

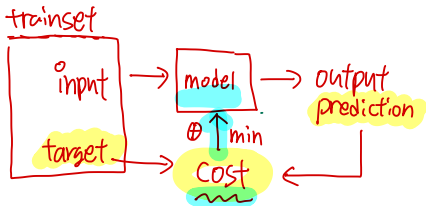
- linear regression
 - normal equation ← 선형방정식
 - gradient descent ← 미분
강화학습

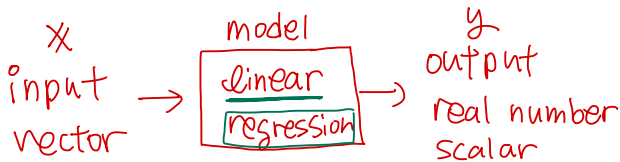
Preclass 01: Linear Regression

[SCS4049] Machine Learning and Data Science

Seongsik Park (s.park@dgu.edu)

School of AI Convergence & Department of Artificial Intelligence, Dongguk University





$$\hat{y} = \underline{\underline{\theta^T x}}$$

Linear regression

Linear regression

output $y \leftarrow$ prediction $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

Linear model

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$
$$= \underline{\theta^T X} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad (1)$$

In this equation,

- $\hat{y}$ is the predicted value (for true y)
- n is the number of features, *input의 특징의 수*
- x_i is the i -th feature value
- θ_j is the j -th model parameter
including the bias term θ_0 and the feature weights $\theta_1, \theta_2, \dots, \theta_n$

Linear regression

This can be written much more concisely using a vectorized form,

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \quad (2)$$

$$= \underline{\underline{\theta^T \mathbf{x}}} \quad (3)$$

In this equation,

input $\rightarrow$ model $\rightarrow$ output
 $\mathbf{x}$ $\uparrow$ $\oplus$ $\hat{y}$

- θ is the model's parameter vector, containing the bias term θ_0 and the feature weights θ_1 to θ_n
- $\mathbf{x}$ is the instance's feature vector, containing x_0 to x_n always equal to 1
input $\quad \quad \quad \begin{matrix} 1 \\ \hline 1 \end{matrix}$ bias
- $\theta \cdot \mathbf{x}$ is the dot product of the vectors θ and $\mathbf{x}$, which is of course equal to $\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$

Linear regression

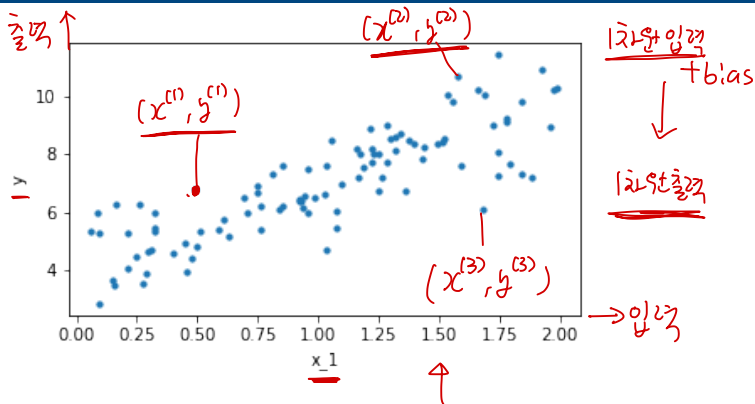


Figure 1: Linear regression: training dataset

Generating training dataset

$$y \approx \theta_0 + \theta_1 x \quad (4)$$

$$y = \theta_0 + \theta_1 x + \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, \sigma^2) \quad (5)$$

Cost function

θ
↓

$(x^{(1)}, y^{(1)}) (x^{(2)}, y^{(2)}) \dots (x^{(n)}, y^{(n)})$

$\Rightarrow \theta?$

That's the linear regression model – but how do we train it?

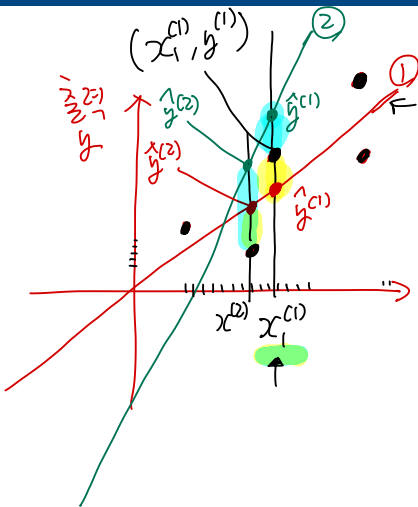
Recall that training a model means setting its parameters so that the model best fits the training set.

We first need a measure of how well (or poorly) the model fits the training data.

