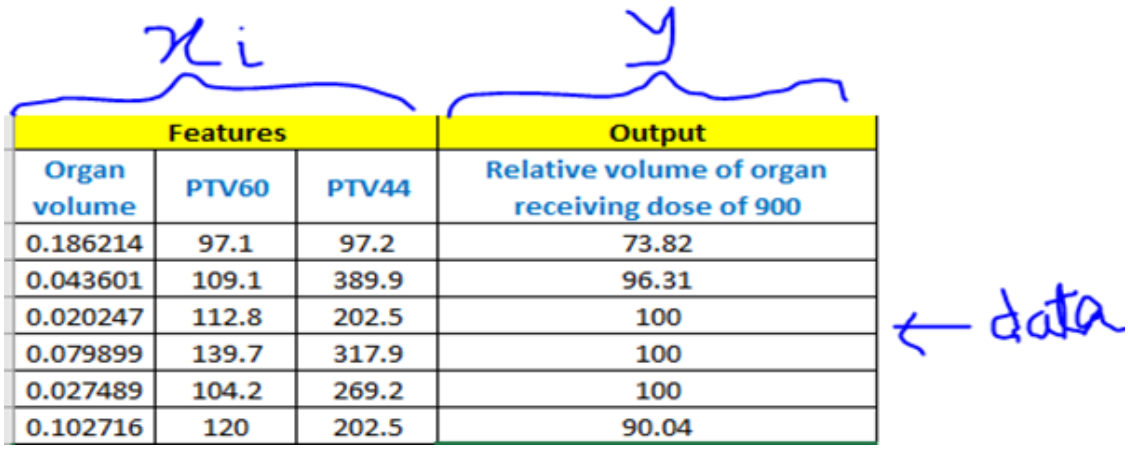
Linear Regression With Multiple Features And Gradient Descent Optimization :

In this project we are told to use gradient descent optimization method on dataset involving one response (i.e., relative volume of organ receiving particular dose value, or y) and multiple variable (i.e., volume of organ, PTV receiving 60Gy radiation, and PTV receiving 44Gy radiation or xi).

We are using a popular machine learning technique known as Multiple Linear Regression.
This algorithm is used for predictive analysis. In classical Statistics and Machine Learning, Regression means a technique to model the relationship between a dependent variable and a given set of independent variables. It can be divided into

1. Simple linear Regression
2. Multiple Linear Regression

Multiple Linear Regression attempts to model the relationship between two or more features and a response by fitting a linear equation to observed data.



There is only one true model which fits our data perfectly. But unfortunately, we don't know that model. So, we have to predict that model using our Machine Learning Algorithm. we make a hypothesis, tuning it until we can check if it is close enough to the true model or it is a wrong one.

Hypothesis Function

In Multivariate Linear Regression, the hypothesis function with multiple variables x and parameters θ is denoted below.

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \quad (1)$$
$$Parameters: \ \theta = \{\theta_0, \theta_1, \theta_2 ..... \theta_n\}$$
$$Features: x = \{x_0, x_1, x_2 .... x_n\}$$

We assume $x_0 = 1$ for convenience of notation.

Simply,

relative volume of organ receiving particular dose value $(y) = f$ (volume of organ(x_1), PTV receiving 60Gy radiation(x_2), PTV receiving 44Gy radiation(x_3))

Notation:

$$n = number\ of\ features$$
$$x^i = input\ (features)\ of\ i^{th} training\ example.$$
$$x_j^i = values\ of\ features\ j\ in\ i^{ih} training\ example.$$

When tuning the hypothesis, our model learns parameters θ which makes hypothesis function a 'good' predictor. A Good predictor means the hypothesis is closed enough to the true model. Here comes the concept of Cost function.

Cost Function:

We can measure the accuracy of our model using cost function. We define cost function using mean square error (MSE).

$$Hypothesis: h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \quad (2)$$
$$Parameters: \theta_0, \theta_1, ..., \theta_n$$

$$Cost\ Function: J(\theta_0, \theta_1, ..., \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2 \quad (3)$$

We square the difference between hypothesis and prediction to make the error positive. This Cost function should be differentiable so that we can apply Gradient Descent Algorithm to it.

Essentially, the cost function $J(\theta)$ is the sum of the square error of each data. The larger the error, the worser the performance of the hypothesis. Therefore, we want to minimize the error, that is, minimize the $J(\theta)$.

