Multiple Linear Regression

Sourav Karmakar

souravkarmakar29@gmail.com

OUTLINE

- Multiple Linear Regression: Hypothesis Function
- Cost Function
- Gradient Descent Algorithm
- Stopping criteria for Gradient Descent
- How to check if Gradient Descent is Working
- Implementation Note
  - Choosing the value of learning rate
  - Feature Scaling

Multiple Linear Regression : Hypothesis

Problem Statement:

Suppose there are $\boldsymbol{k}$ many features or predictor variables $\boldsymbol{x_1, x_2, x_3, \ldots, x_k}$ and a target variable $\boldsymbol{y}$. Now we want to express $\boldsymbol{y}$ as a linear combination of the those predictor variables. We have a vector of features/predictors $\vec{\boldsymbol{x}} = [\boldsymbol{x_1, x_2, x_3, \ldots, x_k}]^T$

Model Representation:

Here, we can express our multiple linear regression problem by means of following probabilistic model.

$$\boldsymbol{y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_k x_k + \varepsilon}$$

Here we have a vector of model parameters $\boldsymbol{\Theta = [\theta_0, \theta_1, \theta_2, \ldots, \theta_k]^T}$ of dimension $k + 1$ and $\boldsymbol{\varepsilon}$ is the random error.

Like in case of simple linear regression, here also we shall use our sample data to estimate model parameter vector $\boldsymbol{\Theta}$ i.e.: $\widehat{\boldsymbol{\Theta}}$.

Multiple Linear Regression : Assumptions

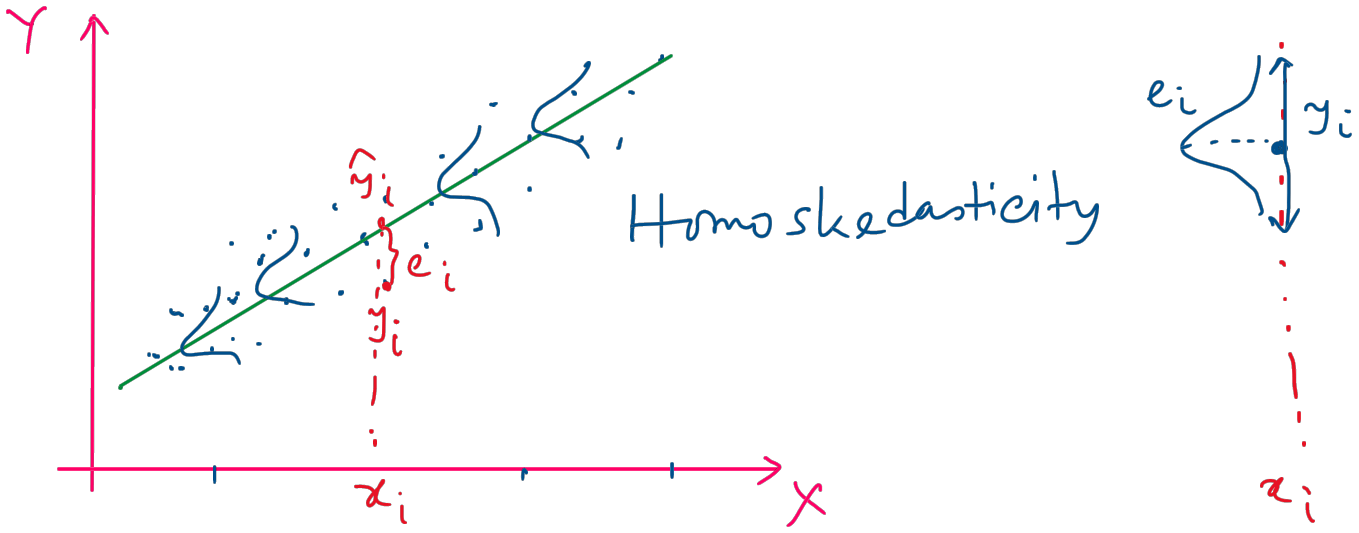
Multiple Linear Regression Assumptions:

- The target variable (y) should be linearly dependent on predictor variables (X).
- The predictors should be uncorrelated and linearly independent of one another.
- The model's error term should have constant variance (Homoscedasticity)
- The error term is preferably normally distributed with zero mean.

