Linear regression with multiple variable

Multiple features

Fundamentals of Machine Learning

Multiple features (variables).

single feature

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

goal

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

hypothesis

Multiple features (variables).

Size (feet^2) 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

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

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

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
…	…	…	…	…

In the training set above, what is $x_1^{(4)}$?

a) The size (in feet^2) of the 1st home in the training set
b) The age (in years) of the 1st home in the training set
c) The size (in feet^2) of the 4th home in the training set
d) The age (in years) of the 4th home in the training set

Hypothesis:

Previously: $\boxed{h_\theta(x) = \theta_0 + \theta_1 x \quad \text{single variable}}$

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

$$E.g. \quad h_\theta(x) = 80 + 0.3x_1 + 1.3x_2 + 4.2x_3 - 0.5x_4$$

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

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

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

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

$$= \theta^T x$$

Multivariate linear regression.

Linear regression with multiple variable

Gradient descent for multiple variables

Fundamentals of Machine Learning

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: $\boxed{\theta_0, \theta_1, \ldots, \theta_n \qquad \theta}$

Cost function:

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

Gradient descent:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n) \qquad J(\theta)\ \text{ - derivative term}}$$

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

When there are *n* features, we define the cost function as

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$$

For linear regression, which of the following are also equivalent and correct definitions of J(θ) ?

(a) $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)} \right)^2$

(b) $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\left(\sum_{j=0}^{n} \theta_j x_j^{(i)} \right) - y^{(i)} \right)^2$ (inner sum starts at 0)

(c) $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\left(\sum_{j=1}^{n} \theta_j x_j^{(i)} \right) - y^{(i)} \right)^2$ (inner sum starts at 1)

(d) $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\left(\sum_{j=0}^{n} \theta_j x_j^{(i)} \right) - \left(\sum_{j=0}^{n} y_j^{(i)} \right) \right)^2$

Gradient Descent

Previously (n=1):

Repeat $\{$

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

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

$x_1^{(i)}$

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

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

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

$\}$

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

=1

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

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

$\ldots$

Linear regression with multiple variable

Gradient descent in practice I: Feature Scaling

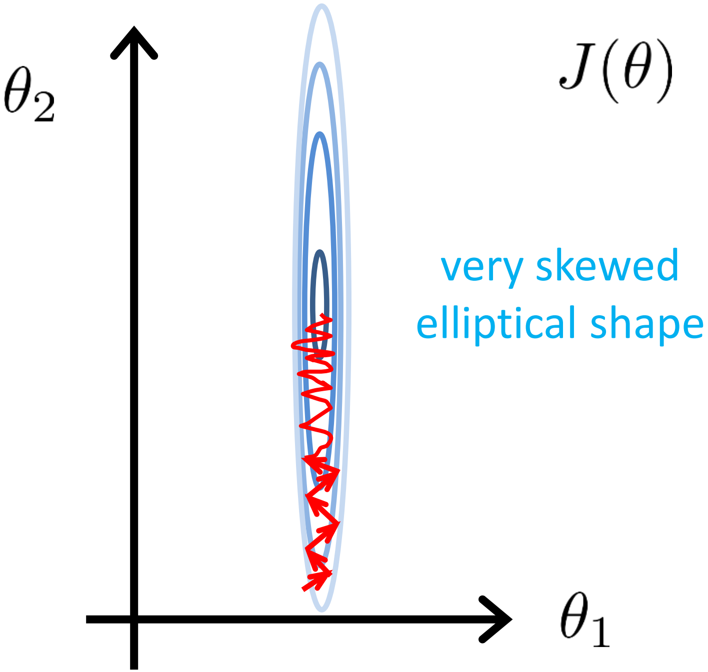
Fundamentals of Machine Learning

Feature Scaling

Idea: Make sure features are on a similar scale.

