Regression

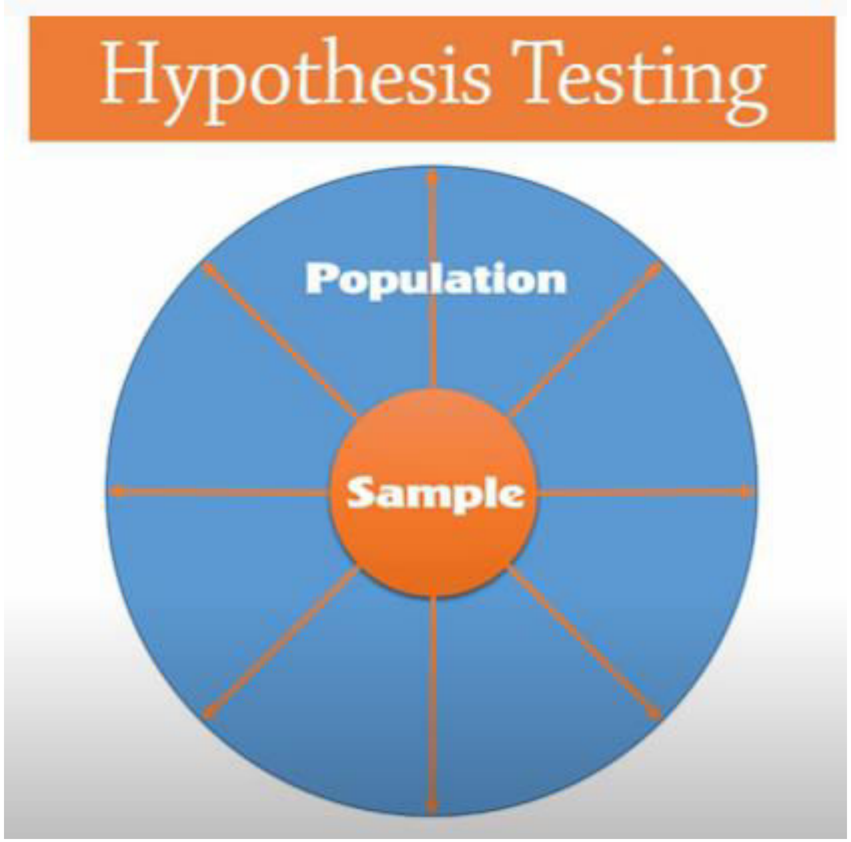
By

Suchita Patil

hypothesis

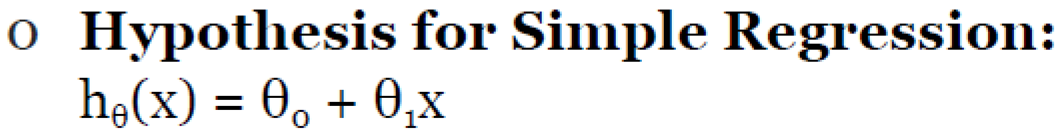
- Is a claim or statement about the population parameter

- Average age of all college students in the city is 23.



Simple regression

- cost functions :



- **Hypothesis for Simple Regression:**

  $h_\theta(x) = \theta_o + \theta_1 x$

- In ML, cost functions are used to estimate how badly models are performing. Put simply, ***a cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and y.***

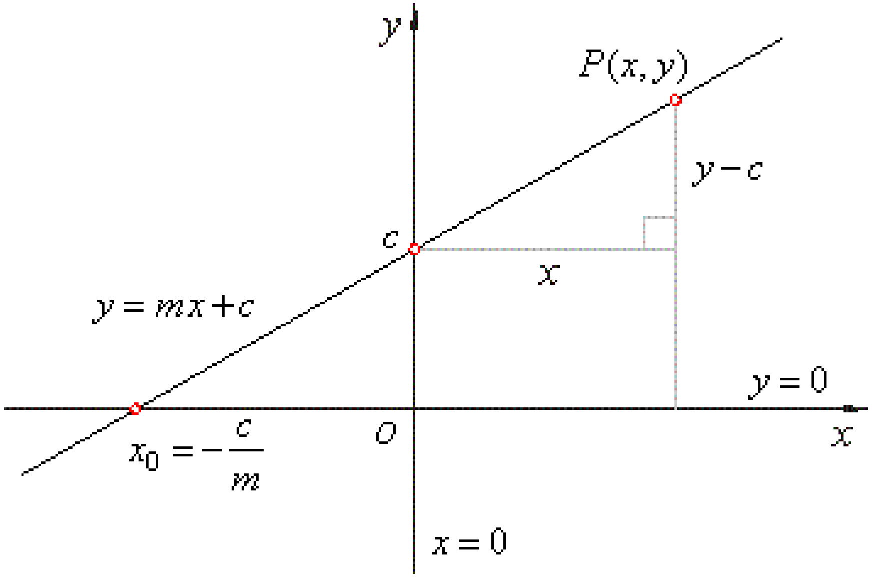
Linear Regression

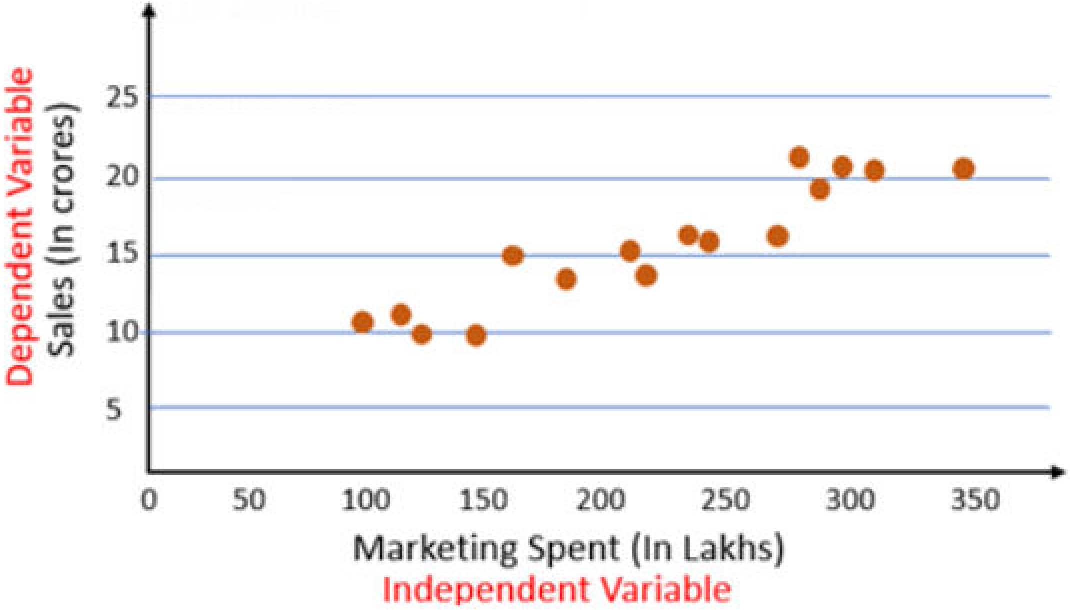
- In statistics, linear regression is a linear approach to modeling the relationship between a dependent variable and one or more independent variables.
- *Where,*
- *y: dependent variable*
- *x: independent variable*
- *m: Slope of the line*

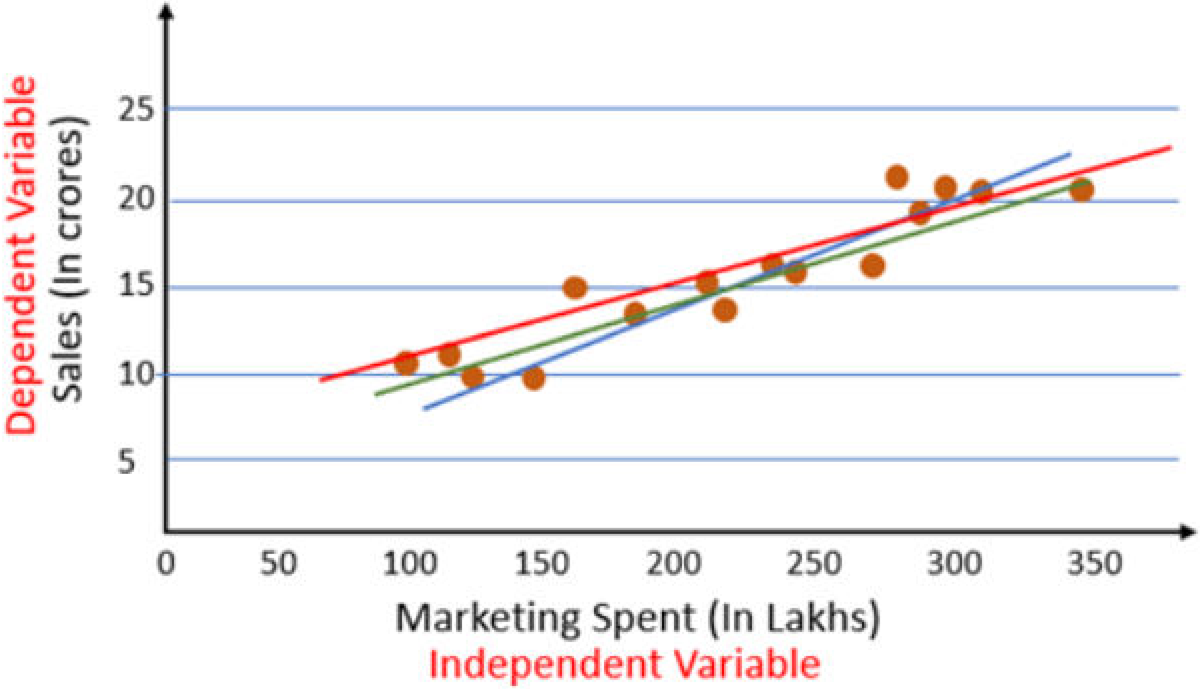
(For a unit increase in the Quantity of X, Y increases by m.1 = m units.)

