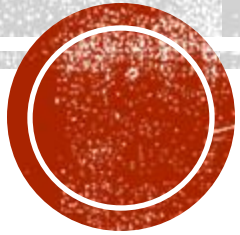
CH3: MULTIPLE LINEAR REGRESSION

Lecturer: NGIN KIMLONG



OBJECTIVES

- Extend our regression model routines to support multiple features Extend data structures to support multiple features
- Rewrite prediction, cost and gradient routines to support multiple features
- Utilize NumPy np.dot to vectorize their implementations for speed and simplicity

CONTENT

1. Multiple feature
2. Vectorization
3. Gradient descent for multipe linear regression
4. Feature scaling

1. MULTIPLE FEATURES (VARIABLES)

The training dataset contains three examples with four features (size, bedrooms, floors and, age) shown in the table below. Note that, unlike the earlier labs, size is in sqft rather than 1000 sqft

(x1) Size (sqft)	(x2) Number of Bedrooms	(x3) Number of floors	(x4) Age of Home	(x5) Price (1000s dollars)
2104	5	1	45	460
1416	3	2	40	232
852	2	1	35	178

X_train = np.array([[2104, 5, 1, 45], [1416, 3, 2, 40], [852, 2, 1, 35]])

y_train = np.array([460, 232, 178])

$x_j = j^{th}$, j = 1, … , 4

$n = number\ of\ features, n = 4$

$\vec{x} = feature\ of\ i^{th}$

Ex: $\vec{x}^2$ is located in row 2
= [1416, 3, 2, 40]

- To get specific feature in i^{th}

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

Example: $x_3^{(2)} = 2$

Model:

Previously: $f_{w,b}(x) = wx + b$ Single featrues

- $f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$

Example: $f_{w,b}(x) = 1.0x_1 + 4x_2 + 10\,x_3 + (-2x_4) + 80$

size bedrooms floors years

N-features: $f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b$

- **Where, w and x are vector. We can write as:**

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b$$

$\vec{w} = [w_1\ w_2\ w_3\ \ldots w_n]$

b is a number

$\vec{x} = [x_1\ x_2\ x_3\ \ldots x_4]$

$$f_{\vec{w},b}(\vec{x}) = \vec{w}.\vec{x} + b = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b$$

dot product from linear algebra of x the vector

In python

import numpy as np

np.dot(…)

2. VECTORIZATION

- The study will focus on vectorized code that allows for arithmetic operations using modern numerical linear algebra libraries.
- Example: parameters and features

$\vec{w} = [w_1\ w_2\ w_3]$

b is a number

$\vec{x} = [x_1\ x_2\ x_3]$

Linear algebra: count from 1

W = np.array([1.0, 2.5, -3.3])

B = 0

X = np.array([10, 20, 30])

❖ **Without vectorization**

$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + w_3x_3 + b$

```
w = np.array([1.0, 2.5, -3.3])
b = 0
x = np.array([10, 20, 30])
```

f = w[0]*x[0] + w[1]*x[1]+w[2]*x[2] + b

```
    f = 0
    for i in range(len(w)):
    f+= w[i] * x[i]
    f = f + b
    print("The hypothesis result is %f " %f)
```

Vectorization

Syntax: fp = np.dot

$f_{\vec{w},b}(\vec{x}) = \vec{w}.\vec{x} + b$

Then,

f = np.dot(w, x) + b

```
    f2 = np.dot(w, x) + b
    print("The hypothesis result using np.dot is %f " %f2)
[23] ✓ 0.0s
```

```
... The hypothesis result using np.dot is -39.000000
```

3. GRADIENT DESCENT FOR MULTIPE LINEAR REGRESSION

Previous notion

Parameters: $w_1, w_2, \dots w_n$

Model: $f_{\vec{w},b}(\vec{x}) = w_1 x_1 + \dots + w_n x_n + b$

Cost function: $J(w_1, \dots w_n, b)$

$\Rightarrow$

Gradient descent

Repeat till convergence {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} (w_1, \dots w_n, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w_1, \dots w_n, b)$$

Vector Notation

$\vec{w} = [w_1, \dots w_n]$

b still a number

$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

$J(\vec{w}, b)$

Repeat till convergence {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$$

}

4. FEATURE SCALING

- Main objective of feature scaling is for helping gradient descent working faster

Example:

$\widehat{price} = w_1 x_1 + w_2 x_2 + b$

X1 = size $(feet^2)$ (range: 300-2000)

X2 = bedrooms (range: 0-5)

Assume that:

x1, x2 = 2000, 5

Size of the parameters w1, w2?

