definition of 2 form:

Arthur Samuel: (ML) field of study that gives computer ability to learn without being explicitly programmed.

Tom Mitchell: A computer program is said to learn from experience E with respect to some task T and some performance measured P, if its performance on T, as measured by P, improves with experience E.

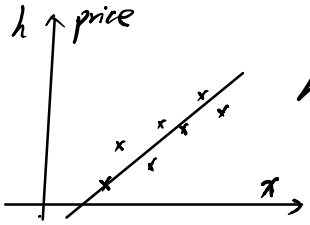
ML algorithms { supervised learning / unsupervised learning }

others: reinforcement learning, recommender system

Linear Regression Problem:

training set → learning algorithm → h (hypothesis 预测)

size of house → h → estimated price



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

house price predict problem, supervised learning

linear regression with one variable

univariate linear regression 单一变量

cost function. 损失函数

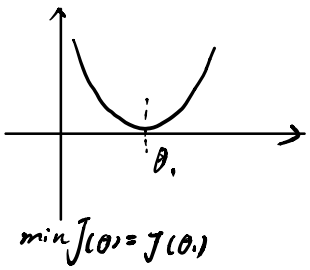
$h_\theta(x) = \theta_0 + \theta_1 x$, choose θ_0, θ_1 to minimize $J(\theta)$

$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$, square error function 方差函数, m is the number of training set

when θ is only one dimention.

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

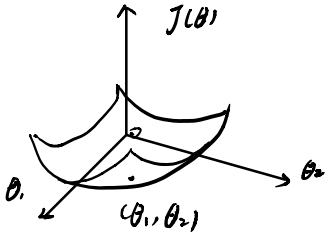
by plotting the graph of $J(\theta)$, is:



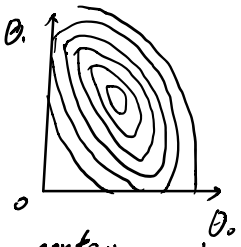
when θ is 2 dimention:

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

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



bowl shaped convex function



contour graph 等高线图

Gradient descent: 梯度下降

a more general method for calculating $\min J(\theta)$, $\theta = \langle \theta_0, \theta_1 \rangle$

outline: - start with some θ_0, θ_1

- keep changing θ_0, θ_1 to reduce $J(\theta)$, until end up at a minimum

repeat until convergence { 收敛

$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ $\quad j = 0, 1$

}

α: learning rate

$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$: partial derivative 偏导数

simultaneously update θ_0, θ_1

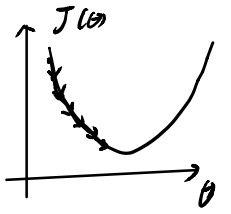
✓

$$\begin{cases} t_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ t_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\ \theta_0 = t_0 \\ \theta_1 = t_1 \end{cases}$$

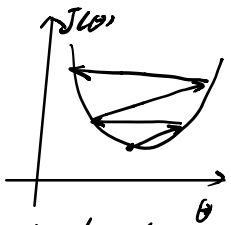
✗

$$\begin{cases} \theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ \theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \end{cases}$$

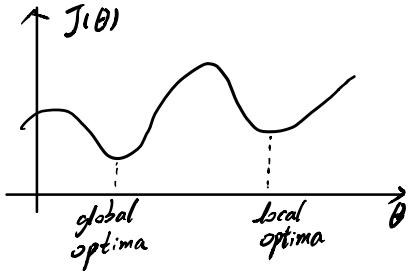
To choose the learning rate:

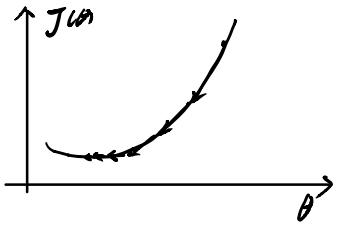


α is too small, gradient descent can be very slow



α too large, gradient descent can overshoot the minimum, it may fail to converge, or even diverge. 发散





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

gradient descent can converge to a local minimum even with the learning rate α fixed.

As we approach a local minimum, gradient descent will automatically take a smaller step, hence no need to decrease α over time.

gradient descent for linear regression:

for a 2 parameter $h_\theta(x)$:

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

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

$= \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$

repeat until convergence {

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

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

}

2 variable linear regression's cost function $J(\theta_0, \theta_1)$ is a bowl shaped convex function, it always has a global minimum which gradient descent can converge to.

