Linear Regression

Single feature

Size (ft^2) x	Price (1k) y
2104	460
1416	232

Use only x_1, 1 feature to predict output y

No. _____

Date _____

Multiple features (variables)

size (ft^2) x_1	Num of bedrooms x_2	Num of floors x_3	Age of home (years) x_4	Price (k) y
2104	5	1	45	460
$x^{(2)}\rightarrow$ 1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
⋮	⋮	⋮	⋮	⋮

$m = 47$

$n = 4$

Notation:

n = num. of features $n=4$

$x^{(i)}$ = input (features) of i^{th} training example $\quad x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$ vector

$x^{(i)}_j$ = value of feature j in i^{th} training example $\quad x^{(2)}_3 = 2$

Previously: $h_\theta(x) = \theta_0 + \theta_1 x$ (single)

Multiple:

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

Eg: $h_\theta(x) = 80 + 0.1 x_1 + 0.01 x_2 + 3 x_3 - 2 x_4$ ← decrease with the age of home

- 80: nominal, the house price is 80k
- $0.1 x_1$: up abit with the size
- $0.01 x_2$: up abit with num of bedroom
- $3 x_3$: up abit more with num of floors
- $-2 x_4$: drop abit more with bedroom

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

For convenience of notation, define $x_0 = 1 \quad (x^{(i)}_0 = 1)$ - zero feature

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

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

$$= \theta^T x \quad (1 \times (n+1) \text{ matrix})$$

$[\theta_0\ \theta_1 \cdots \theta_n]$ = θ^T, $1 \times (n+1)$ matrix (row vector); $\theta^T x$, x

Multivariate linear regression

Note: θ_0 as base price of house, θ_1 as price per ft, θ_2 as price per floor, etc... x_1 be the number of square meters in house, x_2 be num of floor, etc... Can use

For convenience purpose, assume $x^{(i)}_0 = 1$ for $(i \in 1, \dots, m)$. This allows us to do matrix operations with theta and $x^{(i)}$ match each other element-wise (that is, having same num of elements: $n+1$).]

Reason behind setting $x^{(i)}_0 = 1$:

$$X = \begin{bmatrix} x^{(1)}_0 & x^{(1)}_0 & x^{(3)}_0 \\ x^{(1)}_1 & x^{(1)} & x^{(3)}_1 \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

As a result, can calculate hypothesis as a vector with:

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

Gradient Descent for Multiple ~~Features~~ Variables

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

Parameters: ~~$\theta_0, \theta_1, \dots, \theta_n$~~ θ [n+1-dimensional vector]

Cost function:

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

$J(\theta)$

Gradient descent:

Repeat {

$n \geq 1$ $\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

}

Let's say 3 features:

~~θ_0~~ $\quad \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \left(x^{(i)}_0 \right)$ → $=1$ → same as previous equation

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}_1$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}_2$$

MATLAB / OCTAVE

Vectorized form [cost function]

$$\Rightarrow J(\theta) = \frac{1}{2m} (X\theta - \vec{y})^T (X\theta - \vec{y})$$

$(X\theta - y)^2$

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

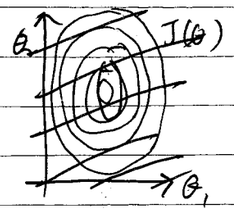
Gradient Descent in Practice I – Feature Scaling

Feature scaling

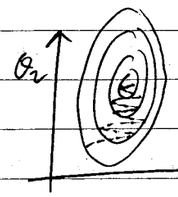
Idea: Make sure features are on a similar scale.

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

x_2 = number of bedrooms (1-5)



scale not similar



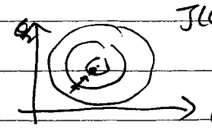
J(θ) ← graph will be very skinny

run gradient descent, gradient will take a long time to find global min.

so want to make it better, make it

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

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



the contour more like circles

gradient descent faster to global min.