w1, w2, b = 50, 0.1, 50

$\widehat{price} \approx$ $100,050,050

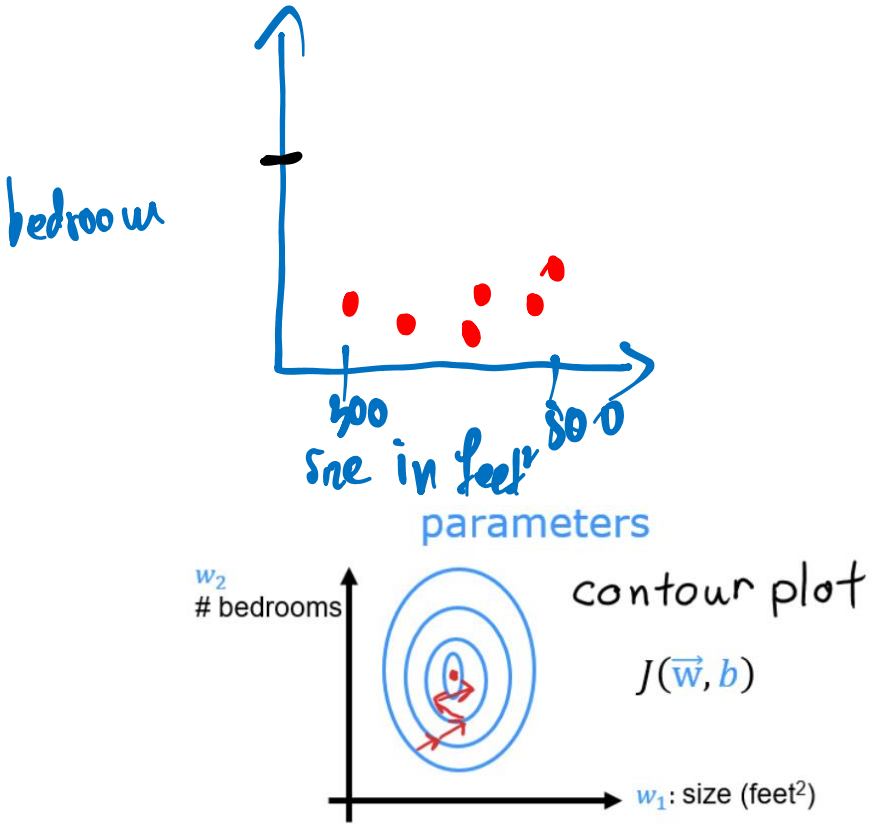
$\widehat{price}$ = 50*2000 + 0.1*5 + 50

w1, w2, b = 0.1, 50, 50

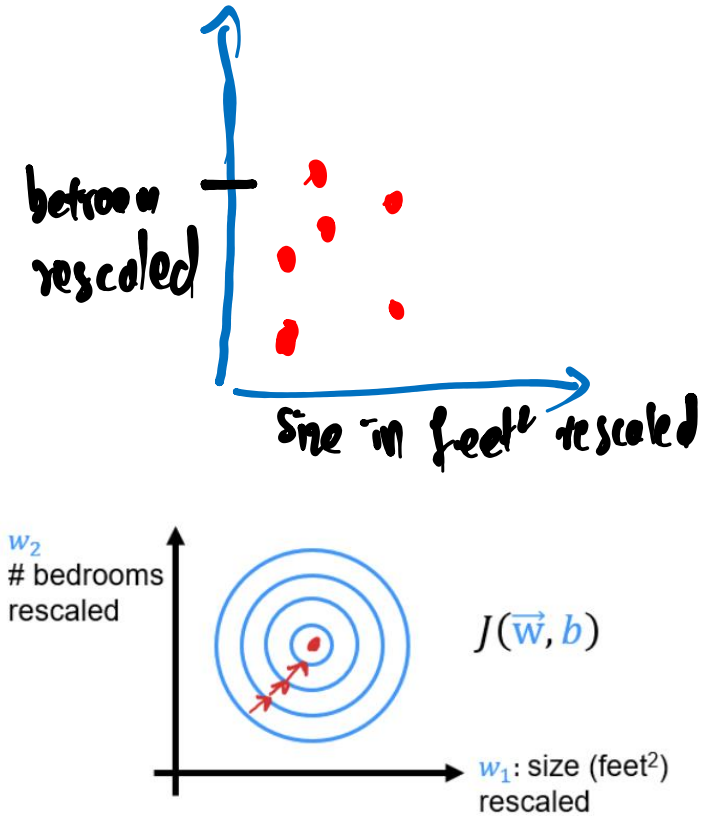
$\widehat{price} = 0.1 * 2000 + 50 * 5 + 50$

$\approx 500k$

- Comparison between feature size and gradient descent:



When feature scales do not match, the plot of cost versus parameters in a contour plot is asymmetric.

