CSM 6405: Symbolic ML II

Lecture 2: Linear and Multivariate Regression

Acknowledgement: Andrew Ng (Stanford University), Coursera

Prof. Dr. Md. Rakib Hassan

Dept. of Computer Science and Mathematics,

Bangladesh Agricultural University.

Email: rakib@bau.edu.bd

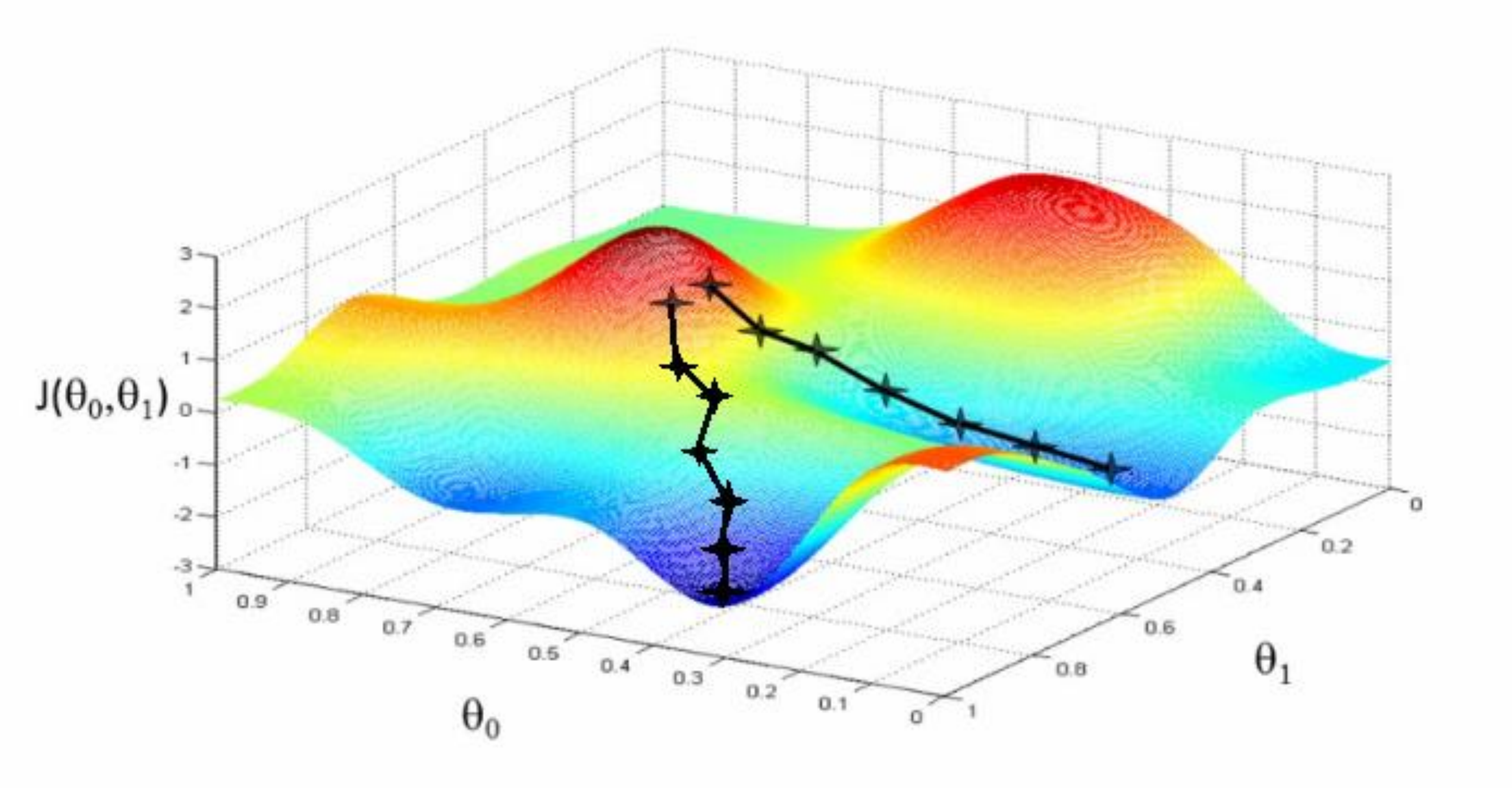
Linear Regression with 1 Variable

Gradient Descent

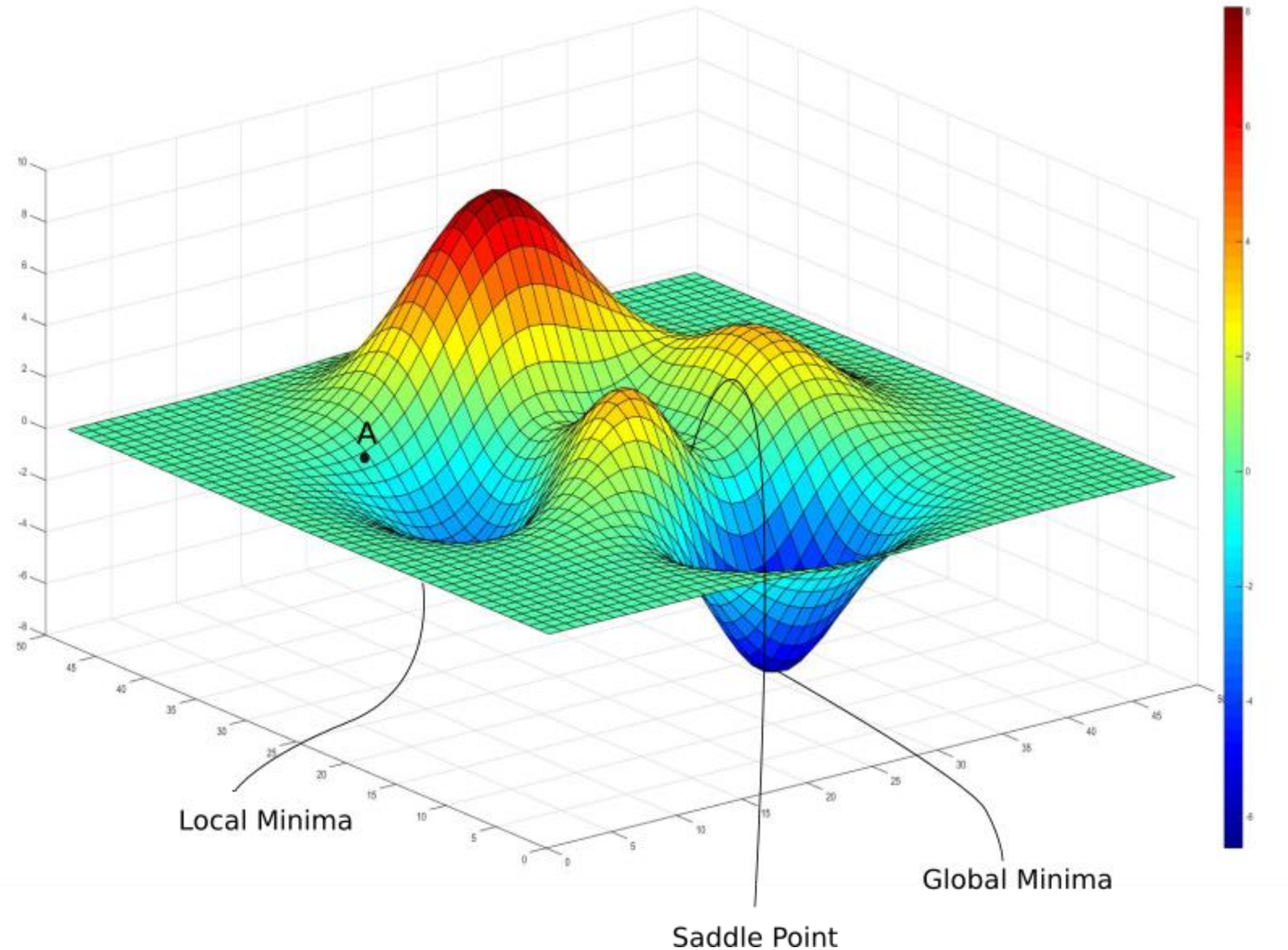
- Have some function $J(\theta_0, \theta_1)$
- Find $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

