LINEAR REGRESSION WITH MULTIPLE VARIABLES

Also known as **multivariable linear regression**.

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$$X^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix} \in \mathbb{R}^4$$

$$X^{(2)}_3 = 2$$

$n = 4 \rightarrow x_1, x_2, x_3, x_4$

Notation:

- n → the number of features
- m → the number of training examples
- $x^{(i)}$ → the input (features) of the i^{th} training example
- $x^{(i)}_j$ → the value of feature j in the i^{th} training example

Hypothesis:

Since we now have multiple features:

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

For convenience of notation, $x^{(i)}_0 = 1$ This allows us to do matrix operations with θ and x.

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

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \cdots & \theta_n \end{bmatrix}$$

$\theta^T x$ → inner product of parameter vector θ and vector x

→ Gradient Descent for multiple variables

Hypothesis

$$h_\theta(x) = \theta_0 \underbrace{x_0}_{=1} + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

Parameters

$\theta_0 \ \theta_1 \ \theta_2 \ \dots \ \theta_n \longrightarrow \theta$ (where θ is a $n+1$ dimensional vector)

Cost Function

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

Gradient Descent

Repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\delta}{\delta\theta_j} J(\theta_0,\dots,\theta_n)$$

}

Simultaneously update for every $j = 0, \dots, n$

So:

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \underbrace{x_0^{(i)}}_{=1}$$

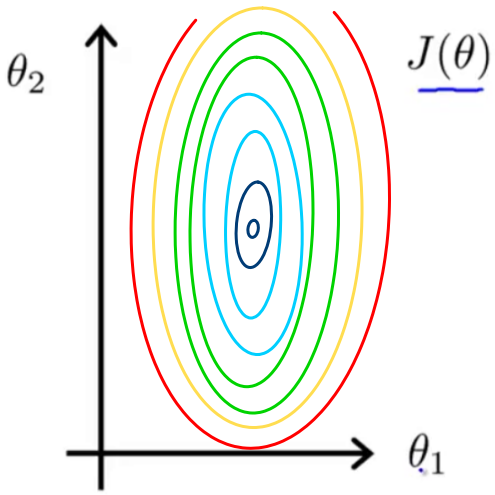
$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 x_2^{(i)}$$

θ will descend quickly on small ranges and slowly on large.

→ Feature Scaling

Feature scaling involves dividing the input values by a range (ex: maximum value − minimum value) of the input variable, resulting in a range of -1 - 1.



Example: x_1 = size (0 - 2000 feet^2)
x_2 = nr of bedrooms (1-5)

$J(\theta)$ function looks very skinny, and gradient descent will take a long time (it's inefficient)

Use instead: x_1 = size (feet^2) / 2000
x_2 = nr of bedrooms / 5

$J(\theta)$ function is more circular, and gradient descent is faster.

Ideally, we use $-1 \leq x_i \leq 1$ or $-0,5 \leq x_i \leq 0,5$

→ Mean Normalization

Mean normalization involves subtracting the average value for an input variable from the values for that input variable, resulting in a new average value of zero (0).

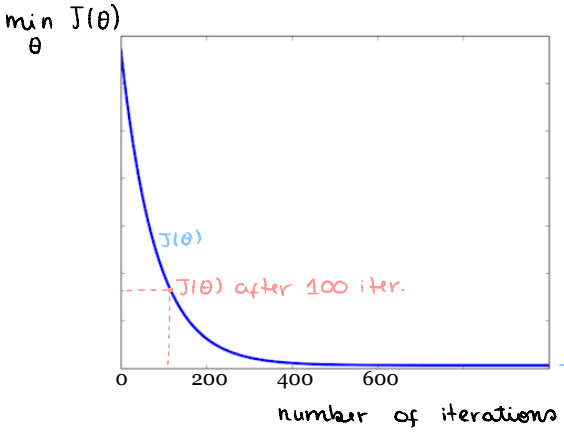
$$x_1 = \frac{x_1 - \mu_1}{s_1}$$

→ average value of x_1

→ Standart Deviation (max - min)

→ "Debugging" gradient descent

- Make a plot with the number of iterations on the x axis.
- Plot the cost function $J(\theta)$ over the number of iterations of gradient descent. If $J(\theta)$ increases, then we need to decrease α.



Automatic convergence test

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

$J(\theta)$ should decrease after every iteration, but after "n" iterations it looks almost flat – it has converged.

Summary

- If α is too small → slow convergence
- If α is too large → $J(\theta)$ may not decrease in every iteration, it may not converge

→ to choose α try: 0,001 ; 0,01 ; 0,1 ; 1 ; ...

→ Features

We can improve our features and the form of our hypothesis function, by combining multiple features into one.

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{frontage}_{x_1} + \theta_2 \times \underbrace{depth}_{x_2}$$

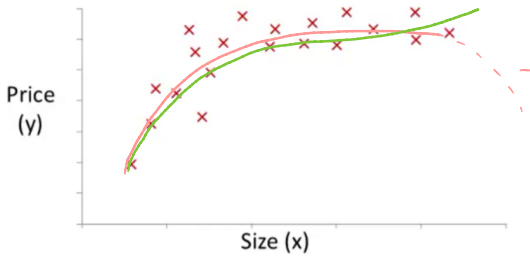
→ we can create our own features like: $x_3 = x_1 \times x_2$ (AREA)

→ Polynomial Regression

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

Quadratic Model

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



→ goes down again

Cubic Model

$\theta_0+\theta_1 x+\theta_2 x^2+\theta_3 x^3$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 =$$

$$= \theta_0 + \theta_1 (\underbrace{size}_{x_1}) + \theta_2 (\underbrace{size}_{x_2})^2 + \theta_3 (\underbrace{size}_{x_3})^3$$

→ necessary to use feature scaling

By having insights into the shape of a function, we can choose different features and get better models.

→ Normal Equation

Method to solve for θ analytically.

Intuition

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

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

Solve for $\theta_0, \theta_1, \dots, \theta_n$ → this will give the values of θ that minimize the cost function.

Examples: $m = 4$.

	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
x_0	x_1	x_2	x_3	x_4	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

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

$m \times (n+1)$ $\qquad$ m-dimensional vector

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

With the Normal Equation, feature scaling is not necessary.

→ When to use Gradient Descent and when to use Normal Equation?

m training examples, n features

Gradient Descent	Normal Equation
• Need to choose α	• No need to choose α
• Needs many iterations	• No need to iterate
• Works well even when n is large	• Need to compute $(X^TX)^{-1}$
	• Slow if n is very large

With the Normal Equation, computing the inversion has complexity $O(n^3)$. So, if we have a large number of features, the normal equation will be slow.

→ If $n > 10000$ we should use Gradient Descent

→ Normal Equation Noninvertibility

What if X^TX is non-invertible?

Two common causes:

- Redundant features (linearly dependent)
  Ex: x_1 = size in feet^2
  x_2 = size in m^2
- Too many features (ex: $m \leq n$)
  → delete some features or use regularization.

Solution (in Octave)

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

→ if we use pinv, Octave will give us the value we want, even if X^TX is non-invertible.