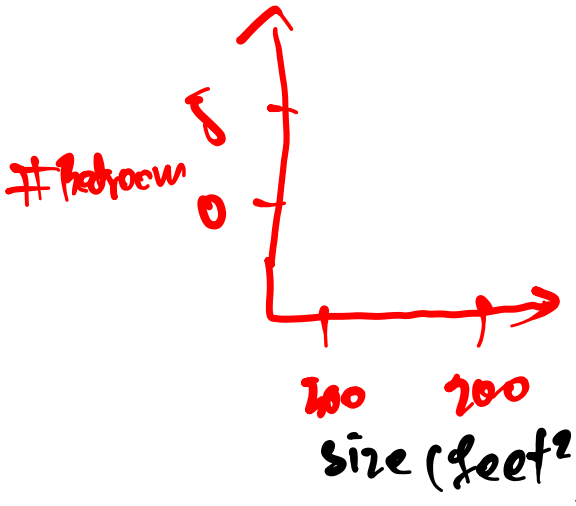


The left plot is the cost contour plot of w[0], the square feet versus w[1], the number of bedrooms before normalizing the features. The plot is so asymmetric, the curves completing the contours are not visible. In contrast, when the features are normalized, the cost contour is much more symmetric. The result is that updates to parameters during gradient descent can make equal progress for each parameter.

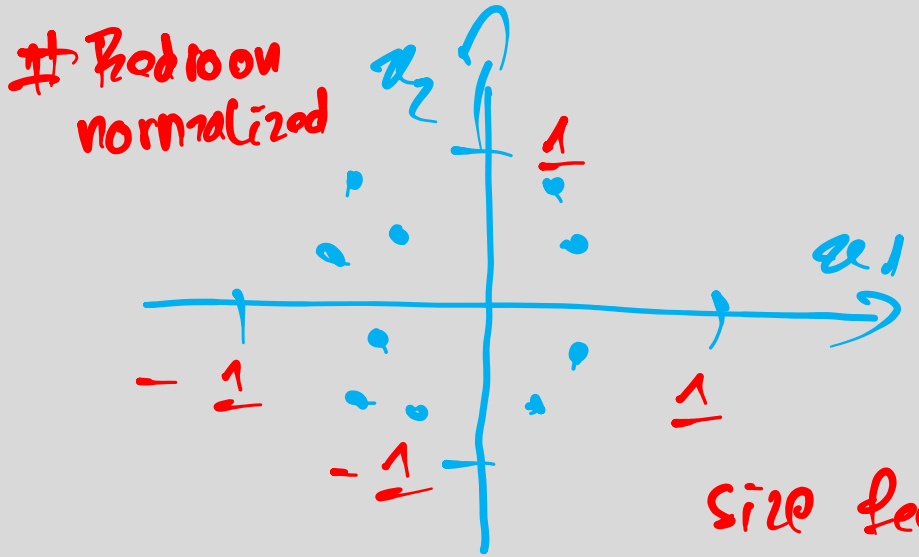
How to scale feature?

a. Mean normalization

- Mu = μ_1 = 600

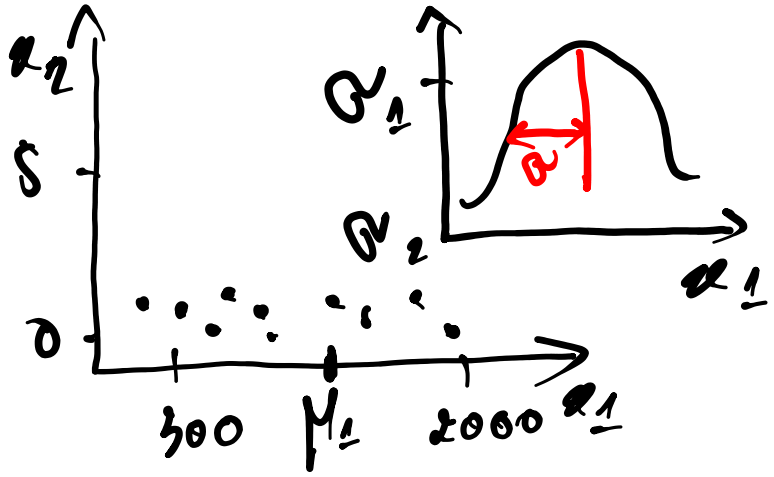


To find mean normailzation of x_1, firstly find average/mean of x_1 on your dataset. We can call it mu_1.



b. Z-score normalization

It's a method that we can use to scale features Z-score normalization, you can calculate standard deviation of each features (σ).



