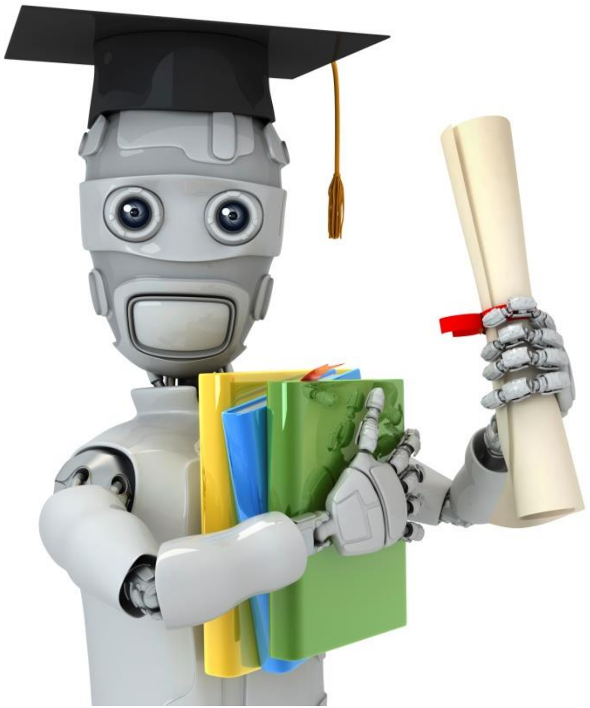


Machine Learning

Linear Regression with multiple variables

Multiple features

Multiple features (variables).

Size (feet²)	Price ($1000)
x	y
2104	460
1416	232
1534	315
852	178
…	…

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

Multiple features (variables).

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	3	2	30	315
852	2	1	36	178
…	…	…	…	…

What is $x_3^{(4)}$?

A. 1
B. 30
C. 2
D. 852

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.

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

$$x_2^{(2)} = 3, x_3^{(2)} = 2$$

Hypothesis:

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

Multiple features:

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

E.g.

$$h_\theta(x) = 80 + 0.1x_1 + 0.01x_2 + 3x_2 - 2x_4$$

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

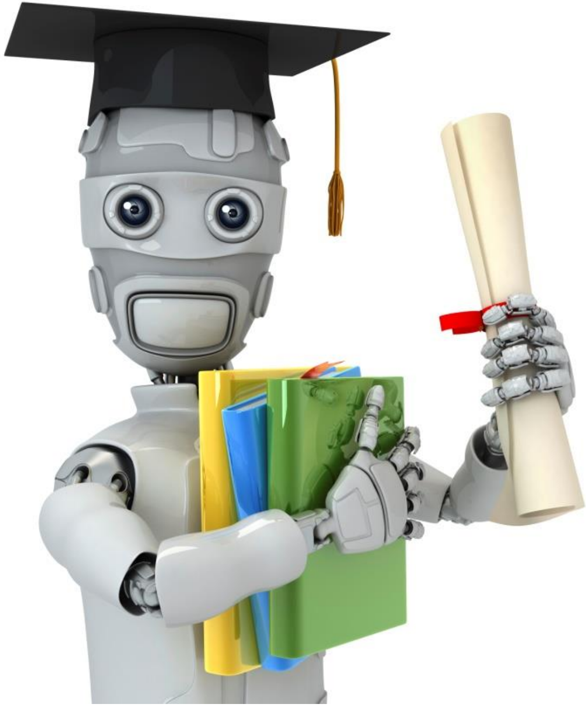
For convenience of notation, define $x_0 = 1$.

$$X = \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\underbrace{[\theta_0, \theta_1, \ldots, \theta_n]}_{\theta^T} \quad \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix}}_{X}$$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n = \theta^T X$$

Multivariate linear regression.