Gradient Descent Algorithm:

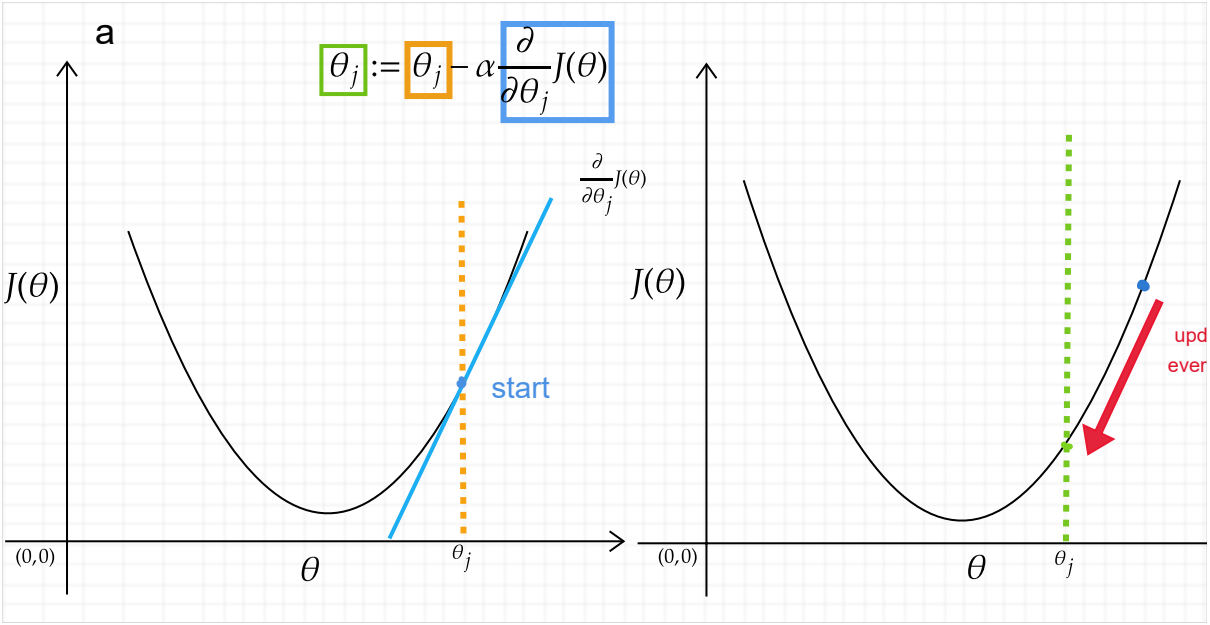
Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function. Gradient descent is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible.

We use this algorithm to estimate the parameters value in such a way so that it minimizes the cost function. When we found the minimum error, our model learns the best value of parametrs. Thus, we can predict more accurate values for new input data.

At each iteration, the parameters needed to be updated simultaneously.

$$Repeat\ until\ converge\ \{$$
$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \quad (4)$$
$$\}$$
$$where\ j\ represents\ the\ feature\ index\ number.$$

In figure (a), the starting point is at orange $(\theta_j, J(\theta_j))$. Calculate its partial differentiation, then multiplied it by a learning rate α and the updated result is at green $(\theta_j, J(\theta_j))$, as figure (b) shows.



Gradient Descent :

Previously(n=1):

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right) \tag{5}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right) . x^i \tag{6}$$

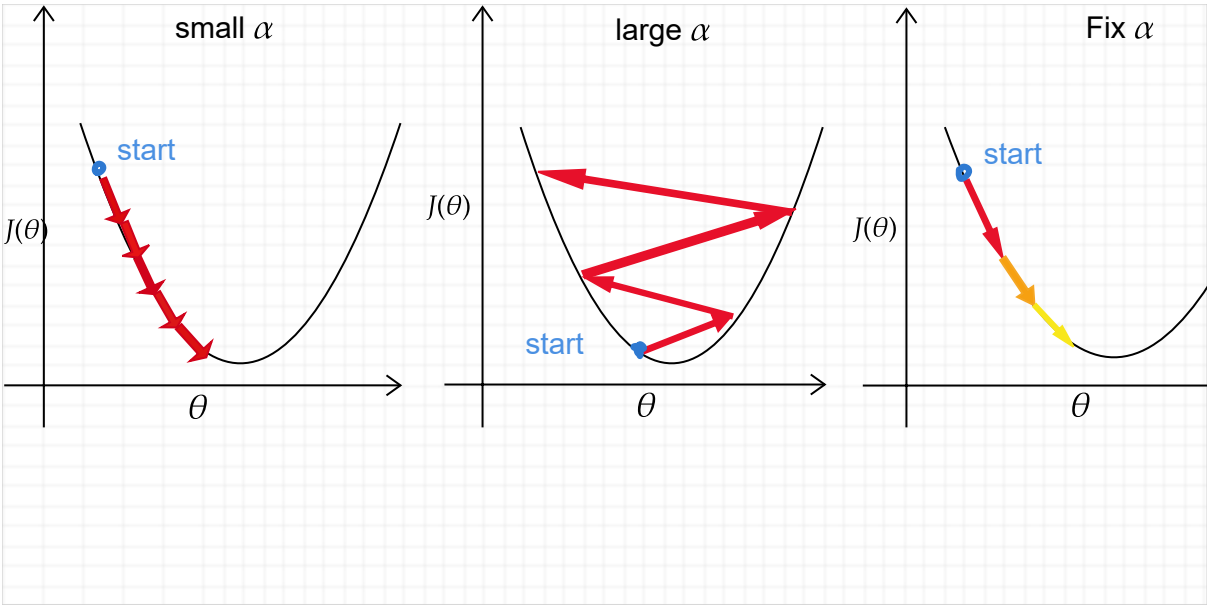
$(simulteneously\ update\ \theta_0, \theta_1)$

