✨

기본 지도 학습 알고리즘(2)

다중 선형 회귀 (Multiple Linear Regression)

- 집 크기, 건물 나이, 층, 지하철 역까지 거리
  - ⇒ 예측 ⇒ 목표 변수
- 시각화하기 힘듦
- **표현 방식**
  - **각 데이터**
    - 1번째 입력 변수 → x^(1)
    - 합치기 → x(j)^(i)
  - **가설 함수**

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

  - **식 표현**

  : **세타, x가 행벡터 형태로 존재할 때**

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

- **입력 변수와 파라미터 표현**

$$X\theta = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \cdots & x_n^{(2)} \\ \vdots & & & \\ x_0^{(m)} & x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} = \begin{bmatrix} \theta_0 x_0^{(1)} + \theta_1 x_1^{(1)} + \cdots + \theta_n x_n^{(1)} \\ \theta_0 x_0^{(2)} + \theta_1 x_1^{(2)} + \cdots + \theta_n x_n^{(2)} \\ \vdots \\ \theta_0 x_0^{(m)} + \theta_1 x_1^{(m)} + \cdots + \theta_n x_n^{(m)} \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix}$$

$$X\theta - y = \begin{bmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

$$error = X\theta - y = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

- **경사 하강법 표현**

$$X^T \times error = \begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \cdots & x_0^{(m)} \\ x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \\ \vdots \\ x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(m)} \end{bmatrix} \times \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\ \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\ \vdots \\ \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_n^{(i)} \end{bmatrix}$$

$$\theta - \alpha\frac{1}{m}(X^T \times error) = \begin{bmatrix} \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\ \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\ \vdots \\ \theta_n - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_n^{(i)} \end{bmatrix}$$

$$\theta \leftarrow \theta - \alpha\frac{1}{m}(X^T \times error) = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \leftarrow \begin{bmatrix} \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\ \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\ \vdots \\ \theta_n - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_n^{(i)} \end{bmatrix}$$

- **경사 하강법 VS 정규 방정식**

경사 하강법	정규 방정식
적합한 학습율 α를 찾거나 정해야 한다.	학습율 α를 정할 필요가 없다.
반복문을 사용해야 한다.	한 단계로 계산을 끝낼 수 있다. ($\theta = (X^T X)^{-1} \cdot X^T y$)
입력 변수의 개수 n이 커도 효율적으로 연산을 할 수 있다.	입력 변수의 개수 n이 커지면 커질수록 월등히 비효율적이다. (행렬 연산을 하는 비용이 경사 하강법을 하는 것보다 크다)
	역행렬 $(X^T X)^{-1}$이 존재하지 않을 수도 있다 (이때는 pseudo inverse를 이용해서 다르게 계산하는 방법이 있기 때문에 큰 문제는 안 됨)