





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
...	...

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

Multiple features (variables).

Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

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.

$n = 4$

$m = 47$

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

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

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. $\underline{h_0(x)} = \underline{80} + \underline{0.1x_1} + \underline{0.01x_2} + 3x_3 - 2x_4$
↑ ↑ ↑
age

$$\rightarrow h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

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

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

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

$$= \theta^T x$$

$$\begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix}$$

θ^T

(n+1) x 1
matrix

$\theta^T x$

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

x

Multivariate linear regression. \leftarrow



Machine Learning

Linear Regression with multiple variables

Gradient descent for multiple variables

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

Handwritten notes: $x_0 = 1$ (with arrow pointing to x_0), θ (underlined), $n+1$ -dimensional vector

Parameters: $\theta_0, \theta_1, \dots, \theta_n$

Handwritten notes: θ (underlined), $n+1$ -dimensional vector

Cost function:

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

Handwritten notes: $J(\theta)$ (underlined), $J(\theta)$ (underlined)

Gradient descent:

Repeat {

$\rightarrow \theta_j := \theta_j - \alpha \left[\frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n) \right]$

Handwritten notes: $J(\theta)$ (underlined), $J(\theta)$ (underlined)

}

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

Gradient Descent

Previously ($n=1$):

Repeat {

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

→ $\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_1^{(i)}$
(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 $j = 0, \dots, n$)

}

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

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

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

...



Machine Learning

Linear Regression with multiple variables

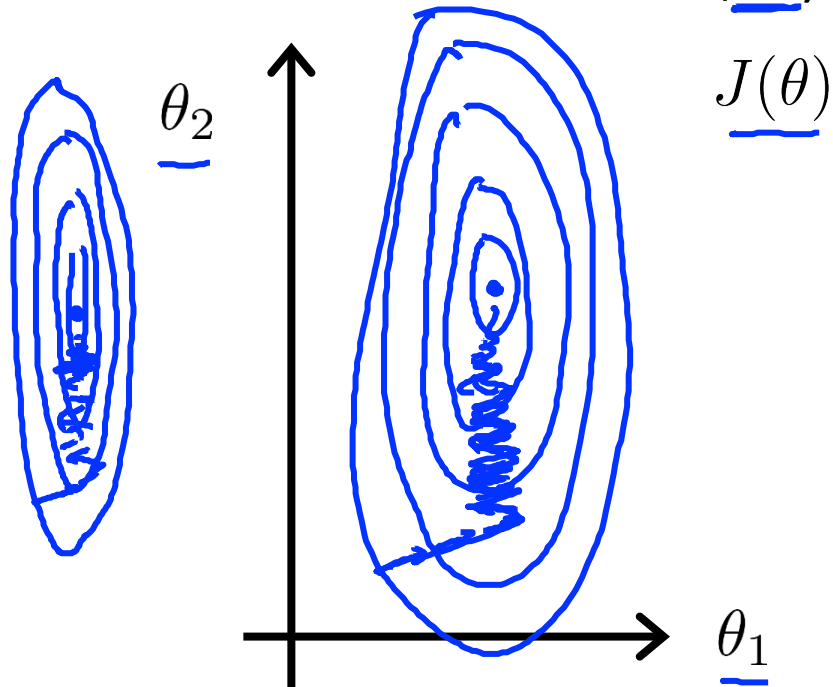
Gradient descent in
practice I: Feature Scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

