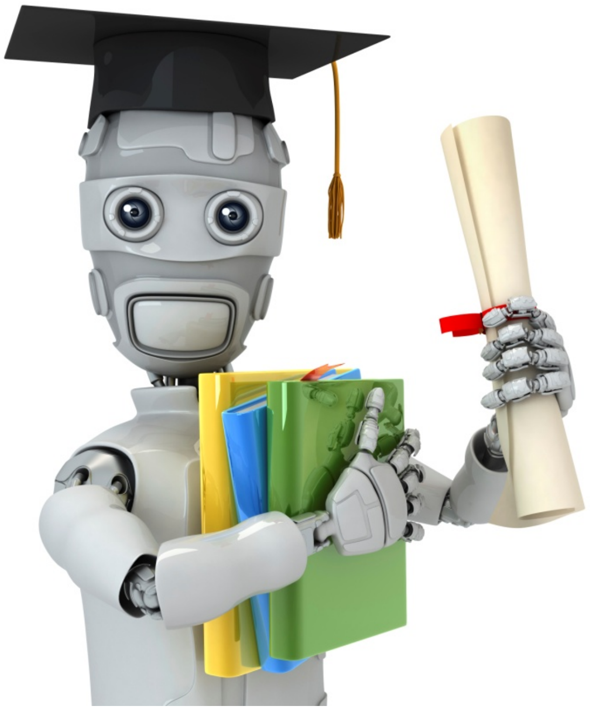


Machine Learning

Linear Regression with multiple variables

Multiple features

Multiple features (variables).

Size (feet^2)	Price (\$1000)
x	y
2104	460
1416	232
1534	315
852	178
…	…

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

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 $\quad n = 4$
- $x^{(i)}$ = input (features) of i^{th} training example.
- $x_j^{(i)}$ = value of feature j in i^{th} training example.

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

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

Andrew Ng

Hypothesis:

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

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

E.g. $h_\theta(x) = 80 + 0.1 x_1 + 0.01 x_2 + 3 x_3 - 2 \underset{\text{age}}{x_4}$

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

For convenience of notation, define $\boxed{x_0 = 1.}$ $(x_0^{(i)} = 1)$

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

$\underbrace{[\theta_0 \ \theta_1 \cdots \theta_n]}_{\theta^T}$ $\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$

$(n+1)\times 1$ matrix $\qquad x$

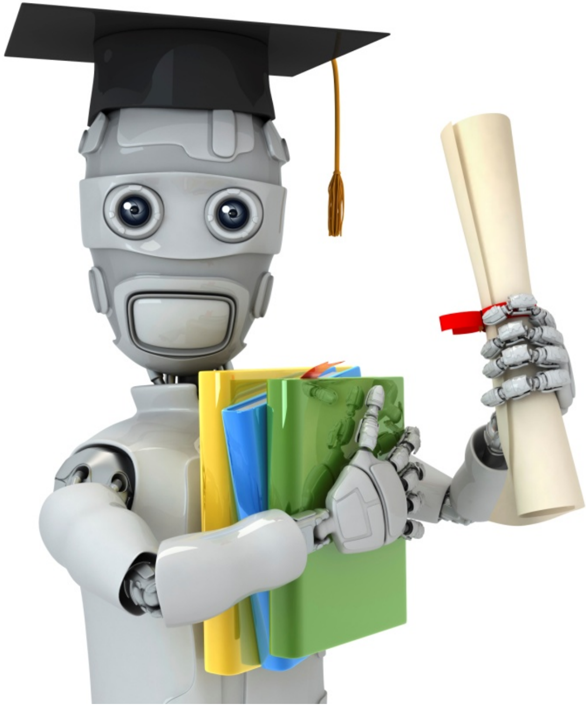
$\theta^T x$

$$h_\theta(x) = \theta_0 \overset{=1}{x_0} + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$= \boxed{\theta^T x}$$

Multivariate linear regression.