Machine Learning

Linear Regression with multiple variables

Gradient descent for multiple variables

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:

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

Gradient descent:

Repeat $\{$

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

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

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 $j = 0, \ldots, n$)

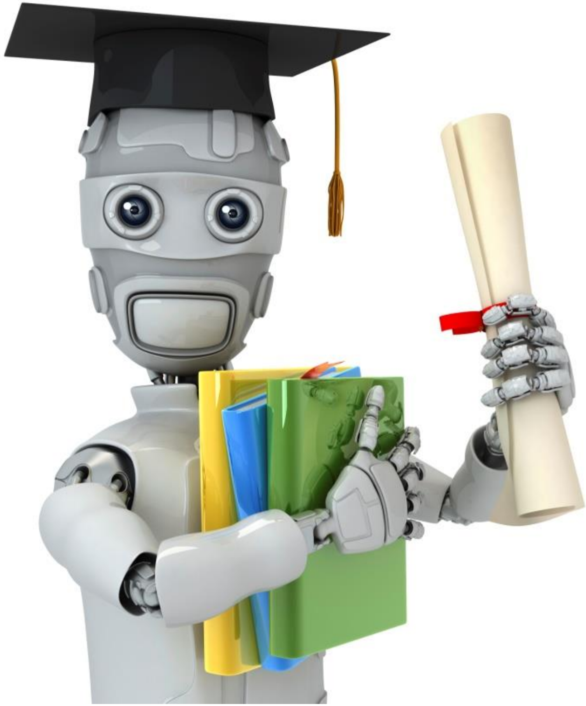
$\}$

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

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

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

$\ldots$



Machine Learning

Linear Regression with multiple variables

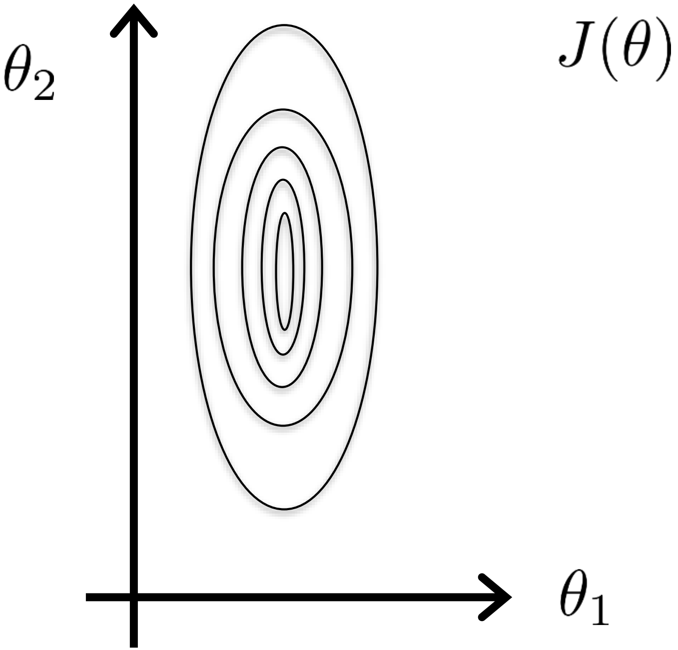
Gradient descent in practice I: Feature Scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

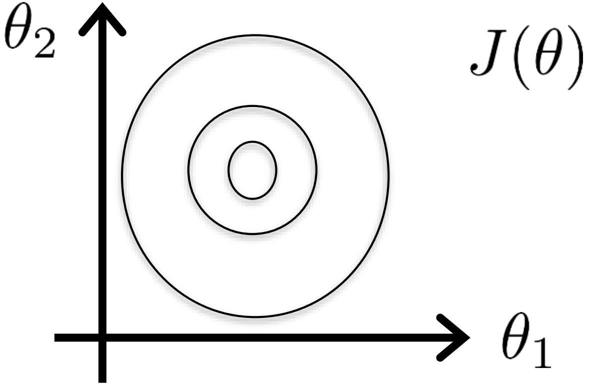
E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)



$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$



Feature Scaling

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

√

$$x_0 = 1$$

$$0 < x_1 < 3$$

$$-2 < x_2 < 0.5$$

×

$$-100 < x_3 < 100$$

$$-0.0001 < x_4 < 0.0001$$

Mean normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g. $$x_1 = \frac{size - 1000}{2000}$$

