Linear Regression with multiple Variables (Multiple features)
AI4003 - Applied Machine Learning

Dr. Mohsin Kamal

Department of Electrical Engineering
National University of Computer and Emerging Sciences, Lahore, Pakistan

Size ($feet^2$) x_1	No. of Bedrooms x_2	No. of floors x_3	Age of home x_4	Price ($k) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Notation:

- m = Number of training examples i.e., 47
- 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
- y's = "output" variable / "target" variable

Hypothesis:

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

How to represent it mathematically?

For convenience of notation, define $x_0 = 1$

let

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

We can write it in multiplication form as:

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \ldots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$h_\theta(x) = \theta^T x$$

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$
Parameters: θ
Cost function: $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Gradient descent:
Repeat {
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$
}
(Simultaneously update for every $j = 0, 1, 2. \dots, n$)

Gradient Descent

Previously (n=1):

Repeat {

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial\theta_0}J(\theta)}$$

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

(simultaneously update θ_0, θ_1)

}

New algorithm $(n \geq 1)$:

Repeat {

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

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

}

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

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

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

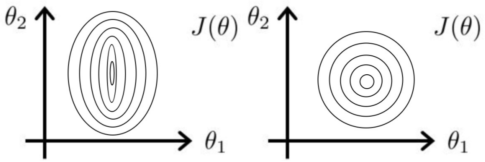
$\ldots$

Feature Scaling

Idea: Make sure features are on a similar scale.

$$x_1 = size\ (0 - 2000 feet^2) \qquad x_1 = \frac{size\ (feet^2)}{2000}$$

$$x_2 = no.\ of\ bedrooms\ (1 - 5) \qquad x_2 = \frac{no.\ of\ bedrooms}{5}$$



Feature Scaling

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 (Do not apply to $x_0 = 1$). For example:

$$x_1 = \frac{size - 1000}{2000} \qquad -0.5 \leq x_1 \leq 0.5$$

$$x_2 = \frac{\#bedrooms - 2}{4} \qquad -0.5 \leq x_2 \leq 0.5$$

$$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1}$$

$$x_2 \leftarrow \frac{x_2 - \mu_2}{s_2}$$

where

- μ_i is the average value of x_i in training set, and
- s_i is the range (max-min) or standard deviation.

Learning rate

Gradient descent

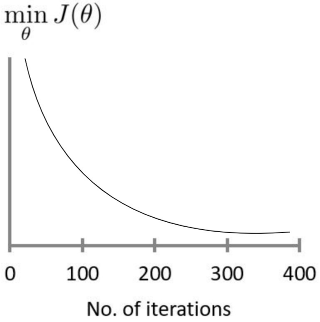
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

We will learn:

- "Debugging": How to make sure gradient descent is working correctly.
- How to choose learning rate α.

Learning rate

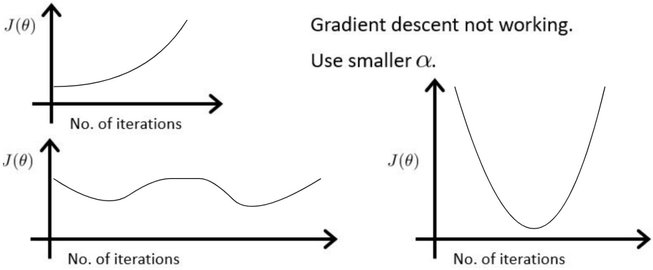
Making sure gradient descent is working correctly.



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

Learning rate

Making sure gradient descent is working correctly.



- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Learning rate

Summary:

- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge.

To choose α, try

$$\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \ldots$$

House price prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times front + \theta_2 \times depth$$

Let $x_1 = front$ and $x_2 = depth$

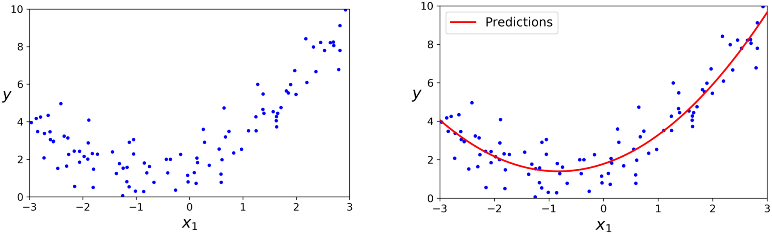
We don't necessarily require only two features. Else, we can create new features as well. e.g.,