Hypothesis Function:

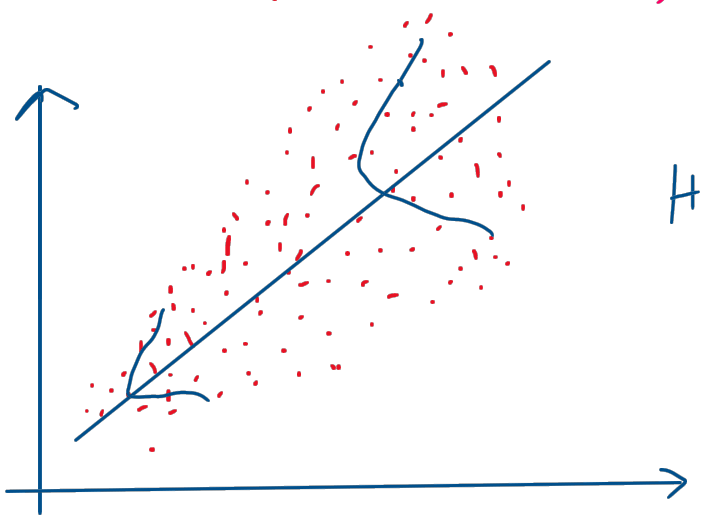
The predicted value of the target variable is obtained from the estimated model parameter as:

$$\hat{y} = \widehat{\theta_0} + \widehat{\theta_1} x_1 + \widehat{\theta_2} x_2 + \cdots + \widehat{\theta_k} x_k = \widehat{\theta_0} + \sum_{j=1}^{k} \widehat{\theta_j} x_j$$



Homoskedasticity

$e_i \sim N(0, \sigma^2)$



Heteroskedasticity

Multiple Linear Regression : Cost Function

Cost Function:

Here, we shall define cost function in the following way:

As, $\hat{y}^{(i)} = \widehat{\theta_0} + \sum_{j=1}^{k}\left(\hat{\theta}_j x_j^{(i)}\right)$

$$J(\hat{\Theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \widehat{\theta_0} - \sum_{j=1}^{k}\left(\hat{\theta}_j x_j^{(i)}\right)\right)^2$$

(Here $\langle \vec{x}^{(i)}, y^{(i)} \rangle$ is the i^{th} data sample and $\hat{y}^{(i)}$ is the corresponding output of the model)

Our Objective:

To estimate the model parameters $\hat{\Theta} = [\widehat{\theta_0}, \widehat{\theta_1}, \widehat{\theta_2}, \dots, \widehat{\theta_k}]^T$ from the given sample of data so that the cost function $J(\hat{\Theta})$ is minimized.

Multiple Linear Regression : Optimization

Gradient Descent algorithm for Cost Function optimization:

- **Gradient Descent** algorithm is an optimization technique for minimizing the cost function $\boldsymbol{J(\hat{\Theta})}$

- **Gradient Descent** is an iterative algorithm and in a nutshell it does the following:
  - Start with some initial values of $\hat{\boldsymbol{\Theta}} = [\widehat{\boldsymbol{\theta_0}}, \widehat{\boldsymbol{\theta_1}}, \widehat{\boldsymbol{\theta_2}}, \dots, \widehat{\boldsymbol{\theta_k}}]^T$
  - Keep Changing $\widehat{\boldsymbol{\theta_0}}, \widehat{\boldsymbol{\theta_1}}, \widehat{\boldsymbol{\theta_2}}, \dots, \widehat{\boldsymbol{\theta_k}}$ to reduce $\boldsymbol{J(\hat{\Theta})}$ until we end up at a minimum.