- 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

Gradient Descent



Gradient Descent



Gradient Descent Algorithm

- Repeat until convergence {

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

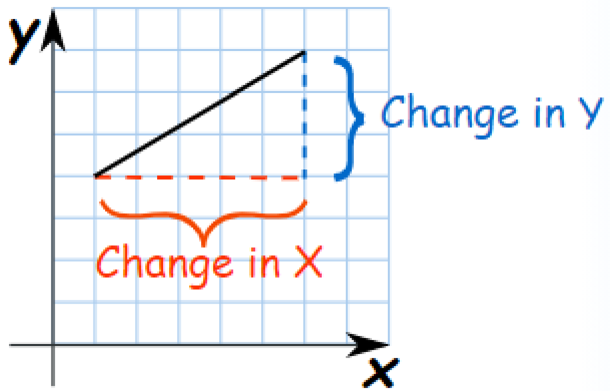
}

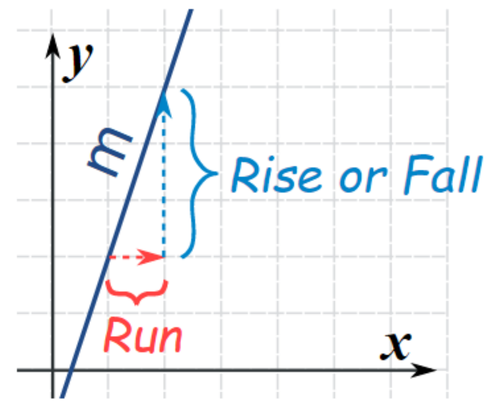
where, α = learning rate

- Simultaneous update
  - temp0 = $\theta_0 - \alpha \frac{\delta}{\delta\theta_j} J(\theta_0, \theta_1)$
  - temp1 = $\theta_1 - \alpha \frac{\delta}{\delta\theta_j} J(\theta_0, \theta_1)$
  - θ_0 = temp0
  - θ_1 = temp1

Gradient or Slope

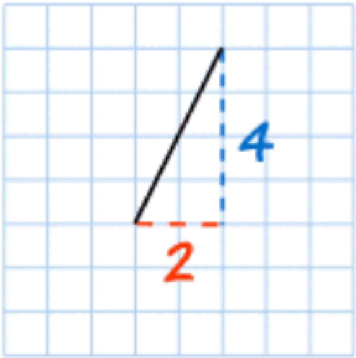
- $$Gradient = \frac{Change\ in\ Y}{Change\ in\ X}$$





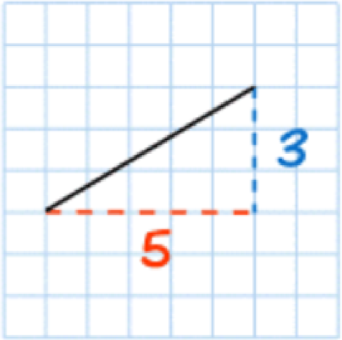
- Starting from the left and going across to the right is positive (but going across to the left is negative).
- Up is positive, and down is negative

Examples



The Gradient = $\frac{4}{2}$ = 2

The line is steeper, and so the Gradient is larger.

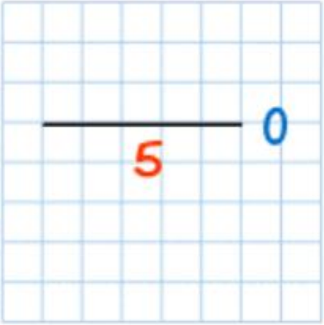


The Gradient = $\frac{3}{5}$ = 0.6

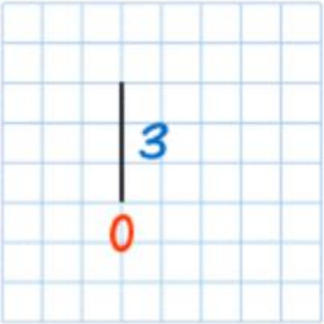
The line is less steep, and so the Gradient is smaller.