E.g. $x_1 = \text{size (0-2000 feet}^2\text{)}$ ←

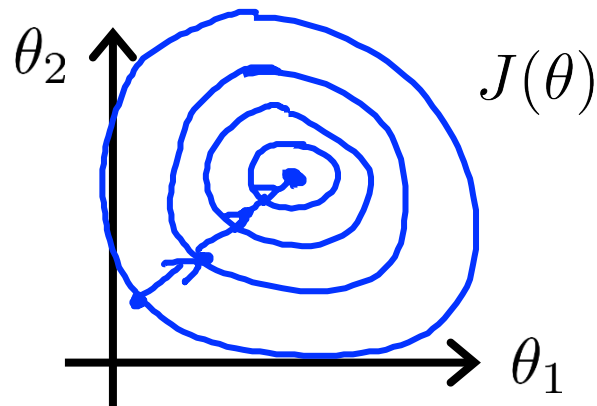
$x_2 = \text{number of bedrooms (1-5)}$ ←



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

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

$0 \leq x_1 \leq 1$ $0 \leq x_2 \leq 1$



Feature Scaling

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

$$x_0 = 1$$

$$0 \leq x_1 \leq 3 \quad \checkmark$$

$$-2 \leq x_2 \leq 0.5 \quad \checkmark$$

$$-100 \leq x_3 \leq 100 \quad \times$$

$$-0.0001 \leq x_4 \leq 0.0001 \quad \times$$

$$-3 \text{ to } 3 \quad \checkmark$$

$$-\frac{1}{3} \text{ to } \frac{1}{3} \quad \checkmark$$

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

Average size = 1000

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

1-5 bedrooms

$$\rightarrow [-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5]$$

$$x_1 \leftarrow \frac{x_1 - \mu_1}{\sigma_1}$$

← avg value of x_1 in training set

range (max-min)
(or standard deviation)

$$x_2 \leftarrow \frac{x_2 - \mu_2}{\sigma_2}$$



Machine Learning

Linear Regression with multiple variables

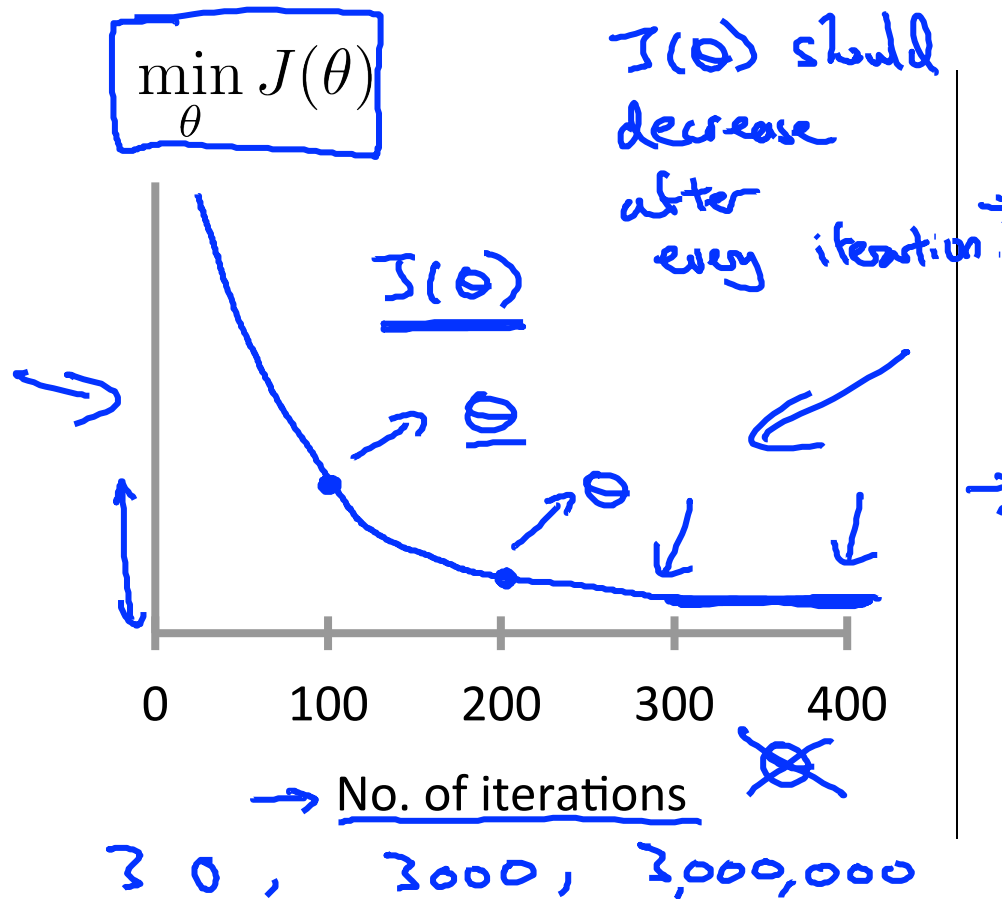
Gradient descent in
practice II: Learning rate

