitting a straight line (as in linear regression), polynomial regression allows us to fit a curve to the data.
- Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y |x)
- In addition to linear terms like b_1x_1, your regression function f can include non-linear terms such as $b_2x_1^2$, $b_3x_1^3$, or even $b_4x_1x_2$, $b_5x_1^2x_2$, and so on.

Mathematical Representation:

The general form of a polynomial regression model is given by:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

where y is the dependent variable, x is the independent variable, $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients to be estimated, n is the degree of the polynomial, and ε is the error term.

Why Polynomial Regression?

- **Non-Linear Relationships:** Linear regression is limited to modeling linear relationships between variables. Polynomial regression is useful when the relationship is more complex and cannot be adequately represented by a straight line.

- **Improved Fit**: In many real-world scenarios, the relationship between variables is better captured by a curve. Polynomial regression provides a flexible approach to fit curves of different shapes, allowing for improved model accuracy.

- **Versatility:** Polynomial regression can be applied to various degrees, allowing flexibility in modeling different levels of complexity in the data.

Polynomial Regression

- The simplest example of polynomial regression has a single independent variable, and the estimated regression function is a polynomial of degree 2: $f(x) = b_0 + b_1x + b_2x^2$.
- Now, remember that you want to calculate b_0, b_1, and b_2, which minimize SSR. These are your unknowns!
- In the case of two variables and the polynomial of degree 2, the regression function has this form: $f(x_1, x_2) = b_0 + b_1x_1 + b_2x_2 + b_3x_1^2 + b_4x_1x_2 + b_5x_2^2$.

Uses of Polynomial Regression: These are basically used to define or describe non-linear phenomenon such as:

- Growth rate of tissues.
- Progression of disease epidemics
- Distribution of carbon isotopes in lake sediments

Polynomial Regression- Example

Let's consider a simple example with a quadratic relationship. We have data points:

$$(x_1, y_1) = (1, 4)$$
$$(x_2, y_2) = (2, 7)$$
$$(x_3, y_3) = (3, 12)$$

We want to fit a quadratic polynomial to these data points.

Polynomial Regression:

The quadratic regression model is given by:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

Polynomial Regression- Example

Calculation:

1. **Matrix Form:**
   - Convert the model into matrix form: $Y = X\beta + \varepsilon$, where Y is the vector of dependent variables, X is the matrix of independent variables, β is the vector of coefficients, and ε is the error term.

$$Y = \begin{bmatrix} 4 \\ 7 \\ 12 \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

2. **Solve for Coefficients:**
   - Solve the normal equations: $X^T X\beta = X^T Y$ to find the coefficients.

$$X^T X = \begin{bmatrix} 3 & 6 & 14 \\ 6 & 14 & 34 \\ 14 & 34 & 94 \end{bmatrix}$$

$$X^T Y = \begin{bmatrix} 23 \\ 53 \\ 137 \end{bmatrix}$$

Solving the system of equations provides the values for $\beta_0, \beta_1, \beta_2$.

Polynomial Regression- Example

Now, we solve the system of equations:

$$X^T X\beta = X^T Y$$

$$\begin{bmatrix} 3 & 6 & 14 \\ 6 & 14 & 34 \\ 14 & 34 & 94 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 23 \\ 53 \\ 137 \end{bmatrix}$$

Solving the System of Equations:

The solution to the system of equations provides the values for $\beta_0, \beta_1, \beta_2$.

$\beta_0 = 2$
$\beta_1 = 1$
$\beta_2 = 1$

Quadratic Model:

The quadratic model is then:

$$y = 2 + x + x^2$$

Issues with Polynomial Regression

- Unfortunately, polynomial regression has a fair number of issues as well. As we increase the complexity of the formula, the number of features also increases which is sometimes difficult to handle. Also, polynomial regression has a tendency to drastically over-fit, even on this simple one dimensional data set.
- There are other issues with polynomial regression. For example, it is inherently non-local, i.e., changing the value of Y at one point in the training set can affect the fit of the polynomial for data points that are very far away.
- Hence, to avoid the use of high degree polynomial on the whole dataset, we can substitute it with many different small degree polynomial functions.
- In order to overcome the disadvantages of polynomial regression, we can use an improved regression technique which, instead of building one model for the entire dataset, divides the dataset into multiple bins and fits each bin with a separate model. **Such a technique is known as Regression spline.**
- Regression splines is one of the most important non linear regression techniques.
- In polynomial regression, we generate new features by using various ***polynomial functions on the existing features*** which imposed a global structure on the dataset***.***
- To overcome this, we can divide the distribution of the data into separate portions and fit linear or low degree polynomial functions on each of these portions.

Metrics for Regression

Metrics for Regression

The various metrics used to evaluate the results of the prediction are

- **Mean Squared Error(MSE)**
- **Root-Mean-Squared-Error(RMSE).**
- **Mean-Absolute-Error(MAE).**
- **R^2 or Coefficient of Determination.**
- **Adjusted R^2**