↓ is a "batch" gradient descent : $\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$ use all the training examples each step iterate

Review of linear algebra:

(only terminologies)

Matrix: rectangular array of numbers

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ 4x2 matrix $\mathbb{R}^{4\times 2}$

vector: $n \times 1$ matrix

$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ 3 dimensional vector $\mathbb{R}^3$

Matrix Addition / substraction

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$ $\quad \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$

Scalar (real number) Multiplication

$3 \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$

Matrix vector multiplication:

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 7 \end{bmatrix}$

Matrix Matrix Multiplication:

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 13 & 16 \\ 29 & 36 \end{bmatrix}$

Matrix Multiplication Properties

$A \times B \neq B \times A$ (not commutative)

Identity Matrix 单位矩阵

denoted as I or $I_{n\times n}$

$I_{2\times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$A = AI = IA$ for any matrix A

matrix transpose:

$B = A^T$

$B_{ij} = A_{ji}$

Matrix Inverse:

$AA^{-1} = I = A^{-1}A$

matrix that can't be inverse is "singular" or "degenerate"

Linear Regression with multiple variables (multiple features)

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_n x_n$ $\quad$ n: number of features

for convience denote $x_0 = 1$ $\quad$ x_i: feature i

$$h_\theta(x) = [\theta_0, \theta_1, \cdots, \theta_n] \cdot \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$= \theta^T x$ (multivariate linear regression)

gradient descent:

repeat until convergence {

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

$j = 0, \cdots, n$

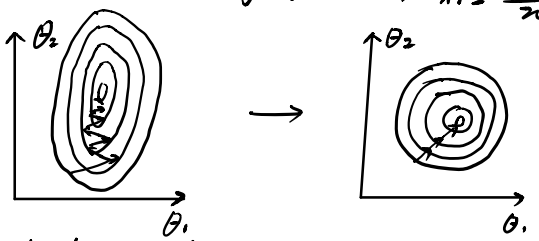
}

to improve the performance of gradient descent:

Feature Scaling: make sure features are on the similiar scale

特征缩放

e.g. x_1 = size (0 ~ 2000 feet^2) $\rightarrow$ $x_1 = \frac{\text{size}}{2000}$



get features into approximately a $-1 \le x_i \le 1$ range

Mean Normalization: replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (not apply to $x_0 = 1$)

均值规范化

e.g. x_1 = size (0 ~ 2000 feet^2) $\rightarrow$ $x_1 = \frac{\text{size} - 1000}{2000}$ $(-0.5 \sim 0.5)$

$x \leftarrow \frac{x - \mu}{s}$ $\quad$ μ: average value of x in training set

s: range, max - min

standardization:

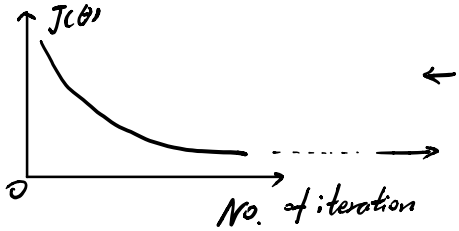
标准化

$x \leftarrow \frac{x - \mu}{\text{std}(x)}$ $\quad$ has zero-mean and unit-variance

零均值 $\quad$ 单位方差

Debug Gradient Descent

to choose the proper α to make it work correctly.



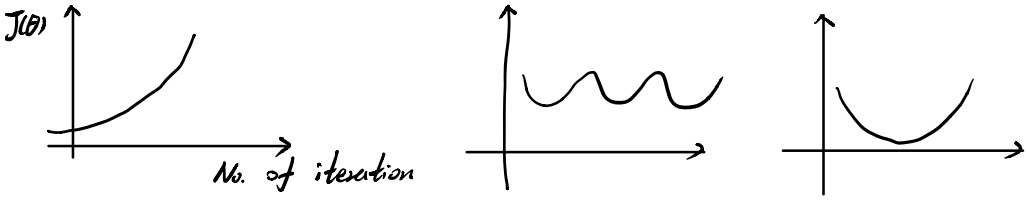
$J(\theta)$ should decrease every iteration

←: example of automatic convergence test

declare convergence if $J(\theta)$ decrease by less than ε in one iteration.