Examples

$$\text{Gradient} = \frac{-4}{2} = -2$$

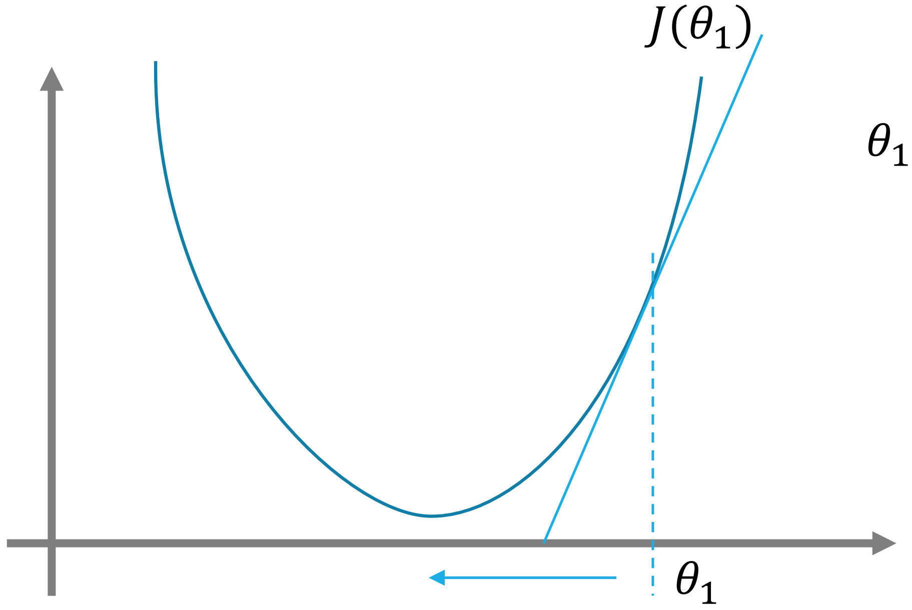


$$\text{Gradient} = \frac{0}{5} = 0$$



$$\text{Gradient} = \frac{3}{0} = \text{undefined}$$

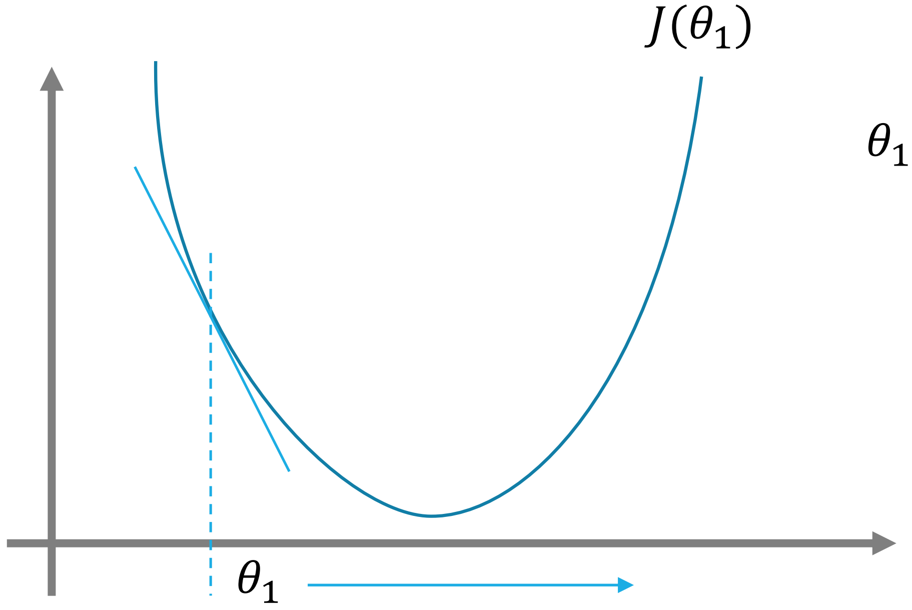
Gradient Descent



$$\theta_1 = \theta_1 - \alpha \frac{\delta}{\delta\theta_1} J(\theta_1)$$

$$= \theta_1 - \alpha(positive\ slope)$$

Gradient Descent

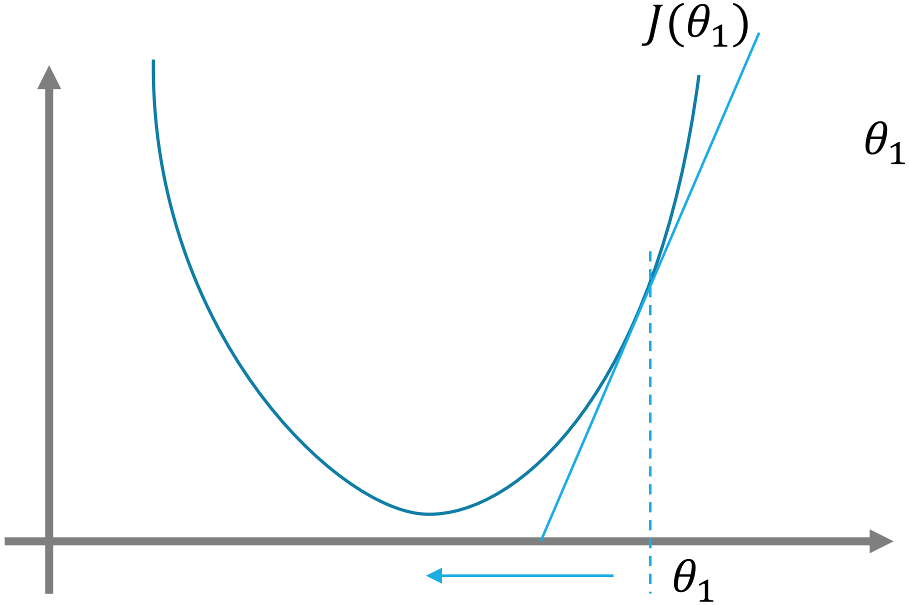


$$\theta_1 = \theta_1 - \alpha \frac{\delta}{\delta\theta_1} J(\theta_1)$$

$$= \theta_1 - \alpha(negative\ slope)$$

Fixed Learning Rate

- Gradient descent can converge to a local minimum, even with the learning rate α fixed.



$$\theta_1 = \theta_1 - \alpha \frac{\delta}{\delta\theta_1} J(\theta_1)$$

$$= \theta_1 - \alpha(positive\ slope)$$

- As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.

