WEEK 2

Multivariate Linear Regression

Notation: n : # of featurs

$x^{(i)}$: input of i^{th} training example

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

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

define $x_0 = 1$, then $x = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$

also: $\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$

then: $h_\theta(x) = \theta^T x$

Cost function $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Gradient descent:

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

Could be due to learning rate α too large, try use small α

Mathematicly, for sufficiently small α, $J(\theta)$ should decrease on every iteration

but if α is too small, gradient decent can be slow to converge

Features and Polynomial Regression

e.g. $h(\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 = \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$

$\hookrightarrow x_1 = (size) \quad x_2 = (size)^2 \quad x_3 = (size)^3$

One should take care of feature scaling, since $x_1 \sim x_3$ takes every different ranges

Normal Equation:

Normal equation: method solve for θ analytically

$$\frac{\partial J(\theta)}{\partial \theta_j} = 0 \implies \theta_0 \cdots \theta_n$$

x_0	feature 1 x_1	feature x_2	feature 3 x_3	feature 4 x_4	results y
1					
1					
1					
$\vdots$					

$$X = \begin{bmatrix} 1 & \cdots \\ 1 & \cdots \\ 1 & \cdots \\ 1 & \cdots \end{bmatrix}_{m \times (n+1)} \qquad y = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}_{m \times 1}$$

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

Generally:

m examples. $(x^{(1)}, y^{(1)}), \cdots\cdots (x^{(m)}, y^{(m)})$

n features

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1} \qquad X = \begin{bmatrix} \text{---} & (x^{(1)})^T & \text{---} \\ & \cdots & \\ \text{---} & (x^{(m)})^T & \text{---} \end{bmatrix}_{m \times (n+1)}$$

$\uparrow$ design matrix

It Does NOT matter whether using feature scaling or not in this case

Compare:

Gradient decent	Normal equation
◦ need to chose α	◦ no need chose α
◦ needs many iterations	◦ No need of iteration
◦ works well even feature n is larger	◦ need to compute $[X^TX]^{-1}$ (slow when n is large)
◦ more general	◦ liner regression only
	◦ $[X^TX]^{-1}$ may non-invertible

what does $[X^TX]$ is non-invertible mean?

- Redundant features
- Too many features ($m \le n$)

Vectorization:

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j = \theta^T x$$