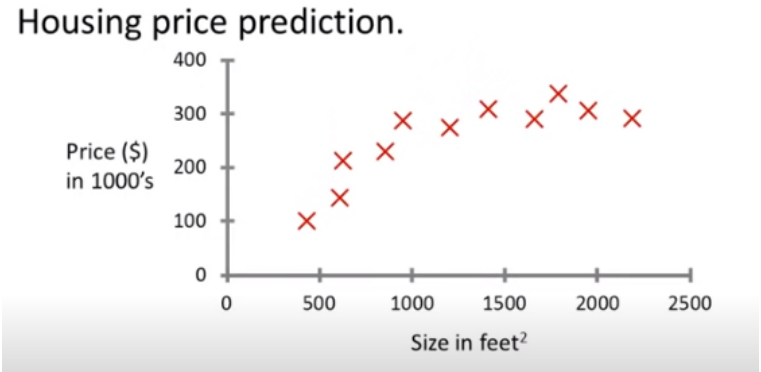
Linear regression

A learning algorithm in supervised learning section.



Training sets. Input X, Output Y, (x, y) is one training example , $(x^{(i)}, y^{(i)})$ is the i^{th} training example

The training set is feeded to the learning algorithm and it gives a hypothesis or an estimate for the required output function.

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

Linear regression with one variable.

Let our data set or training set has m entries or m training examples and the cost function

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

θ_i are the parameters of the model and the work is to find the values of these parameters.

The major work here is to fit a line to the given data set. A line which best fits the data. Or choose the value of the parameters θ_0 and θ_1 so that the value of $h_\theta(x)$ is close to y for our training set.

So the actual problem is

$$Min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$$

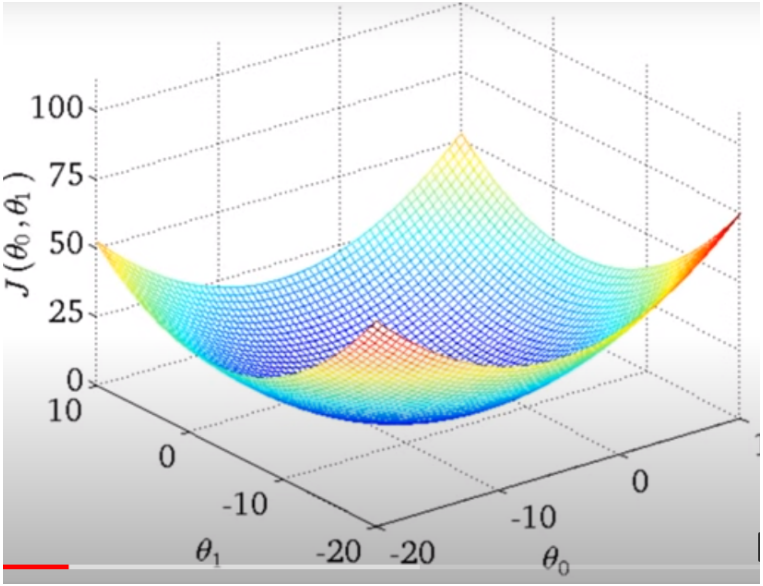
where

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

The mean square error will be small if the predicted responses are very close to the true responses, and will be large if for some of the observations, the predicted and true responses differ substantially.

If we fix θ_0 to 0 and try to solve this problem by fixing some values for θ_1 say 0.5,1,2 etc for a data where input and output is same for all observations, it can be seen that the minimum value of $J(\theta_1)$ is minimum when θ_1 is 1.

If we arbitrarily take values for θ_0 and θ_1 and find the values of $J(\theta_0, \theta_1)$ then we get,


100
75
50
25
0
$J(\theta_0, \theta_1)$
10
0
-10
-20
θ_1
-20
-10
0
θ_0

Gradient Descent Method

We have some function $J(\theta_0, \theta_1)$(called the cost function) and we want to $\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$

Outline: - Start with some θ_0, θ_1

- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

Algorithm,

repeat until convergence $\{$

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

$\}$

Syntax

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{ temp 0}$$

$$\theta_1 := \text{ temp 1}$$

$\alpha > 0$ is called a learning rate. α is usually taken small.