Gradient Descent Algorithm

- Repeat until convergence {

$$\theta_j = \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta_0, \theta_1)$$

}

where, α = learning rate

- Linear regression model

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

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

Gradient Descent Algorithm

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

- $j = 0 : \frac{\delta}{\delta \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$
- $j = 1 : \frac{\delta}{\delta \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) . x^{(i)}$

Gradient Descent Algorithm

- Repeat until convergence {

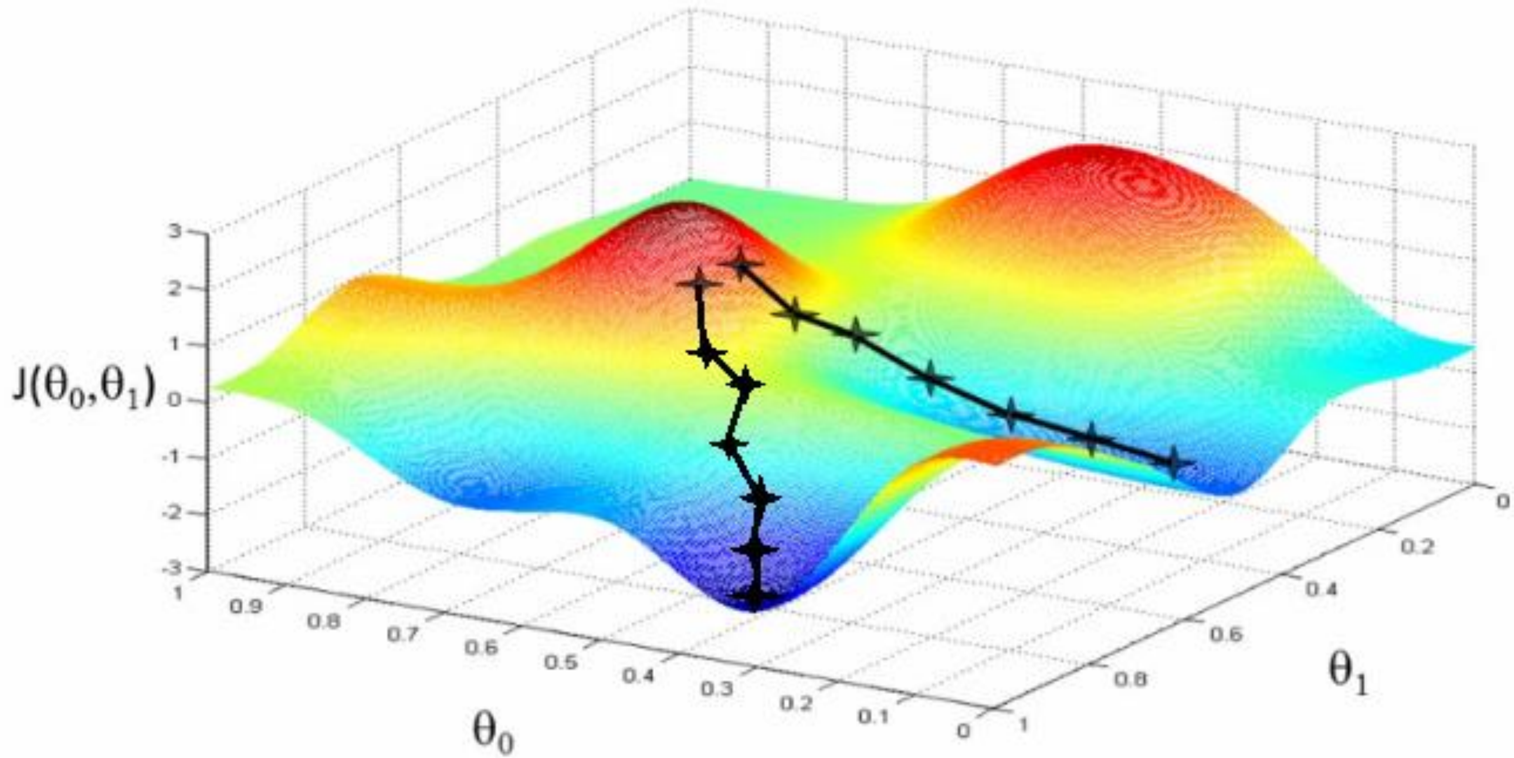
$$\theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$