- *c: y intercept (The value of Y is c when the value of X is 0)*

$$Y = mX + c$$

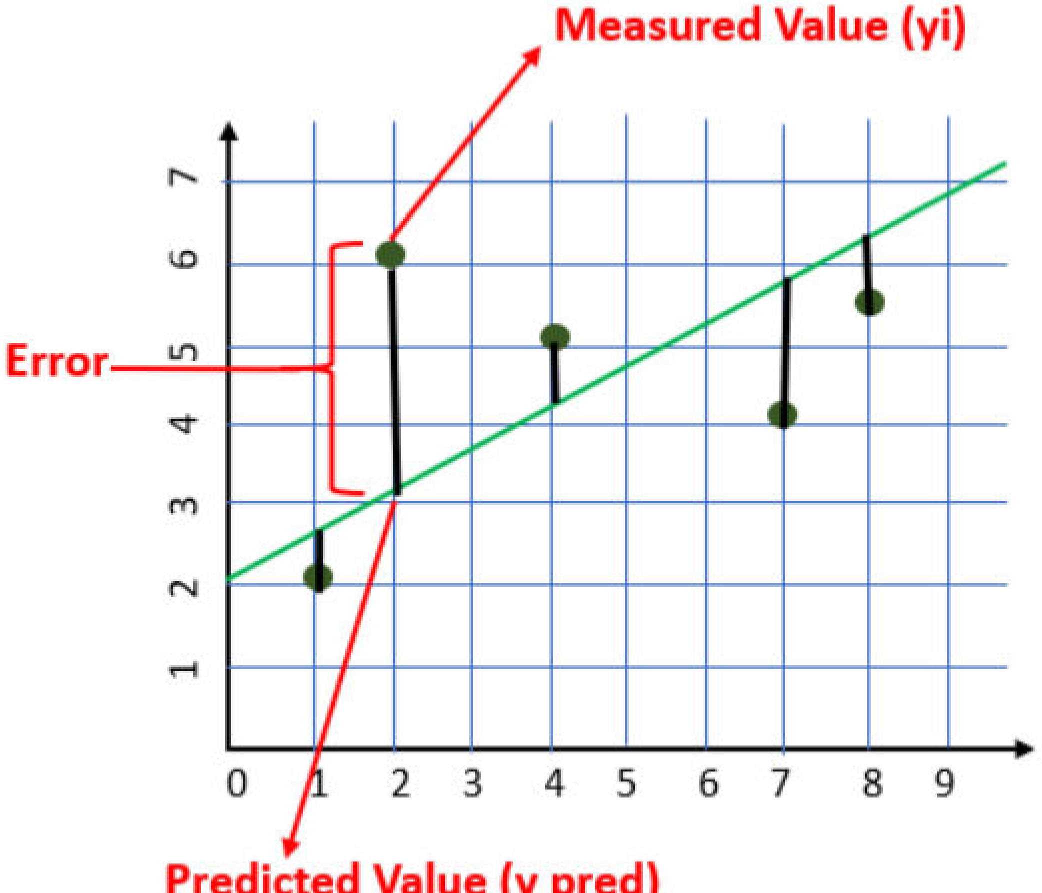



Dependent Variable
Sales (In crores)
25
20
15
10
5
0
50
100
150
200
250
300
350
Marketing Spent (In Lakhs)
Independent Variable


Dependent Variable
Sales (In crores)
25
20
15
10
5
0
50
100
150
200
250
300
350
Marketing Spent (In Lakhs)
Independent Variable

Cost Function

- The cost is the error in our predicted value. We will use the Mean Squared Error function to calculate the



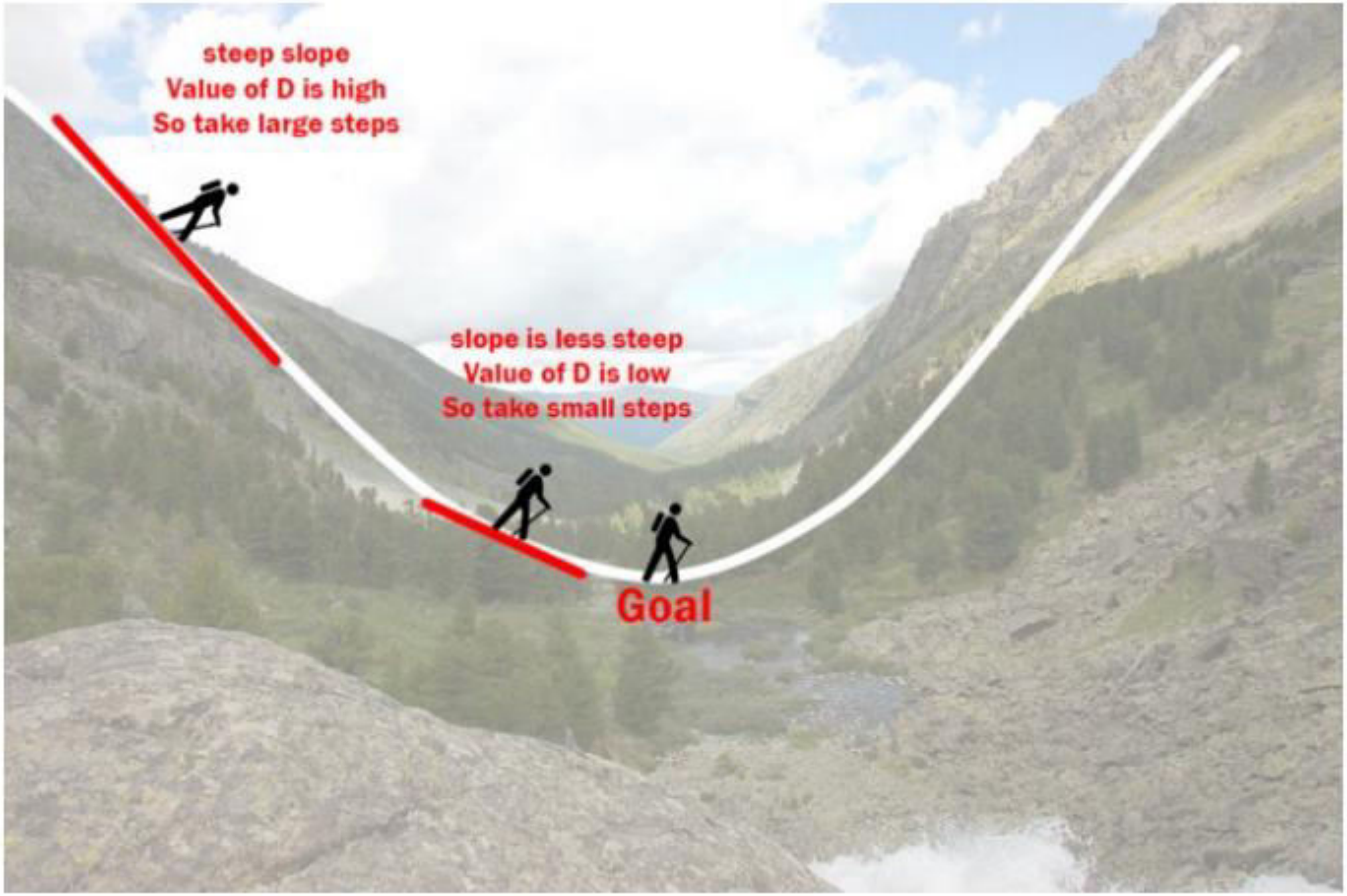
$$Cost\ Function(MSE) = \frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})^2$$

Replace $y_{i\,pred}$ with $mx_i + c$

$$Cost\ Function(MSE) = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$$

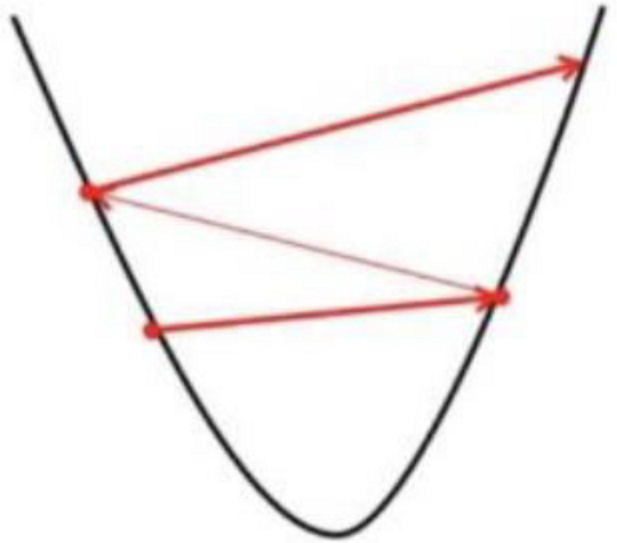
parameter learning with gradient descent

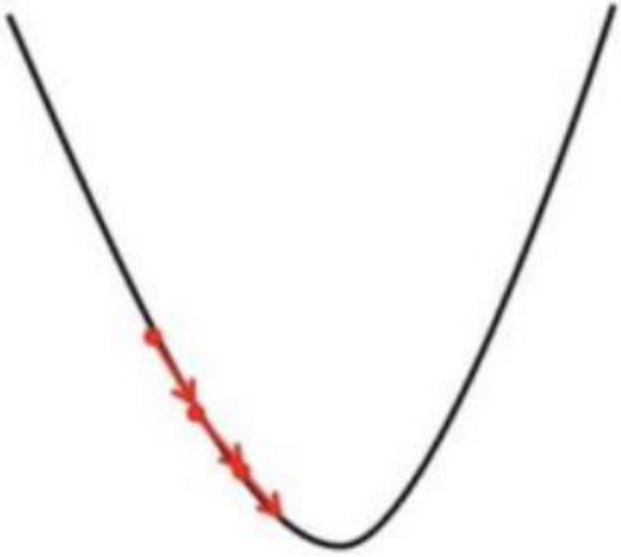
- Gradient descent is an iterative optimization algorithm to find the minimum of a function



