Linear Regression with multiple variables

Multiple features

Machine Learning

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
...	...	...	...	...

$m = 47$

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

$x_3^{(2)} = 2$

Notation:

n = number of features $n = 4$

$x^{(i)}$ = input (features) of ith training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

Hypothesis:

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

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

e.g. $h_\theta(x) = 80 + 0.1 x_1 + 0.01 x_2 + 3 x_3 - 2 x_4$

age

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

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

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

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

$(n+1) \times 1$ matrix

$\theta^T x$

x

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

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

Multivariate linear regression. $\leftarrow$

Machine Learning

Linear Regression with multiple variables

Gradient descent for multiple variables

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

$\rightarrow x_0 = 1$

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

Cost function:

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

$J(\theta)$

Gradient descent:

Repeat $\{$

$\rightarrow \quad \theta_j := \theta_j - \alpha \dfrac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n) \quad J(\theta)$

$\}$

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

Gradient Descent

Previously (n=1):

Repeat {

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

$$\underbrace{\qquad}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

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

$$x^{(i)}$$

(simultaneously update θ_0, θ_1)

}

New algorithm $(n \geq 1)$:

Repeat {

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

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

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

}

$$x_0^{(i)} = 1$$

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

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

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

…

Linear Regression with multiple variables

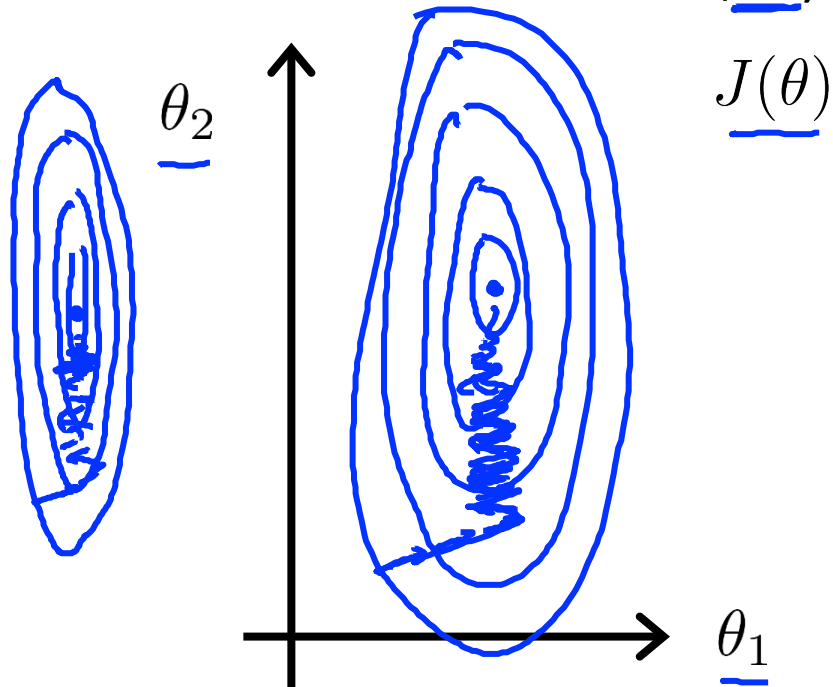## Gradient descent in practice I: Feature Scaling

Machine Learning

Feature Scaling

Idea: Make sure features are on a similar scale.
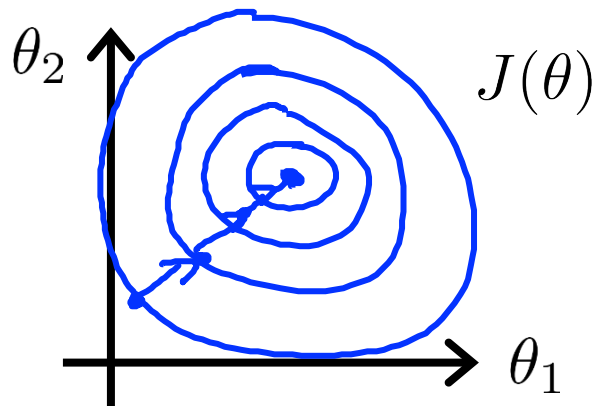
E.g. x_1 = size (0-2000 feet²)

x_2 = number of bedrooms (1-5)