$0 \le x_i \le 1$

Feature Scaling

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

$x_0 = 1$

Still close to $-1 \le x_i \le 1$

$0 \le x_1 \le 3$ ✓

$-2 \le x_2 \le 0.5$ ✓

not close → $-100 \le x_3 \le 100$ ✗

$-0.0001 \le x_4 \le 0.0001$ ✗

$-3 \le x \le 3$ ✓

$-\frac{1}{3} \le x \le \frac{1}{3}$ ✓

Mean normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean

(Do not apply to $x_0 = 1$)

Eg. $x_1 = \frac{\text{size} - 1000}{2000}$ → find, average size = 1000

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

$-0.5 \le x_1 \le 0.5$, $-0.5 \le x_2 \le 0.5$

$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1}$ ← μ_1: average value of x_1 in training set

s_1: ~~range of value of x_1~~ (max − min) (or standard deviation)

→ become nearer to scale

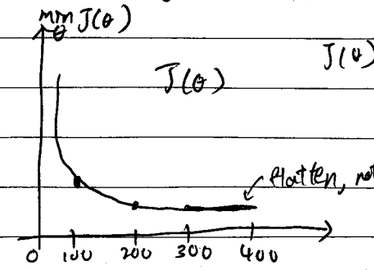
Gradient descent in practice II : Learning rate (α)

Gradient descent update

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

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

Make sure gradient descent is working correctly



Example automatic convergence test:

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

ε

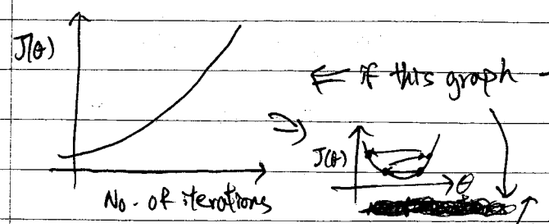
better to check the gradient descent's converge

than this

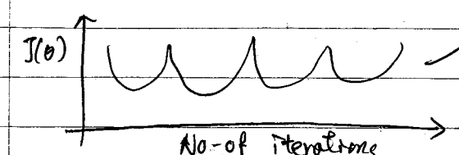
No.

Date

Making sure gradient descent is working correctly.



⇐ If this graph → Gradient descent not working. Use smaller α.



if this no happen, probably mean α too big, should make it smaller.

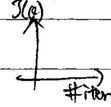
α too small ⇒ J(θ) need decrease

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

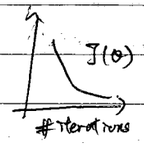
Summary:

- If α is too small cause → slow convergence (take a lot of iterations to reach global min)
- If α is too large: J(θ) may not decrease on every iteration; may not even converge. (slow converge also possible)

In order to debug all of these things, often plotting that J(θ) as a function of number of iterations can help you figure out what's going on.



plot over num. of iterations of gradient descent. If J(θ) ever increases, need to decrease α



To choose α, try running

0.001, 0.01, 0.1, 1

0.003 → 0.03 → 0.3 →

keep trying the values until found one value that's too small! and made sure found one value that's too large

Linear Regression with multiple variables

— Features and polynomial regression

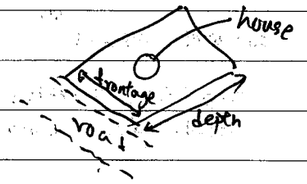
No.

Date

Housing prices prediction — 2 features

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

frontage: how wide the area of house



Area

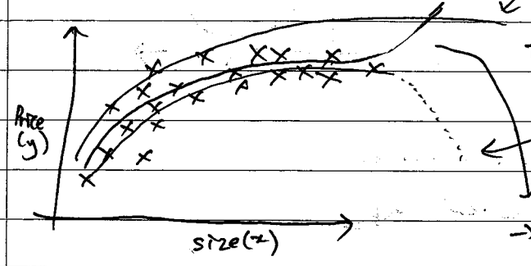
new feature → $x = \text{frontage} * \text{depth}$

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

land area

Polynomial regressions

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



try quadratic equation

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

graph will go well first but come down, in this case, price should go up when size increase, so this not suit

try cubic function $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$ is better as it doesn't come back down eventually.

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

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

$x_1 = \text{size}$

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

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

size	1 – 1000
size^2	1 – 1,000,000
size^3	1 – 10^9

Note: - We can improve our features and form of our hypothesis function in a couple different ways.

- combine features into one. For example, combine x_1 and x_2 into a new feature x_3 by taking $x_1 \cdot x_2$.

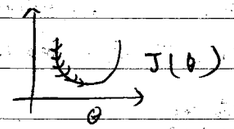
Polynomial Regression

- Our hypothesis function need not be linear (a straight line) if it does not fit the data well.
- Change the behavior or curve of our hypothesis function by making it a quadratic, cubic or square root function (or any other form)
- Important thing, if you choose features this way then feature scaling becomes very important.