Gradient descent

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

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

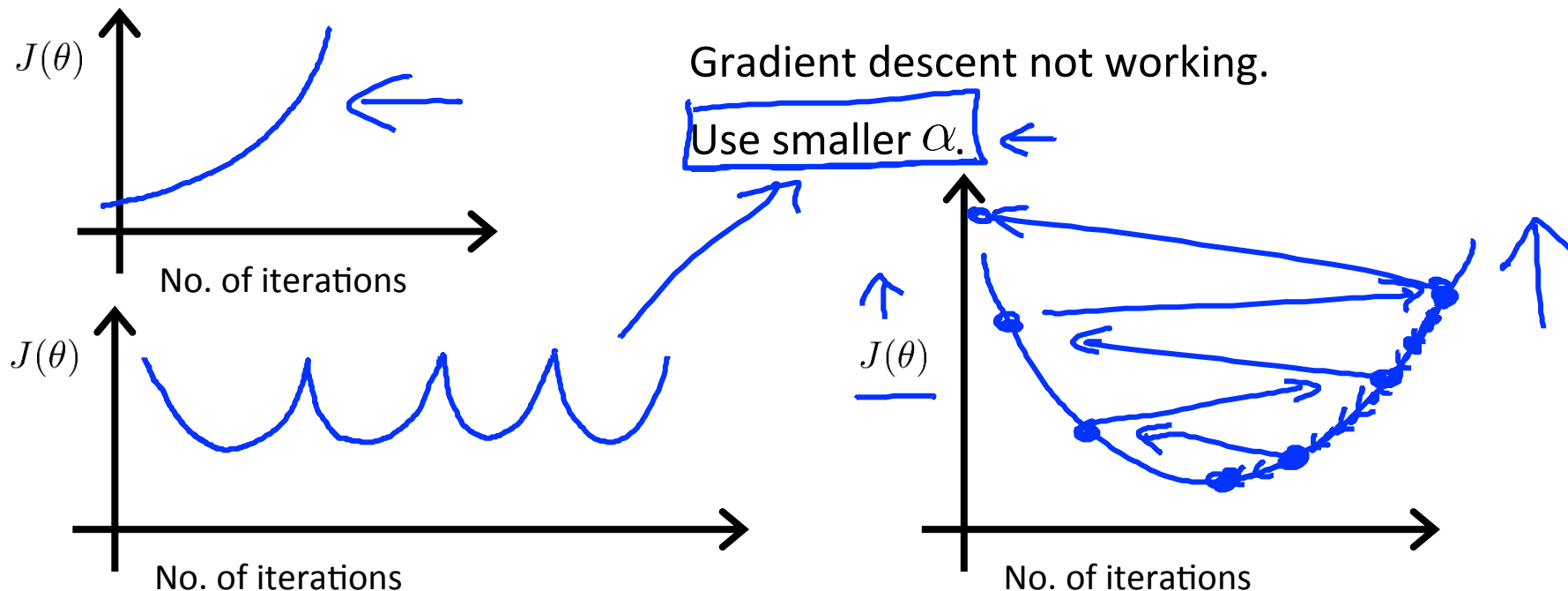
Making sure gradient descent is working correctly.