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

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

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

$J(\theta)$

$J(\theta)$

Feature Scaling

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

$x_0 = 1$

$0 \leq x_1 \leq 3$ ✓

$-2 \leq x_2 \leq 0.5$ ✓

$-100 \leq x_3 \boxed{100}$ ✗

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

$-3 \text{ to } 3$ ✓

$-\frac{1}{3} \text{ to } \frac{1}{3}$ ✓

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. $\quad x_1 = \dfrac{size - 1000}{2000}$

$x_2 = \dfrac{\#bedrooms - 2}{5 \quad 4}$

$-0.5 \leq x_1 \leq 0.5, \quad -0.5 \leq x_2 \leq 0.5$

Average size $= 100$

$1 - 5$ bedrooms

$$x_1 \leftarrow \frac{x_1 - \mu_1}{S_1}$$

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

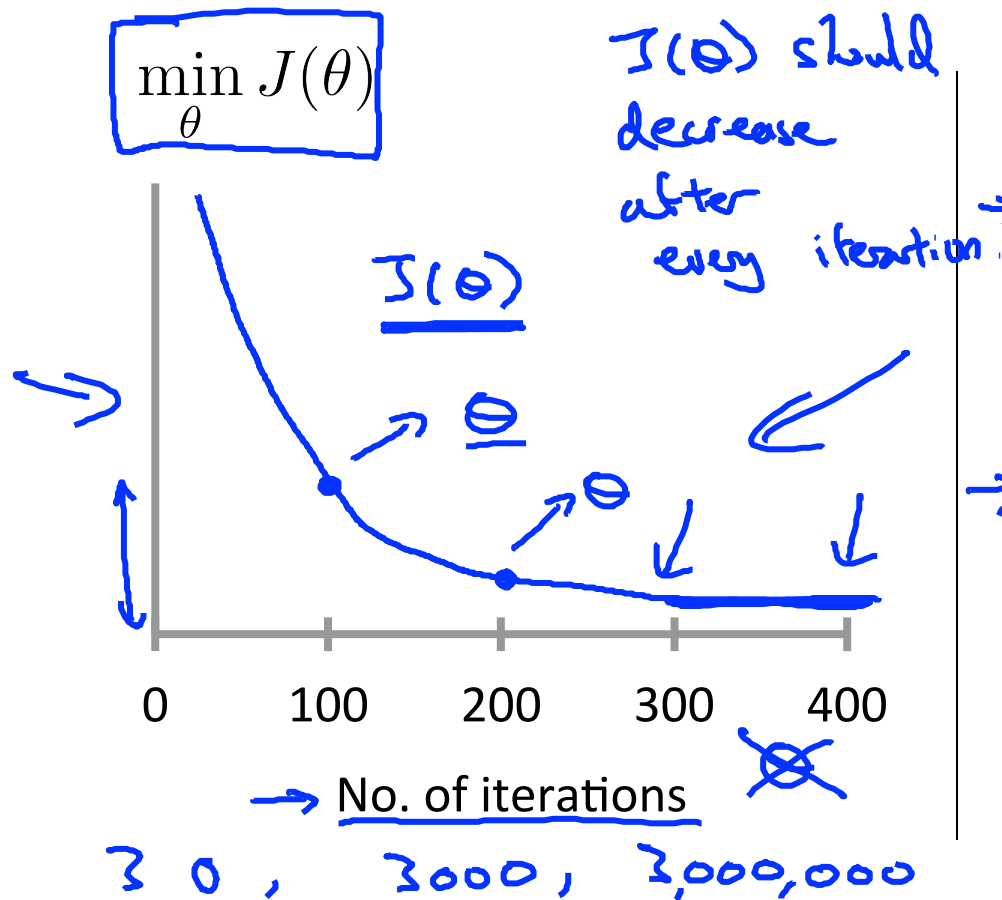# Linear Regression with multiple variables