Gradient Descent Algorithm

- **Algorithm:** Gradient Descent
  - **Input:** Feature set ($X = [\vec{x}^{(i)}]_{i=1}^{m}$) and corresponding values of target variable ($Y = [y^{(i)}]_{i=1}^{m}$).
  - **Output:** Model parameters $\widehat{\boldsymbol{\Theta}} = [\widehat{\boldsymbol{\theta_0}}, \widehat{\boldsymbol{\theta_1}}, \widehat{\boldsymbol{\theta_2}}, \dots, \widehat{\boldsymbol{\theta_k}}]^T$ for which the cost function $\boldsymbol{J}(\widehat{\boldsymbol{\Theta}})$ is minimum.
  - **Initialisation:** Initialise the model parameters $\widehat{\boldsymbol{\Theta}} = [\widehat{\boldsymbol{\theta_0}}, \widehat{\boldsymbol{\theta_1}}, \widehat{\boldsymbol{\theta_2}}, \dots, \widehat{\boldsymbol{\theta_k}}]^T$ with random values.
    iteration number: $t \leftarrow 1$
  - Repeat until convergence {
    - Compute the cost function $\boldsymbol{J}(\widehat{\boldsymbol{\Theta}})$
    - $\widehat{\boldsymbol{\theta_j}}(t+1) = \widehat{\boldsymbol{\theta_j}}(t) - \boldsymbol{\alpha} \frac{\partial}{\partial \widehat{\theta}_j} \left(\boldsymbol{J}(\widehat{\boldsymbol{\Theta}}) \right)$, simultaneously update for every $j = 0,1,2, \dots, k$
    - $t \leftarrow t+1$

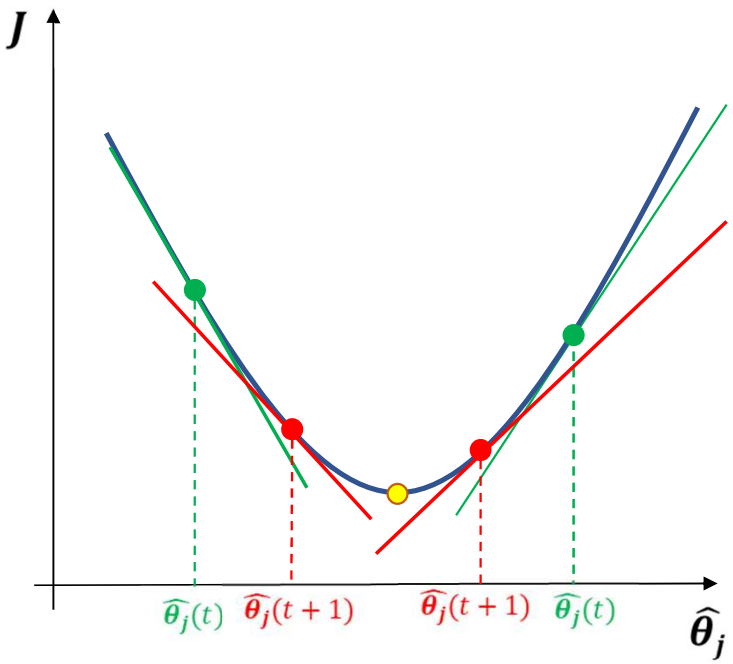
    }

α** is a small positive user defined parameter called the **Learning Rate

Gradient Descent Algorithm : Intuition

Intuition of Gradient Descent algorithm:

- The plot of cost function with any of the parameters (considering the other parameters fixed) is of the following form.



- Now as per gradient descent rule:

$$\widehat{\boldsymbol{\theta_j}}(t+1) = \widehat{\boldsymbol{\theta_j}}(t) - \boldsymbol{\alpha}\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\widehat{\boldsymbol{\theta}}_{\boldsymbol{j}}}\left(\boldsymbol{J}\left(\widehat{\boldsymbol{\Theta}}\right)\right)$$

- Now if the gradient $\frac{\partial J}{\partial \widehat{\theta}_j}$ is *positive*, then the update in the value of $\widehat{\boldsymbol{\theta}_j}$ which is calculated as $\widehat{\boldsymbol{\theta}_j}(t+1) - \widehat{\boldsymbol{\theta}_j}(t)$ is *negative*. Indicating that, in the next update $\widehat{\boldsymbol{\theta}_j}$ would be smaller than its previous value.
- On the other hand if the gradient $\frac{\partial J}{\partial \widehat{\theta}_j}$ is *negative*, then the update in the value of $\widehat{\boldsymbol{\theta}_j}$ is *positive*. Indicating that, in the next update $\widehat{\boldsymbol{\theta}_j}$ would be greater than its previous value.