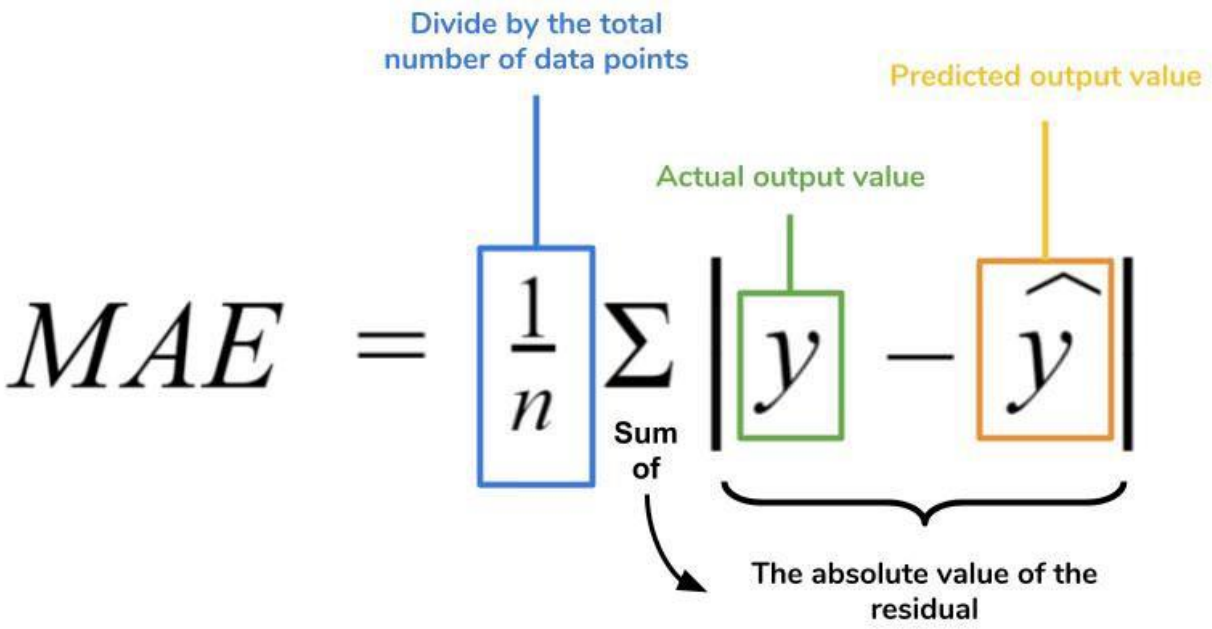
▪**Residual = actual value — predicted value**

$$e = y - \hat{y}$$

▪One of the assumptions of a linear regression model is that the errors must be normally distributed. This means, make sure your residuals are distributed around zero for the entire range of predicted values. Thus, if the residuals are evenly scattered, then your model may perform well.

Mean Absolute Error

- MAE is the **absolute difference** between the **target value** and the **value predicted** by the model. The MAE is more **robust to outliers** and **does not penalize the errors** as extremely as mse. MAE is a linear score which means all the individual differences are weighted equally. **It is not suitable for applications where you want to pay more attention to the outliers.**



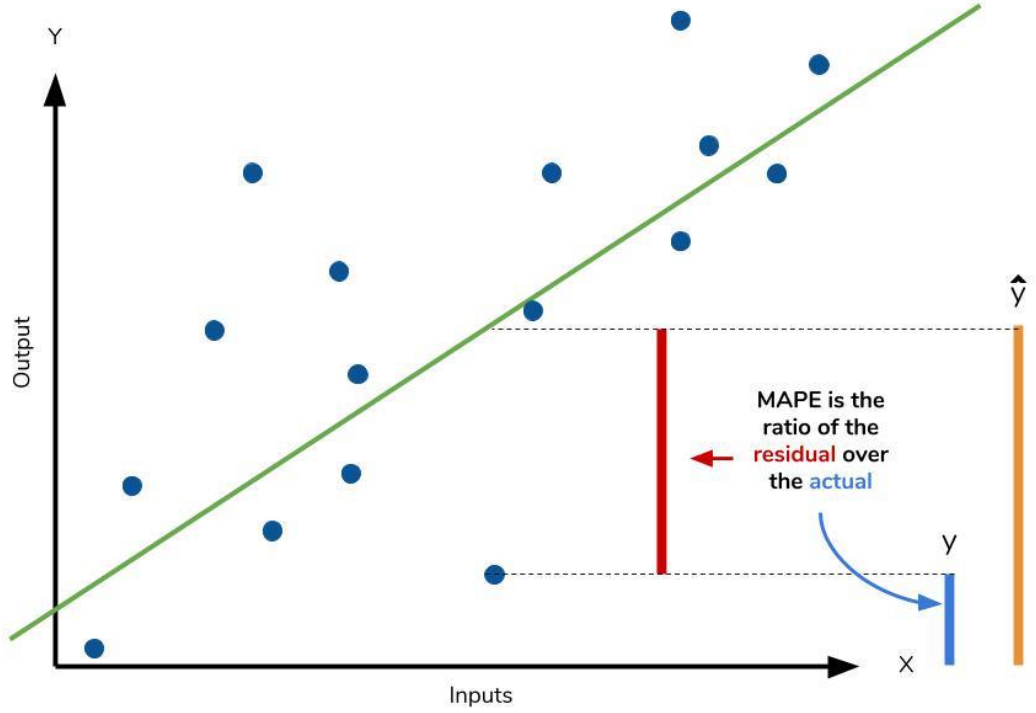
Mean Squared Error

- MSE or Mean Squared Error is one of the most preferred metrics for regression tasks. It is simply the average of the squared difference between the **target value** and the **value predicted** by the regression model. As it squares the differences, it penalizes even a **small error** which leads to over-estimation of how bad the model is. It is preferred more than other metrics because it is **differentiable** and hence can be optimized better.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

Mean absolute percentage error

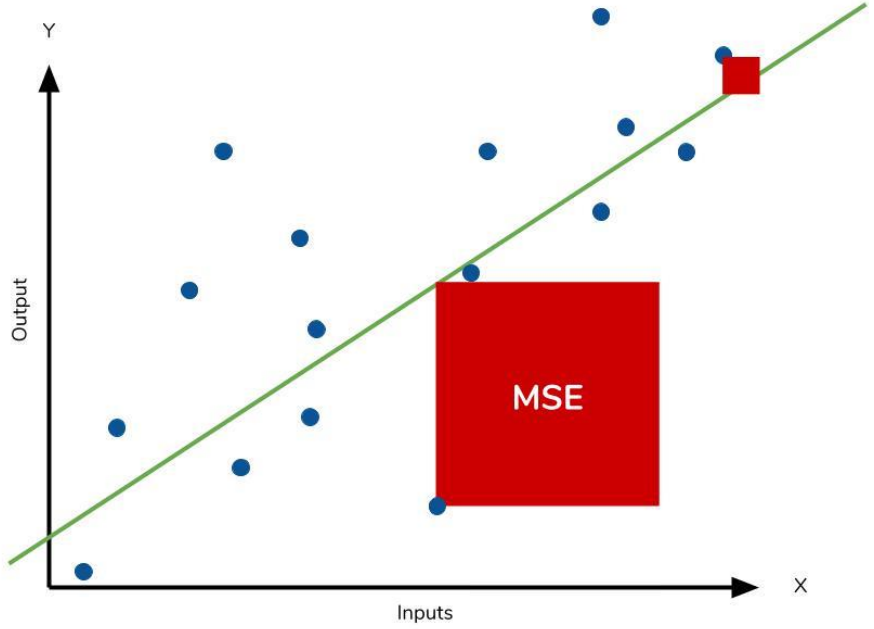
- The **mean absolute percentage error** (MAPE) is the percentage equivalent of MAE. The equation looks just like that of MAE, but with adjustments to convert everything into percentages. Just as MAE is the average magnitude of error produced by your model, the MAPE is how far the model's predictions are off from their corresponding outputs on average. Like MAE, MAPE also has a clear interpretation since percentages are easier for people to conceptualize