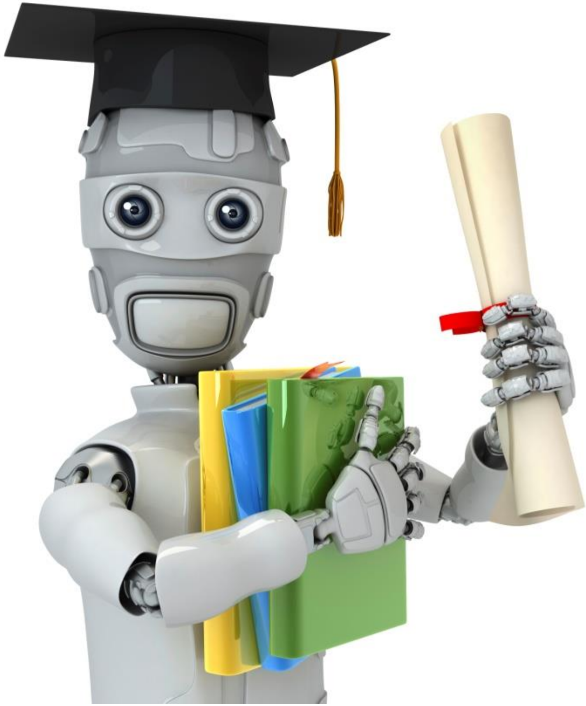
$$x_2 = \frac{\#bedrooms - 2}{5}$$

$$-0.5 \le x_1 \le 0.5, -0.5 \le x_2 \le 0.5$$

Scaling Rule： $x_n \leftarrow \dfrac{x_n - \mu_n}{s_n}$

μ_n: *Average value of* x_i *in training set*

s_n: *Standard deviation*



Machine Learning

Linear Regression with multiple variables

Gradient descent in practice II: Learning rate

Gradient descent

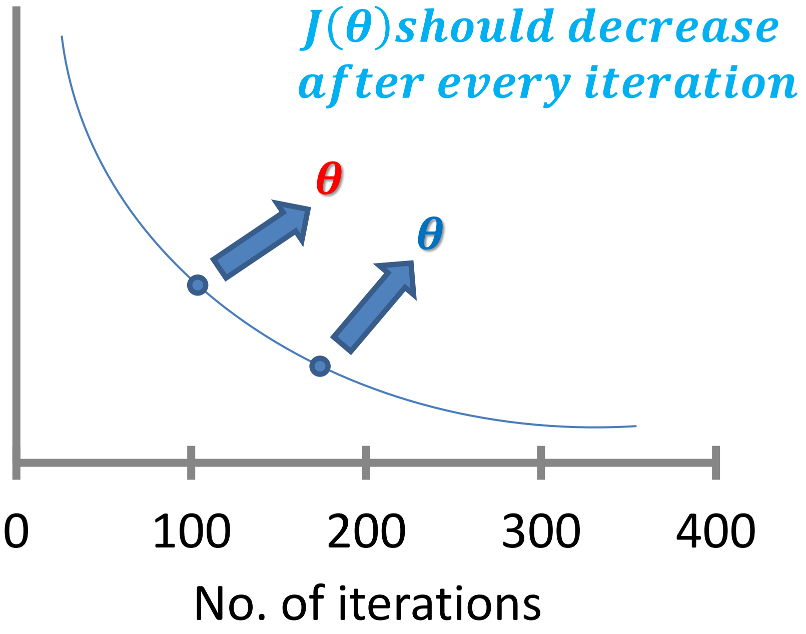
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- "Debugging": How to make sure gradient descent is working correctly.

- How to choose learning rate α.

Making sure gradient descent is working correctly.