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

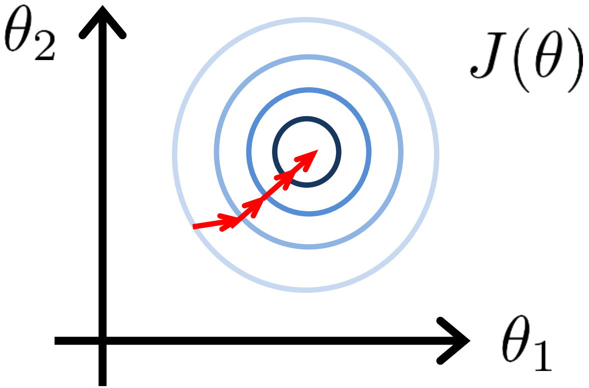
x_2 = number of bedrooms (1-5)



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

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

$0 \le x_1 \le 1$ $\quad$ $0 \le x_2 \le 1$



Feature Scaling

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

$x_0 = 1$

$0 \le x_1 \le 2$ ✓

$-2 \le x_2 \le 0.7$ ✓

$-100 \le x_3 \le 100$ ✗

$0.003 \le x_4 \le 0.01$ ✗

-3 to 3 ✓

-1/3 to 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. $x_1 = \frac{size - 1000}{2000}$ average size = 1000

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

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

$$x_i \leftarrow \frac{x_i - \mu_i}{S_i}$$

average value of x_i in the training set

Range of values of that feature

"maximum – minimum" or standard deviation

Suppose you are using a learning algorithm to estimate the price of houses in a city. You want one of your features x_i to capture the age of the house. In your training set, all of your houses have an age between 30 and 50 years, with an average age of 38 years. Which of the following would you use as features, assuming you use feature scaling and mean normalization?

$$(a) \quad x_i = age\ of\ house$$

$$(b) \quad x_i = \frac{age\ of\ house}{50}$$

$$(c) \quad x_i = \frac{age\ of\ house - 38}{50}$$

$$(d) \quad x_i = \frac{age\ of\ house - 38}{20}$$

Linear regression with multiple variable

Gradient descent in practice II: Learning rate

Fundamentals of Machine Learning

Gradient descent

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

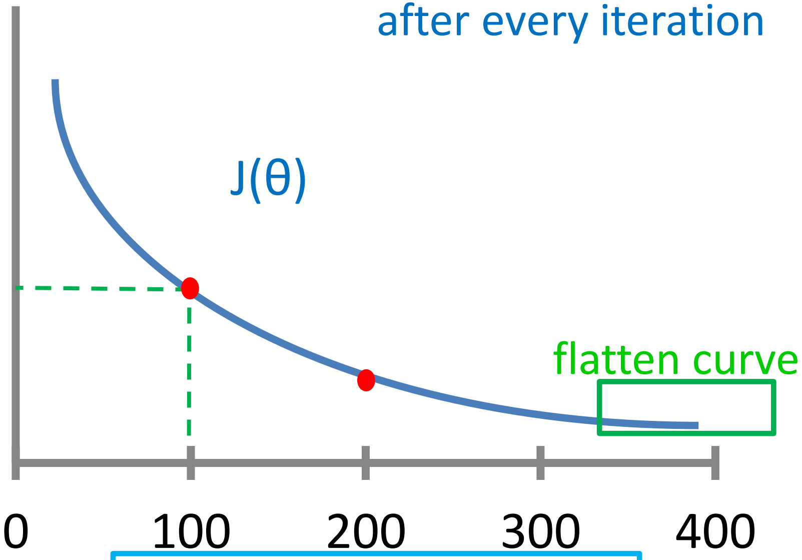
- "Debugging": How to make sure gradient descent is working correctly.

- 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



No. of iterations

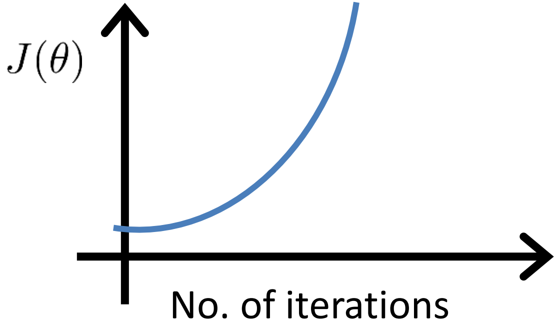
30 , 3.000 , 3.000.000

Example automatic convergence test:

