ML OnLince Course

Linear Regression with multiple Variables

Linear Regression with multiple variables

$x_j^{(i)}$ = value of feature j in i^{th} training example. $\begin{bmatrix} x_1^1 \\ x_1^2 \\ x_1^3 \end{bmatrix}$

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

Hypothesis:

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_3 x_3 \cdots + \theta_n x_n$

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

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

$$[\theta_0 \theta_1 \cdots \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

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

$= \theta^T x$

Multivariate linear Regression

$$X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} \\ x_0^{(2)} & x_1^{(2)} \\ x_0^{(3)} & x_1^{(3)} \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \qquad h_\theta(X) = X\theta$$

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

Gradient Decent

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: θ $\quad$ $n+1$-dimensional vector

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

Gradient descent:

Repeat {

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

}

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

For (n≥1):

Repeat {

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

}

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

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

Tricks to converge well

Feature Skills: Feature Scaling

Idea: Make sure features are on a similar scale

(in order to make GD converge more quickly)

如果scale不similar.
那么contour会特别的扁
这样的话converge特别的慢

on a similar scale的话
contour接近一个圆.

Feature scaling: Diving the input values by the range of input variable, resulting in a new range of just 1.

Get every feature 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.

(Do not apply to $x_0=1$)

$x_1 \leftarrow \frac{x_1-\mu_1}{S_1}$ — μ_1: avg value of x_1 in training set; S_1: range (max − min), standard deviation

$x_2 \leftarrow \frac{x_2-\mu_2}{S_2}$

Matrix Notation

$\theta := \theta - \alpha\nabla J(\theta)$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\theta_0} \\ \frac{\partial J(\theta)}{\partial\theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\theta_n} \end{bmatrix}$$

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

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

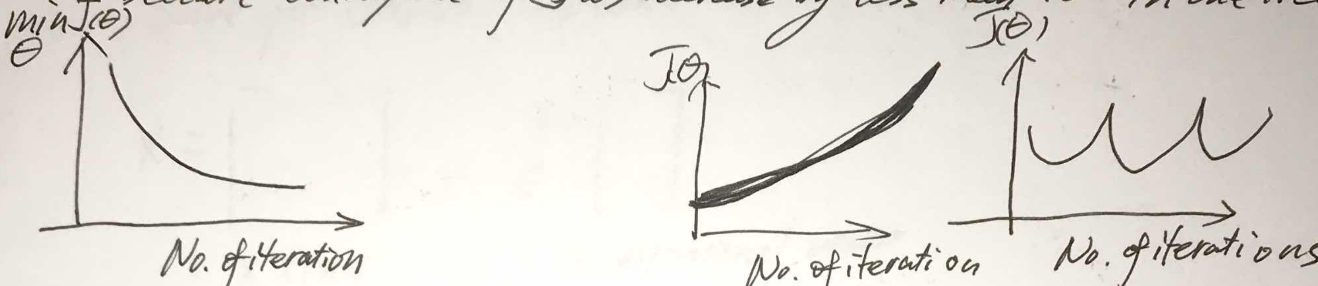
$$= \frac{1}{m}\vec{x_j}^T(X\theta-\vec{y})$$

$$\nabla J(\theta) = \frac{1}{m}X^T(X\theta-\vec{y}) \Rightarrow \theta := \theta - \frac{\alpha}{m}X^T(X\theta-\vec{y})$$

- "Debugging": How to make sure gradient decent is working correctly
- How to choose learning rate α

Convergence test:

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



For sufficient small α, $J(\theta)$ should drease on every iteration

But if α is too small, GD 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.

To choose α, try

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

Polynomial Regression:

Our hypothesis function need not be linear if it does not fit the data well.

Feature scaling would be important in the case when u use GD.

No Need to do Feature Scaling

Normal equation: Method to solve for θ analytically

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

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

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

$X = \begin{bmatrix} \\ \\ \end{bmatrix}$ $y = \begin{bmatrix} \\ \\ \end{bmatrix}$

$X \times (n+1$ m-dimensional vector

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

For example.

$x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

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

(design matrix

$m \times (n+1)$

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

$\theta = (X^T X)^{-1} X^T y$ Octave: pinv(x'*x)*x'*y

m training examples, n features

GD	Normal Equation
Need to choose α	~~Do not~~ No need to choose α
Needs many iterations	Don't need to iterate
Works well even when n is large	Need to compute $(X^T X)^{-1}$ $O(n^3)$
$O(kn^2)$	Slow if n is very large

Normal equation and non-invertibility

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

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

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

Matlab

点乘 .* .^ ./

V= linspace(0,3,8)
V= 2:0.2:3

V=c'
~~pol~~ plot(x,y)

'm:s' magenta, dotted line with square markers
'g--*' green dashedline with star markers
'r-' red, solid line with no markers

xlabel('time [s]')
ylabel('amplitude')
title('my plot')
legend('y(t)')
grid
annotation

---

matrices

A=[-4,1.9,-3.2,-12; -0.25 ,2,9,0.3; 0.1,7,-1,5]

$$\begin{bmatrix} -4 & 1.9 & -3.2 & -12 \\ -0.25 & 2 & 9 & 0.3 \\ 0.1 & 7 & -1 & 5 \end{bmatrix}$$

Array Creation

rand
ones
eye
randi   randi([0,2],3,4) random integers between 0 and 2
randn   normal random variables
zeros   zeros(10,3)
toeplitz
vander   vandermonde matrix
diag
magic   hilb

$S1 = M(3,2)$

$S2 = M([2,3],2)$

$S3 = M(2:4,2)$

$S4 = M(:,2)$

u vector

$L1 = length(u)$ — length(m) will return the max of col and rows

matrix

$s = size(M)$

$C = [A,B]$ horizontal concatenating

$C = [A;B]$ vertical concatenating

$A*x$

matrix multiplication

~ = not equal to

vector

$I = V < 0.05$

$V(I) = 0$

```
for n=1:6
y(n+1) = y(n)-0.1*y(n)
end
```

```
while ...
...
end
```

```
IF
Else
End
```

1==2 % false

1~=2

1&&0 %And

1||0 % or

xor(1,0)

---

who

whos

~~hello~~ save hello.mat v 存v进hello.mat

save hello.tex v -ascii

A(3,2)

A(2,:)

A(:,2)

A([1 3],:)

a=3; semicolon supressing count

format(long)

format(short)

V=[1 2 3]

V=[1;2;3]

V=1:0.1:2 1→2 0.1一间隔

1:6 1→6 1间隔

ones(2,3)

zeros(1,3)

rand(1,3)

rand(3,3)

randn(1,3) Gaussian Distribution

hist(w)

eye( )

~~help(eye)~~ help eye

size(A) size(A,1) size(A,2)

V=[1 2 3 4]

length(V) ans=4

length(A) ans=3

length([1;2;3;4;5])

pwd cd ls

load filename.dat

load('__.dat')