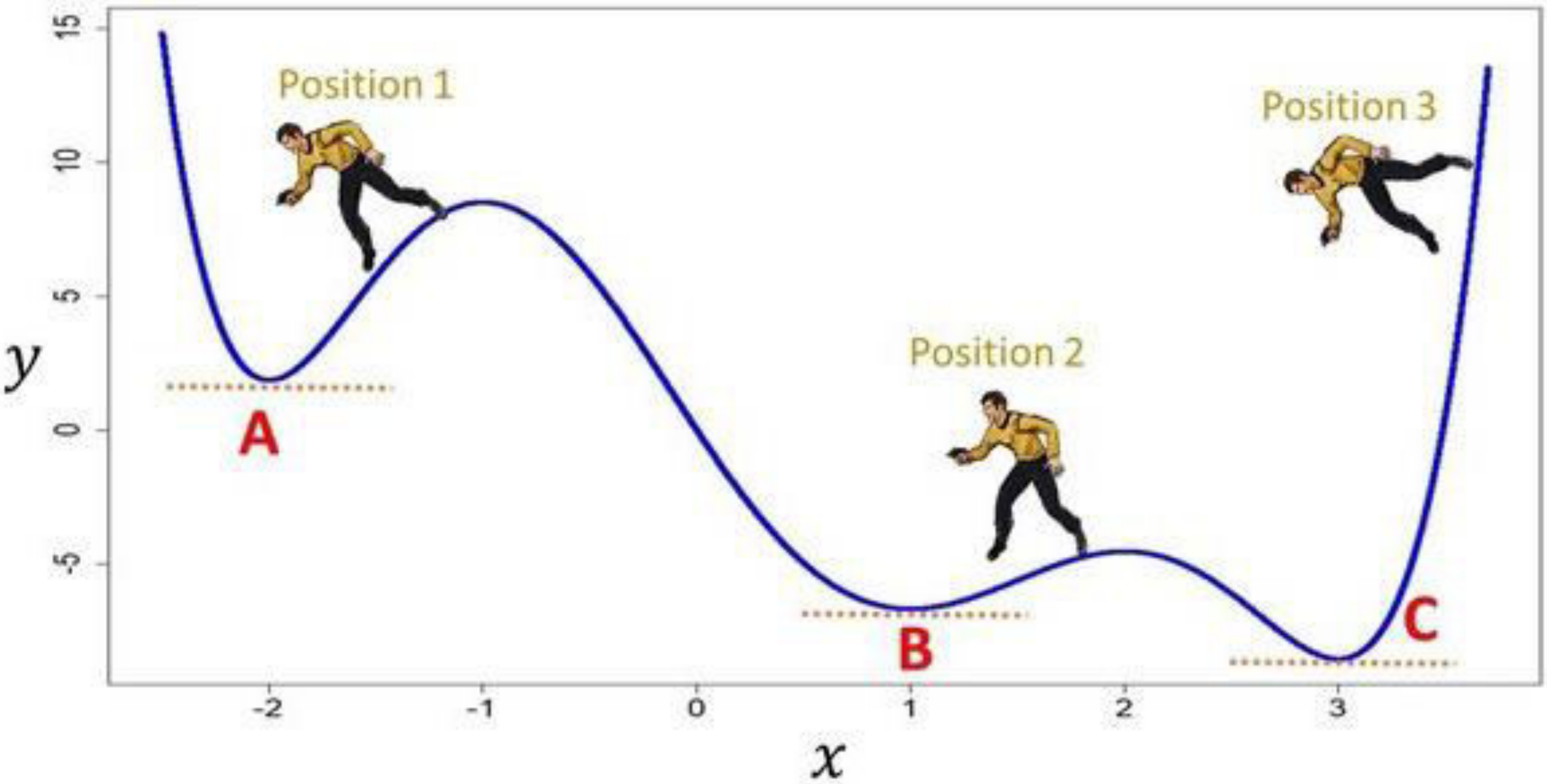
learning rate

- The choice of correct learning rate is very important as it ensures that Gradient Descent converges in a reasonable time. :
- If we choose learning rate **to be very large**, Gradient Descent can overshoot the minimum. It may fail to converge or even diverge.
- If we choose learning rate to be very small, Gradient Descent will take small steps to reach local minima and will take a longer time to reach minima.





Direction



- 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.

Gradient Descent for linear regression:

- We have $J(\theta_o, \theta_1)$
- We want to get **min $J(\theta_o, \theta_1)$**
- Gradient descent is used all over machine learning for minimizatio

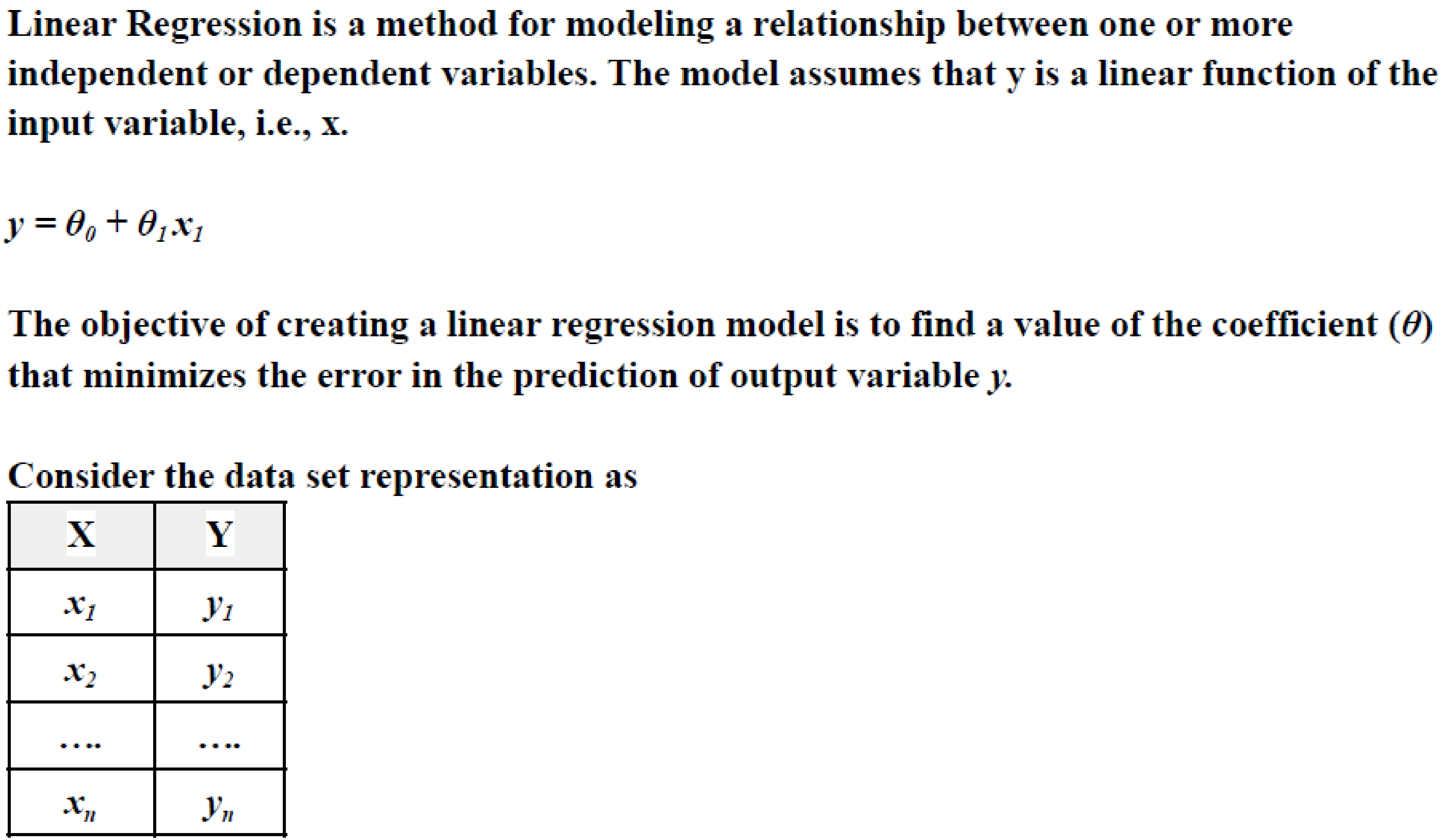
Do until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

Where, α (alpha)

- Is a number called the **learning rate**

simple regression in matrix form



Linear Regression is a method for modeling a relationship between one or more independent or dependent variables. The model assumes that y is a linear function of the input variable, i.e., x.

$y = \theta_0 + \theta_1 x_1$

The objective of creating a linear regression model is to find a value of the coefficient (*θ*) that minimizes the error in the prediction of output variable *y*.