}

Learning rate α:

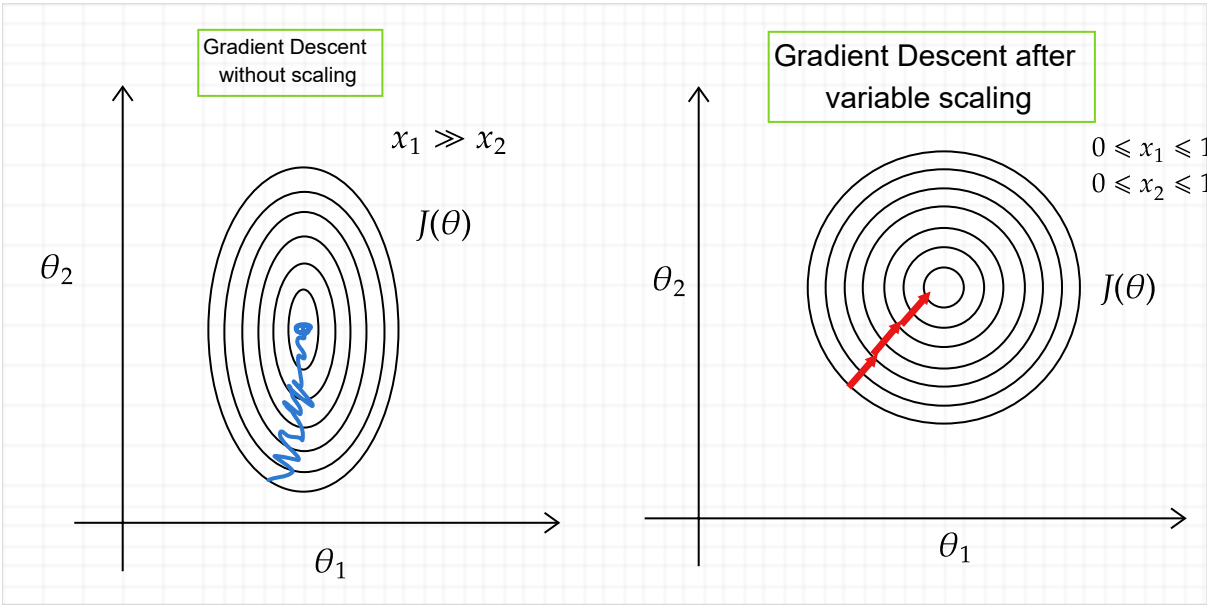
We use learning rate α to control how much we update at one iteration. If α is too small, it makes the gradient descent update too slow, whereas the update may overshoot the minimum and won't converge.

Note that we set a fixed learning rate α in the beginning since the gradient descent will update slowly and automatically until it reaches the minimum. Hence, there is no need to change the learning rate α at each iteration by ourselves.