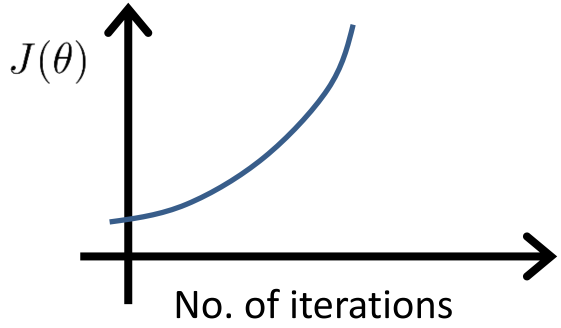
$$\min_{\theta} J(\theta)$$



Example automatic convergence test:

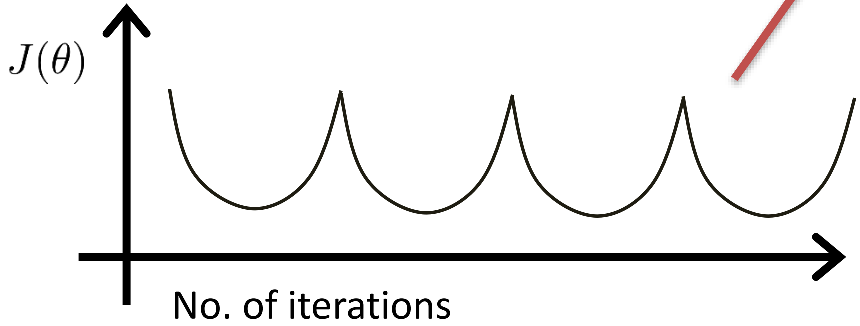
Declare convergence if $J(\theta)$ decreases by less than $\underset{\varepsilon}{10^{-3}}$ in one iteration.

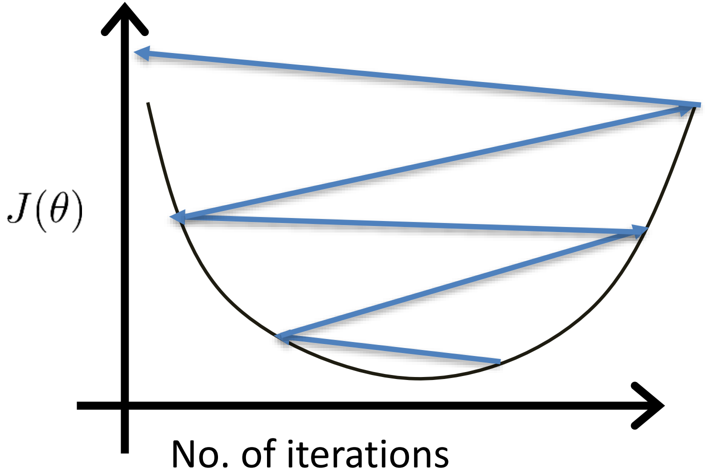
Making sure gradient descent is working correctly.



Gradient descent not working.

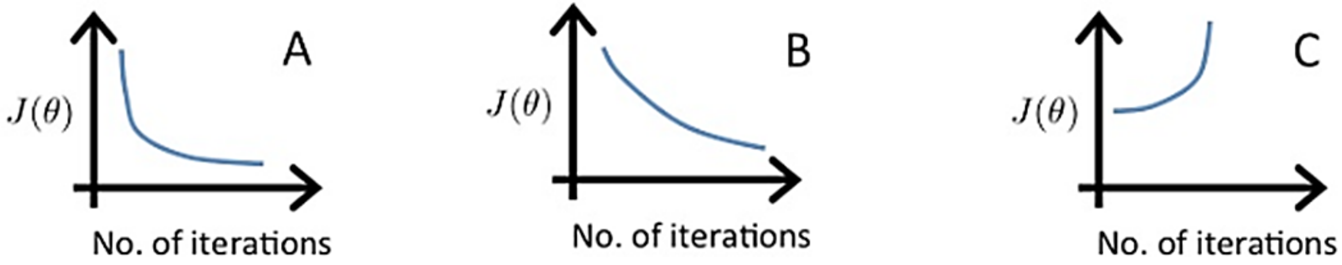
Use smaller α.