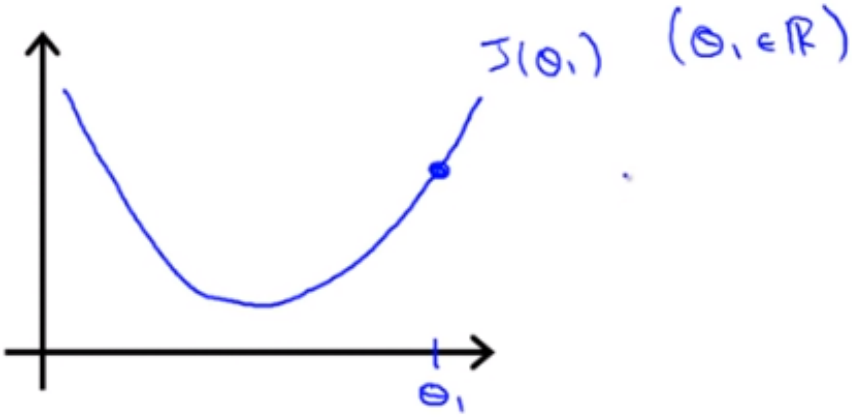
repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{2}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)$$

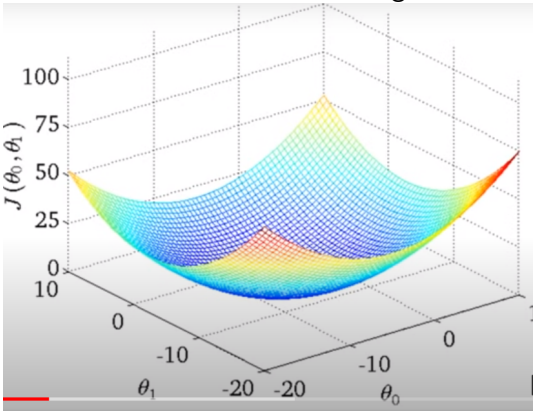
$$\theta_1 := \theta_1 - \alpha \frac{2}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) \cdot x^{(i)}$$

}

gradient descent in 1-d



The cost function for a linear regression is always a function of this kind



convex function, or a bowl shaped function. So only one global optima.

Convex function

Let f be a function of many variables defined on a convex set S. Then f is - convex if for all $x \in S$, all $x' \in S$, and all $\lambda \in (0,1)$ we have

$$f\left((1-\lambda)x + \lambda x'\right) \leq (1-\lambda)f(x) + \lambda f\left(x'\right).$$

Multiple Linear Regression.

A Different situation.

Y, that is the Price depends upon 4 variables say, X_1, X_2, X_3 and X_4

Size (feet²) X_1	Number of bedrooms X_2	Number of floors X_3	Age of home (years) X_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$-value of feature j in i^{th} training example.

Here, $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

we also use $x_0^{(i)} = 1$ for all i

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\begin{aligned} h_\theta(x) &= \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n \\ &= \theta^T x. \end{aligned}$$

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

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

Gradient descent: Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n) \qquad \}$$

(simultaneously update for every $j = 0, \ldots, n$)

New algorithm ($n \geq 1$) : Repeat $\{$

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

(simultaneously update θ_j for

$j = 0, \ldots, n$)

that is,

$$\theta_0 := \theta_0 - \alpha \frac{2}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{2}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_1^{(i)}$$

....

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

Suppose m=4 and we need to fit a multiple linear regression model $h_\theta(x) = \theta^T x$ with the given data.

$\downarrow$	Size (feet ^{2})	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$\text{A}=\begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \text{ and } y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \text{ and solve for } \theta$$