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

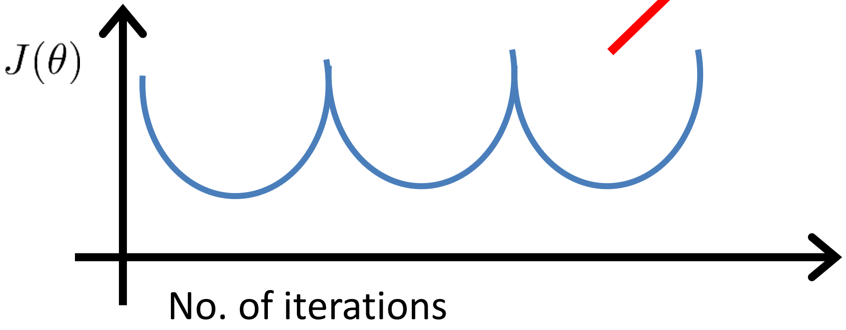
ε

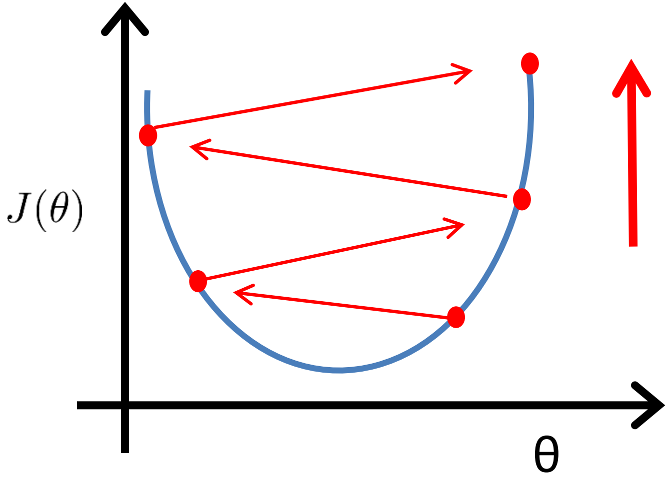
Making sure gradient descent is working correctly.



Gradient descent not working.

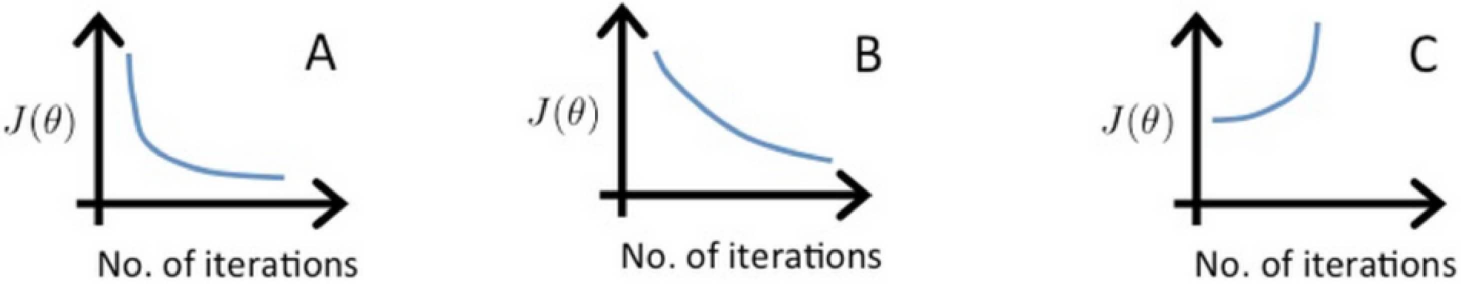
$\boxed{\text{Use smaller } \alpha.}$





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

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

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 is also possible

To choose α, try

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

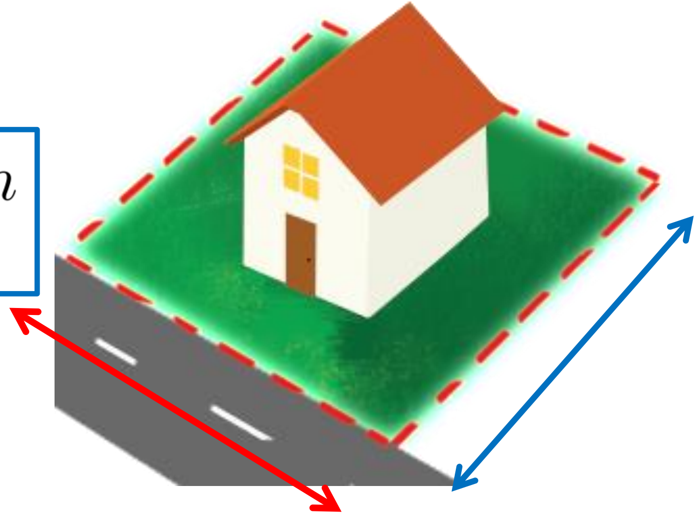
Linear regression with multiple variable