Consider the data set representation as

X	Y
x_1	y_1
x_2	y_2
….	….
x_n	y_n

$$\begin{array}{ccc} \mathbf{Y} & = & \mathbf{X} \qquad \theta_0 \\ \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ y_n \end{bmatrix} & = & \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ . & . \\ . & . \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \end{array}$$

Matrix formation of Linear Regression (LR) is given as

Y = Xθ

Where

X is an input data, and each column is a data feature

θ is a vector of coefficients

Y is a vector of output variables for each row in x

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} \theta_0 + \theta_1 x_1 \\ \theta_0 + \theta_1 x_2 \\ \theta_0 + \theta_1 x_3 \\ . \\ . \\ \theta_0 + \theta_1 x_n \end{bmatrix}$$

Our initial equation -

Y = Xθ

Multiply both sides with X$^{\mathrm{T}}$

$$X^T Y = X^T X\theta \qquad \text{............}\textit{equation (1)}$$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & \dots & n \\ x_1 & x_2 & x_3 & \dots & x_n \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ . & . \\ . & . \\ 1 & x_n \end{bmatrix} = \begin{bmatrix} n & \Sigma(x_i) \\ \Sigma(x_i) & \Sigma(x_i)^2 \end{bmatrix}$$

$$X^T Y = \begin{bmatrix} 1 & 1 & 1 & \dots & n \\ x_1 & x_2 & x_3 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} \Sigma(y_i) \\ \Sigma(x_i\, y_i) \end{bmatrix}$$

From equation (1)

$X^TY = X^TX\theta$

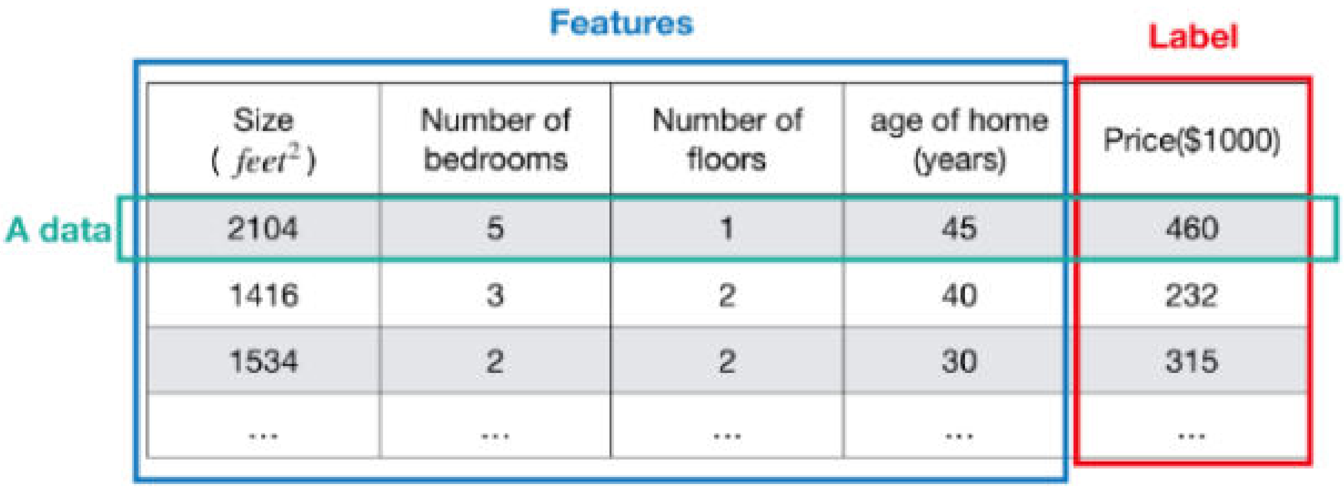
$\theta = (X^TX)^{-1} X^TY$

$$\begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} n & \Sigma(x_i) \\ \Sigma(x_i) & \Sigma(x_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} \Sigma(y_i) \\ \Sigma(x_i y_i) \end{bmatrix}$$

Example: Calculate coefficient (θ) for given data points (1,1) (2,3) (3,3) (4,5). Find the equation of a line. Also find MSE,MAE,RMSE

Multivariate Linear Regression

1. Linear Regression with multiple features is called Multivariate Linear Regression.
2. If we have a continuous labeled data with more than one feature, we can use the multivariate linear regression to make a machine learning model.
3. The label is the answer to a data and the value is continuous (i.e. real numbers). With below housing price example, the label is price and the features are size, number of bedrooms, number of floors and age.


Features
Label
A data

Size ($feet^2$)	Number of bedrooms	Number of floors	age of home (years)	Price($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	2	2	30	315
...	...	...	...	...

Hypothesis function:

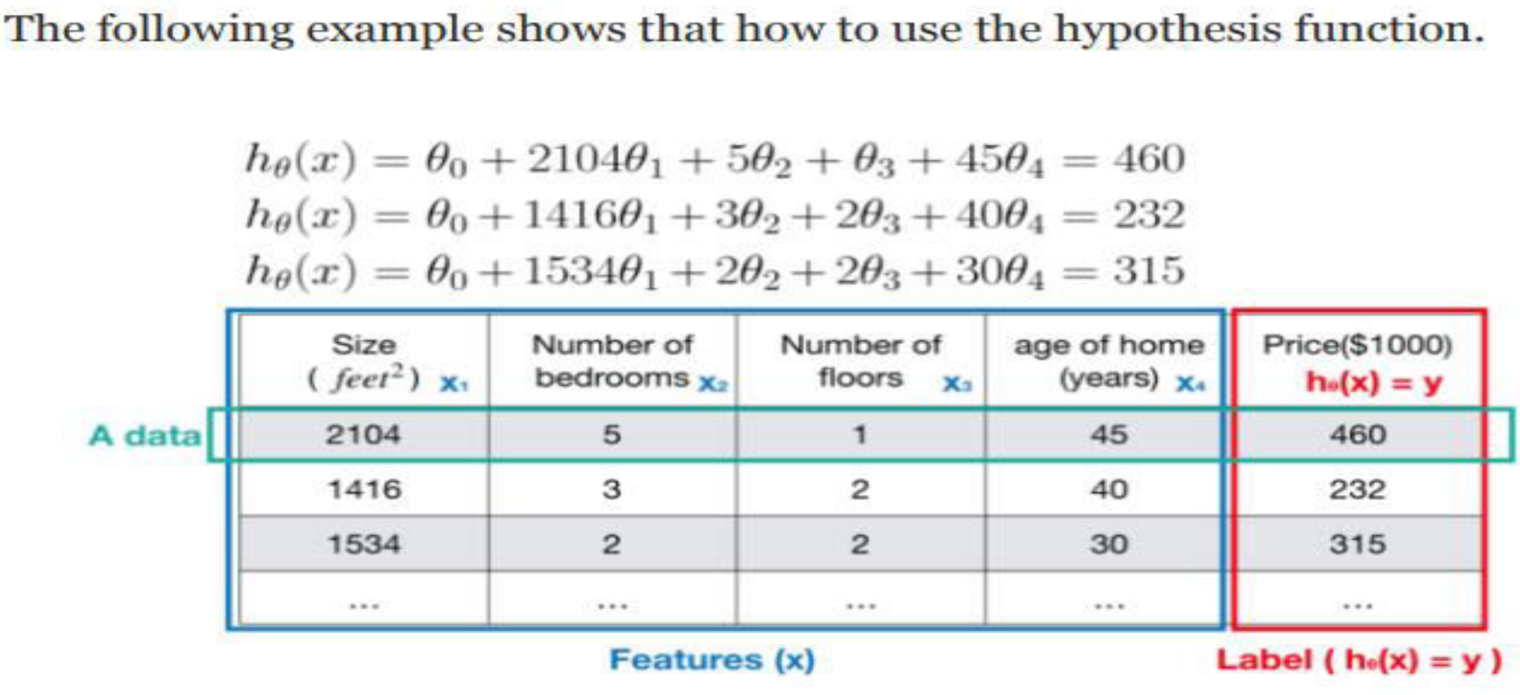
The hypothesis in case of univariate linear regression was,

$$h_\theta(x) = \theta_0 + \theta_1\, x$$

Extending the above function to multiple features, hypothesis of multivariate linear regression is given by,

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1\, x_1 + \theta_2\, x_2 + \cdots + \theta_n\, x_n \\ &= \theta_0\, x_0 + \theta_1\, x_1 + \theta_2\, x_2 + \cdots + \theta_n\, x_n, \text{ where } x_0 = 1 \\ &= \theta^T x, \text{ vectorizing above equation} \end{aligned}$$

- Where,
  - $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$, and $x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$



The following example shows that how to use the hypothesis function.

$$h_\theta(x) = \theta_0 + 2104\theta_1 + 5\theta_2 + \theta_3 + 45\theta_4 = 460$$
$$h_\theta(x) = \theta_0 + 1416\theta_1 + 3\theta_2 + 2\theta_3 + 40\theta_4 = 232$$
$$h_\theta(x) = \theta_0 + 1534\theta_1 + 2\theta_2 + 2\theta_3 + 30\theta_4 = 315$$

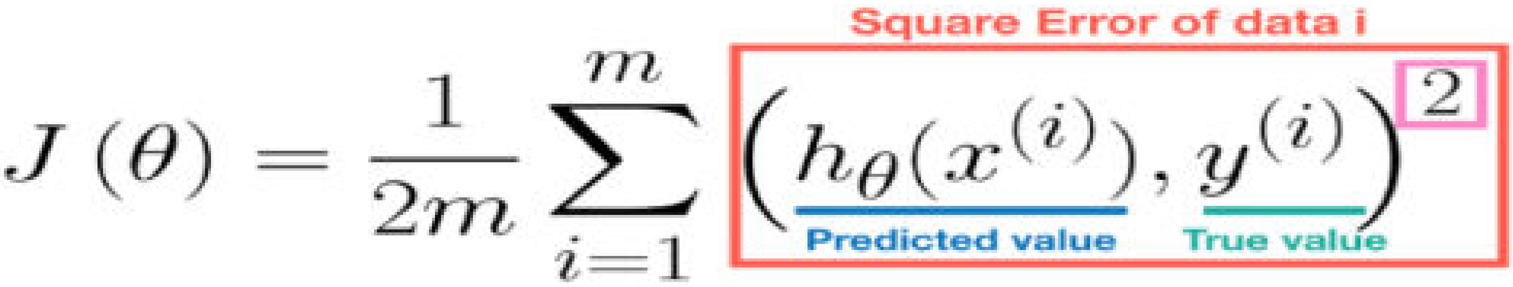
Size ($feet^2$) x_1	Number of bedrooms x_2	Number of floors x_3	age of home (years) x_4	Price($1000) $h_\theta(x) = y$
2104	5	1	45	460
1416	3	2	40	232
1534	2	2	30	315
...	...	...	...	...