Andrew Ng



Machine Learning

Linear Regression with multiple variables

Gradient descent for multiple variables

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

$x_0 = 1$

Parameters: $\cancel{\theta_0, \theta_1, \ldots, \theta_n}$ θ n+1 - dimensional vector

Cost function:

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

$J(\theta)$

Gradient descent:

Repeat $\{$

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

$\}$

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

Andrew Ng

Gradient Descent

Previously (n=1):

Repeat $\{$

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

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

$x^{(i)}$

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

$\frac{\partial}{\partial\theta_j}J(\theta)$

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

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

$\}$

$x_0^{(i)} = 1$

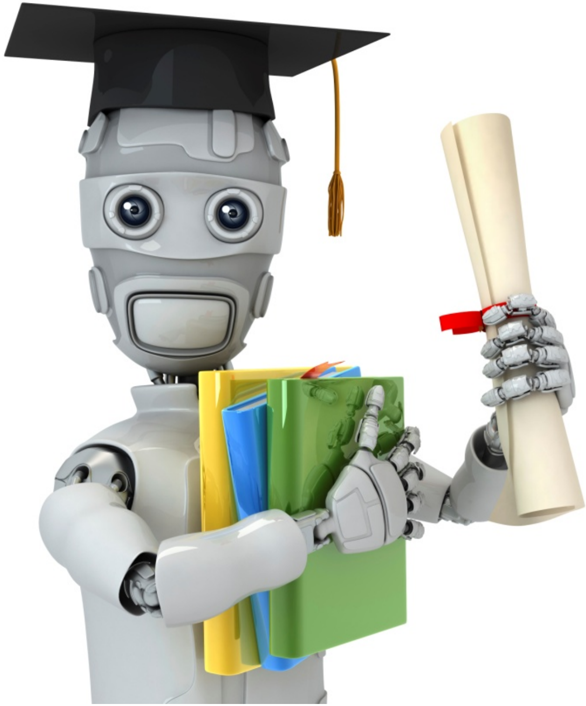
$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

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

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

$\ldots$

Andrew Ng



Machine Learning

Linear Regression with multiple variables

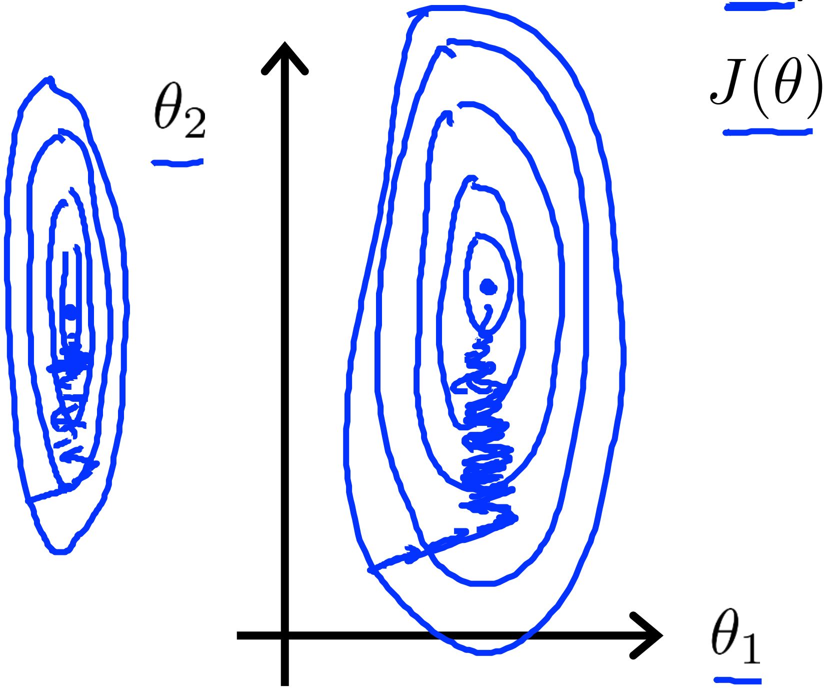
Gradient descent in practice I: Feature Scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