Consequences of the Square Term

- The effect of the square term in the MSE equation is most apparent with the presence of outliers in our data. While each residual in MAE contributes **proportionally** to the total error, the error grows **quadratically** in MSE. This ultimately means that outliers in our data will contribute to much higher total error in the MSE than they would the MAE.

Similarly, our model will be penalized more for making predictions that differ greatly from the corresponding actual value. This is to say that large differences between actual and predicted are punished more in MSE than in MAE



Root Mean Squared Error

RMSE is the most widely used metric for regression tasks and is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors. This implies that **RMSE is useful when large errors are undesired**.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

R Square Method – Goodness of Fit

R–squared value is the statistical measure to show how close the data are to the fitted regression line

- **y = actual value**
- **$\bar{y}$ = mean value of y**
- **Yp = predicted value of y**

$$R^2 = 1 - \frac{\sum (y_p - \bar{y})^2}{\sum (y - \bar{y})^2}$$

- Larger R^2 indicates a better fit and means that the model can better explain the variation of the output with different inputs.
- The value R^2 = 1 corresponds to SSR = 0, that is to the **perfect fit** since the values of predicted and actual responses fit completely to each other.

R Square Method – Goodness of Fit

- ***If R^2 is high (say 1), then the model represents the variance of the dependent variable.***
- ***If R^2 is very low, then the model does not represent the variance of the dependent variable and regression is no better than taking the mean value, i.e. you are not using any information from the other variables.***
- ***A Negative R^2 means you are doing worse than the mean value.***
- **Thus R^2 evaluates the scattered data points about the regression line.**
- **It is not possible to see a model with an R^2 of 1. In that case, all predicted values are the same as actual values and this essentially means that all values fall on the regression line.**

Adjusted R^2

- The main difference between **adjusted R-squared** and **R-square** is that **R-squared** describes the amount of variance of the dependent variable represented by every single independent variable, while **adjusted R-squared** measures variation explained by only the independent variables that actually affect the dependent variable.

$$R^2_{adjusted} = \left[\frac{(1-R^2)(n-1)}{n-k-1}\right]$$

In the equation above, n is the number of data points while k is the number of variables in your model, excluding the constant.

R^2 tends to increase with an increase in the number of independent variables. This could be misleading. Thus, the adjusted R-squared penalizes the model for adding furthermore independent variables (k in the equation) that do not fit the model.

Underfitting and Overfitting

Underfitting and Overfitting

- **Underfitting** occurs when a **model can't accurately capture the dependencies among data**, usually as a consequence of its own simplicity.
- when it cannot capture the underlying trend of the data
- Its occurrence simply means that our model or the algorithm does not fit the data well enough.
- It usually happens when we have **less data** to build an accurate model and also when we try to build a linear model with a non-linear data
- It often yields a low R^2 with known data and bad generalization capabilities when applied with new data.
- In such cases the rules of the machine learning model are too easy and flexible to be applied on such minimal data and therefore the model will probably make a lot of wrong predictions.
- Underfitting can be avoided by using **more data** and also **reducing the features** by feature selection.

Underfitting and Overfitting

- A statistical model is said to be **overfitted**, when we train it with a lot of data *(just like fitting ourselves in oversized pants!)*.
- When a model gets trained with so much of data, it starts learning from the **noise** and **inaccurate data** entries in our data set
- **Overfitting** happens when a model learns both **dependencies among data** and **random fluctuations**. In other words, a model learns the existing data too well.
- Then the model does not categorize the data correctly, because of too many details and noise.
- Complex models, which have many features or terms, are often prone to overfitting.
- When applied to known data, such models usually yield high R^2. However, they often don't generalize well and have significantly lower R^2 when used with new data.
- In a nutshell, **Overfitting – High variance and low bias**

Bias VS Variance

- The performance of the model on the task can be described in terms of the prediction error on all examples not used to train the model. We will refer to this as the model error.
- Error(Model): The model error can be decomposed into three sources of error: the **variance** of the model, the **bias** of the model, and the variance of the **irreducible error** in the data.
- **Error(Model) = Variance(Model) + Bias(Model) + Variance(Irreducible Error)**
- Bias is one type of error which occurs due to wrong assumptions about data such as assuming data is linear when in reality, data follows a complex function.
- On the other hand, variance gets introduced with high sensitivity to variations in training data. This also is one type of error since we want to make our model robust against noise.

Bias & Variance

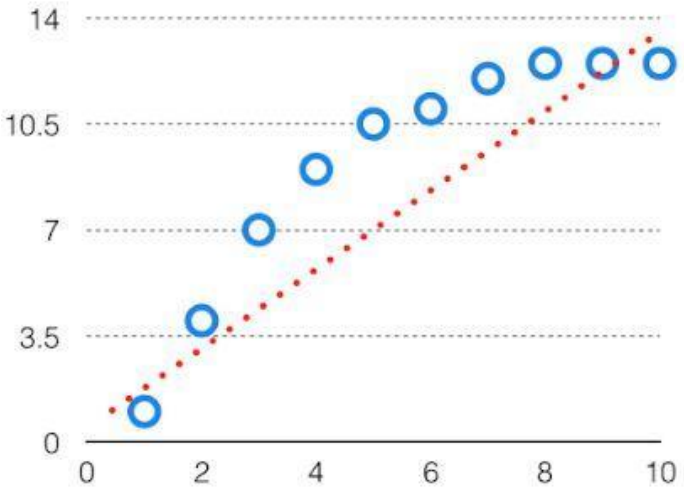
- **Bias** –Bias occurs when an algorithm has limited flexibility to learn the true signal from the dataset
- bias is an error from erroneous assumptions in the learning algorithm. **High bias** can cause an algorithm to miss the relevant relations between features and target outputs **(underfitting).**
- **Variance** – If you train your data on training data and obtain a very low error, upon changing the data and then training the same previous model you experience high error, this is variance.