When tuning the hypothesis, our model learns parameters θ which makes hypothesis function h(x) a ‘good’ predictor. A Good predictor means the hypothesis is closed enough to the true model. The closer, the better.

But how do we know the hypothesis h(x) is good enough or not? The answer is using a cost function, which we defined to measure the error that the hypothesis made.

Cost Function :

The accuracy of Hypothesis Function h(x) can be measured by using Cost Function. In the Regression Problem, the most popular cost function is Mean Error Square(MSE). The formulation is defined as below.

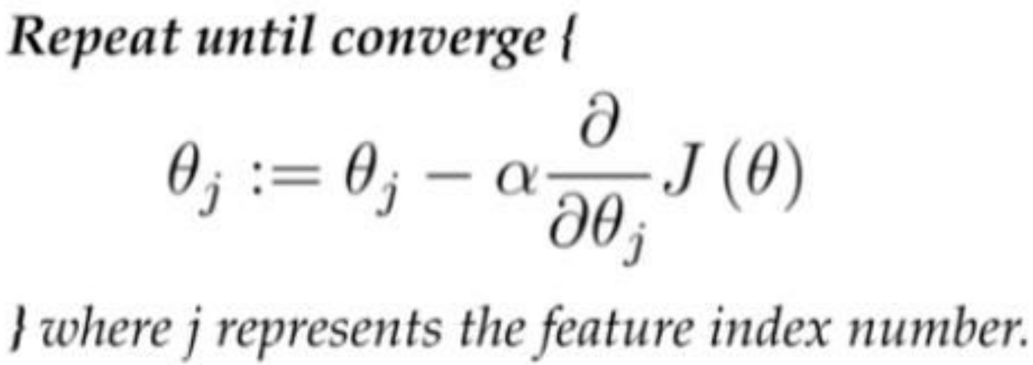


$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \overbrace{\boxed{\left(\underbrace{h_\theta(x^{(i)})}_{\text{Predicted value}}, \underbrace{y^{(i)}}_{\text{True value}} \right)^{\boxed{2}}}}^{\text{Square Error of data i}}$$

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

Essentially, the cost function J(θ) 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(θ).

Gradient Descent for Multiple Variables: Gradient Descent Algorithm is a common method for minimizing cost function. Once the minimum error is found, the model learns the best parameters θ in the meanwhile. Thus, we probably find the good predictor, a hypothesis function with the best parameters θ that makes the minimum error.

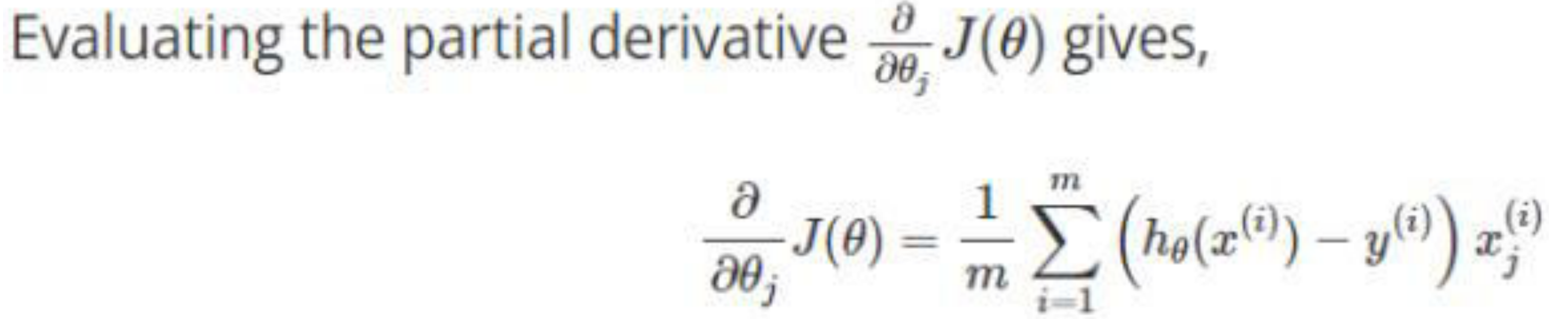


Repeat until converge {

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

} where j represents the feature index number.

At each iteration, the parameters θ need to be updated simultaneously.



Evaluating the partial derivative $\frac{\partial}{\partial\theta_j}J(\theta)$ gives,

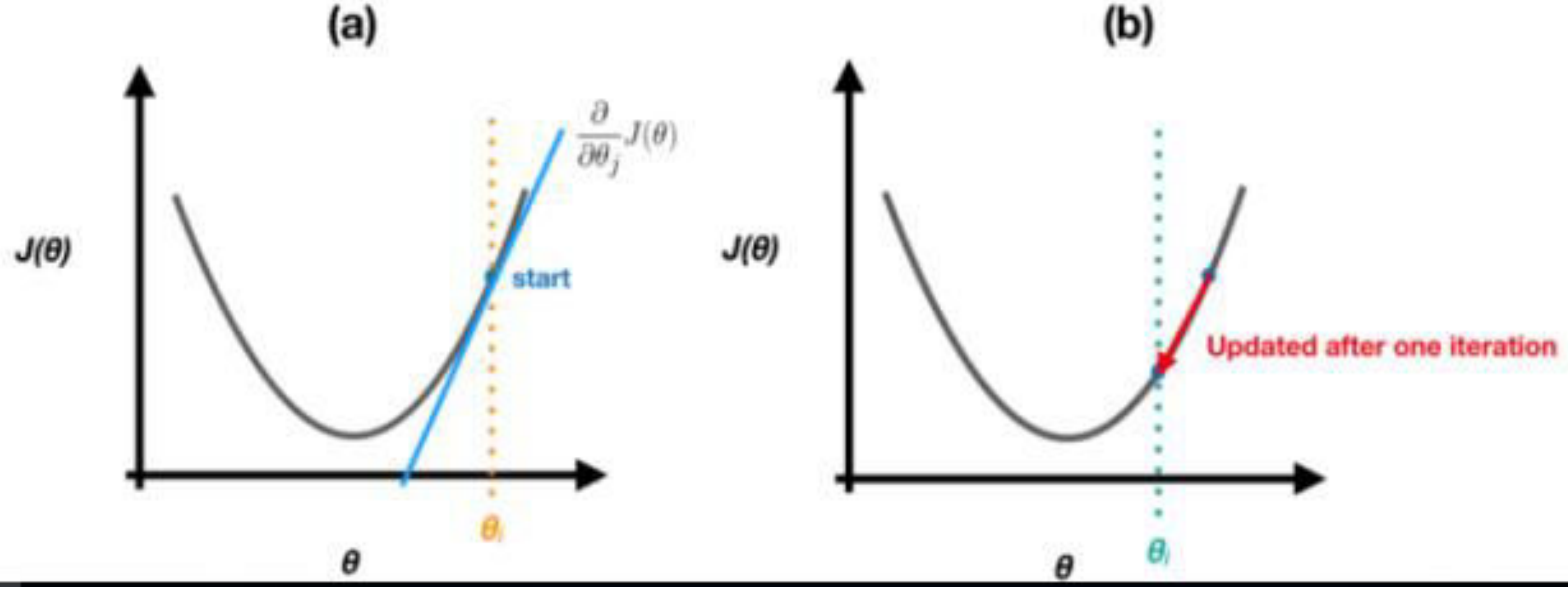
$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

In order to get a better understanding of the Gradient Descent Algorithm easily, we visualized the steps in following figure. In figure (a), the starting point is at orange (θj, J(θj)). Calculate its partial differentiation, then multiplied it by a learning rate α and the updated result is at green (θj, J(θj)), as figure (b) shows.

Repeat until converge {

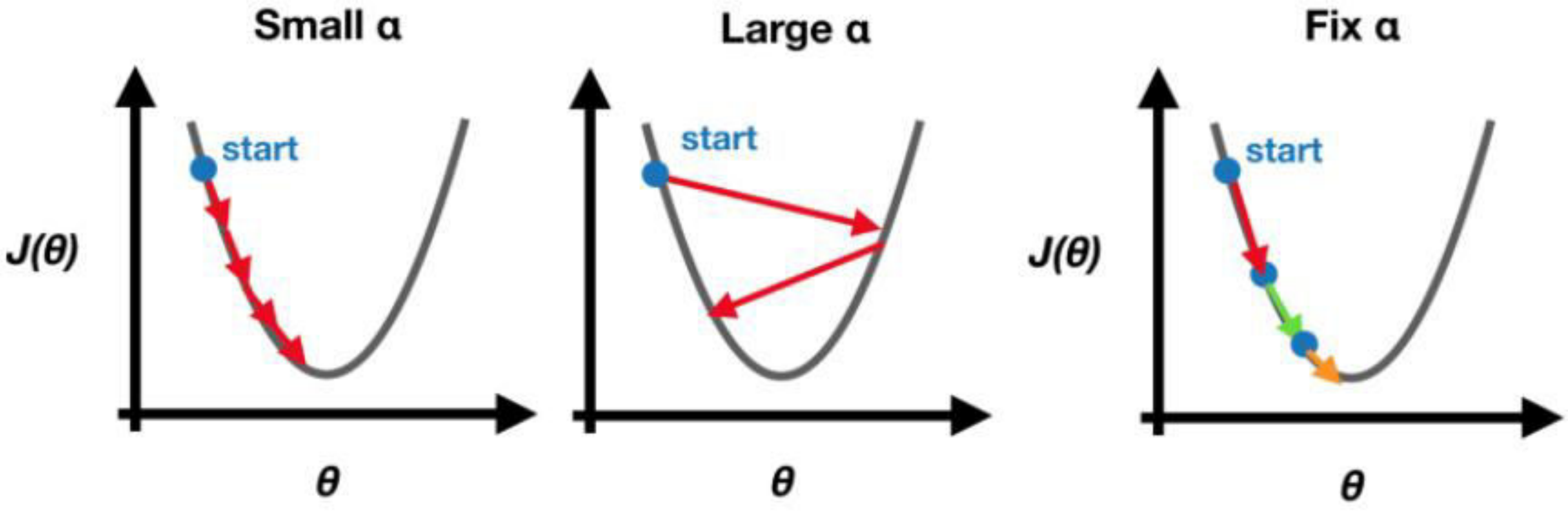
$$\boxed{\theta_j} := \boxed{\theta_j} - \alpha \boxed{\frac{\partial}{\partial \theta_j} J(\theta)}$$

} where j represents the feature index number.


