Linear regression with multiple variables

4.1) Multiple features

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

{will be a number}

{will be n dimentional vector}

$x^{(i)} \in \mathbb{R}^n$

⇒ New hypothesis function: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_n x_n$

⇒ For convenience of notation define $x_0^{(i)} = 1$

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

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$h_\theta(x) = \theta^T x \qquad \left\{ x = \begin{bmatrix} x_0 \\ x^{(i)} \end{bmatrix} \right\}$$

⇒ Multivariate linear regression

4.2) Gradient descent for multiple Variables

Cost function

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

Vector Vector

Gradient descent

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta) \quad \{ \forall \; j = 0, 1 \cdots n \}$$

}

⇓

Repeat {

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

(Simultaneously update θ_j for $j = 0 \cdots n$)

}

13) Feature Scaling

If different features takes different range of values then it becomes very difficult for algorithm to find minimum or it takes way more time for algorithm to find minimum.

↳ To avoide this we scale every feature so that they are in similar range.

⇒ Get every feature into approximately a $-1 \leq x_i \leq 1$ range.

Mean normalization

⇒ Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean.

General rule $x_i \leftarrow \frac{x_i - \mu_i}{s_i}$

μ_i → avg value of x_i in training set

s_i → range of x_i (i.e. Max x_i - Min x_i)

⇒ To make sure gradient descent is working correctly.

↳ Plot $J(\theta)$ Vs (Number of iteration)

↓

{If it is continuously decreasing then its going well.}

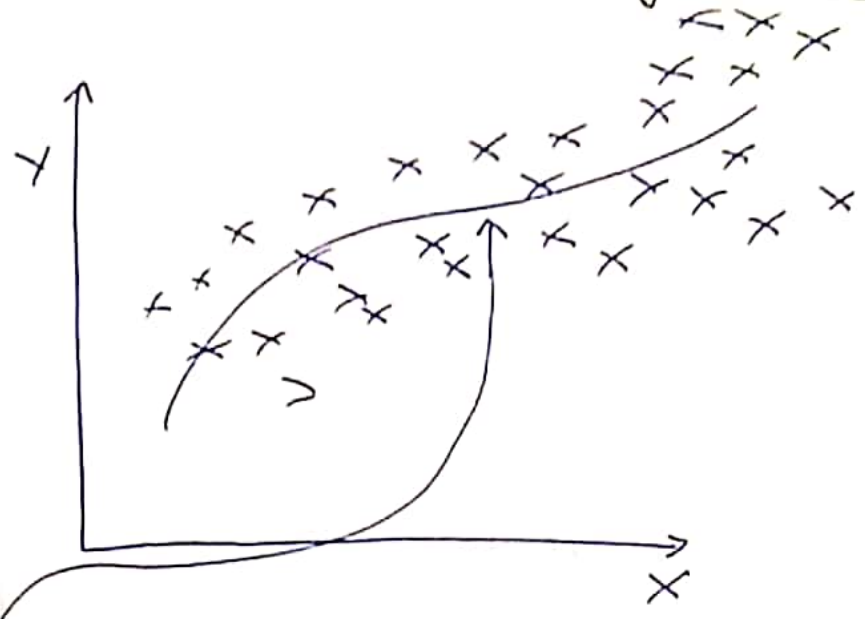
↓

{If it is almost flat than you can assume that it has converged}

⇒ To choose α, try

... 0.001 , 0.01 , 0.1 , 1 ...

4.5> Polynomial regression



$h_\theta(x) = \theta_0 + \theta_1 x$ { ✗ doesn't give good fit }

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

{ Very good fit }

So $x_1 = x$
$x_2 = x^2$
$x_3 = x^3$

{ define three new features }

4.6) Normal Equations

Method to solve for θ analytically.

Let $X = \begin{bmatrix} 1 & x_1^1 & x_2^1 \cdots x_n^1 \\ 1 & x_1^2 & x_2^2 \cdots x_n^2 \\ \vdots & \vdots & \vdots \\ 1 & x_1^m & x_2^m \cdots x_n^m \end{bmatrix}_{m \times (n+1)}$

- m → number of training set
- $(n+1)$ → number of features
- Matrix that contains all the training examples

Similarly $Y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}_{m \times 1}$ { Collection of all the target variable in training example }

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

→ This gives you the value of θ that minimizes the cost function

Note

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

Gradient Descent	Normal Equation
⇒ Need to choose α.	⇒ No need to choose α.
⇒ Need many iterations.	⇒ Don't need to iterate.
⇒ Works well even when n is large	⇒ Slow if n is very large. $O(n^3)$
$n > 10000$	$n = 100$ OK $n = 1000$ OK $n = 10000$ ← Not greater than this