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

Suppose a friend ran gradient descent three times, with $\alpha = 0.01$, $\alpha = 0.1$, and $\alpha = 1$, and got the following three plots (labeled A, B, and C):



Which plots corresponds to which values of α?

A. A is $\alpha = 0.01$, B is $\alpha = 0.1$, C is $\alpha = 1$.

B. A is $\alpha = 0.1$, B is $\alpha = 0.01$, C is $\alpha = 1$.

C. A is $\alpha = 1$, B is $\alpha = 0.01$, C is $\alpha = 0.1$.

D. A is $\alpha = 1$, B is $\alpha = 0.1$, C is $\alpha = 0.01$.

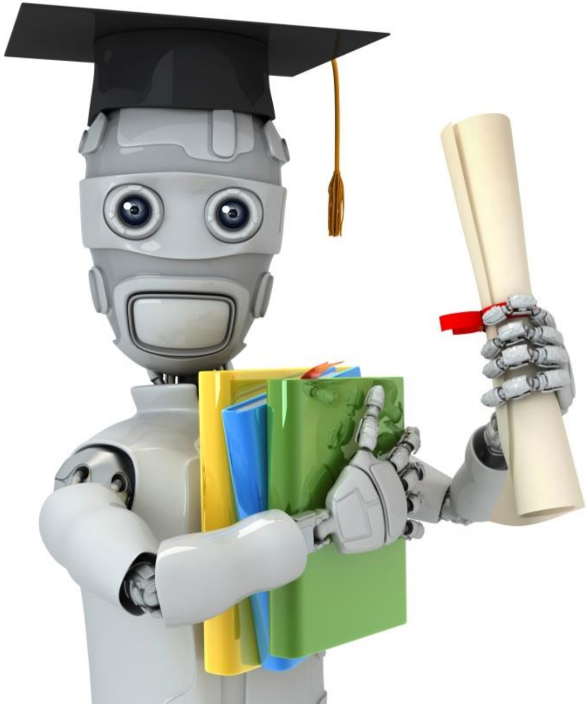
B

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

$$\ldots, 0.001, \qquad , 0.01, \qquad , 0.1, \qquad , 1, \ldots$$



Machine Learning

Linear Regression with multiple variables

Features and polynomial regression

Housing prices prediction

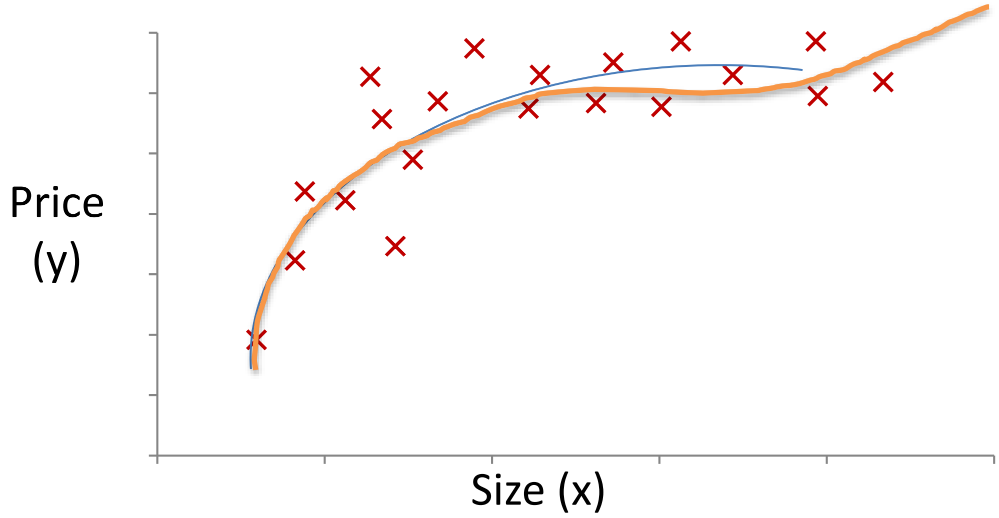
$$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$$