Gradient of Cost Function

- The Cost Function for Linear Regression with multiple predictor variable is:

$$J(\hat{\Theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \widehat{\theta_0} - \sum_{j=1}^{k}\left(\hat{\theta}_j x_j{}^{(i)}\right)\right)^2$$

- We can calculate the derivative of the cost function with respect to $\widehat{\theta_0}$ and $\hat{\theta}_j$ ($for\ j = 1,2,3,\dots,k$) as following:

$$\frac{\partial}{\partial\widehat{\theta_0}}\left(J(\hat{\Theta})\right) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \widehat{\theta_0} - \sum_{j=1}^{k}\left(\hat{\theta}_j x_j{}^{(i)}\right)\right)(-1) = \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{\theta_0} + \sum_{j=1}^{k}\left(\hat{\theta}_j x_j{}^{(i)}\right) - y^{(i)}\right)$$

$$\frac{\partial}{\partial\hat{\theta}_j}\left(J(\hat{\Theta})\right) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \widehat{\theta_0} - \sum_{j=1}^{k}\left(\hat{\theta}_j x_j{}^{(i)}\right)\right)\left(-x_j{}^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{\theta_0} + \sum_{j=1}^{k}\left(\hat{\theta}_j x_j{}^{(i)}\right) - y^{(i)}\right)x_j{}^{(i)}$$

$$for\ j = 1,2,3,\dots,k$$

Gradient Descent update rule

- Hence, the update rule in Gradient descent algorithm would be (for t^{th} iteration):

$$\widehat{\theta_0}(t+1) := \widehat{\theta_0}(t) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(\overbrace{\widehat{\theta_0}(t) + \sum_{j=1}^{k}\left(\widehat{\theta_j}(t)\, x_j^{(i)}\right)}^{\widehat{y}^{(i)}(t)} - y^{(i)}\right)$$

$$\textbf{and} \qquad \widehat{\theta_j}(t+1) := \widehat{\theta_j}(t) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(\overbrace{\widehat{\theta_0}(t) + \sum_{j=1}^{k}\left(\widehat{\theta_j}(t)\, x_j^{(i)}\right)}^{\widehat{y}^{(i)}(t)} - y^{(i)}\right) x_j^{(i)}$$

$$for\ j = 1,2,3,\dots,k$$

- **Note:**

If we consider another predictor x_0 whose value is always 1 (i.e. $x_0^{(i)} = 1, \forall i$), then we can use the last equation as a generalized gradient descent update rule.

Gradient Descent update rule

- Training Data Set can be written as following:

Sl. No.	x_0	x_1	⋯	x_k	y
1	1	$x_1^{(1)}$	⋯	$x_k^{(1)}$	$y^{(1)}$
2	1	$x_1^{(2)}$	⋯	$x_k^{(2)}$	$y^{(2)}$
⋮	⋮	⋮	⋯	⋮	⋮
m	1	$x_1^{(m)}$	⋯	$x_k^{(m)}$	$y^{(m)}$

- There are total k many features and m many training examples.
- Notice that we have added one extra feature column $\boldsymbol{x_0}$ with all values 1 in the left.

- Hence, the Gradient Descent update rule for the parameters of linear regression (for t^{th} iteration) is:

$$\widehat{\boldsymbol{\theta}_j}(\boldsymbol{t+1}) := \widehat{\boldsymbol{\theta}_j}(\boldsymbol{t}) - \frac{\boldsymbol{\alpha}}{\boldsymbol{m}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \left(\hat{\boldsymbol{y}}^{(\boldsymbol{i})}(\boldsymbol{t}) - \boldsymbol{y}^{(\boldsymbol{i})}\right) \boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{i})}$$

$$for\ j = 0, 1, 2, 3, \dots, k\ and\ x_0^{(i)} = 1, \forall i \in [m]$$

Gradient Descent : Stopping Criteria

- **Stopping criteria for gradient descent:**

  We can consider following two methods as stopping criteria for the gradient descent algorithm.