Features and polynomial regression

Fundamentals of Machine Learning

Housing prices prediction

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

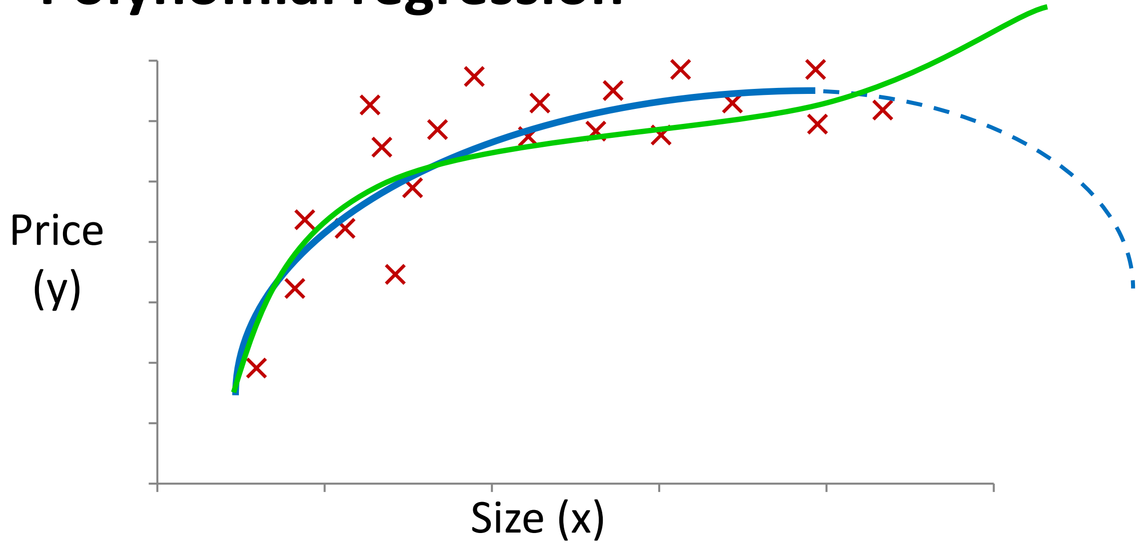


Area (land)

x = frontage * depth

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

Polynomial regression



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

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

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

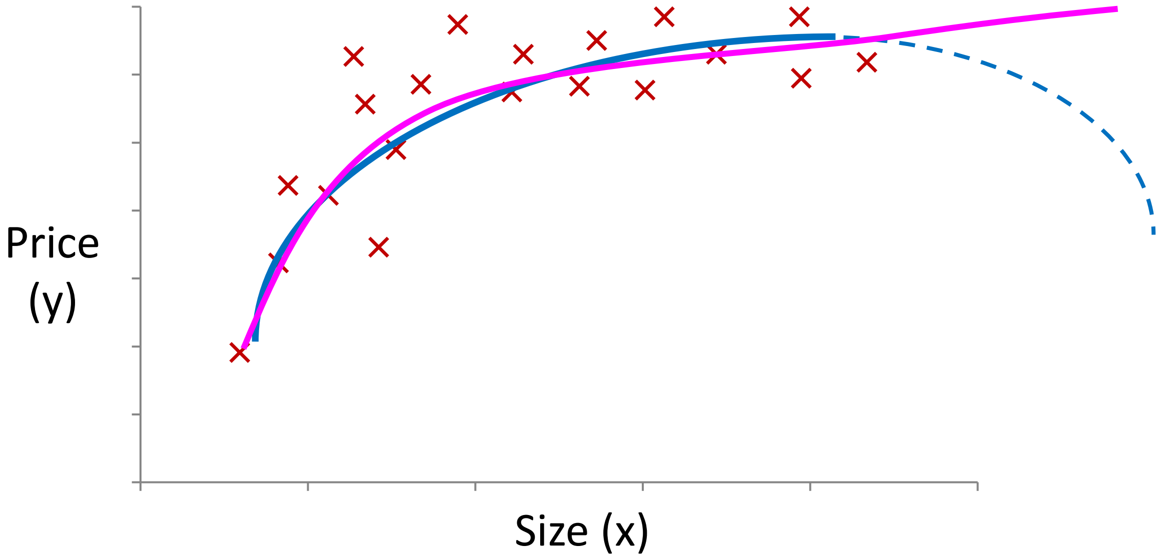
$$\boxed{x_1} = (size)$$
$$\boxed{x_2} = (size)^2$$
$$\boxed{x_3} = (size)^3$$

Size : 1 – 1000
$Size^2$: 1 – 1.000.000
$Size^3$: 1 – 10^9

Feature scaling

Andrew Ng

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: $\sqrt{1000} \approx 32$.)