$$\boldsymbol{Area: X = frontage * depth}$$

$$\boldsymbol{h_\theta(x) = \theta_0 + \theta_1 x}$$

Land Area

Polynomial regression



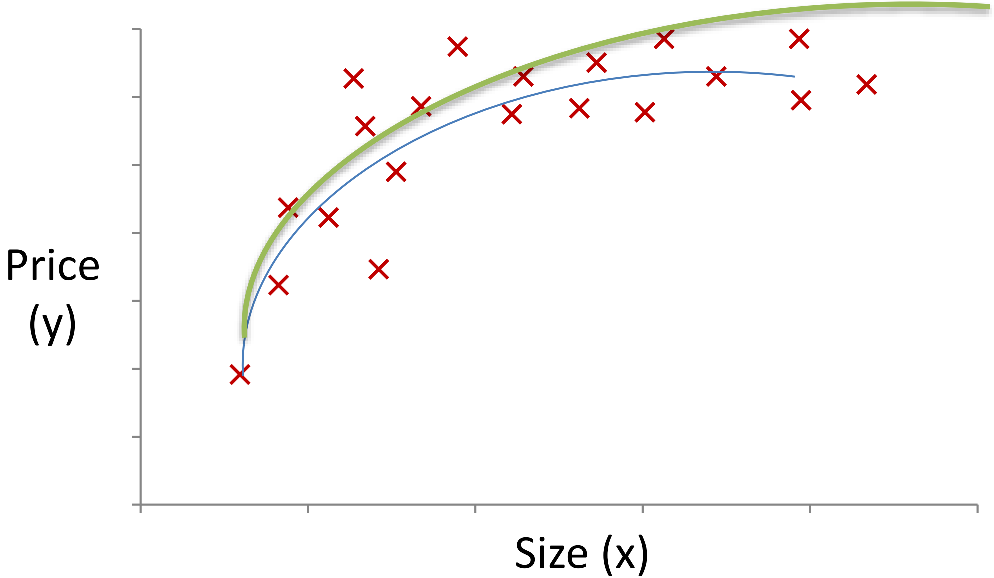
$$\theta_0 + \theta_1 x + \theta_2 x^2$$

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

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3 \end{aligned}$$

$$x_1 = (size)$$
$$x_2 = (size)^2$$
$$x_3 = (size)^3$$

Choice of features



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

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

Suppose you want to predict a house's price as a function of its size. Your model is

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

Suppose size ranges from 1 to 1000 (feet^2). You will implement this by fitting a model

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2.$$

Finally, suppose you want to use feature scaling (without mean normalization).

Which of the following choices for x_1 and x_2 should you use? (Note: $\sqrt{1000} \approx 32$.)

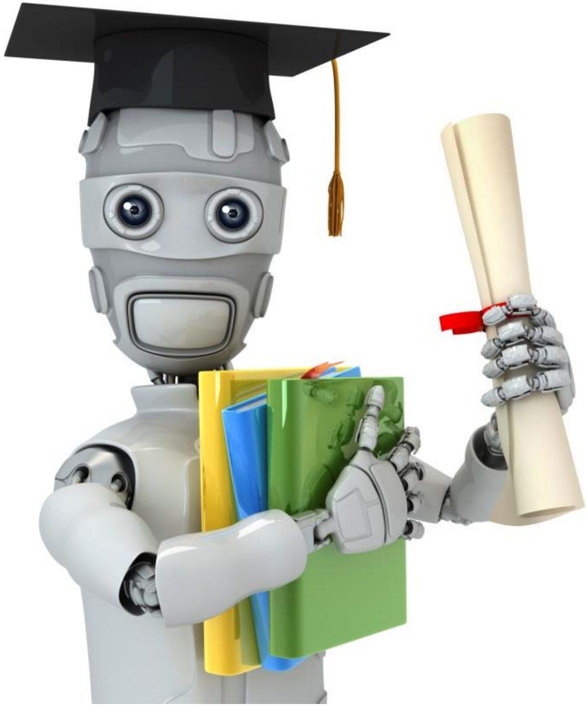
A. $x_1 = \text{size},\ x_2 = 32\sqrt{(\text{size})}$