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

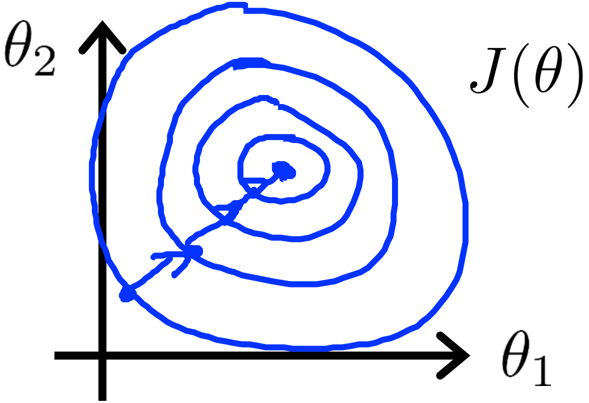
x_2 = number of bedrooms (1-5)



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

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

$0 \le x_1 \le 1 \qquad 0 \le x_2 \le 1$



Andrew Ng

Feature Scaling

Get every feature into approximately a $\boxed{-1 \le x_i \le 1}$ range.

$x_0 = 1$

$0 \le x_1 \le 3$ ✓

$-2 \le x_2 \le 0.5$ ✓

$-100 \le x_3 \boxed{100}$ ✗

$-0.0001 \le x_4 \le \boxed{0.0001}$ ✗

-3 to 3 ✓

$-\frac{1}{3}$ to $\frac{1}{3}$ ✓

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

Average size = 100

$\boxed{x_2 = \frac{\#bedrooms - 2}{5}}$ 4

1-5 bedrooms

$\boxed{-0.5 \leq x_1 \leq 0.5}$, $\boxed{-0.5 \leq x_2 \leq 0.5}$

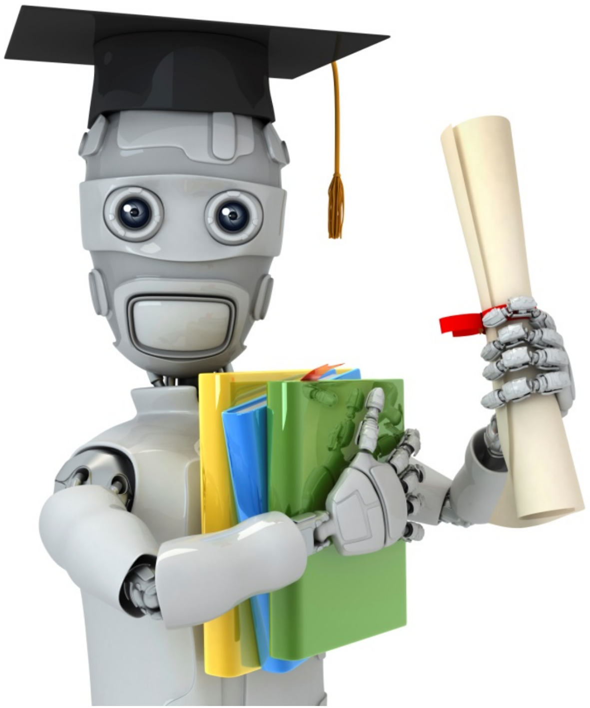
$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1}$

avg value of x_1 in training set

range (max - min)

(or standard deviation)

