multiple linear regression (multiple variables / features)

n : number of all features. (列) m : the number of training examples. (行)

$x^{(i)}$: input (features) of i^{th} training example → index of traing sets (examples).

$x_j^{(i)}$: value of feature j in i^{th} traing examples.

$x_n^{(m)}$ 第m个样本的第n个特征.

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n \qquad n: \text{the number of features.}$$

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

(the zero feature)

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

feature vector.

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

parameter vector.

$$\theta^T = [\theta_0, \theta_1, \theta_2 \cdots \theta_n] \qquad X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$\theta^T \cdot x$.

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

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

Multivariate linear regression.

parameters: $\theta_0, \theta_1, \theta_2 \cdots \theta_n$ θ: is a $n+1$-dimensional vector.

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

$J(\theta)$.

Gradient Descent:

Repeat {

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} \cdot J(\theta_0, \theta_1, \theta_2 \cdots \theta_n)$$
$$= \theta_j - \alpha \frac{\partial}{\partial \theta_j} \cdot J(\theta)$$

}

simultaneously update for every $j = 0, 1, \cdots n$

$n = 1$

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

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

$$\begin{cases} \frac{\partial J}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\ \frac{\partial J}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \end{cases}$$

$n > 1$

Repeat {

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

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

}

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

e.g.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_0^{(i)}) - y_0^{(i)}) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_1^{(i)}) - y_1^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_2^{(i)}) - y_2^{(i)}) \cdot x_2^{(i)}$$

Feature Scaling

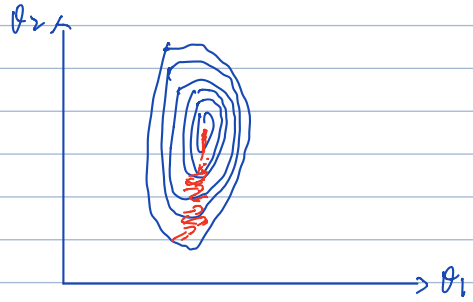
Idea: make sure features are on a similiar scale, then the GD will be converged faster.

e.g. x_1 = size ($0 \sim 2000$ feet^2)
x_2 = number of bedrooms (1-5).

$x_1 = \frac{\text{size}}{2000}$ $\quad 0 \le x_1 \le 1$

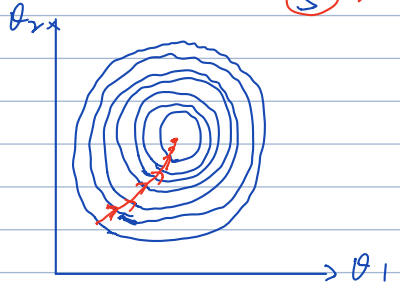
$x_2 = \frac{\text{number}}{5}$ $\quad 0 \le x_2 \le 1$

the range of feature (max - min)



ignore x_0.

converge slowly



converge faster and more directly.

normally, we get every feature into approximately a $-1 \le x_i \le 1$ range or near it.

$x_0 = 1$ ✓

$0 \le x_1 \le 3$. ✓

$-2 \le x_2 \le 0.5$ ✓

$-100 \le x_3 \le 100$ ✗ too large.

$-0.0001 \le x_4 \le 0.0001$ ✗ too small.

people have different notate on the range of feature scaling according to different applications.

Mean Normalization: 平均归一化.

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

mean normalization involves subtracting the average value for an input variables from the values for that input variable resulting in a new average value for the input variables of just zero.

$$x_i = \frac{x_i - \mu_i}{s_i}$$

μ_i: average value of that variable

s_i: the range of value (max - min) or standard deviation.

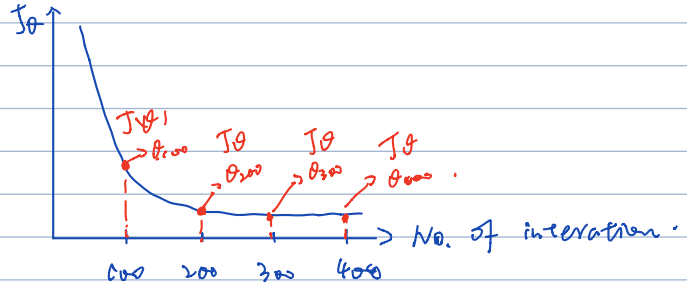
用range和方差有不同的效果.

Debugging Gradient Descent

Make a plot with the number of iteration on the x-axis, the value of $J(\theta)$ on the y-axis.

If $J(\theta)$ is increasing, then you probably need to decrease the α.

Normally, $J(\theta)$ should decrease after every iteration,

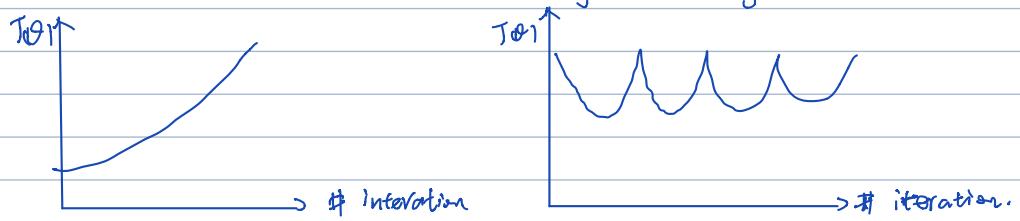


Automatic convergance test:

Declare convergence if $J(\theta)$ decrease by less than ε in one iteration where ε is some small value such as 10^{-3}.

However, it's difficult to choose this threshold value.

Gradient Descent is not working correctly:



way because the learning rate is too large.

Summary:

- α is too small: slow convergence
- α is too large: may not decrease on every iteration and thus may not converge.

Feature and Polynomial Regression

Our hypothesis function need not be linear (a straight) line if that does not fit the data well.

We can use quadratic (二次) cubic (三次) or square root function.

e.g.

original: $h_\theta(x) = \theta_0 + \theta_1 x_1$ we can create additional features based on x_1

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 \quad \Leftarrow \text{quadratic}$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 \quad \Leftarrow \text{Cubic}$$

$x_1^2 = x_2 \quad x_1^3 = x_3$.

Notice: if you choose your hypothesis function like this, feature scaling is increasingly important.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$$

if $x_1 \in [1, 1000]$ $x_1^2 \in [1, 10^6]$ $x_1^3 \in [1, 10^9]$

Normal Equation: Don't need to do feature scaling.

Method to solve for θ analytically.

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

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

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

solve for θ.

$(n+1)$ features:

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

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

solve for $\theta_0, \theta_1, \cdots \theta_n$.

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

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \quad (n+1) \times 1$$

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix} \quad \text{design matrix} \quad m \times (n+1)$$

$x_0^{(i)} = 1$

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

$\theta = (X^T X)^{-1} X^T y \Rightarrow \min J(\theta)$.

method to solve for θ analytically. ↙ linear regression.

Gradient Descent	Normal Equation
• Need to choose α.	• No need to choose α.
• Need many iterations	• Don't need to iterate.
• works well even when n is very large.	• Need to compute $(X^T X)^{-1}$
(more than 10,000)	• slow if n is very large.

Non-invertibility (singular / degenerate)

pinv → even $(X^T X)$ is non-invertible.
inv

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

① Redundant features (linearly dependent).
E.g. x_1 = size in feet^2. $1m = 3.28$ feet
x_2 = size in m^2.
x_1, x_2 always satisfy this equation: $x_1 = (3.28)^2 x_2$

② Too many features (with less training examples). ($m \leq n$)
- Delete some features, or use regularization.
cannot learn features from fewer training examples