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

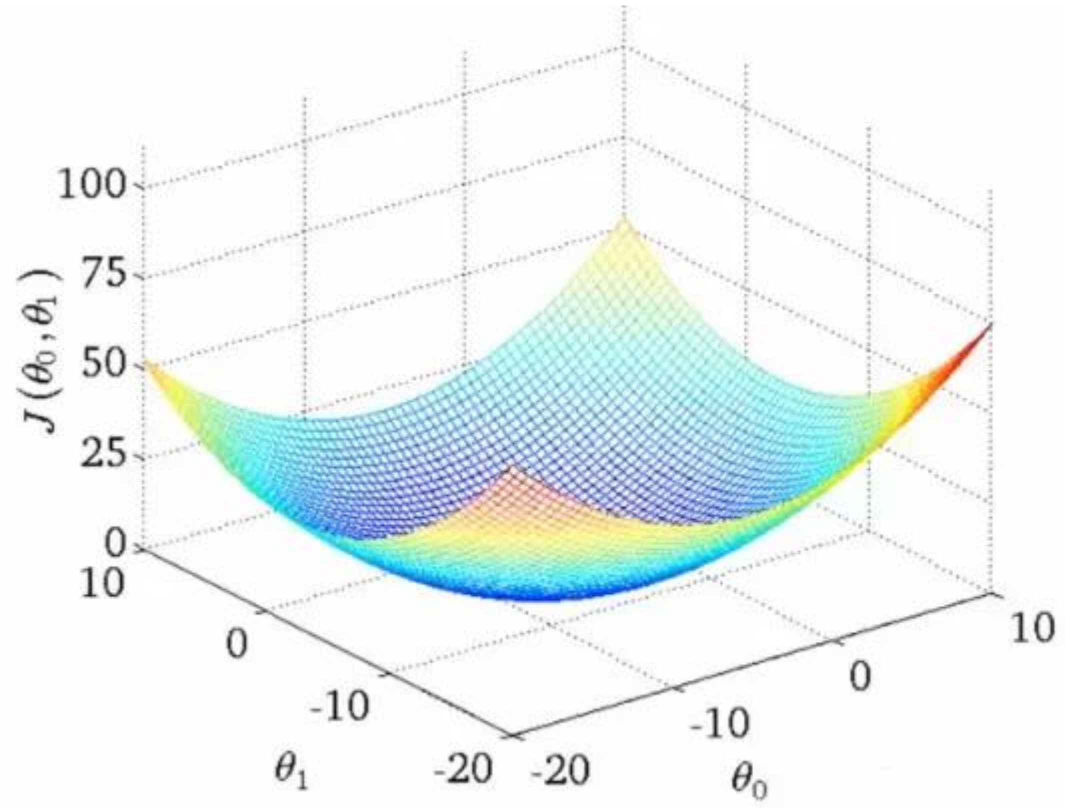
}

- Update θ_0 and θ_1 simultaneously

Convergence



Convex Function



Batch Gradient Descent

- "Batch": Each step of gradient descent uses all the training examples.

Linear Regression with Multiple Variables

MULTIPLE FEATURES

Multiple Features (Variables)

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
…	…	…	…	…

$m = 47$

- 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

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_3^{(2)} = 2$$

Hypothesis

- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3\ x_3 + \cdots + \theta_n x_n$
- $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3\ x_3 + \cdots + \theta_n x_n$
- For convenience of notation, define $x_0 = 1$.
- $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3\ x_3 + \cdots + \theta_n x_n$
- $= \theta^T X$

$$\theta_T = [\theta_0 \quad \theta_1 \quad \ldots]$$

$$X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \end{bmatrix}$$

Gradient Descent

- Repeat until convergence {

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

(simultaneously update θ_j for $j = 0,1, \dots, n$)

}

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

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

$$\theta_2 = \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) . x_2^{(i)}$$

Feature Scaling

- Make sure features are on a similar scale
- Get every feature into approximately a $-1 \le x_i \le 1$ range.
- Examples:
  - x_1= size (0-2000 feet^2)
  - x_2= number of bedrooms (1-5)

- Method 1:
  - $x_1 = \frac{\text{size}\left(\text{feet}^2\right)}{2000}$
  - $x_2 = \frac{\text{number of bedrooms}}{5}$