$x_2 \leftarrow \frac{x_2 - \mu_1}{s_2}$



Machine Learning

Linear Regression with multiple variables

Gradient descent in practice II: Learning rate

Gradient descent

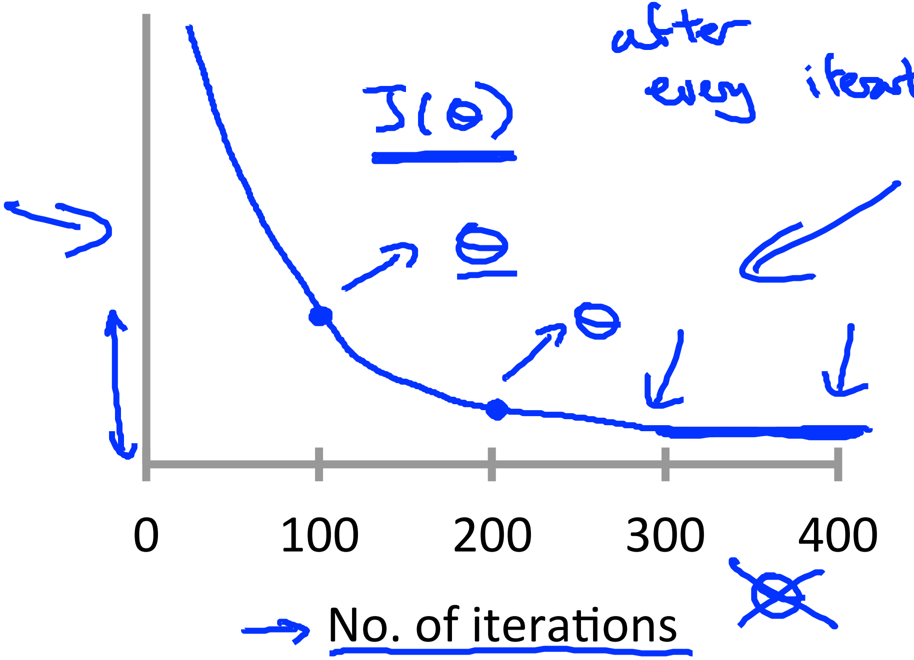
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- "Debugging": How to make sure gradient descent is working correctly.
- How to choose learning rate $\boxed{\alpha.}$

Making sure gradient descent is working correctly.

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

J(θ) should decrease after every iteration



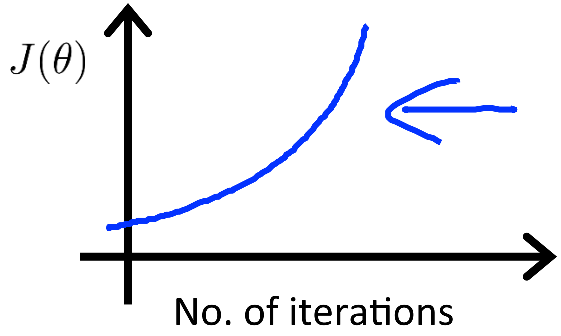
30, 3000, 3,000,000

→ Example automatic convergence test:

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

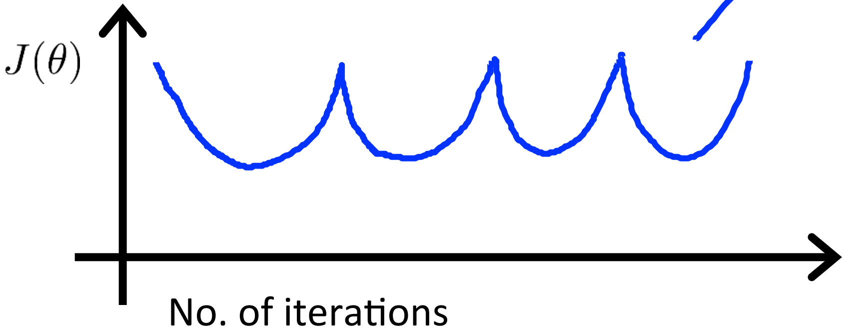
ε

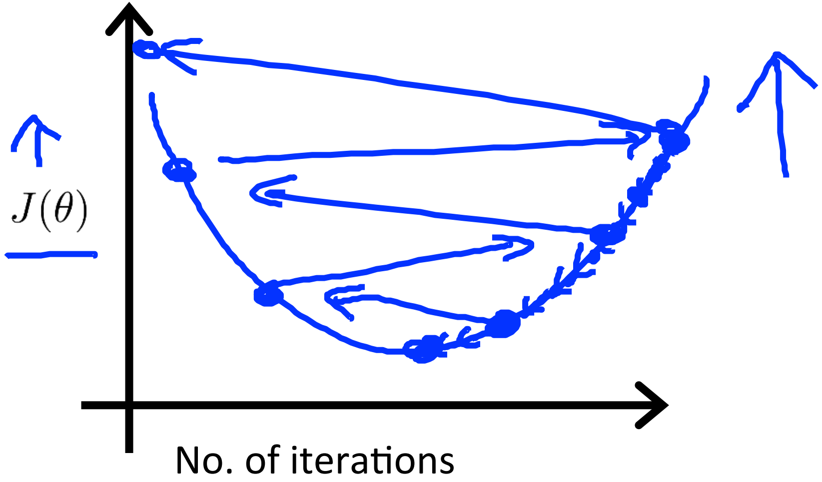
Making sure gradient descent is working correctly.



