

Week2

Multi-variate linear regression

Multiple features

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

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

m = the number of training examples

n = the number of features

multivariable form of the hypothesis function

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

or

$$h_{\theta}(x) = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

cost function

Cost function:

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

Gradient descent

Gradient descent:

Repeat {

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

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

Comparison

Previously ($n=1$):

Repeat {

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

$\underbrace{\hspace{10em}}_{\frac{\partial}{\partial \theta_0} J(\theta)}$

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

(simultaneously update θ_0, θ_1)

}

New algorithm ($n \geq 1$):

Repeat {

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

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

}

Gradient descent in practice I

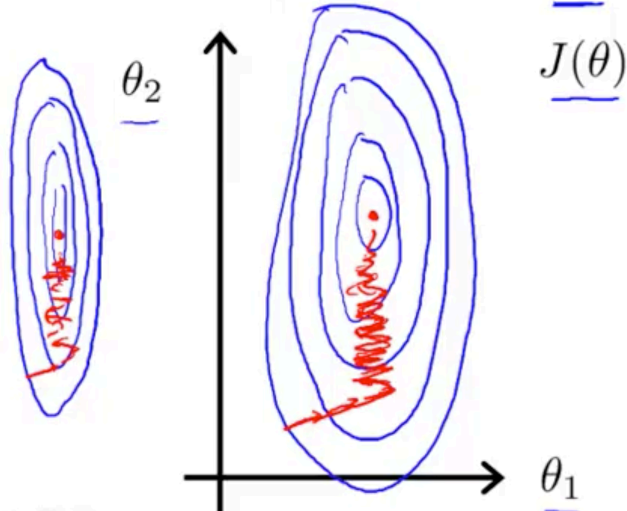
Goal: speed things up

feature scaling: dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1.

- Idea: Make sure features are on a similar scale

E.g. $x_1 = \text{size (0-2000 feet}^2\text{)}$ ←

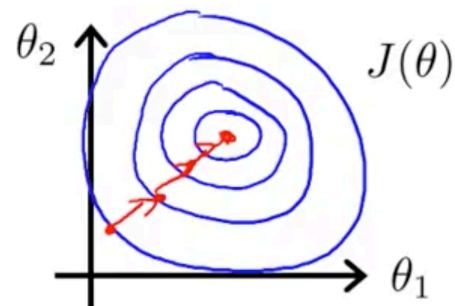
$x_2 = \text{number of bedrooms (1-5)}$ ←



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

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

$$0 \leq x_1 \leq 1 \quad 0 \leq x_2 \leq 1$$



Notes: faster to approach the center point

- Get every feature (x_i) into an approximately range of (-1,1), slightly off the range is okay (for example, -3 to 3) but large range like -100 to 100 is not acceptable.

Mean normalization: subtracting the average value for an input variable from the values for that input variable resulting in a new average value for the input variable of just zero.

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

Where μ_i is the **average** of all the values for feature (i) and s_i is the range of values (max - min), or s_i is the standard deviation.

Example: Suppose you are using a learning algorithm to estimate the price of houses in a city. You want one of your features x_i to capture the age of the house. In your training set, all of your houses have an age between 30 and 50 years, with an average age of 38 years. Which of the following would you use as features, assuming you use feature scaling and mean normalization?

$$x_i = \frac{\text{age of house} - 38}{20}$$

Gradient descent in practice II

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

Automatic convergence test: Declare convergence if $J(\theta)$ decreases by less than E in one iteration, where E is some small value such as 10^{-3} . However, in practice it's difficult to choose this threshold value.

Polynomial Regression

Our hypothesis function need not be linear (a straight line) if that does not fit the data well. We can change the behavior or curve of our hypothesis function by making it a quadratic, cubic or square root function (or any other form).

Example:

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$$

Compute Parameters Analytically

Normal function

explicitly taking its derivatives with respect to the θ_j 's, and setting them to zero. This allows us to find the optimum theta without iteration.

Examples: $m = 4$.

	Size (feet ²)	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

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

$m \times (n+1)$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

m -dimensional vector

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

Notes: no need to do feature scaling with the normal equation.

Gradient Descent	Normal Equation
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
$O(n^2)$	$O(n^3)$, need to calculate inverse of $X^T X$
Works well when n is large	Slow if n is very large

Normal Equation Non-invertibility

If $X^T X$ is noninvertible, the common causes might be having:

- Redundant features, where two features are very closely related (i.e. they are linearly dependent)
- Too many features (e.g. $m \leq n$). In this case, delete some features or use "regularization"