ε: 10^{-3} ? hard to determine

in the following case, gradient descent works incorrectly for α is too large:



if α is small enough, $J(\theta)$ should decrease every iteration

choose α: usually from 0.001 to 1

... 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, ...

×3 ×3 ×3 ×3 ×3 ×3

Features And Polynomial Regression:

e.g.1: house pricing prediction

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

$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} \times \underbrace{\text{deepth}}_{x_2}$ is not linear function

$= \theta_0 + \theta_1 x_1 x_2$

but if let $x = x_1 x_2$, $h_\theta(x) = \theta_0 + \theta_1 x$ it turns linear again

e.g.2: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

$= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x^3$

$x_1 = x$ $1 \sim 1000$

$x_2 = x^2$ $1 \sim 1000000$

$x_3 = x^3$ $1 \sim 1000000000$

it must use feature scaling or mean normalization

Normal Equation

- method to solve for θ analytically.

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

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

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

e.g.

$m = 4$

	size	number of bedrooms	number of floors	age of home	price
x_0	x_1	x_2	x_3	x_4	x_5
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

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

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \quad m \times 1$$

$h_\theta(x) = \theta^T x$

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

no need to choose α
no need to iterate
no need to do feature scaling
$O(n^3)$, be slow if n is large

when has many features ($>10^6$), use gradient descent

when $X^T X$ is non-invertible, two reason

1: redundant features (linearly dependent) e.g. ($x_1 = k x_2 \ldots$)

2. too many features ($n \geq m$), just delete some features or use regularization.

Normal Equation proof:

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

$$h_\theta(x) = \theta^T x = \sum_{i=0}^{n}\theta_i x_i$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = 0$$

assume that we have n features and m training items

training set is:

$$X = \begin{bmatrix} x_0^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(m)} & & x_n^{(m)} \end{bmatrix}_{m \times (n+1)}$$

feature parameters:

$$\theta = [\theta_0, \theta_1, \cdots, \theta_n]^T$$

output y:

$$Y = [y^{(1)}, y^{(2)}, \cdots, y^{(m)}]^T$$

$$X\theta - Y = \begin{bmatrix} (\theta_0 x_0^{(1)} + \theta_1 x_1^{(1)} + \cdots \theta_n x_n^{(1)}) - y^{(1)} \\ \vdots \\ \theta_0 x_0^{(m)} + \theta_1 x_1^{(m)} + \cdots \theta_n x_n^{(m)} - y^{(m)} \end{bmatrix}_{m \times 1} = \begin{bmatrix} \theta^T x^{(1)} - y^{(1)} \\ \theta^T x^{(2)} - y^{(2)} \\ \vdots \\ \theta^T x^{(m)} - y^{(m)} \end{bmatrix}_{m \times 1}$$

$$= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}_{m \times 1}$$

$$\therefore \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 = (X\theta - Y)^T(X\theta - Y)$$

cost function.

$$J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{2m}(X\theta - Y)^T(X\theta - Y)$$

$$= \frac{1}{2m}(Y^TY + \theta^TX^TX\theta - Y^TX\theta - \theta^TX^TY$$

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{2m}\left(\underbrace{\frac{\partial Y^TY}{\partial \theta}}_{①} + \underbrace{\frac{\partial \theta^TX^TX\theta}{\partial \theta}}_{②} - \underbrace{\frac{\partial Y^TX\theta}{\partial \theta}}_{③} - \underbrace{\frac{\partial \theta^TX^TY}{\partial \theta}}_{④}\right)$$

$\because \frac{\partial AB}{\partial B} = A^T \qquad \frac{\partial x^T A x}{\partial x} = 2Ax$

① $\frac{\partial Y^T Y}{\partial \theta} = 0$

③ $\because Y^T X\theta = [y^{(1)}, y^{(2)}, \cdots, y^{(m)}] \begin{bmatrix} x_0^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} [\theta_0, \theta_1, \cdots, \theta_n]^T$

$= (x_0^{(1)} y^{(1)} + \cdots + x_0^{(m)} y^{(m)})\theta_0 + (x_1^{(1)} y^{(1)} + \cdots + x_1^{(m)} y^{(m)})\theta_1 + \cdots$