Gradient descent not working.

$\boxed{\text{Use smaller } \alpha.}$

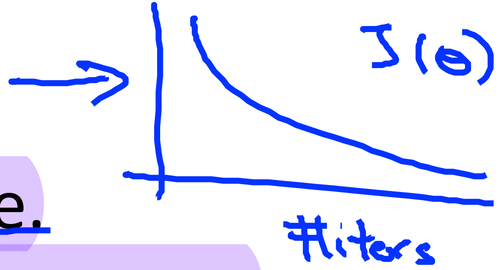




- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Summary:

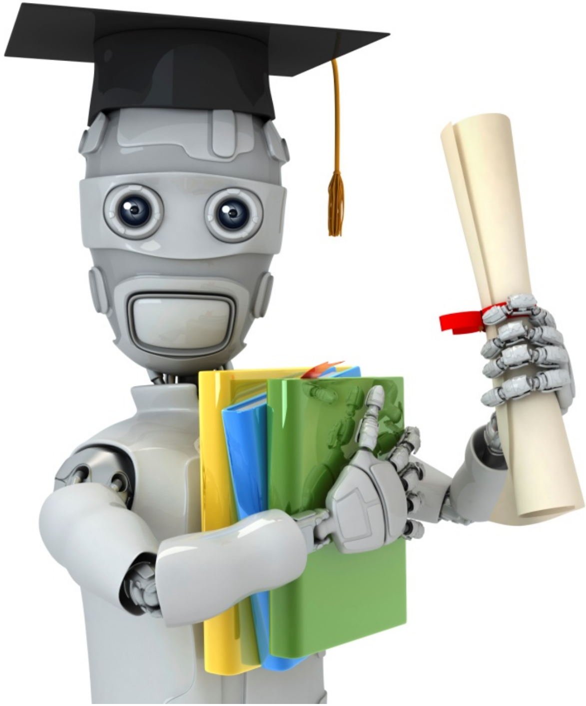
- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge. (Slow converge also possible)



To choose α, try

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

3x ≈3x 3x ≈3x



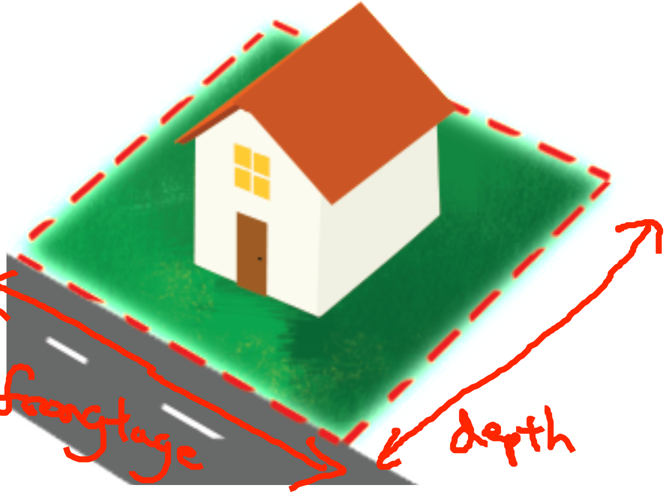
Machine Learning

Linear Regression with multiple variables

Features and polynomial regression

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\boxed{frontage}}_{\underline{x}_1} + \theta_2 \times \underbrace{\boxed{depth}}_{\underline{x}_2}$$