B. $x_1 = 32(\text{size}),\ x_2 = \sqrt{(\text{size})}$

C. $x_1 = \frac{\text{size}}{1000},\ x_2 = \frac{\sqrt{(\text{size})}}{32}$

D. $x_1 = \frac{\text{size}}{32},\ x_2 = \sqrt{(\text{size})}.$

C

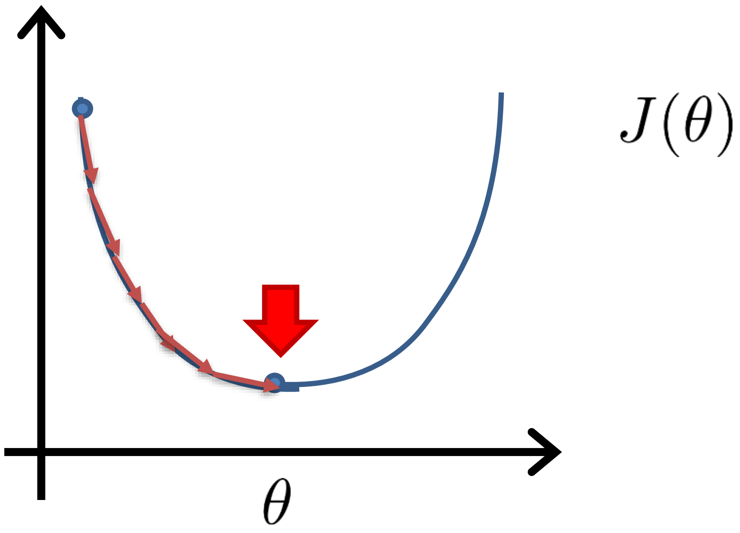


Machine Learning

Linear Regression with multiple variables

Normal equation

Gradient Descent



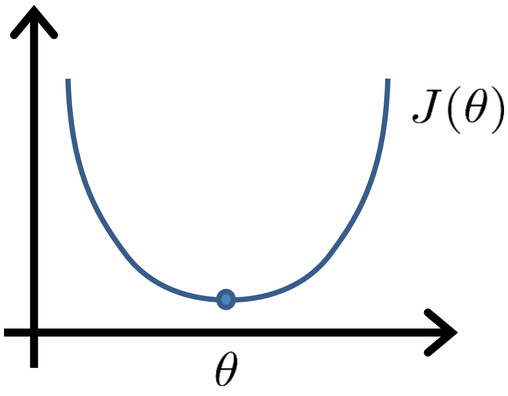
Normal equation: Method to solve for θ analytically.

Intuition: If 1D $(\theta \in \mathbb{R})$

$$J(\theta) = a\theta^2 + b\theta + c$$

$$set: \frac{\partial}{\partial \theta_j} J(\theta_j) = 0$$

$$solve\ for\ \theta_j$$



$\theta \in \mathbb{R}^{n+1}$ $\qquad J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

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

Solve for $\theta_0, \theta_1, \ldots, \theta_n$

Examples: $m = 4.$

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) 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

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

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

Examples: $m = 5.$

	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
1	3000	4	1	38	540

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ 1 & 3000 & 4 & 1 & 38 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \\ 540 \end{bmatrix}$$

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

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

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

$$X = \begin{bmatrix} -\left(x^{(1)}\right)^T - \\ -\left(x^{(2)}\right)^T - \\ \cdots \\ \cdots \\ \cdots \\ -\left(x^{(\mathrm{m})}\right)^T - \end{bmatrix} \qquad \textit{design matrix}$$

E.g. If $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

Suppose you have the training in the table below:

age (x_1)	height in cm (x_2)	weight in kg (y)
4	89	16
9	124	28
5	103	20

You would like to predict a child's weight as a function of his age and height with the model

$$\text{weight} = \theta_0 + \theta_1 \text{age} + \theta_2 \text{height}.$$

What are X and y?