  1. Convergence of the model parameters:

     If in two successive iterations the change in the model parameters is *not significant* then we can say that the algorithm has converged. As the model parameters are vectors, to compare the values of the model parameters in two successive iterations we can either take norm or absolute differences. Alternatively if at least one of the parameters is not converged then we keep on running the algorithm until all the parameters are converged.

  2. Predefined number of iterations:

     We run the algorithm for predefined number of iterations.

Convergence of Model Parameters:-

Norm of a vector $\vec{V} = [v_1, v_2, v_3, \ldots, v_k]^T$

$$\|\vec{V}\| = \sqrt{v_1^2 + v_2^2 + v_3^2 + \cdots + v_k^2}$$

$\hat{\Theta} = [\hat{\theta}_0, \hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_k]^T \longrightarrow$ Estimated model parameter vector.

$\hat{\Theta}(t+1) \longrightarrow$ Estimated model parameters after $(t+1)^{th}$ iteration of grad. desc.

$\hat{\Theta}(t) \rightarrow$ Estimated model parameters after t^{th} iteration of gradient desc.

$$\Delta\hat{\Theta}(t) = \hat{\Theta}(t+1) - \hat{\Theta}(t)$$

$\Delta\hat{\Theta}(t) \rightarrow$ Change in the model parameters vector.

$$\Delta\hat{\Theta} = [\Delta\theta_0, \Delta\theta_1, \Delta\theta_2, \ldots, \Delta\theta_k]$$

Method-1

To check the convergence :—

tolerance = 0.0001

$$\left\| \Delta \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_k \end{pmatrix} \right\| = \sqrt{\Delta\theta_0^2 + \Delta\theta_1^2 + \Delta\theta_2^2 + \cdots + \Delta\theta_k^2} < \text{tolerance}$$

Convergence has achieved.

Method-2

$\Delta\theta_0, \Delta\theta_1, \Delta\theta_2, \ldots, \Delta\theta_k$

$|\Delta\theta_0| < tol$

$|\Delta\theta_1| < tol$

$|\Delta\theta_2| < tol$

$\vdots$

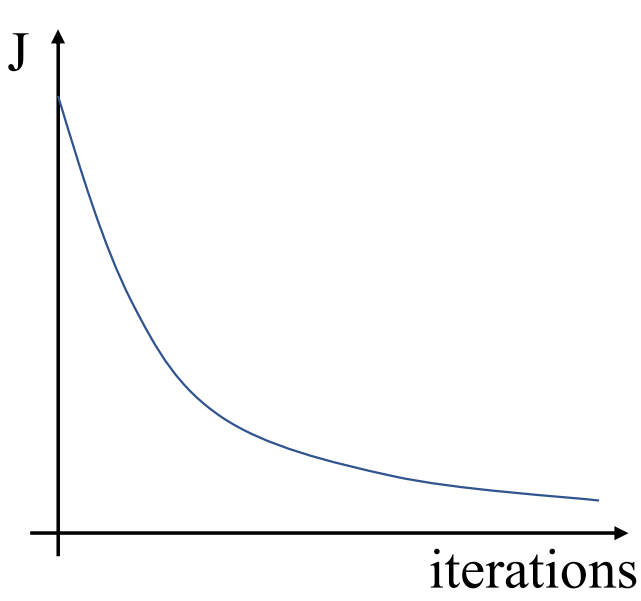
$|\Delta\theta_k| < tol$

Convergence is achieved if all of them are true, if any one of them is false then we continue grad. desc.

