Gradient Descent for Multiple variable:

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

way 1:
Feature scaling:

→ variable/feature range can vary

→ if the scale is almost similar, it gives a better result.

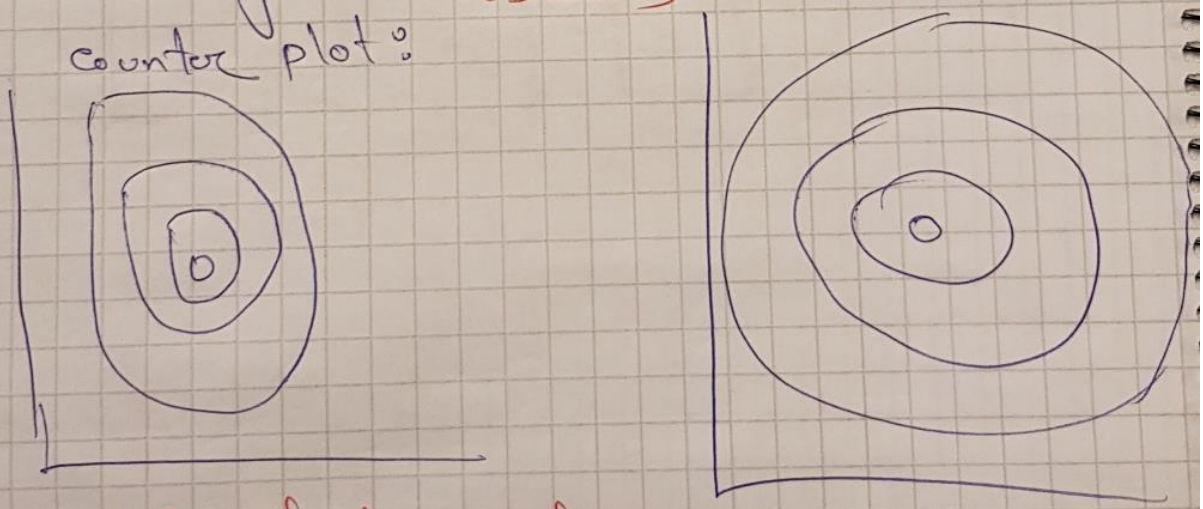
For example,

x_1 range: 0 to 20,000 → room size

x_2 range: 1 to 5 → bedrooms

if scaled:

Counter plot:



calculate feature scaling:

$$x_1 = \frac{\text{size}}{20,000}, \quad x_2 = \frac{\text{rooms}}{5}$$

range

we get range normally: $-1 \leq x \leq 1$.

Way 2 → Mean normalization:

$$\text{Normalized value} := \frac{\text{input value} - \text{mean value}}{\text{range}}$$

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

Example:

mean = 1000

Range: 100 to 2000 = 1900

Price = ?

$$x_i := \frac{\text{price} - 1000}{1900}$$

* How to improve features?

(1) combine multiple feature into one.

eg > x_1 = highest, x_2 = width, $\boxed{x_3 = x_1 \times x_2}$

(2) Polynomial regression: $y = a + bx + cx^2$ ....

so we better do the square root:

eg. $y = a + bx_1 + c \cdot \sqrt{x_1}$.

Gradient Descent vs. Normal equation:

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

→ need to choose α → no need of α

α = learning rate.

→ need many iterations → no need to iterate

→ complexity $O(kn^2)$ → complexity $O(n^3)$

→ good if $n \times n$ is large, or lots of features,

eg. 10^6

inverse of $(X^T X)$

→ Good if n is small ($n \times n$ matrix)

eg. 10,000

it may happen that $(X^T X)$ is not invertible.

why?

① may be the units are different

x_1 = size in $feet^2$

x_2 = " " m^2

② too many features.