A. $X = \begin{bmatrix} 4 & 89 \\ 9 & 124 \\ 5 & 103 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$

B. $X = \begin{bmatrix} 1 & 4 & 89 \\ 1 & 9 & 124 \\ 1 & 5 & 103 \end{bmatrix}, y = \begin{bmatrix} 1 & 16 \\ 1 & 28 \\ 1 & 20 \end{bmatrix}$

C. $X = \begin{bmatrix} 4 & 89 & 1 \\ 9 & 124 & 1 \\ 5 & 103 & 1 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$

D. $X = \begin{bmatrix} 1 & 4 & 89 \\ 1 & 9 & 124 \\ 1 & 5 & 103 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$

D

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

$(X^T X)^{-1}$ is inverse of matrix $X^T X$.

$$Set\ A = X^T X$$

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

Matlab\Octave:

```
pinv(X'*X)*X'*y
```

Python:

```python
import numpy as np
def normalEqn(X, y):
    theta = np.linalg.inv(X.T@X)@X.T@y
    return theta
```

m **training examples,** n **features.**

Gradient Descent

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

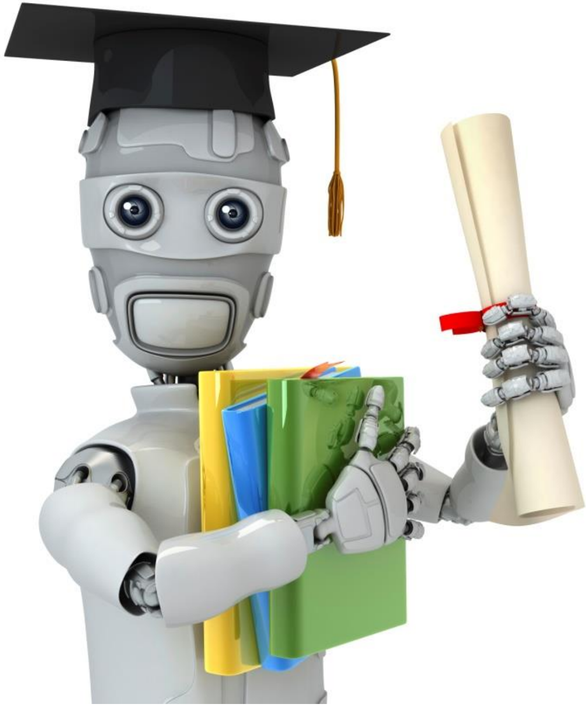
Normal Equation

- No need to choose α.
- Don't need to iterate.
- Need to compute $(X^TX)^{-1}$
- Slow if n is very large.

对于那些不可逆的矩阵（通常是因为特征之间不独立，如同时包含英尺为单位的尺寸和米为单位的尺寸两个特征，也有可能是特征数量大于训练集的数量），正规方程方法是不能用的。

梯度下降与正规方程的比较：

梯度下降	正规方程
需要选择学习率α	不需要
需要多次迭代	一次运算得出
当特征数量n大时也能较好适用	需要计算$\left(X^TX\right)^{-1}$ 如果特征数量n较大则运算代价大，因为矩阵逆的计算时间复杂度为$O\left(n^3\right)$，通常来说当n小于10000 时还是可以接受的
适用于各种类型的模型	只适用于线性模型，不适合逻辑回归模型等其他模型



Machine Learning

Linear Regression with multiple variables

Normal equation and non-invertibility (optional)

Normal equation

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

- What if $X^T X$ is non-invertible? (singular/ degenerate)

What if $X^T X$ is non-invertible?

- Redundant features (linearly dependent).
  E.g. $x_1 =$ size in feet^2
  $x_2 =$ size in m^2

- Too many features (e.g. $m \leq n$).
  - Delete some features, or use regularization.