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 α.
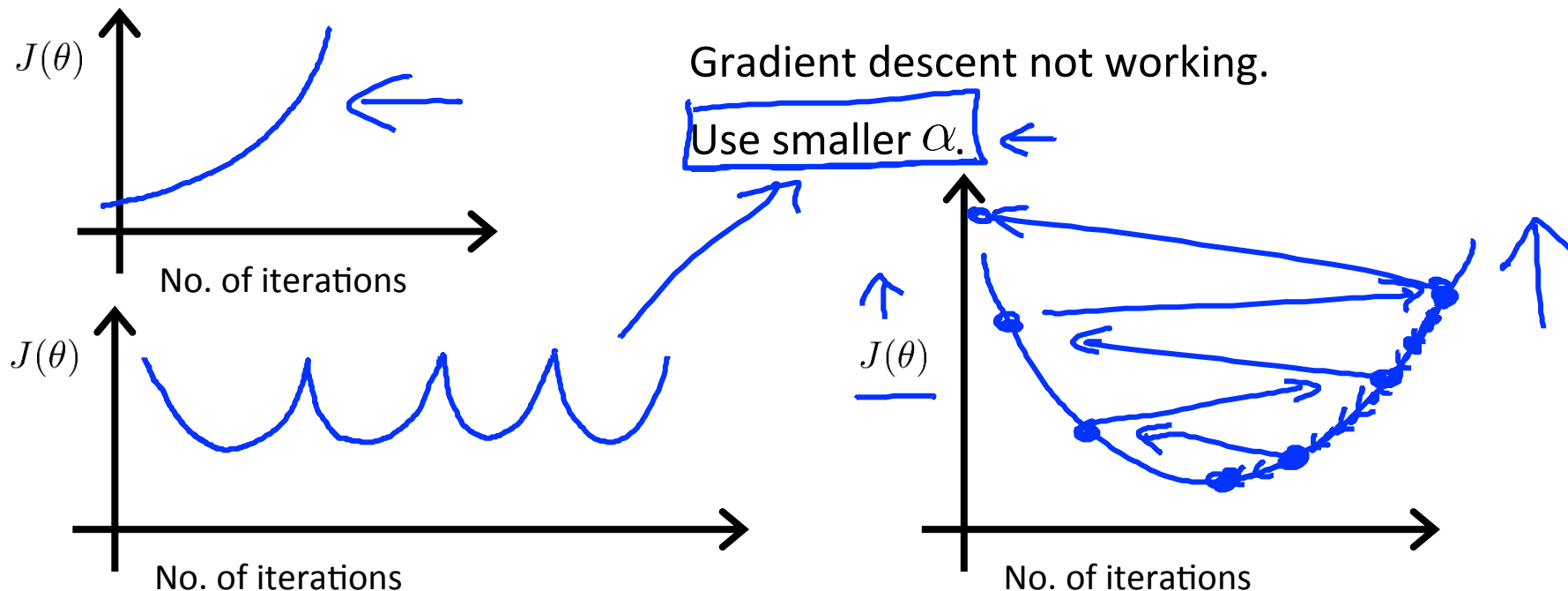
Making sure gradient descent is working correctly.

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

$J(\theta)$ should decrease after every iteration.

$J(\theta)$

θ

θ

No. of iterations

3 0 ,   3000,   3,000,000

→ Example automatic convergence test:

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

ε

Making sure gradient descent is working correctly.

Gradient descent not working.

Use smaller α.

- For <u>sufficiently small α</u>, $J(\theta)$ should decrease on <u>every iteration.</u>
- 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)