→ Example automatic convergence test:

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

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.

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

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

Arrows indicate the sequence of values, with labels $\approx 3\times$ and $\approx 10\times$ indicating the relative increase in α between consecutive values.



Machine Learning

Linear Regression with multiple variables

Features and
polynomial regression

Housing prices prediction

$$h_{\theta}(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} + \theta_2 \times \underbrace{\text{depth}}_{x_2}$$

Area

$$x = \underline{\text{frontage} \times \text{depth}}$$

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

↖ land area



Polynomial regression



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

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

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

$$= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3$$

$$\rightarrow x_1 = (\text{size})$$

$$\rightarrow x_2 = (\text{size})^2$$

$$\rightarrow x_3 = (\text{size})^3$$

Size: 1-1000

Size²: 1-1,000,000

Size³: 1-10⁹

Choice of features



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

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



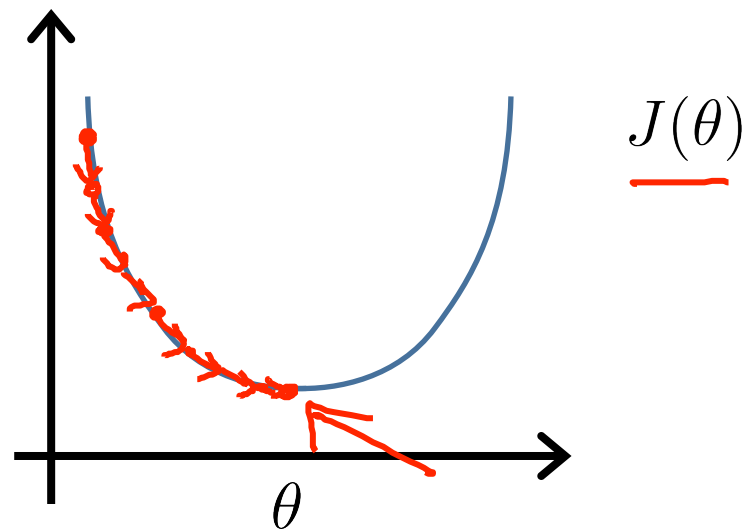


Machine Learning

Linear Regression with multiple variables

Normal equation

Gradient Descent



Normal equation: Method to solve for θ
analytically.

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

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

$\frac{\partial}{\partial \theta} J(\theta) = \dots \stackrel{\text{set}}{=} 0$

Solve for θ



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

$\frac{\partial}{\partial \theta_j} J(\theta) = \dots \stackrel{\text{set}}{=} 0$ (for every j)

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

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

$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$
 $m \times (n+1)$

$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$
 m -dimensional vector

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

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

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

X
(design matrix)

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

$m \times (n+1)$

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

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

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

$m \times 2$

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

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

Set $A = X^T X$

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

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

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

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

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

X'	X^T
	Feature Scaling
	$0 \leq x_1 \leq 1$
	$0 \leq x_2 \leq 1000$
	$0 \leq x_3 \leq 10^{-5}$ ✓

m training examples, n features.

Gradient Descent

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

↗
 $n = 10^6$

← -

Normal Equation

- • No need to choose α .
- • Don't need to iterate.
- Need to compute
- • $(X^T X)^{-1}$ $\overset{n \times n}{\text{}} \quad \underline{O(n^3)}$
- 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

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

$$\underline{X^T X}$$

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

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

θ

$\boxed{\text{pinv}}$
inv

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

- Redundant features (linearly dependent).

E.g. $x_1 = \text{size in feet}^2$

~~$x_2 = \text{size in m}^2$~~

$$x_1 = (3.28)^2 x_2$$

$$1\text{m} = 3.28\text{ feet}$$

$$\rightarrow m = 10 \leftarrow$$

$$\rightarrow n = 100 \leftarrow$$

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

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

- Delete some features, or use regularization.

↓ later