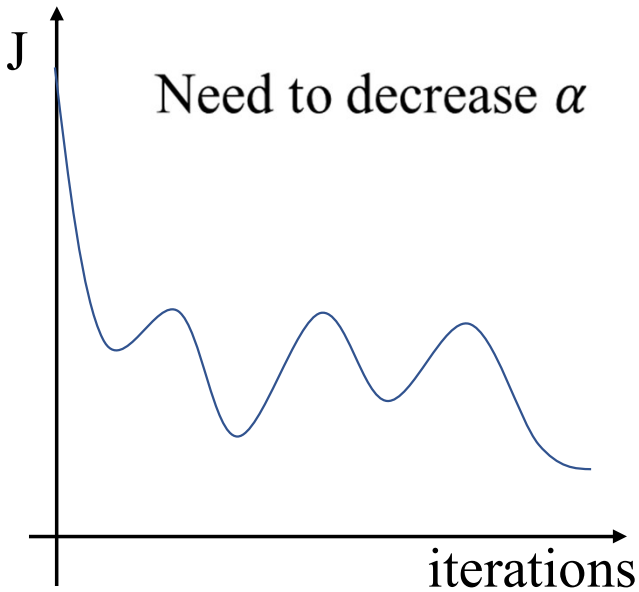
Gradient Descent : Troubleshooting

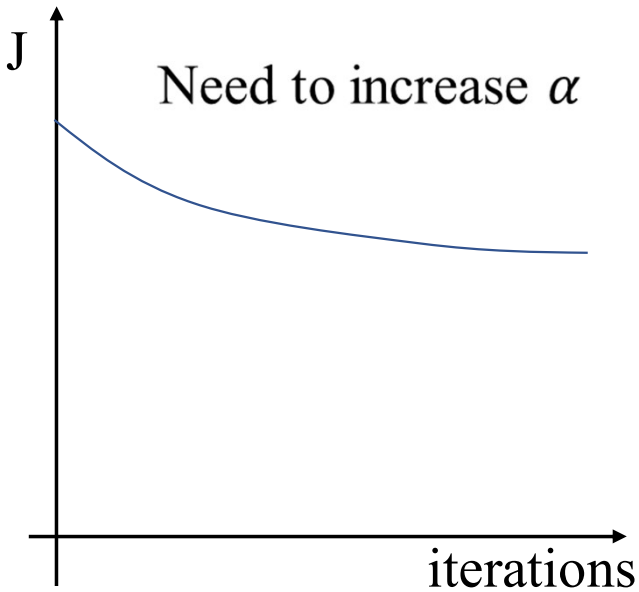
- **How to check whether gradient descent is working or not:**

  We can plot cost function versus number of iterations. If it's continuously decreasing then we can say that gradient descent is working. However we may have to fine-tune the value of "learning rate" (α) depending upon how fast or slow the cost function is converging.



Gradient Descent working properly





Gradient Descent not working properly

Gradient Descent : Implementation Note

- **Implementation Note-1: Choosing the value of Learning Rate α**

  Higher value of learning rate $\boldsymbol{\alpha}$, may cause Gradient Descent algorithm to oscillate around the minimum point. Too low value of $\boldsymbol{\alpha}$ will make Gradient Descent algorithm sluggish (i.e. too slow to converge). Hence, choosing a right value of $\boldsymbol{\alpha}$ is important. There is no general *apply-to-all* value of learning rate.

  As a rule of thumb to choose $\boldsymbol{\alpha}$ people tries different values like:
  $\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, \ldots$
  Again this is not a exhaustive list of values of learning rate. One may choose different values like 0.005 as learning rate $\boldsymbol{\alpha}$.

Gradient Descent : Implementation Note

- **Implementation Note-2: Feature Scaling:**

  Linear Regression and most of the other Machine Learning algorithms performs well when the range of the predictor variables are roughly in the same range. This is because, if the range of one predictor variable is significantly higher than another predictor variable, then the first one starts to dominate over the second one, making Gradient Descent algorithm sluggish. Hence, Feature Scaling is necessary for faster convergence of Gradient Descent algorithm. This done by either of the following two methods.

  1. Standardization: Consider a predictor variable/ feature $\boldsymbol{x_j}$ whose mean is $\bar{\boldsymbol{x}}_{\boldsymbol{j}}$ and standard deviation is $\boldsymbol{\sigma_{x_j}}$, then for feature scaling replace $\boldsymbol{x_j}$ with $\frac{x_j - \bar{x}_j}{\sigma_{x_j}}$, i. e. $\boldsymbol{x_j} := \frac{x_j - \bar{x}_j}{\sigma_{x_j}}$, This ensures that after scaling $\boldsymbol{x_j}$ will have zero mean and unit standard deviation.

  2. Min-Max Scaling: $\boldsymbol{x_j} := \frac{x_j - x_j^{min}}{x_j^{max} - x_j^{min}}$, This ensures that after scaling x_j will be between 0 to 1

Thank You