Bias

- The bias is known as the difference between the prediction of the values by the ML model and the correct value.
- Being high in biasing gives a large error in training as well as testing data.
- Its recommended that an algorithm should always be low biased to avoid the problem of underfitting.
- By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set.
- Such fitting is known as **Underfitting of Data**. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below for an example of such a situation.

In such a problem, a hypothesis looks like follows.

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$



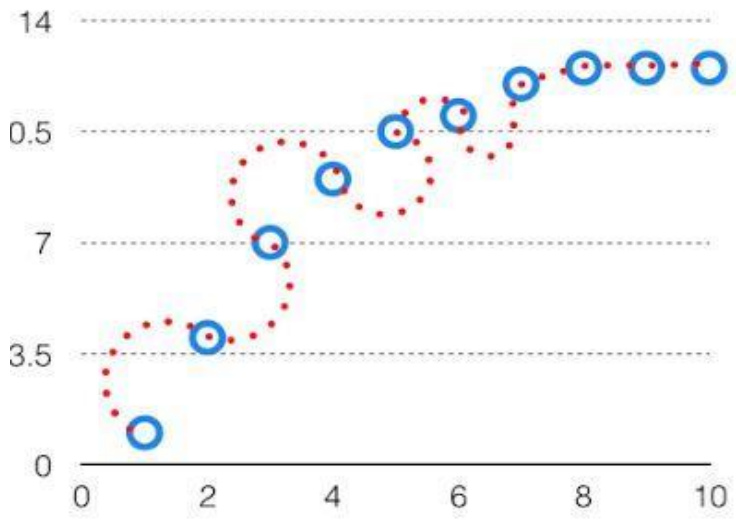
Low Bias: Weak assumptions regarding the functional form of the mapping of inputs to outputs.
High Bias: Strong assumptions regarding the functional form of the mapping of inputs to outputs.
The bias is always positive.

Variance

- The variability of model prediction for a given data point which tells us spread of our data is called the variance of the model.
- The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before.
- As a result, such models perform very well on training data but has high error rates on test data.
- When a model is high on variance, it is then said to as **Overfitting of Data**.
- Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high. While training a data model variance should be kept low.

The high variance data looks like follows.

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



Model Variance

- The variance of the model is the amount the performance of the model changes when it is fit on different training data.
- It captures the impact of the specifics the data has on the model.
- *Variance refers to the amount by which [the model] would change if we estimated it using a different training data set.*
- A model with high variance will change a lot with small changes to the training dataset. Conversely, a model with low variance will change little with small or even large changes to the training dataset.
- **Low Variance**: Small changes to the model with changes to the training dataset.
- **High Variance**: Large changes to the model with changes to the training dataset.
- **The variance is always positive.**