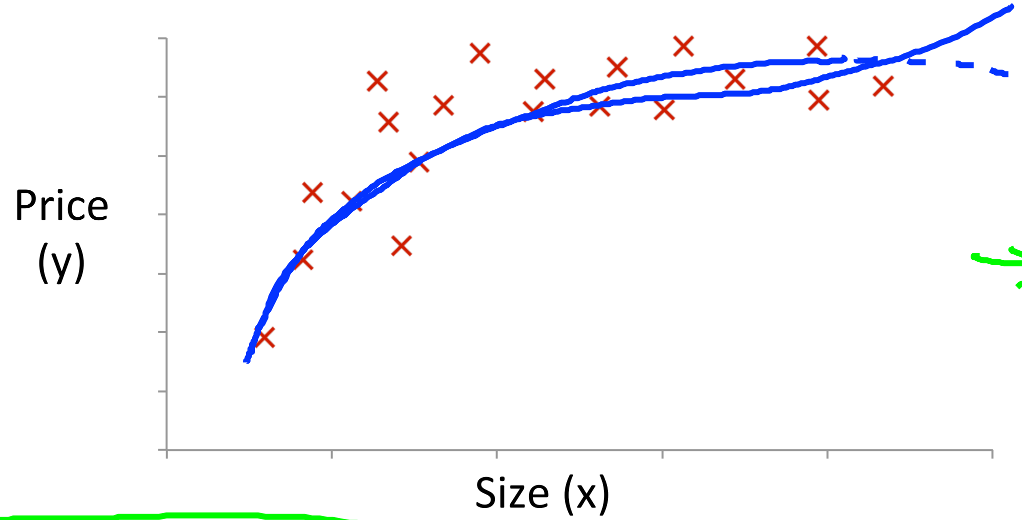
Area

$x = \underline{frontage * depth}$

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

land area

Polynomial regression



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

$$\boxed{\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3}$$

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

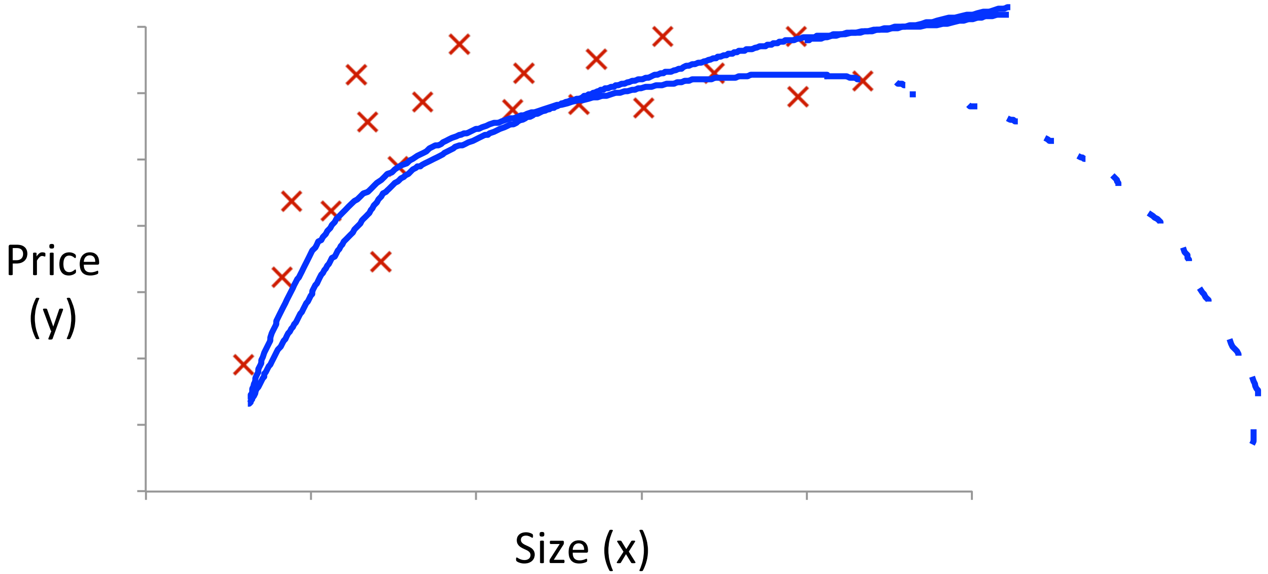
$$x_1 = (size)$$
$$x_2 = (size)^2$$
$$x_3 = (size)^3$$