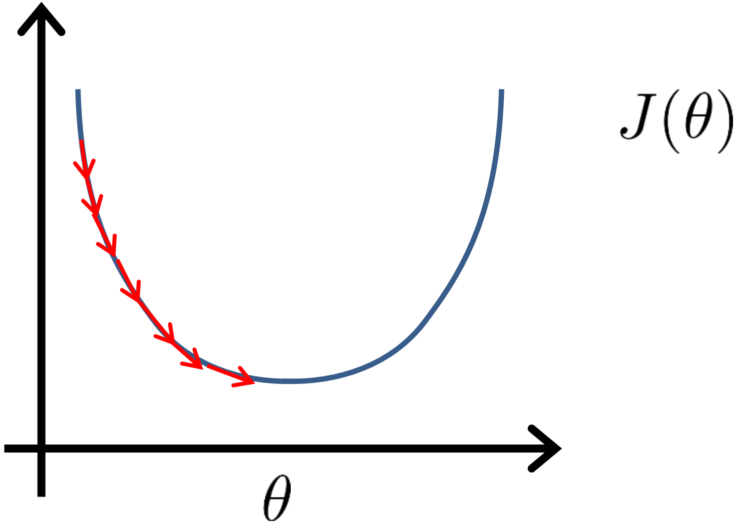
- ○ $x_1 = \text{size},\ x_2 = 32\sqrt{(\text{size})}$
- ○ $x_1 = 32(\text{size}),\ x_2 = \sqrt{(\text{size})}$
- ○ $x_1 = \frac{\text{size}}{1000},\ x_2 = \frac{\sqrt{(\text{size})}}{32}$
- ○ $x_1 = \frac{\text{size}}{32},\ x_2 = \sqrt{(\text{size})}.$

Linear regression with multiple variable

Normal equation

Fundamentals of Machine Learning

Gradient Descent



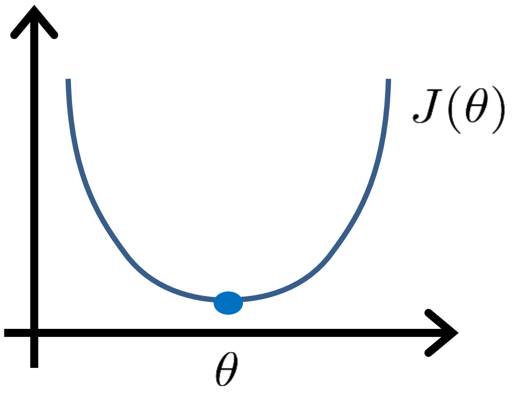
Normal equation: Method to $\boxed{\text{solve for } \theta}$ analytically.

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

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

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

Solve for θ



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

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

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

Andrew Ng

Examples: $m = 4.$

extra 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

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

m x (n+1)

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

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.**

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

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

(design matrix)

m x (n+1) dimensional matrix

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

$$X = \begin{bmatrix} 1 & x^{(1)}_1 \\ 1 & x^{(1)}_2 \\ & \vdots \\ 1 & x^{(1)}_n \end{bmatrix}$$

m x 2

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

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

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

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

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

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

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

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

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

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

Matlab: `pinv(X'*X)*X'*y`   X' – transpose of X (X^T)

Feature scaling is not necessary

$0 \leq x_1 \leq 1$

$0 \leq x_2 \leq 1000$

$0 \leq x_1 \leq 10^{-6}$

....

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 $\boxed{(X^TX)^{-1} \quad n \times n \quad O(n^3)}$
- Slow if n is very large.

$n = 100$
$n = 1.000$
$n = 10.000$

Linear regression with multiple variable

Normal equation and non-invertibility

Fundamentals of Machine Learning

Normal equation

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

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

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

pinv – pseudo-inverse
inv – inverse

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

1m = 3.28 feet
x_1=(3.28) x_2
delete one of the feature

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

m=10
n=100