$+ (x_n^{(1)} y^{(1)} + \cdots x_n^{(m)} y^{(m)})\theta_n$

$\therefore \frac{\partial Y^T X\theta}{\partial \theta} = \begin{bmatrix} \frac{\partial Y^T X\theta}{\partial \theta_0} \\ \vdots \\ \frac{\partial Y^T X\theta}{\partial \theta_n} \end{bmatrix} = \begin{bmatrix} x_0^{(1)} y^{(1)} + \cdots + x_0^{(m)} y^{(m)} \\ \vdots \\ x_n^{(1)} y^{(1)} + \cdots + x_n^{(m)} y^{(m)} \end{bmatrix} = X^T Y$

④ $\theta^T X^T Y = [\theta_0, \theta_1, \cdots, \theta_n] \begin{bmatrix} x_0^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(m)} & \cdots & x_n^{(m)} \end{bmatrix}^T [y^{(1)}, y^{(2)} \cdots y^{(m)}]^T$

$= (x_0^{(1)}\theta_0 + \cdots + x_n^{(1)}\theta_n) y^{(1)} + (x_0^{(2)}\theta_0 + \cdots + x_n^{(2)}\theta_n) y^{(2)}$

$+ \cdots + (x_0^{(m)}\theta_0 + \cdots + x_n^{(m)}\theta_n) y^{(m)}$

$\frac{\partial \theta^T X^T Y}{\partial \theta} = \begin{bmatrix} \frac{\partial \theta^T X^T Y}{\partial \theta_0} \\ \vdots \\ \frac{\partial \theta^T X^T Y}{\partial \theta_n} \end{bmatrix} = \begin{bmatrix} x_0^{(1)} y^{(1)} + \cdots + x_0^{(m)} y^{(m)} \\ x_n^{(1)} y^{(1)} + \cdots + x_n^{(m)} y^{(m)} \end{bmatrix} = X^T Y$

② $\theta^T X^T X\theta = (X^T X)(\theta_0^2 + \theta_1^2 + \cdots \theta_n^2)$

$\frac{\partial \theta^T X^T X\theta}{\partial \theta} = \begin{bmatrix} \frac{\partial \theta^T X^T X\theta}{\partial \theta_1} \\ \vdots \\ \frac{\partial \theta^T X^T X\theta}{\partial \theta_n} \end{bmatrix} = 2(X^T X) \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix} = 2X^T X\theta$

$\therefore \frac{\partial J(\theta)}{\partial \theta} = \frac{1}{2m}(-2X^T Y + 2X^T X\theta) = 0$

$\therefore X^T X\theta = X^T Y$

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

Another Proof Not Using Matrix Calculus

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

$$\begin{cases} \dfrac{\partial J(\theta)}{\partial \theta_0} = \dfrac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} = 0 & ① \\ \vdots & \\ \dfrac{\partial J(\theta)}{\partial \theta_n} = \dfrac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_n^{(i)} = 0 & ⓝ \end{cases}$$

$$①: \frac{1}{m} \begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \cdots & x_0^{(m)} \end{bmatrix} \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix} = 0$$

$$\begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \cdots & x_0^{(m)} \end{bmatrix} \left(\begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \theta - y \right) = 0$$

$$\begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \cdots & x_0^{(m)} \end{bmatrix} (X\theta - y) = 0$$

$$\therefore \quad \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \end{bmatrix} (X\theta - y) = 0$$

$$\vdots$$

$$\begin{bmatrix} x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(m)} \end{bmatrix} (X\theta - y) = 0$$

$$\therefore \quad \begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \cdots & x_0^{(m)} \\ x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(m)} \end{bmatrix} (X\theta - y) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\| \quad X^T$$

$$X^T(X\theta - y) = 0$$

$$X^T X\theta - X^T y = 0$$

$$X^T X\theta = X^T y$$

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

vectorized :

e.g.

formula :

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i \quad \rightarrow \quad h_\theta(x) = \theta^T x$$

octave code.

```
prediction = 0.0
for i = 1: n+1
  prediction = prediction + theta(i) * x(i)
end.
```

→

```
prediction = theta' * x
```

vectorize can use GPU to accelerate running, and shorten the code.