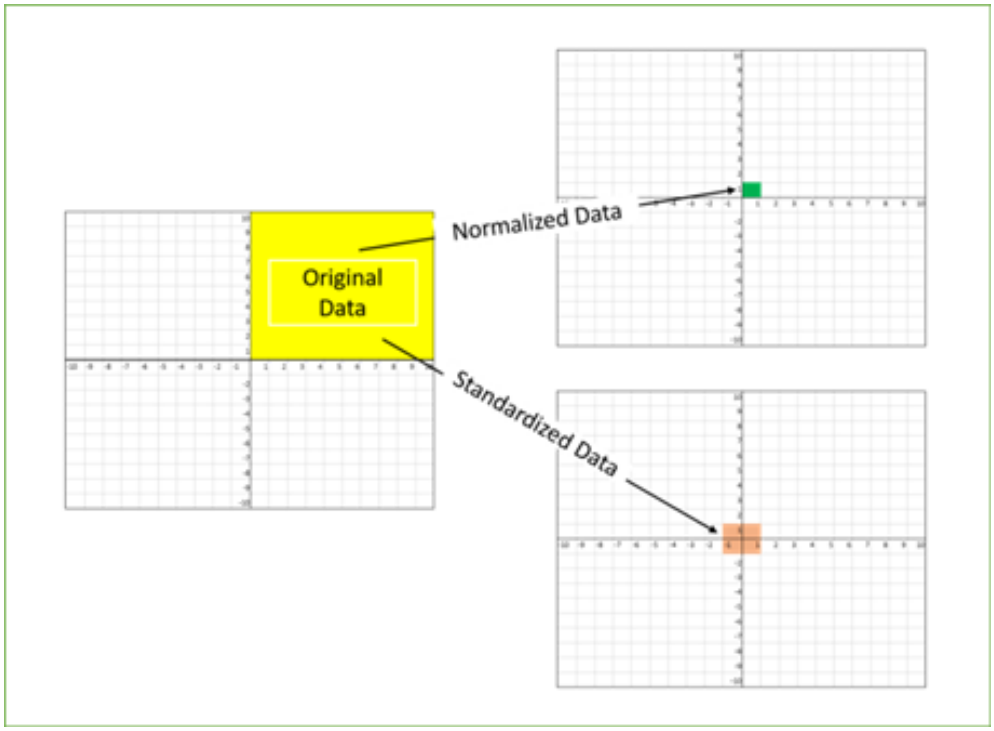
Feature Scaling:

Feature scaling is a method used to normalize the range of independent variables or features of data.



The most common techniques of feature scaling are Normalization and Standardization.

Normalization is used when we want to bound our values between two numbers, typically, between [0,1] or [-1,1]. While Standardization transforms the data to have zero mean and a variance of 1, they make our data unitless. Refer to the below diagram, which shows how data looks after scaling in the X-Y plane.



We can speed up gradient descent by scaling because θ descends quickly on small ranges and slowly on large ranges, and oscillates inefficiently down to the optimum when the variables are very uneven.

Also, we can use Mean Normalization in Feature scaling.

$$x' = \frac{x - average(x)}{max(x) - min(x)}$$

where x is an original value, x′ is the normalized value.

Here, we will implement gradient descent optimization method to fit the dataset using linear regression model $(y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3)$
, wherein $\theta_0, \theta_1, \theta_2$ and θ_3 need to be optimized to get a good fitting function for the given dataset. Hence, we define a predictive function $(h_\theta(x))$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \tag{7}$$

$$or, h(\theta) = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \theta_3 \end{bmatrix} \times \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{8}$$

Now, we will provide a guess value of $\theta_{old} = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \theta_3 \end{bmatrix}_{old}$ and perform an iteration till objective function minimizes to minimum value. Let us define an objective function as

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^i) - y^i\right)^2 \tag{9}$$

To update the new value of $\theta_{new} = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \theta_3 \end{bmatrix}_{new}$, we will implement following steps

$$\begin{bmatrix} \theta_{new} \end{bmatrix} = \begin{bmatrix} \theta_{old} \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \\ \frac{\partial J}{\partial \theta_3} \end{bmatrix} \tag{10}$$

where, is the correction factor or learning rate

$$or, \begin{bmatrix} \theta_{0,new} \\ \theta_{1,new} \\ \theta_{2,new} \\ \theta_{3,new} \end{bmatrix} = \begin{bmatrix} \theta_{0,old} - \alpha\frac{\partial J(\theta)}{\partial \theta_0} \\ \theta_{1,old} - \alpha\frac{\partial J(\theta)}{\partial \theta_1} \\ \theta_{2,old} - \alpha\frac{\partial J(\theta)}{\partial \theta_2} \\ \theta_{3,\,old} - \alpha\frac{\partial J(\theta)}{\partial \theta_3} \end{bmatrix} \tag{11}$$

$$or, \begin{bmatrix} \theta_{0,new} \\ \theta_{1,new} \\ \theta_{2,new} \\ \theta_{3,new} \end{bmatrix} = \begin{bmatrix} \theta_{0,old} - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^i) - y^i\right).1 \\ \theta_{1,old} - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^i) - y^i\right).x_1 \\ \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^i) - y^i\right).x_2 \\ \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^i) - y^i\right).x_3 \end{bmatrix} \tag{12}$$

Next, $\begin{bmatrix} \theta_{old} \end{bmatrix} = \begin{bmatrix} \theta_{new} \end{bmatrix}$ and above iteration will continue till the termination condition reached (i.e., no of iteration).