$$\left(A^T A\right)\theta = A^T y$$
$$\theta = \left(A^T A\right)^{-1} A^T y$$

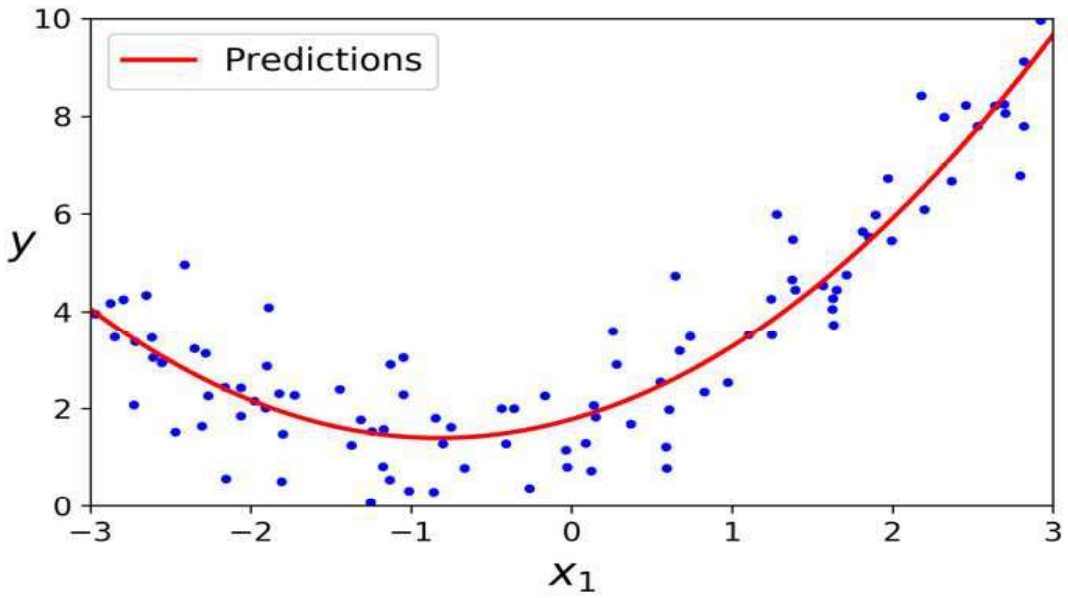
where

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$$

Polynomial Regression

- The best fit line in Polynomial Regression that passes through all the data points is not a straight line, but a curved line, which depends upon the power of X or value of n.
- Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial. The Polynomial Regression equation is given below:

$$y = \beta_0 + \beta_1 x^1 + \beta_2 x^2 + \beta_3 x^3 + \cdots \ldots \beta_n x^n$$



Polynomial Regression

- It is also called the special case of Multiple Linear Regression in ML. Because we add some polynomial terms to the Multiple Linear regression equation to convert it into Polynomial Regression.
- It is a linear model with some modification in order to increase the accuracy.
- The dataset used in Polynomial regression for training is of non-linear nature.
- It makes use of a linear regression model to fit the complicated and non-linear functions and datasets.
- Hence, "In Polynomial regression, the original features are converted into Polynomial features of required degree (2,3,..,n) and then modeled using a linear model."

Polynomial Regression

Order = n

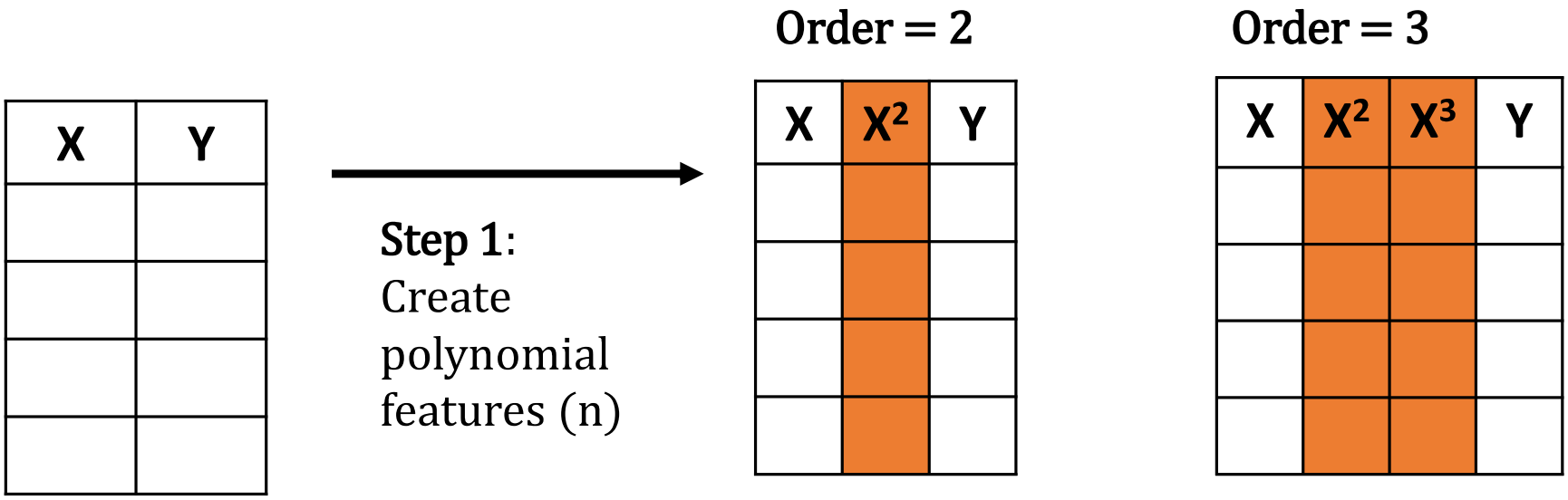
- $y = \beta_0 + \beta_1 x^1 + \beta_2 x^2 + \beta_3 x^3 + \cdots \ldots \beta_n x^n$

Order = 2

- $y = \beta_0 + \beta_1 x^1 + \beta_2 x^2$

Order = 3

- $y = \beta_0 + \beta_1 x^1 + \beta_2 x^2 + \beta_3 x^3$



Step 2: Process according to linear regression