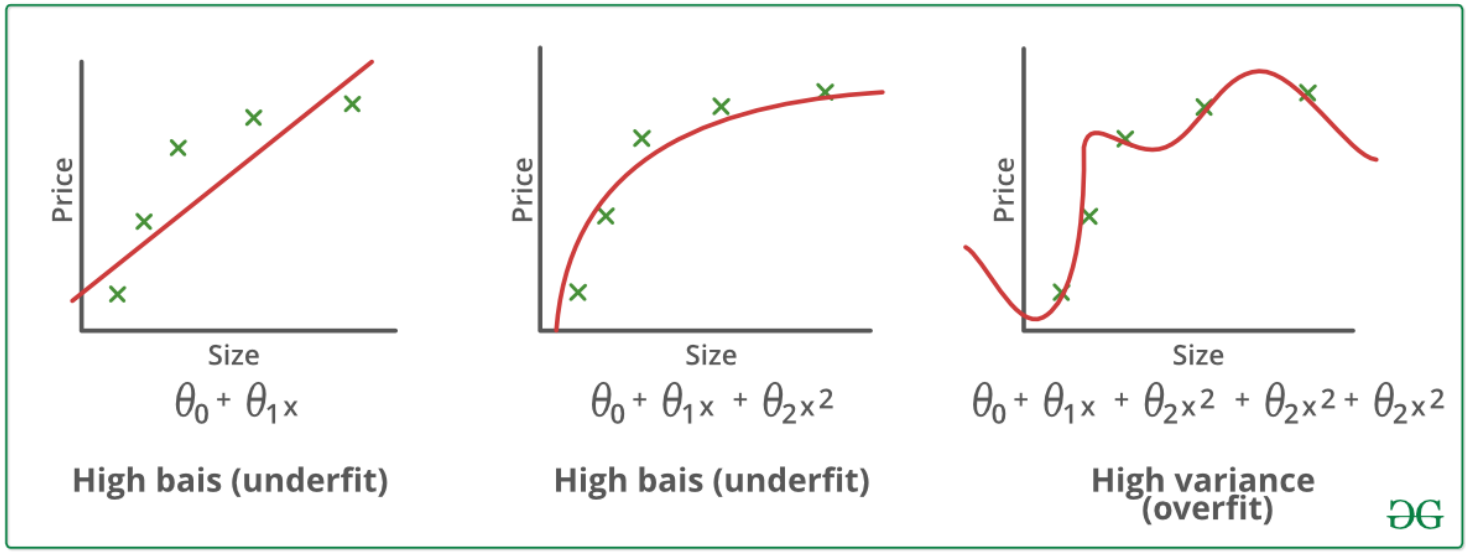
Irreducible Error

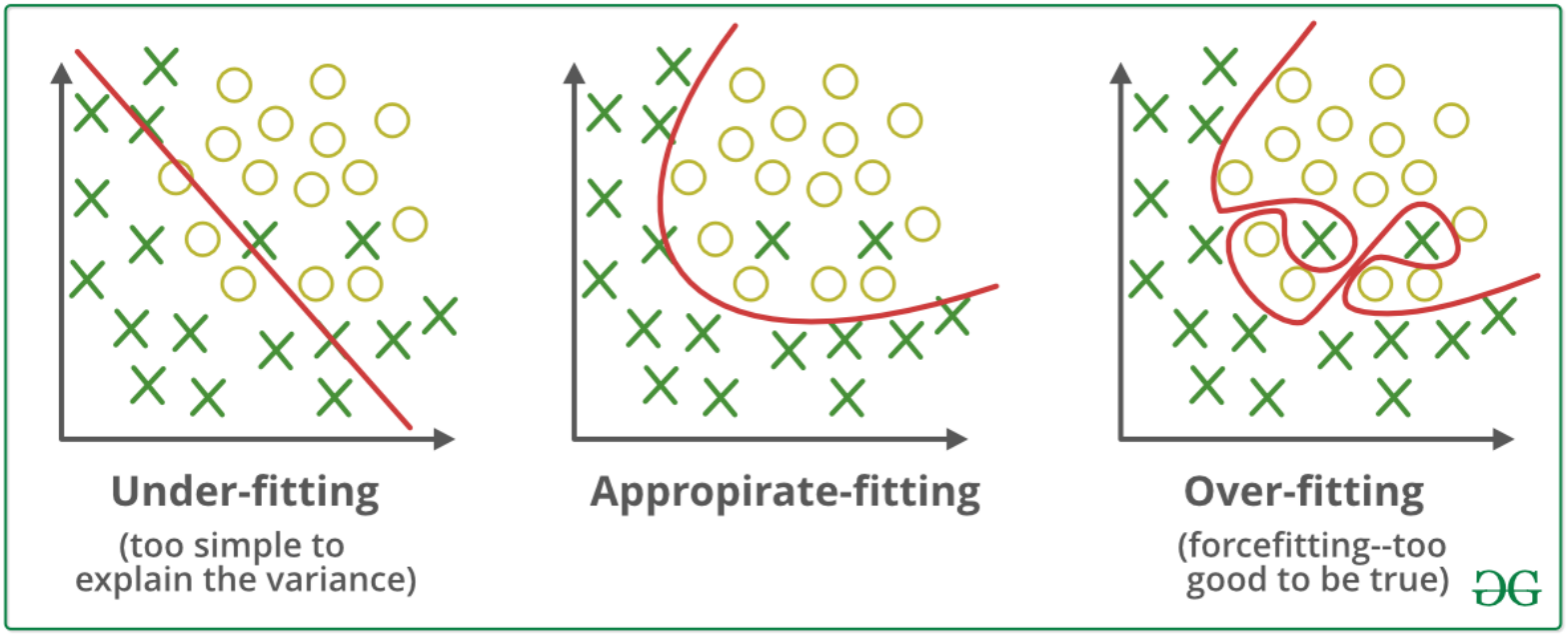
- On the whole, the error of a model consists of reducible error and irreducible error.
- Model Error = Reducible Error + Irreducible Error
- The reducible error is the element that we can improve. It is the quantity that we reduce when the model is learning on a training dataset and we try to get this number as close to zero as possible.
- The irreducible error is the error that we can not remove with our model, or with any model.
- The error is caused by elements outside our control, such as statistical noise in the observations.
- *It is important to keep in mind that the irreducible error will always provide an upper bound on the accuracy of our prediction for Y. This bound is almost always unknown in practice.*

Bias-Variance Trade-off

- Ideally, we would prefer a model with **low bias** and **low variance**, although in practice, this is very challenging.
- In fact, this could be described as the goal of applied machine learning for a given predictive modeling problem,
- Reducing the bias can easily be achieved by increasing the variance.
- Conversely, reducing the variance can easily be achieved by increasing the bias.
- This relationship is generally referred to as the **bias-variance trade-off**. It is a conceptual framework for thinking about how to choose models and model configuration.
- We can choose a model based on its bias or variance.
- Simple models, such as linear regression and logistic regression, generally have a high bias and a low variance.
- Complex models, such as random forest, generally have a low bias but a high variance.

Underfitting and Overfitting





Techniques to reduce overfitting :

- **Simple Techniques to Prevent Overfitting**
- Hold-out (data)
- Cross-validation (data)
- Data augmentation (data)
- Feature selection (data)
- L1 / L2 regularization (learning algorithm)
- Remove layers / number of units per layer (model)
- Dropout (model)
- Early stopping (model)

Techniques to reduce overfitting :

1. **Collect/Use more data:** This approach is, however, considered costly, and consumers should also ensure that the data used is relevant and safe.
2. **Data augmentation:** Data augmentation lets a sample data appear subtly different each time the algorithm processes it. The approach makes each data set look unique to the model and stops the model from understanding the data sets' characteristics.

Adding noise to the input and output data is another option. Adding noise to the input keeps the model robust without compromising the accuracy and privacy of information, thus adding noise to the output makes the information more varied. This must, however, be performed with moderation.

3. **Simplify the data/Remove features**

Even though this method may lead to some loss in information, we could just reduce the hierarchy and complexity of the data. Pruning, reducing the parameters in a neural network, and using dropouts are some of the techniques that can be introduced.

4. **Ensemble Learning**

- EL is a technique of machine learning that operates by integrating two or more different models' predictions.

Techniques to reduce overfitting :

- The most common strategies for assembly include **boosting** and **bagging**.
- **Boosting** – works to increase its overall complexity by using simple base models. It teaches a large number of poor learners structured in a series, such that each learner learns from the learner's errors before him in the series.
- **Bagging** – is the opposite of boosting and is the other ensemble process. Bagging operates by teaching a huge number of powerful learners arranged in a parallel pattern to optimize their forecasts and then merging them.