(a)
(b)
J(θ)
θ
$\frac{\partial}{\partial \theta_j} J(\theta)$
start
θ_j
J(θ)
θ
Updated after one iteration
θ_j

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.



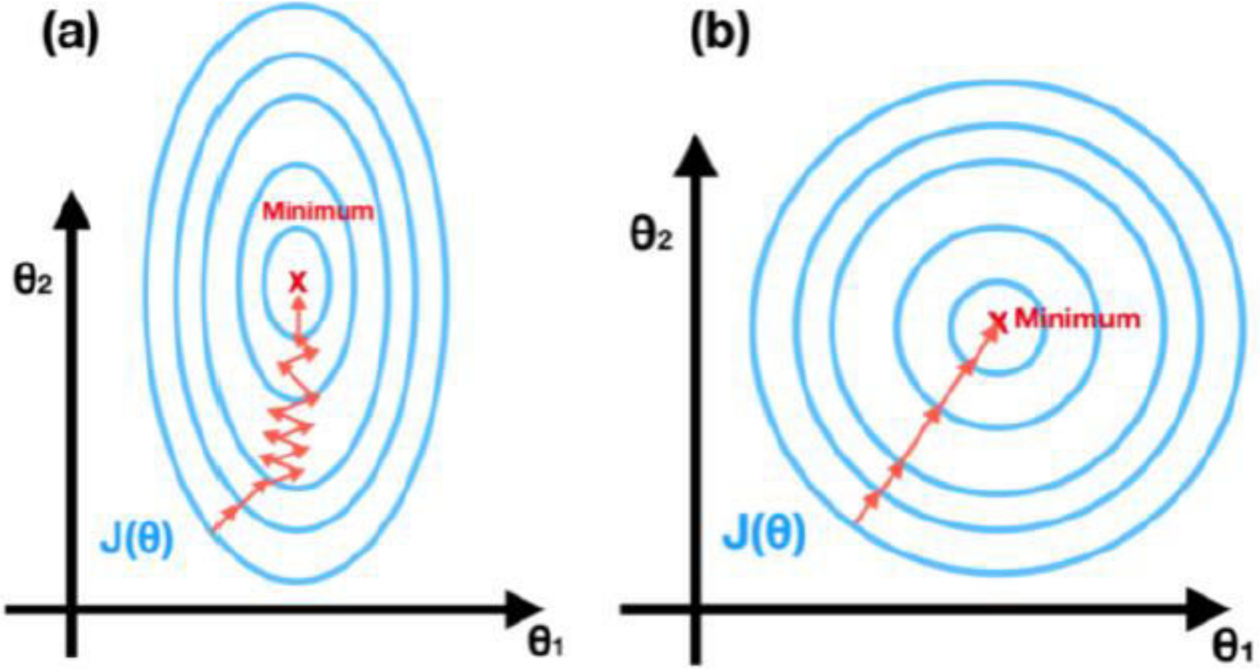
Small α

Large α

Fix α

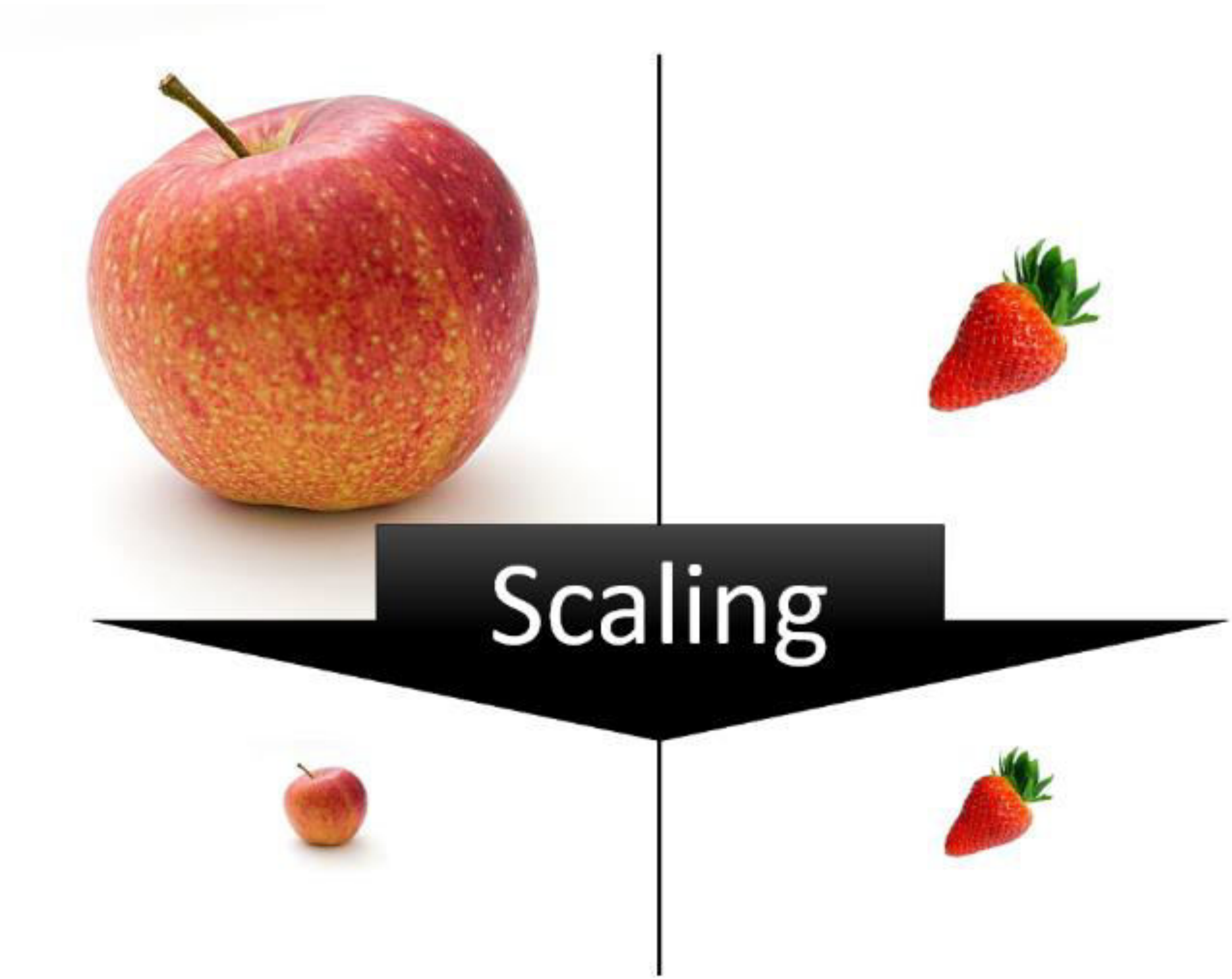
Feature Scaling(Make Gradient Descent Well)

- It is found that during gradient descent if the features are on the **same scale** then the algorithm tends to work better than when the features are not appropriately scaled in the same range.
- **Feature Scaling, also called normalized features.**

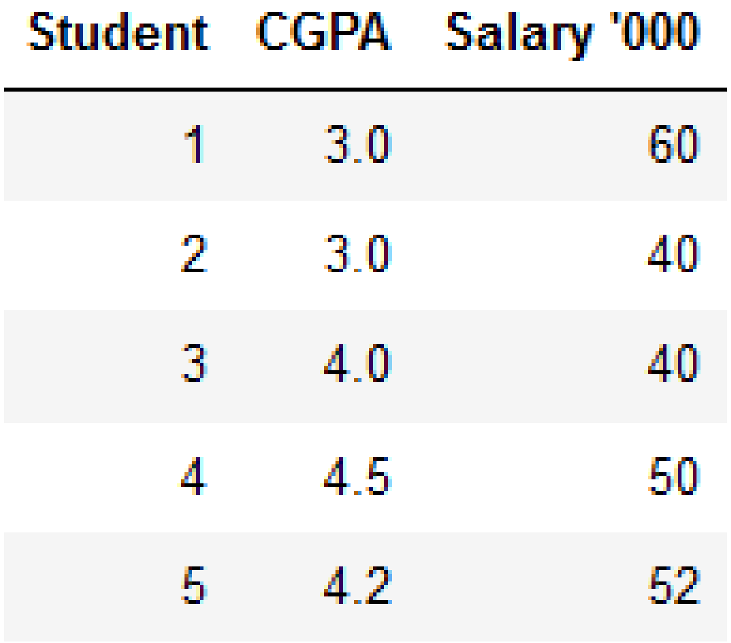


- As seen above, if the **contours are skewed then learning steps would take** longer to converge as the steps would be more prone to oscillatory behavior as shown in the left plot. Whereas if the features are properly scaled, then the plot is evenly distributed and the steps of gradient descent have better profile of convergence.