Size: 1 - 1000

size^2: 1 - 1000,000

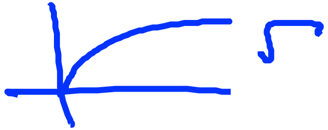
Size^3: 1 - 10^9

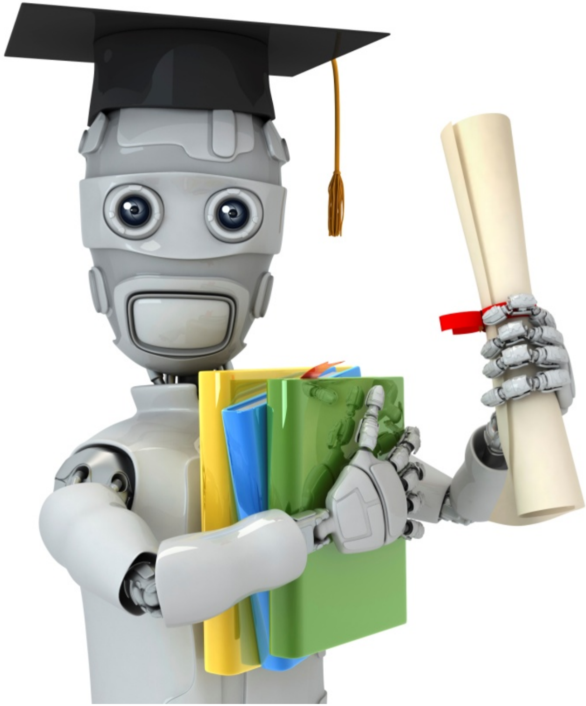
Choice of features



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

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



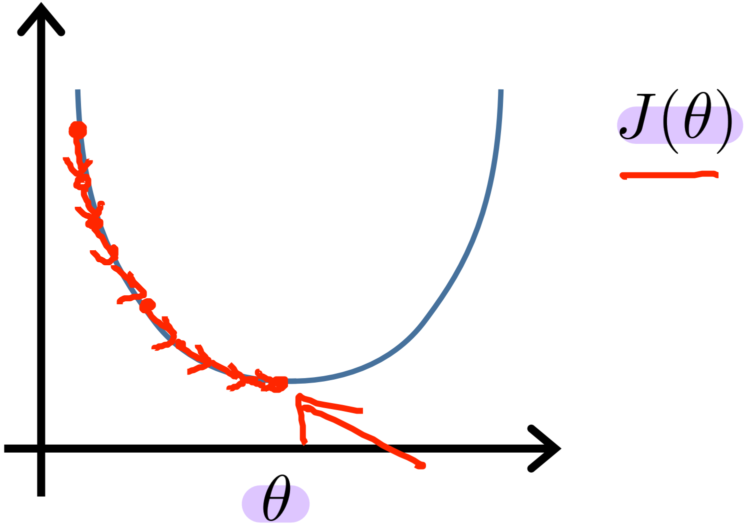


Machine Learning

Linear Regression with multiple variables

Normal equation

Gradient Descent



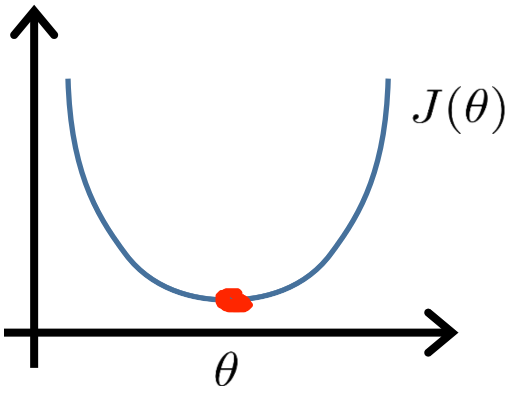
Normal equation: Method to solve for θ analytically.