5. Cross-validation: CV is a powerful technique to avoid overfitting.

We partition the data into k subsets, referred to as folds, in regular k-fold cross-validation. Then, by using the remaining fold as the test set (called the "holdout fold"), we train the algorithm iteratively on k-1 folds. This helps us to use only the initial training set to tune hyperparameters. This helps us to retain our test collection for choosing our final model as a truly unknown dataset.

6. Early stopping : This method is kind of intuitive. The problem we have is that our model trains too long, and overfits. What's the solution?

Don't train too long!

Before the learner passes the stage, we stop the training phase. Simple, right?

Techniques to reduce overfitting :

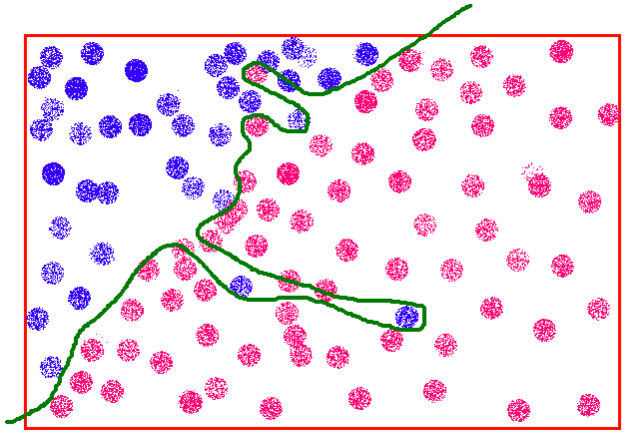
7. Regularization

Regularization is a whole class of similar methods that are used to force the model to simplify itself with the least loss in information.

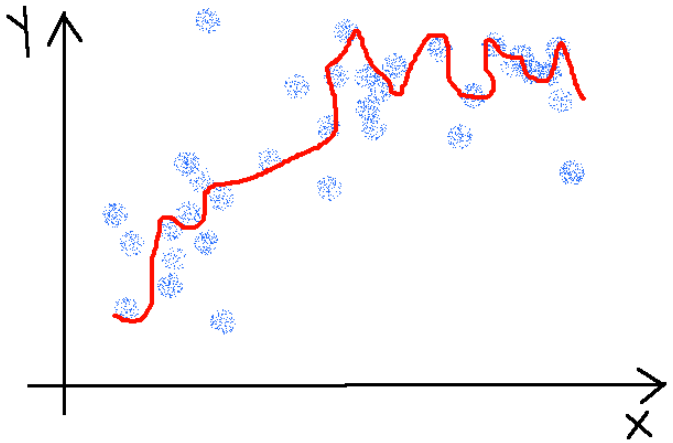
The types of regularization are:

L1: A type of regularization that penalizes weights in proportion to the **sum of the absolute values** of the weights.

L2: A type of regularization that penalizes weights in proportion to the **sum of the squares** of the weights.



Overfit Classification



Overfitted regression model

Techniques to reduce underfitting :

innovate achieve lead

1. Increase model complexity
2. Increase number of features, performing feature engineering
3. Remove noise from the data.
4. Increase the number of epochs or increase the duration of training to get better results.

Regularization

- **Overfitting** is one of the most serious kinds of problems related to machine learning. It occurs when a model learns the training data too well.
- The model then learns not only the relationships among data but also the noise in the dataset.
- Overfitted models tend to have good performance with the data used to fit them (the training data), but they behave poorly with unseen data (or test data, which is data not used to fit the model).
- **Overfitting usually occurs with complex models**.
- **regularization** is the process of adding information in order to solve an ill-posed problem or to prevent over fitting
- **Regularization** normally tries to reduce or penalize the complexity of the model. Regularization techniques applied with logistic regression mostly tend to penalize large coefficients b_0, b_1, …, b_r:

Regularized Regression

- Such a model will not generalize well on the unseen data. To overcome this shortcoming, we do regularization which penalizes large coefficients.
- **L1 regularization** penalizes the LLF(log likelihood) with the scaled sum of the absolute values of the weights: $|b_0|+|b_1|+\cdots+|b_r|$.
- **L2 regularization** penalizes the LLF with the scaled sum of the squares of the weights: $b_0^2+b_1^2+\cdots+b_r^2$.
- **Elastic-net regularization** is a linear combination of L1 and L2 regularization.
- **Regularization can significantly improve model performance on unseen data.**

Regularization

- A regression model that uses L1 regularization technique is called ***Lasso Regression*** and model which uses L2 is called ***Ridge Regression***
- *The key difference between these two is the penalty term.*
- **Ridge regression** adds ***"squared magnitude"*** of coefficient as penalty term to the loss function. Here the *highlighted* part represents L2 regularization element.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Here, if lambda is zero then you can imagine we get back OLS. However, if lambda is very large then it will add too much weight and it will lead to over-fitting. Having said that it's important how lambda is chosen. This technique works very well to avoid over-fitting issue.

Regularization

- **Lasso Regression** (**Least Absolute Shrinkage and Selection Operator**) adds "***absolute value of magnitude***" of coefficient as penalty term to the loss function.

Again, if *lambda* is zero then we will get back OLS whereas very large value will make coefficients zero hence it will under-fit.

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

The **key difference** between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for **feature selection** in case we have a huge number of features.

Elastic Net

- In elastic Net Regularization we added the both terms of L_1 and L_2 to get the final loss function. This leads us to reduce the following loss function:

$$L_{enet}(\hat{\beta}) = \frac{\sum_{i-1}^{n}(y_i - x_i'\hat{\beta})^2}{2n} + \lambda\left(\frac{1-\alpha}{2}\sum_{j-1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j-1}^{m}|\hat{\beta}_j|\right),$$

where α is the mixing parameter between ridge ($\alpha = 0$) and lasso ($\alpha = 1$).

How to Detect Overfitting

- we can't know how well our model will perform on new data until we actually test it.
- To address this, we can split our initial dataset into separate *training* and *test* subsets.
- This method can approximate of how well our model will perform on new data.
- **If our model does much better on the training set than on the test set, then we're likely overfitting.**
- For example, it would be a big red flag if our model saw 99% accuracy on the training set but only 55% accuracy on the test set.