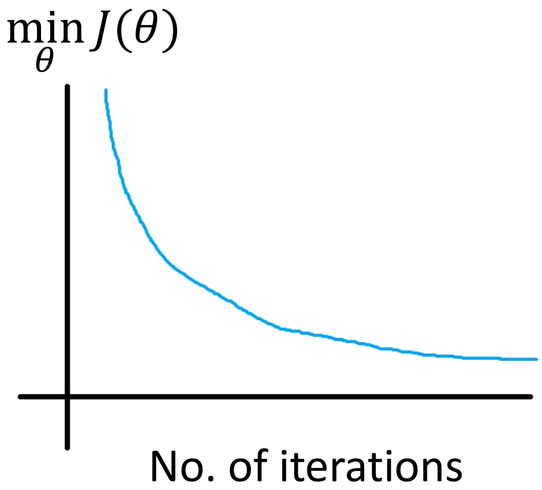
Method 2 – Mean Normalization

- $x_i = \frac{x_i - \mu}{\sigma}$
  - μ = mean
  - σ = standard deviation

- Do not apply to $x_0 = 1$

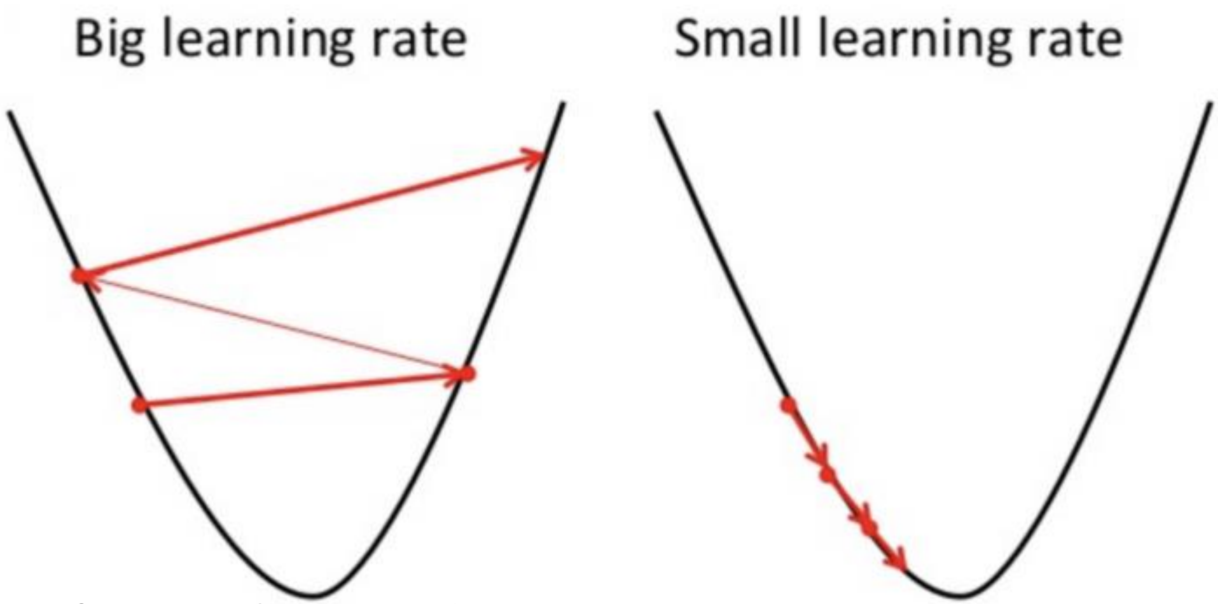
How to Choose Learning Rate?

- Make sure gradient descent is working correctly.
- Example automatic convergence test:



- Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

Learning Rate



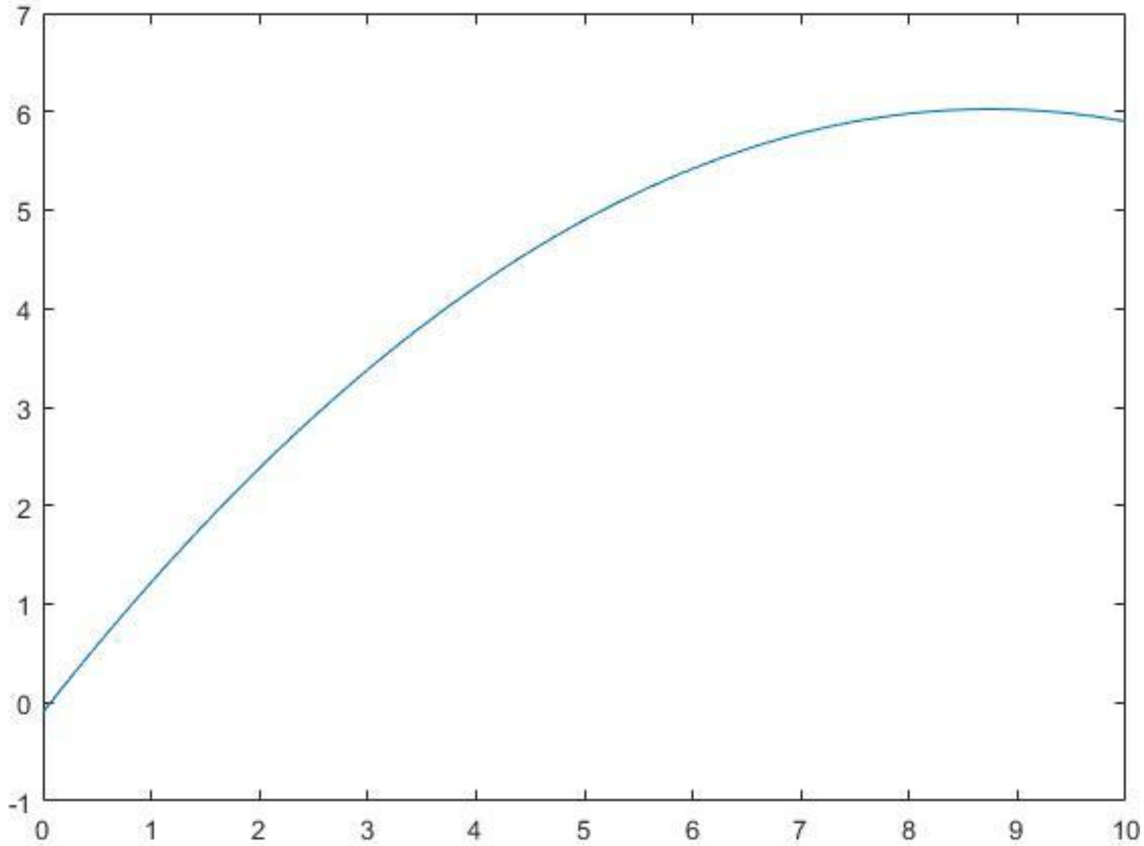
If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

If α is too small, gradient descent can be slow

- Try:
  - …, 0.001, 0.003, …, 0.01, 0.03, … , 0.1, 0.3, …

Polynomial Regression

- $y = -0.08x^2 + 1.4x - 0.1$



Analytical Solution of Linear Regression

- Examples: m = 4

	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

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

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

Normal Equation

- m examples $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$
- n featues

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

Example: If $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$,

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ \vdots & \vdots \\ 1 & x_1^{(m)} \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

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

Plots of Sample Data

Using
- Gradient descent
- Normal equation

Gradient Descent vs Normal Eq.

- Gradient Descent
  - Need to choose α
  - Needs many iterations
  - Works well even when n is large

- Normal equation
  - No need to choose α
  - No need to iterate
  - Need to compute $(X^T X)^{-1}$
    - Complexity: $O(n^3)$
  - Slow if n is very large

m = training examples
n = features

Non-Invertible/ Singular/ Degenerate

- Matrices that cannot be inverted, are called Non-invertible/ Singular/ Degenerate matrices
  - Normal equation
    - $\theta = (X^T X)^{-1} X^T y$

- Causes
  - Redundant features (Linearly dependent)
    - Example: $x_1 =$ size in feet^2, $x_2 =$ size in m^2
  - Too many features (e.g., $m \leq n$): more features than training examples
    - Delete some features, or use regularization.

- Solution:
  - Use pinv(X) in Matlab
    - Moore-Penrose pseudo inverse

Assignment

- Find the values of θ to best fit the sample data using
  - Gradient descent:
    - Size -> price
    - Size, beds -> price
  - Normal equation
    - Size -> price
    - Size, beds -> price

- Plots
  - Best fit lines
  - Cost functions (2D and 3D)
- Do’s and don’ts
  - Do not use any library
  - Only use gradient descent algorithm and normal equation given in the slides

Assignment (Cont.)

- Any Tool
  - Java, Python, Matlab, Octave, etc.
- Submit
  - What files:
    - Source codes with proper documentation
  - To: rakib@bau.edu.bd
  - Deadline: 05-Nov-2020
- Marks:
  - 20
- Marks deduction:
  - 10 marks deduction per day for submitting after deadline
  - X% deduction for X% similarity.
  - 0 for > 50% similarity.