Feature scaling



- Machine learning algorithm just sees number — if there is a vast difference in the range say few ranging in thousands and few ranging in the tens, and it makes the underlying assumption that higher ranging numbers have superiority of some sort. So these more significant number starts playing a more decisive role while training the model.
- For example, let’s say we have data containing high school CGPA scores of students (ranging from 0 to 5) and their future incomes (in thousands Rupees):



Student	CGPA	Salary '000
1	3.0	60
2	3.0	40
3	4.0	40
4	4.5	50
5	4.2	52

- If features do not have the same numerical scale in values, will cause issues in training a mode.
- If the scale of one independent variable (feature) is greater than another independent variable, the model will give more importance (skew) to the independent variable with the larger range.
- To eliminate this problem, one converts all the independent variables to use the same scale.
- Normalization (0 to 1)
- Standardization (-1 to 1)

Normalization or Standardization

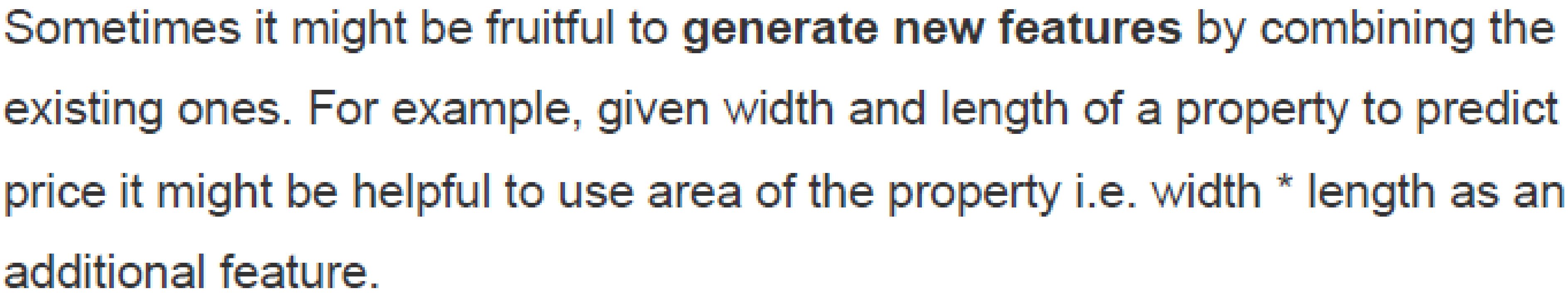
- Feature Scaling means scaling features to the same scale.
- Normalization scales features between 0 and 1, retaining their proportional range to each other.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

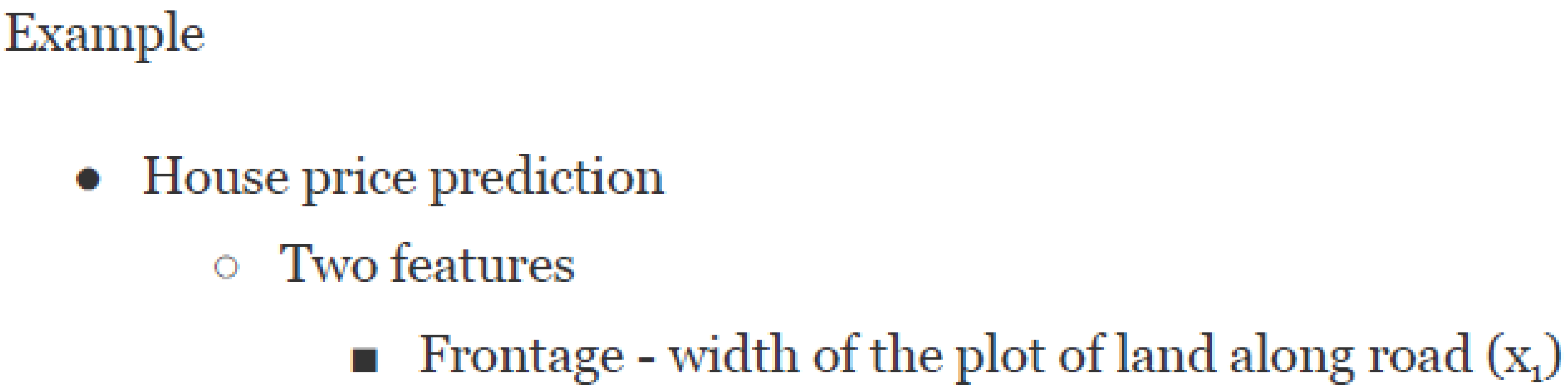
- Standardization scales features to have a is the mean of the feature values and is the standard deviation of the feature values

$$X' = \frac{X - \mu}{\sigma}$$

polynomial regression: Feature Engineering



Sometimes it might be fruitful to **generate new features** by combining the existing ones. For example, given width and length of a property to predict price it might be helpful to use area of the property i.e. width * length as an additional feature.