How to Prevent Overfitting

Cross-validation

- Cross-validation is a powerful preventative measure against overfitting.
- The idea is clever: Use your initial training data to generate multiple mini train-test splits. Use these splits to tune your model.
- In standard k-fold cross-validation, we partition the data into k subsets, called folds. Then, we iteratively train the algorithm on k-1 folds while using the remaining fold as the test set (called the "holdout fold").

Cross Validation

- Model evaluation method
- An approximate estimate of how well the learned model will do on “unseen” data
- Slightly better than ‘residuals’
- Residuals do not give a clear indication when predicting unseen data
- Types of Cross validation
  - **Validation Set Approach**
  - **Leave-P-out cross-validation**
  - **Leave one out cross-validation**
  - **K-fold cross-validation**
  - **Stratified k-fold cross-validation**

Methods of Cross Validation

Validation Set Approach

- We divide our input dataset into a training set and test or validation set in the validation set approach. Both the subsets are given 50% of the dataset.
- But it has one of the big disadvantages that we are just using a 50% dataset to train our model, so the model may miss out to capture important information of the dataset. **It also tends to give the underfitted model**.

Leave-P-out cross-validation

- In this approach, the p datasets are left out of the training data. It means, if there are total n datapoints in the original input dataset, then n-p data points will be used as the training dataset and the p data points as the validation set. This complete process is repeated for all the samples, and the average error is calculated to know the effectiveness of the model.
- There is a disadvantage of this technique; that is, it can be computationally difficult for the **large p**.

Methods of Cross Validation

Leave one out cross-validation

- This method is similar to the leave-p-out cross-validation, but instead of p, we need to take 1 dataset out of training. It means, in this approach, for each learning set, only one datapoint is reserved, and the remaining dataset is used to train the model. This process repeats for each datapoint. Hence for n samples, we get n different training set and n test set.

It has the following features:

- In this approach, the bias is minimum as all the data points are used.
- The process is executed for n times; hence execution time is high.
- This approach leads to high variation in testing the effectiveness of the model as we iteratively check against one data point.

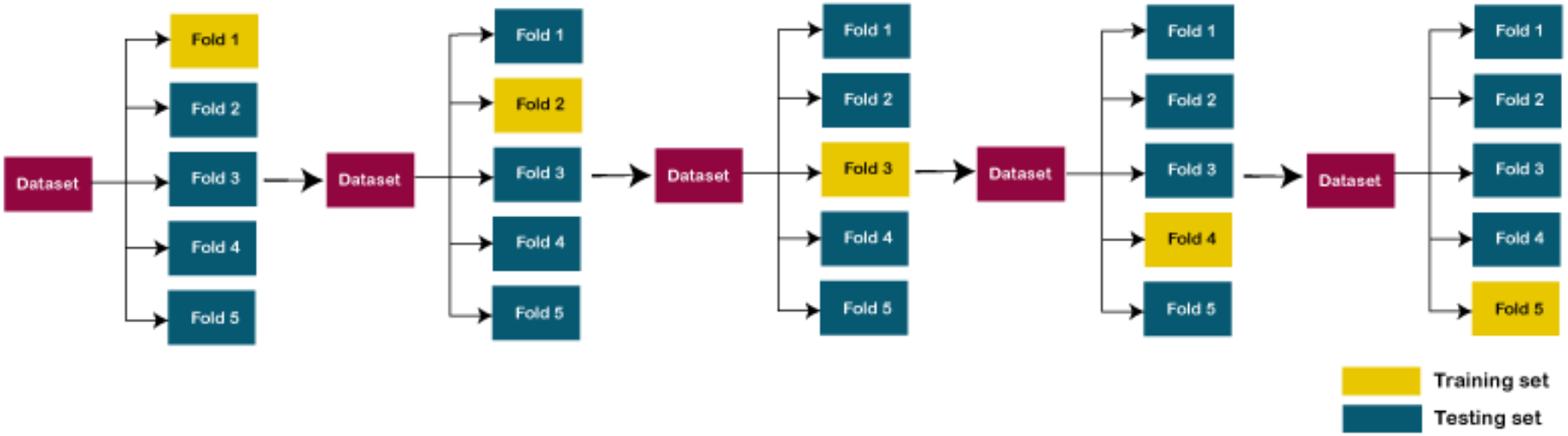
Methods of Cross Validation

K-Fold Cross-Validation

- K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called **folds**. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the **output is less biased than other methods.**
- The steps for k-fold cross-validation are:
- Split the input dataset into K groups
- For each group:
  - Take one group as the reserve or test data set.
  - Use remaining groups as the training dataset
  - Fit the model on the training set and evaluate the performance of the model using the test set.

Methods of Cross Validation

- **Let's take an example of 5-folds cross-validation. So, the dataset is grouped into 5 folds. On 1st iteration, the first fold is reserved for test the model, and rest are used to train the model. On 2nd iteration, the second fold is used to test the model, and rest are used to train the model. This process will continue until each fold is not used for the test fold.**



Methods of Cross Validation

Stratified k-fold cross-validation

- This technique is similar to k-fold cross-validation with some little changes. This approach works on stratification concept, it is a process of rearranging the data to ensure that each fold or group is a good representative of the complete dataset. To deal with the bias and variance, it is one of the best approaches.
- It can be understood with an example of housing prices, such that the price of some houses can be much high than other houses. To tackle such situations, a stratified k-fold cross-validation technique is useful.

Holdout Method

- This method is the simplest cross-validation technique among all. In this method, we need to remove a subset of the training data and use it to get prediction results by training it on the rest part of the dataset.
- The error that occurs in this process tells how well our model will perform with the unknown dataset. Although this approach is simple to perform, it still faces the issue of high variance, and it also produces misleading results sometimes.