$J(\theta)$

#iters

To choose α, try

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

3x   ≈3x   3x   ≈3x

Linear Regression with multiple variables

Features and polynomial regression

Machine Learning

Housing prices prediction

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

x_1

x_2

Area

$x = \underline{frontage \; * \; depth}$

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

$\curvearrowleft$ land area

frontage

depth

Andrew Ng
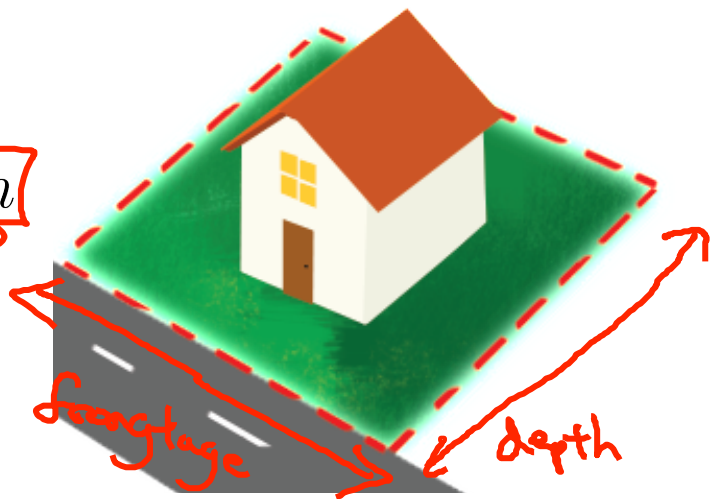
Polynomial regression
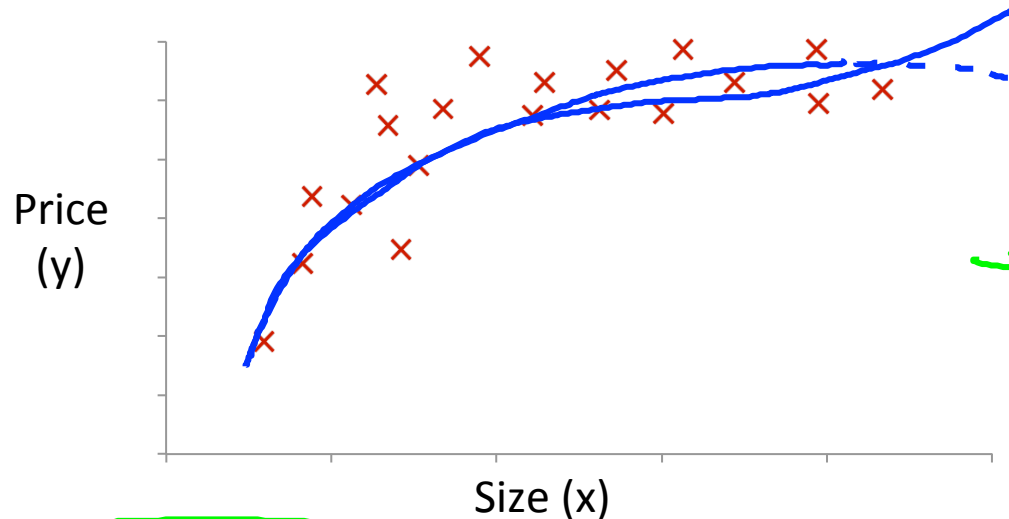
$$\theta_0 + \theta_1 x + \theta_2 x^2$$

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

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

$x_1 = (size)$

$x_2 = (size)^2$

$x_3 = (size)^3$

Size: $1 - 1000$

$Size^2$: $1 - 1000,000$

$Size^3$: $1 - 10^9$

Andrew Ng

Choice of features
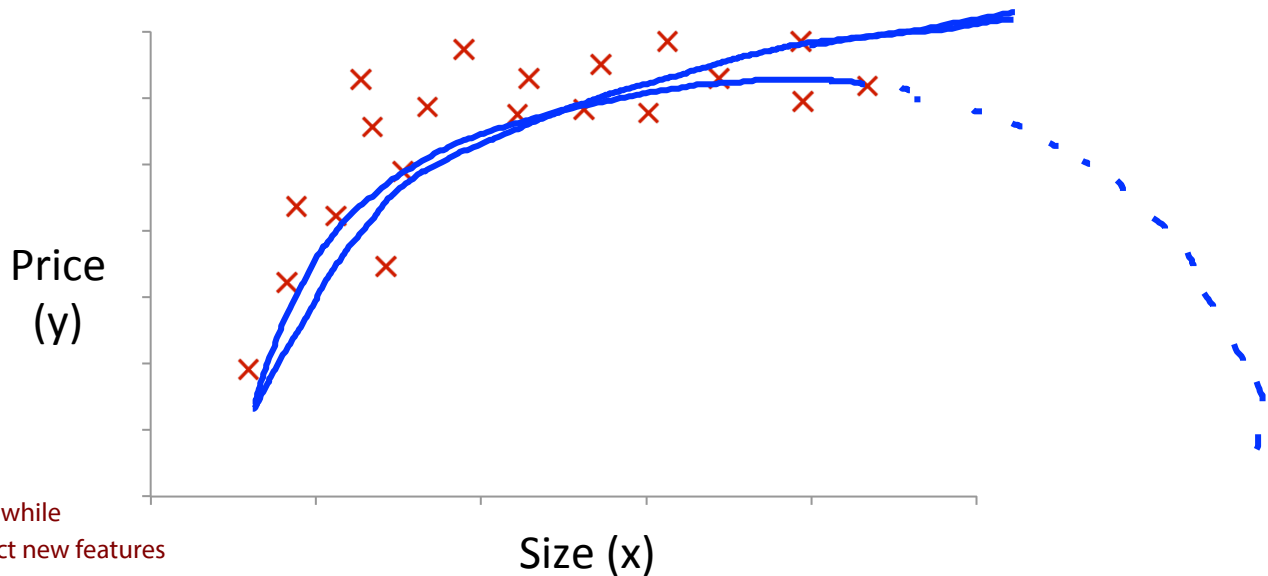
Price
(y)