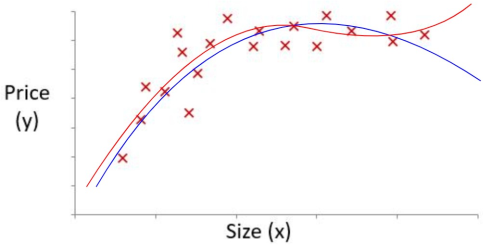
$$\text{Area: } x = front \times depth$$

Hence,

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

- What if your data is actually more complex than a simple straight line?
- Surprisingly, you can actually use a linear model to fit nonlinear data.
- A simple way to do this is to add powers of each feature as new features, then train a linear model on this extended set of features.
- This technique is called **Polynomial Regression**.





$\theta_0 + \theta_1 x + \theta_2 x^2$

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$$

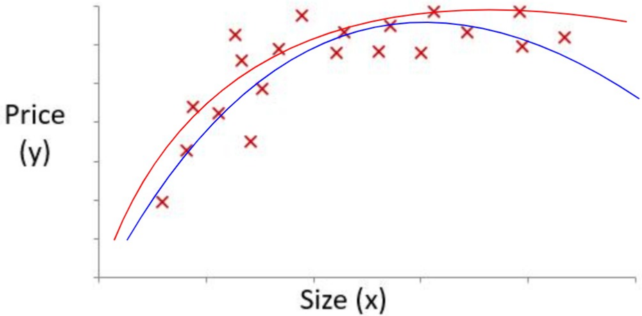
We will require feature scaling because:

size: 1 - 1000

$size^2$: 1 - 1000000

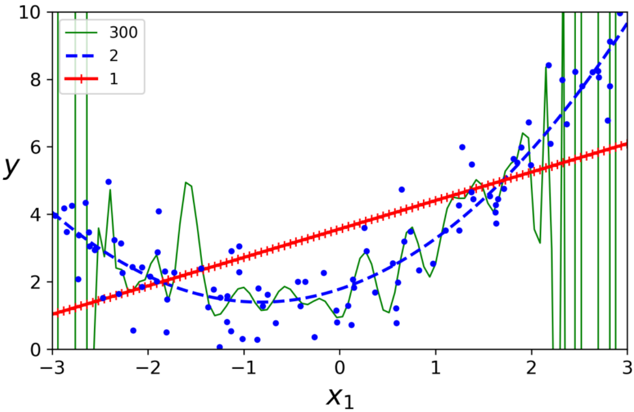
$size^3$: 1 - 10^9

Choice of features



$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$
$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$$

High-degree Polynomial Regression



Normal equation: Method to solve for θ analytically.
When to use it?

Intuition: If 1D ($\theta \in \Re$) $J(\theta) = a\theta^2 + b\theta + c$

how to minimize quadratic function?,

$$\frac{d}{d\theta} J(\theta) = \cdots = 0$$

Solve for θ

When $\theta \in \Re^{n+1}$

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

do,

$$\frac{\partial}{\partial\theta} J(\theta) = \cdots = 0 \text{ (for every j)}$$

Solve for $\theta_0, \theta_1, \cdots, \theta_m$

Training examples: $m = 4$.

	Size ($feet^2$)	Bedrooms	Floors	Age (years)	Price (k)
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

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

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

m examples $(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})$; n **features.**

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \Re^{n+1} \qquad i.e., \qquad x^{(2)} = \begin{bmatrix} x_0^{(2)} = 1 \\ x_1^{(2)} = 1416 \\ x_2^{(2)} = 3 \\ x_3^{(2)} = 2 \\ x_4^{(2)} = 40 \end{bmatrix}$$

Then,

$$X(design\ matrix) = \begin{bmatrix} \cdots (x^{(1)})^T \cdots \\ \cdots (x^{(2)})^T \cdots \\ \vdots \\ \cdots (x^{(m)})^T \cdots \end{bmatrix} = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

Feature scaling is not required in Normal equations method!

m training examples, n features.

Gradient Decent	Normal Equation
Need to choose α	No need to choose α
Needs many iterations	Doesn't need to iterate
Works well even when n is large	Need to compute $(X^T X)^{-1}$
	Slow if n is very large