Comparison of Cross-validation to train/test split in Machine Learning

- **Train/test split:** The input data is divided into two parts, that are training set and test set on a ratio of 70:30, 80:20, etc. It provides a **high variance**, which is one of the biggest disadvantages.
  - **Training Data:** The training data is used to train the model, and the dependent variable is known.
  - **Test Data:** The test data is used to make the predictions from the model that is already trained on the training data. This has the same features as training data but not the part of that.
- **Cross-Validation dataset:** It is used to overcome the disadvantage of **train/test split** by splitting the dataset into groups of train/test splits, and averaging the result. It can be used if we want to optimize our model that has been trained on the training dataset for the best performance. It is more efficient as compared to train/test split as every observation is used for the training and testing both.

Methods of Cross Validation

Limitations of Cross-Validation

- For the ideal conditions, it provides the optimum output. **But for the inconsistent data, it may produce a drastic result**. So, it is one of the big disadvantages of cross-validation, as there is no certainty of the type of data in machine learning.
- In predictive modeling, the data evolves over a period, due to which, it may face the differences between the training set and validation sets. Such as if we create a model for the prediction of stock market values, and the data is trained on the previous 5 years stock values, but the realistic future values for the next 5 years may drastically different, so it is difficult to expect the correct output for such situations.

Applications of Cross-Validation

- This technique can be used to compare the performance of different predictive modeling methods.
- It has great scope in the medical research field.
- It can also be used for the meta-analysis, as it is already being used by the data scientists in the field of medical statistics.

Interpolation and Extrapolation

- Prediction *within* the range of values in the dataset used for model-fitting is known informally as **interpolation**.
- Prediction *outside* this range of the data is known as **extrapolation**. Performing extrapolation relies strongly on the regression assumptions.
- Suppose that we are given the following tabulated values of the function f(x) corresponding to a discrete set of values of x:

x	x_0	x_1	x_2	.....	x_n
f(x)	y_0	y_1	y_2	.....	y_n

- **Interpolation** is the process of finding the value of f(x) corresponding to any **untabulated value of x between x_0 and x_n**.
- The process of finding the value of f(x) for **some value of x outside the given range [x0, xn]** is called **extrapolation**.
- It assumes that the behaviour of f(x) outside the given range is identical to the behaviour of f(x) inside the given range and this may not always be valid.

Interpolation methods used in Machine Learning

Linear Interpolation: Linear interpolation assumes a linear relationship between data points. Given two data points (x1, y1) and (x2, y2), the value at any point x between x1 and x2 can be estimated using the formula:

$$y = y1 + \frac{(x - x1) \cdot (y2 - y1)}{(x2 - x1)}$$

Polynomial Interpolation: Polynomial interpolation involves fitting a polynomial function to the given data points. The Lagrange polynomial and Newton's divided differences are common techniques for polynomial interpolation.

Suppose you have three data points (1, 10), (2, 15), and (3, 30). You can use polynomial interpolation to find a quadratic function that fits these points.

The Lagrange polynomial for these points is:

$$P(x) = \frac{(x-2)(x-3)}{(1-2)(1-3)} \cdot 10 + \frac{(x-1)(x-3)}{(2-1)(2-3)} \cdot 15 + \frac{(x-1)(x-2)}{(3-1)(3-2)} \cdot 30$$

Simplifying this polynomial would give you the quadratic interpolation function.

Spline Interpolation: Spline interpolation uses piecewise-defined polynomial functions (splines) to interpolate between data points. This method often provides a smoother interpolation compared to a single polynomial.

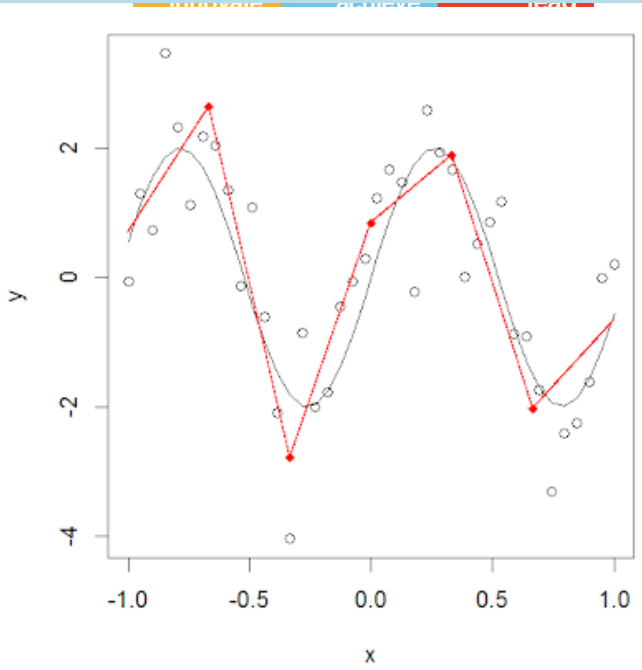
Consider the data points (1, 10), (2, 15), and (3, 30) again. Spline interpolation might involve fitting a piecewise cubic polynomial to these points.

Thank You

Regression spline

The points where the division occurs are called Knots. Functions which we can use for modelling each piece/bin are known as Piecewise functions. There are various piecewise functions that we can use to fit these individual bins.

Piecewise Step Functions



- One of the most common piecewise functions is a Step function. Step function is a function which remains constant within the interval. We can fit individual step functions to each of the divided portions in order to avoid imposing a global structure. Here we break the range of X into bins, and fit a different constant in each bin.
- In greater detail, we create cut points C1 , C2, . . . , Ck in the range of X, and then construct K + 1 new variables.

Regression spline

$$
\begin{aligned}
C_0(X) &= I(X < c_1), \\
C_1(X) &= I(c_1 \leq X < c_2), \\
C_2(X) &= I(c_2 \leq X < c_3), \\
&\vdots \\
C_{K-1}(X) &= I(c_{K-1} \leq X < c_K), \\
C_K(X) &= I(c_K \leq X),
\end{aligned}
$$

- where I( ) is an indicator function that returns a 1 if the condition is true and returns a 0 otherwise. For example, I(cK ≤ X) equals 1 if cK ≤ X, otherwise it equals 0. For a given value of X, at most only one of C1, C2, . . ., CK can be non-zero, as X can only lie in any one of the bins.