$300 \leq x_1 \leq 2000$

$x_1 = \frac{x_1 - \mu_1}{\sigma_1}$

$0 \leq x_2 \leq 5$

$x_2 = \frac{x_2 - \mu_2}{\sigma_2}$

$-0.67 \leq x_1 \leq 3.1$ $\quad -1.6 \leq x_2 \leq 1.9$

#Bedroom normalized

Size in feet normalized

- Make sure all features are on a similar scale
- Solution:
  - Replace x_j with $\frac{x_j - \mu_j}{s_j}$

- mean $\mu_j = \frac{1}{m}\sum_{i=1}^{m} \mu_j^{(i)}$
- s_j standard deviation or simply $\max - \min$

- Aceptable range for each x_j:
  - $-1 <= x_j <= 1$
  - $-3 <= x_j <= 3$
  - $-0.3 <= x_j <= 0.3$

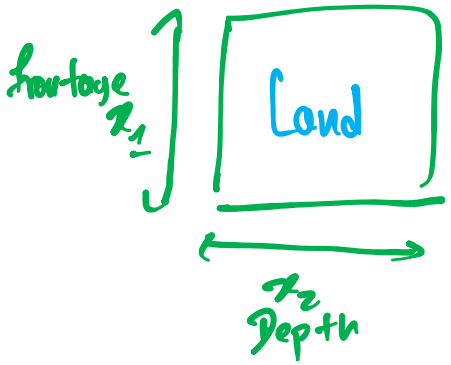
- Other case:
  - $0 \leq x_j \leq 3 \Rightarrow Okay\ , no\ rescaling$
  - $-2 \leq x_j \leq 0.5 \Rightarrow Okay\ , no\ rescaling$

  - $-100 \leq x_j \leq 100 \Rightarrow Too\ large\ , rescaling$
  - $-0.001 \leq x_j \leq 0.001 \Rightarrow Too\ small\ , rescaling$

c. Feature Engineering

Choosing feature can impact on operations of your learning algorithm .

- Feature engineering use intuition to design new a new feature by **changing** or **combining** feature.

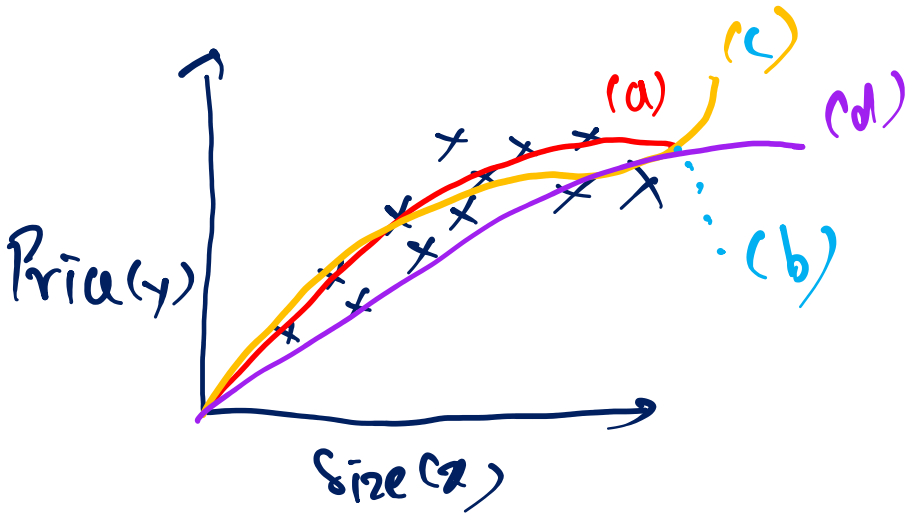


- $f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + b$
- $Area = Frontage \times depth$
- $New\ feature \Rightarrow x_3 = x_1x_2$

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + w_3x_3 + b$$

d. Polynomial Regression

- For polynomial regression algorithm will make line curve.
- Example:



1. For (a) cannot make straight line but is a curve line

$$f_{w,b}(x) = w_1x_1 + w_2x^2 + b$$

2. If quardratic of x^2, the curve line will continue down (b). Mean that size of house is large, but low price
3. If x^3 (cube) then the line will curve up and down. Mean that It's better than x^2

$$f_{w,b}(x) = w_1x_1 + w_2x^2 + w_3x^3 + b$$

- Noted: When x feature has power, the feature scaling will more and more significance for model

PRACTICE

Description: By using house price dataset. Please Convert house price into polynomial form. Each feature will quadratic 1, 2, 3, 4 respectively.

1. Finding gredient descent and it's cost for normal value with quadratic value.