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

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

$$\frac{\partial}{\partial \theta} J(\theta) = \cdots \overset{\text{set}}{=} 0$$

Solve for θ



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

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

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

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

$$X = \underbrace{\begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}}_{m \times (n+1)} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

m-dimensional vector

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

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

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

$$X = \begin{bmatrix} \text{---} & (x^{(1)})^T & \text{---} \\ \text{---} & (x^{(2)})^T & \text{---} \\ & \vdots & \\ \text{---} & (x^{(m)})^T & \text{---} \end{bmatrix}$$

(design matrix)

$m \times (n+1)$

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

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

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

$$\boxed{\theta = \boxed{(X^TX)^{-1}X^Ty}}$$

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

Set $A = X^TX$

$$\boxed{(X^TX)^{-1}} = A^{-1}$$

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

X' X^T

pinv$(X^T*X)*X^T*y$

~~Feature Scaling~~

$0 \le x_1 \le 1$

$0 \le x_2 \le 1000$

$0 \le x_3 \le 10^{-5}$ ✓

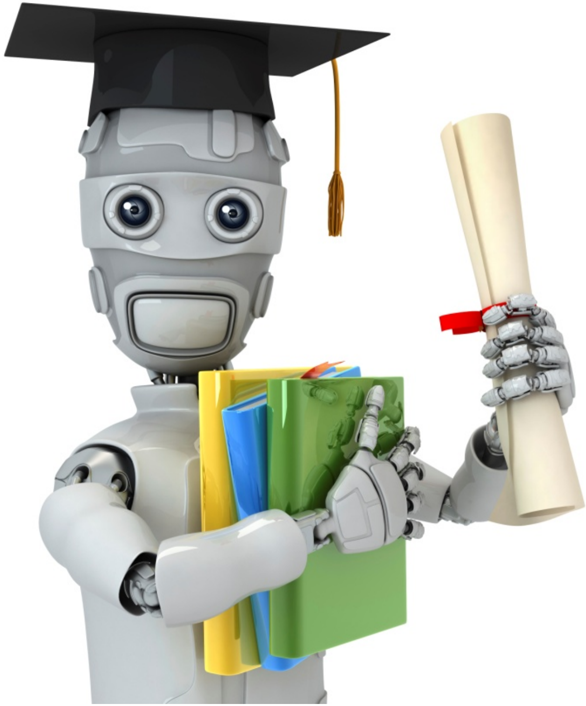
$\theta = (X^TX)^{-1}X^Ty$

$\min_\theta J(\theta)$

Andrew Ng

m training examples, n features.

<u>Gradient Descent</u>	<u>Normal Equation</u>
• Need to choose α.	• No need to choose α.
• Needs many iterations.	• Don't need to iterate.
• Works well even when n is large.	• Need to compute $\boxed{(\boxed{X^TX})^{-1}}$ $n \times n$ $O(n^3)$
$n = 10^6$	• Slow if n is very large.
	$n = 100$
	$n = 1000$
	$n = 10000$



Machine Learning

Linear Regression with multiple variables

Normal equation and non-invertibility (optional)

Normal equation

$$\theta = (X^TX)^{-1}X^Ty$$

X^TX

- What if $\boxed{X^TX}$ is non-invertible? (singular/ degenerate)
- Octave: `pinv(X'*X)*X'*y`

θ

pinv

inv

What if $\boxed{X^T X}$ is non-invertible?

- Redundant features (linearly dependent).

  E.g. x_1 = size in feet^2

  ~~x_2 = size in m^2~~

  $x_1 = (3.28)^2 x_2$

  1m = 3.28 feet

  $m = 10$

  $n = 100$

  $\theta \in \mathbb{R}^{101}$

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

  later