Normal Equation

No.
Date

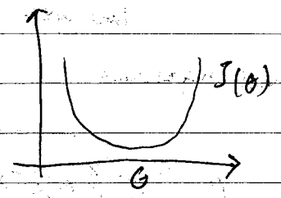
Gradient Descent
- keep taking iterations to reach global minimum



Normal equation: Method to solve for θ analytically.

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

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



To minimize $J(\theta)$: $\frac{d}{d\theta}J(\theta) = \dots \overset{\text{set}}{=} 0$

Solve for θ

↑ This is just for real number

we should deal with theta no longer a real number

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

$(n+1)$-dimensional parameter vector

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

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

How to minimize $J(\theta)$
- take partial derivative of J, with respect to every parameter of θ of J in turn, and then, to set all of these to 0

x^i_j i=row, j=column

No.
Date

Examples: $m=4$

x_0	Size (ft²) x_1	Num. of bedrooms x_2	Num. 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

Add 1 column and name as x_0

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$m \times (n+1)$ dimensional matrix

m-dimensional vector

$\theta = (X^T X)^{-1} X^T y$ (~~minimize~~ theta that minimize cost function)

Let's say

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

$$x^{(1)} = \begin{bmatrix} x_0^{(1)} \\ x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \\ \vdots \\ x_n^{(1)} \end{bmatrix} \in \mathbb{R}^{n+1} \qquad X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}$$

(design matrix)

$m \times (n+1)$

$x_0^{(i)} = 1$

Eg. If $x^{(1)} = \begin{bmatrix} 1 \\ x_1^{(1)} \end{bmatrix}$ → Only 1 features

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ \vdots & \\ 1 & x_m^{(1)} \end{bmatrix} \quad m \times 2 \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

vector y

No. _____
Date _____

Normal Equation and Noninvertibility

$\theta = (X^TX)^{-1}X^Ty$ ← Using this, it's ok not to use this (not necessary)

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

Set $A = X^TX$

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

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

$x' = x^T$ in Octave

$\min_\theta J(\theta)$

Feature scaling (gradient descent)

$0 \le x_1 \le 1$

$0 \le x_2 \le 1000$

$0 \le x_3 \le 10^{-5}$

m training examples, n features.

Gradient Descent

- Need to choose α
- Needs many iterations

Disadvantage

→ you need to choose α, and keep running diff. α to see the best, and so that is sort of extra work and extra hassle

→ needs many more iterations, may make it slower.

Advantage → even have millions of features, you can run gradient descent and it still efficient, it will do something reasonable.

- Work well even when n is large

Normal Equation

- No need to choose α → convenient, simple to implement
- Don't need to iterate
  ↳ no need to plot J of θ or check convergence or take all those extra steps.

Disadvantage

- Need to compute $(X^TX)^{-1}$ $n \times (n+1)$ matrix
- Slow if n is very large

n large

$n = 10^6$

n small

$n = 10000$ actually a modern computer will work not that fast.

$n = 100$ ✓
$n = 1000$ ✓

$O(kn^2)$

$O(n^3)$ need to calculate inverse of X^TX

With normal eqn, computing inversion has complexity $O(n^3)$. So if we have very large num of features, normal eqn will be slow. In practice, when n exceeds 10k, it might be a good time to go from a normal solution to an iterative process.

Normal equation

$\theta = (X^TX)^{-1}X^Ty$

- What if X^TX is non-invertible? (singular / degenerate)
- Octave: `pinv(X'*X)*X'*y`

non-invertible matrices

pinv (pseudo-inverse)
inv (inverse)

What if X^TX is non-invertible?

2 common causes?

first • Redundant features (linearly dependent).

E.g. x_1 = size in feet^2

x_2 = size in m^2 ← related, $1m = 3.28$ ft

$x_1 = (3.28)^2 x_2$

second • Too many features (e.g. $m \le n$). → example: $m = 10$, $n = 100$ (not enough data)

- Delete some features or use regularization (later topic in this course)

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

sometimes work but not always be a good idea

Solutions to above problems include deleting a feature that is linearly dependent with another or deleting one or more features when there are too many features.