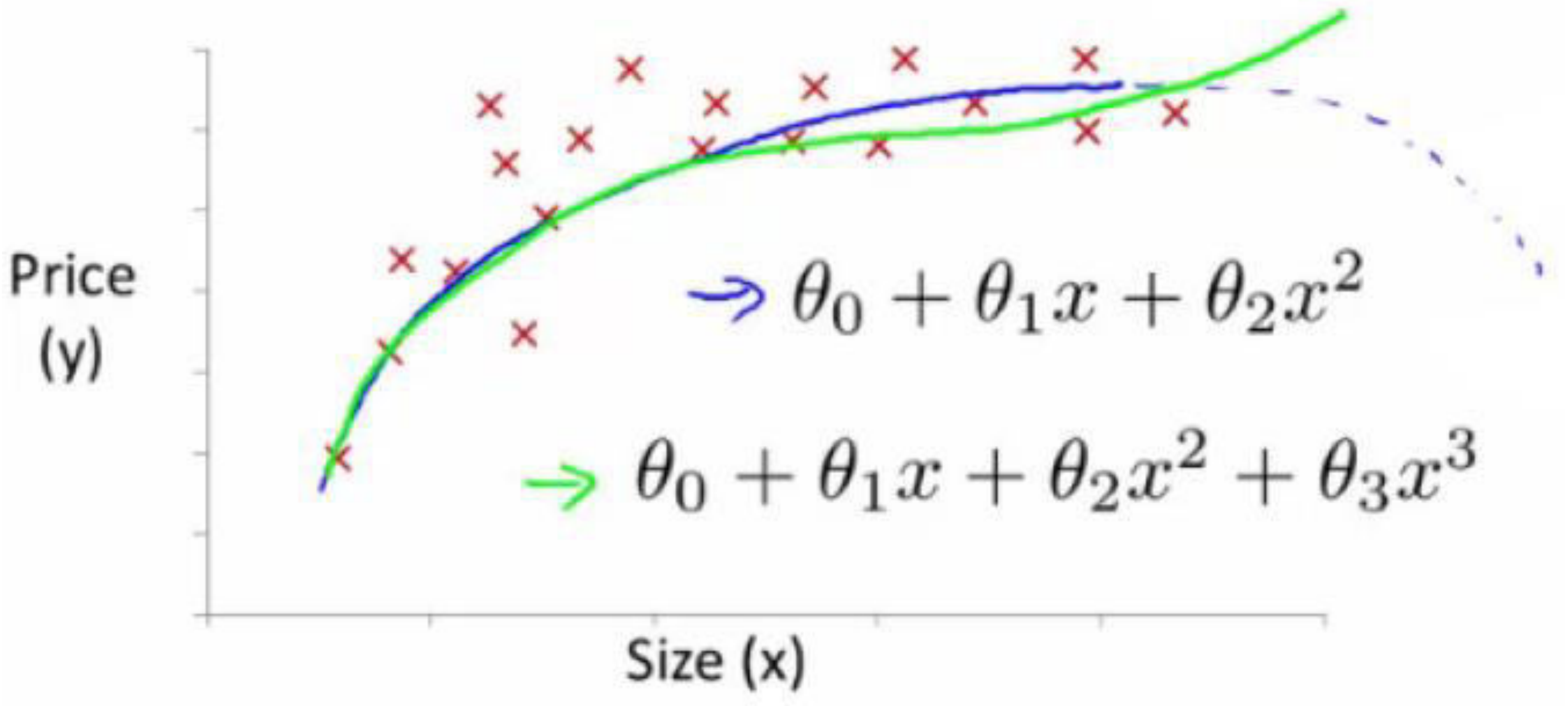


Example

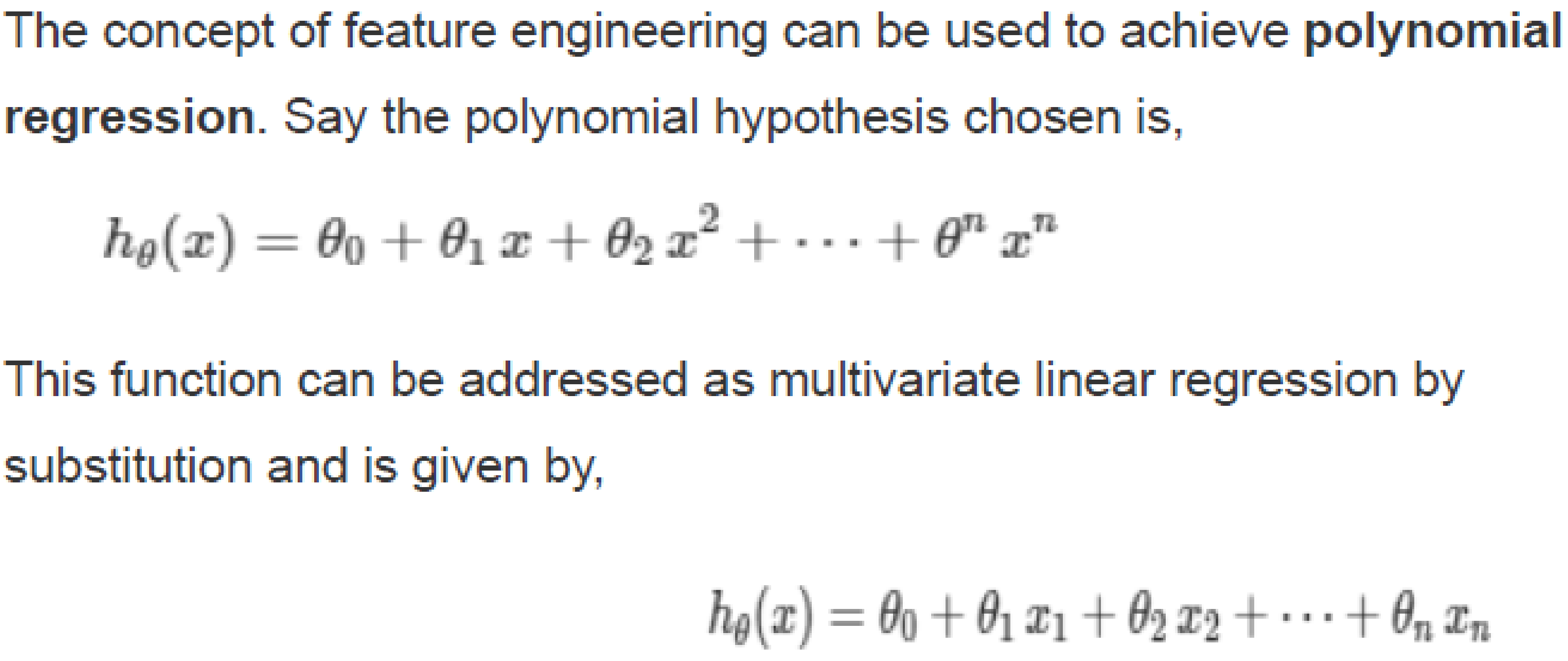
- House price prediction
    - Two features
        - Frontage - width of the plot of land along road (x_1)

- Depth - depth away from road (x_2)

- You don't have to use just two features
    - **Can create new features**
- Might decide that an important feature is the land area
    - So, create a new feature = frontage * depth (x_3)
- Often, by defining new features you may get a better model
- Polynomial regression
    - May fit the data better
    - $\theta_0 + \theta_1x + \theta_2x^2$ e.g. here we have a quadratic function
    - For housing data could use a quadratic function
        - But may not fit the data so well - inflection point means housing prices decrease when size gets really big
        - So instead must use a cubic function



- To map our old linear hypothesis and cost functions to these polynomial descriptions the easy thing to do is set
  - $x_1 = x$
  - $x_2 = x^2$
  - $x_3 = x^3$
- By selecting the features like this and applying the linear regression algorithms you can do polynomial linear regression
- Remember, feature scaling becomes even more important here



The concept of feature engineering can be used to achieve **polynomial regression**. Say the polynomial hypothesis chosen is,

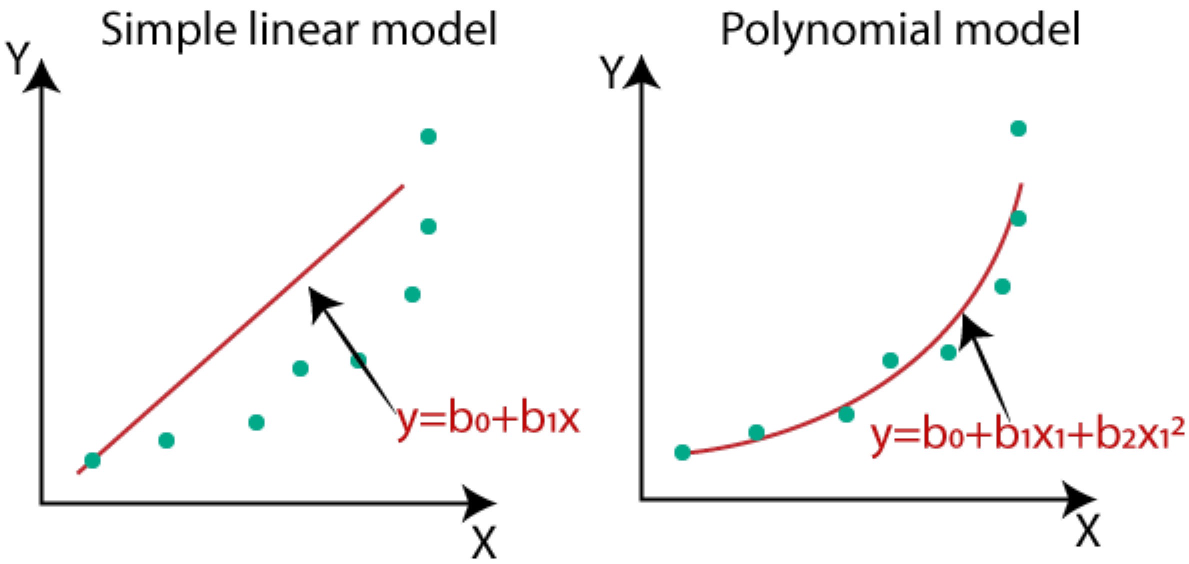
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta^n x^n$$

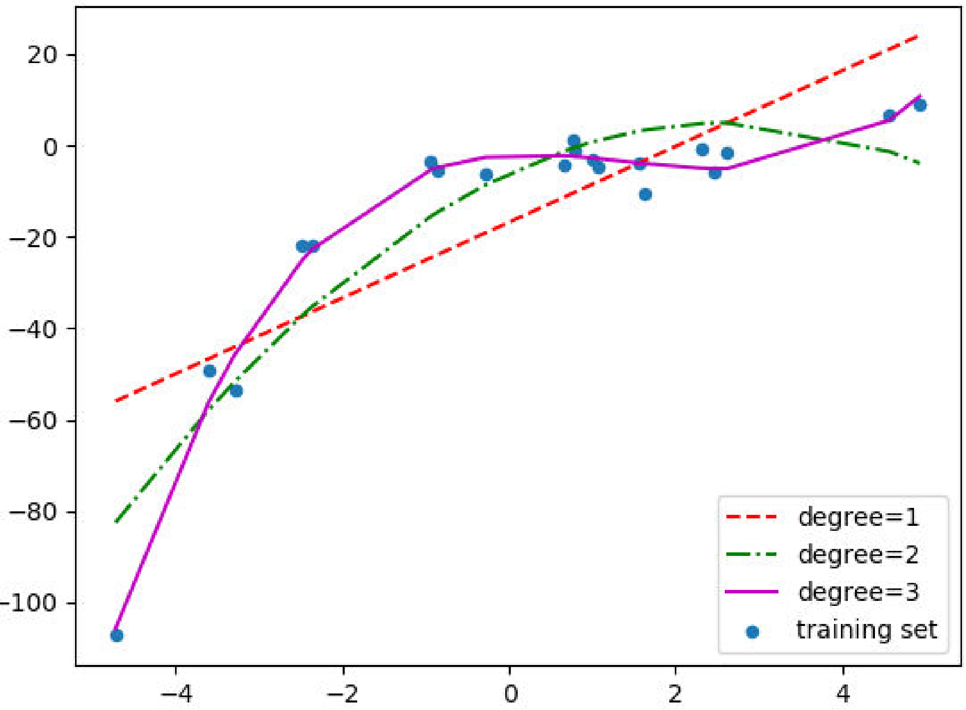
This function can be addressed as multivariate linear regression by substitution and is given by,

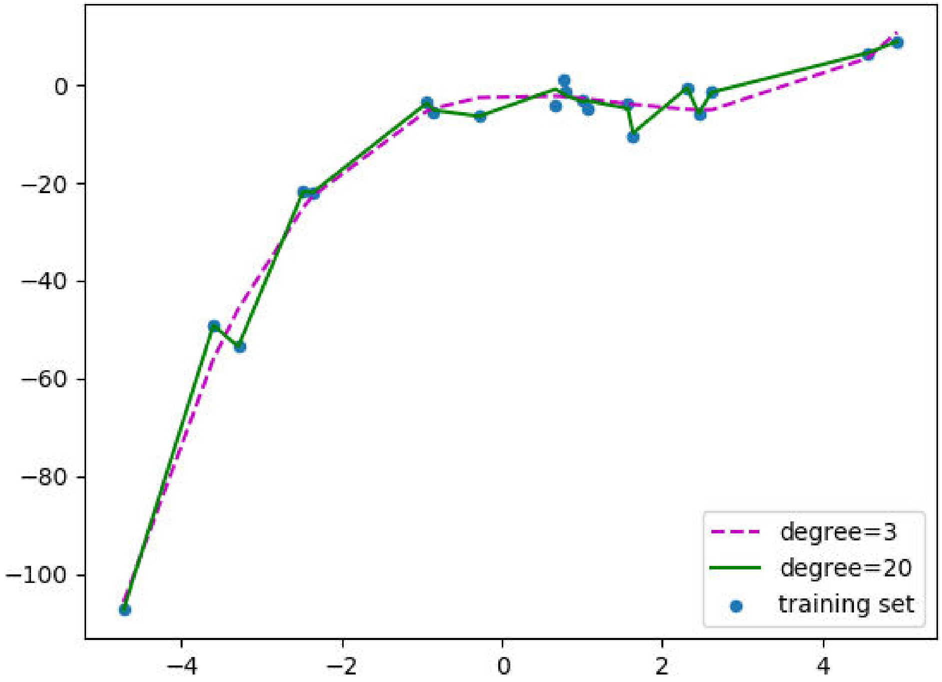
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- Where $x_n = x^n$

Need for Polynomial Regression:







- We can see that the straight line is unable to capture the patterns in the data. This is an example of **under-fitting**.
- For degree=20, the model is also capturing the noise in the data. This is an example of **over-fitting**. Even though this model passes through most of the data, it will fail to generalize on unseen data.