Size (x)

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

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

Linear Regression with multiple variables

Normal equation

Machine Learning

Gradient Descent

$J(\theta)$

θ
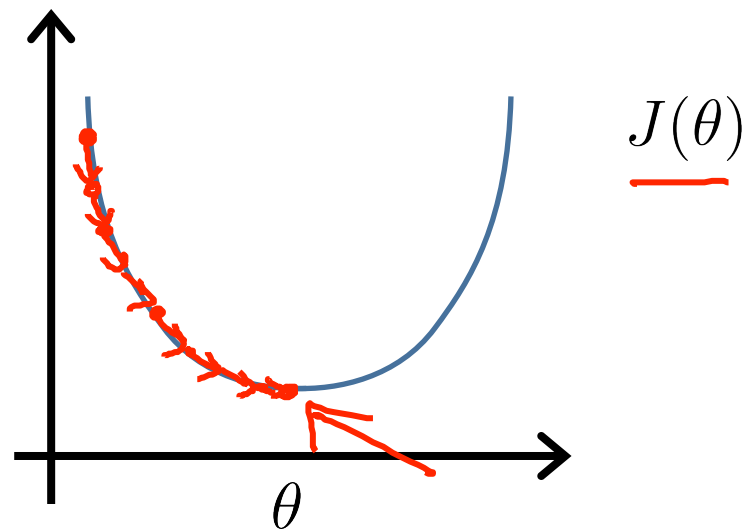
Normal equation: Method to solve for θ analytically.

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

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

$\dfrac{d}{d\theta} J(\theta) = \ldots \overset{set}{=} 0$

Solve for θ

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

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

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

Examples: $m = 4$.

	Size (feet²)	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

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

$M \times (n+1)$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

m-dimensional vector

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

Andrew Ng

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 (design matrix) $= \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{bmatrix}$

$m \times (n+1)$

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

$X = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \\ 1 & x_m^{(i)} \end{bmatrix}$

$m \times 2$

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

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

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

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

$$\text{Set} \quad \underline{A} = \underline{X^T X}$$

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

Octave: **pinv$\boxed{\text{(X'*X)}}$*X'*y**

$$\text{pinv}(X^T * X) * X^T * y$$

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

X' X^T

~~Feature Scaling~~

$$0 \leq x_1 \leq 1$$
$$0 \leq x_2 \leq 1000$$
$$0 \leq x_3 \leq 10^{-5} \quad \checkmark$$

m **training examples, n features.**

Gradient Descent	Normal Equation
• 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 $(X^T X)^{-1}$ $n \times n$ $O(n^3)$
	• Slow if n is very large.

$n = 10^6$

$n = 100$
$n = 1000$
$n = 10000$

There is no need to do feature scaling with the normal equation.

Machine Learning

Linear Regression with multiple variables

Normal equation and non-invertibility (optional)

Normal equation

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

$X^T X$

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

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

pinv

inv

peudo-inverse (will inverse even not invertible matrix)

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

  $X_1 = (3.28)^2 X_2$

  $1m = 3.28$ feet

  $\rightarrow m = 10 \leftarrow$

  $\rightarrow n = 100 \leftarrow$

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

- Too many features (e.g. $m \leq n$).

  - Delete some features, or use regularization.

  $\downarrow$ later