ফেসন রেসলিং গোল ← θ_1, θ_2

The most common performance measure of a regression model is the Root Mean Square Error (RMSE).

$$\text{RMSE}(\mathbf{X}, \boldsymbol{\theta}) = \sqrt{\frac{1}{m} \sum_{i=1}^m \left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)} \right)^2} \quad (6)$$



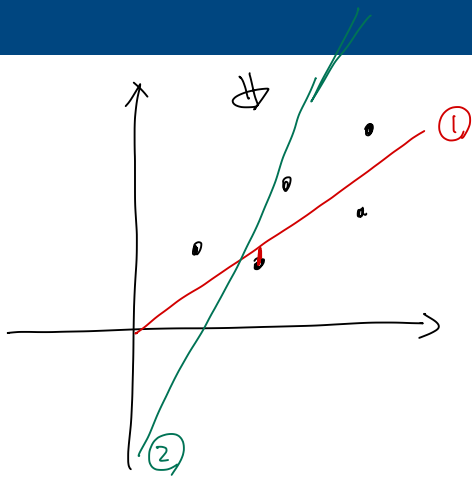
$$\hat{y} = \theta_0 + \theta_1 x_1$$

① $\theta = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

② $\theta = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

입력 x_1

error⁽ⁱ⁾ = $\frac{\text{실제값}}{y^{(i)}} - \frac{\text{예측값}}{\hat{y}^{(i)}}^2 \geq 0$



$$\sum_{i=1}^5 \text{error}^{(i)2}$$

$$= \sum_{i=1}^5 (y^{(i)} - \hat{y}^{(i)})^2$$

$$\textcircled{1} \sum 1^2 = 20$$

$$\textcircled{2} \sum 1^2 = 100$$

직접 계산
 $\text{error} \textcircled{1} \geq \textcircled{2}$

$$\Rightarrow \sum \text{error}^2$$

$$\textcircled{1} \leq \textcircled{2}$$

Cost function

regression problem $\rightarrow$ suitable measure, error index
cost function.

We need to find the value of θ that minimizes the RMSE. In practice, it is simpler to minimize the sum of squared error (SSE) than the MSE or the RMSE.

$$\hat{\theta} = \arg \min \text{RMSE}(\mathbf{X}, \theta) = \arg \min \sqrt{\frac{1}{m} \sum_{i=1}^m (y^{(i)} - \theta^T \mathbf{x}^{(i)})^2} \quad (7)$$

$$= \arg \min \text{MSE}(\mathbf{X}, \theta) = \arg \min \frac{1}{m} \sum_{i=1}^m (y^{(i)} - \theta^T \mathbf{x}^{(i)})^2 \quad (8)$$

Sum of Squared Error

$$= \arg \min_{\theta} \text{SSE}(\mathbf{X}, \theta) = \arg \min \sum_{i=1}^m (y^{(i)} - \theta^T \mathbf{x}^{(i)})^2 \quad (9)$$

$(\mathbf{X}, y, \theta)$
 $\uparrow \quad \uparrow \quad \uparrow$

$y^{(i)}$

Design matrix

Design matrix (regressor matrix, model matrix, data matrix)

- 훈련 데이터(sample, example)이 m 개
- feature vector $\mathbf{x}$ 의 차원이 n 일 때,
- m 개의 sample을 row vector로 한 design matrix $\mathbf{X}$ 로 표시할 수 있음

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad (10)$$

$$\mathbf{X} = \begin{bmatrix} \text{1st sample } \mathbf{x}^{(1),T} \\ \text{2nd sample } \mathbf{x}^{(2),T} \\ \dots \\ \text{m-th sample } \mathbf{x}^{(m),T} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad (11)$$

model $\hat{y} = \theta^T x$

training dataset $(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$ $(x^{(3)}, y^{(3)})$
.....
 $(x^{(m)}, y^{(m)})$

↓

model
prediction

$$\hat{y}^{(1)} = \theta^T x^{(1)}$$

$$\hat{y}^{(2)} = \theta^T x^{(2)}$$

$$\hat{y}^{(3)} = \theta^T x^{(3)}$$

⋮

$$\hat{y}^{(m)} = \theta^T x^{(m)}$$

$$\begin{aligned} SSE = & (y^{(1)} - \theta^T x^{(1)})^2 \\ & + (y^{(2)} - \theta^T x^{(2)})^2 \\ & + (y^{(3)} - \theta^T x^{(3)})^2 \\ & \vdots \\ & + (y^{(m)} - \theta^T x^{(m)})^2 \end{aligned}$$

~~$m \times n$~~
 design
 matrix
 ↑

$$= \begin{bmatrix} \text{--- } X^{(1)T} \text{ ---} \\ \text{--- } X^{(2)T} \text{ ---} \\ \text{--- } X^{(3)T} \text{ ---} \\ \vdots \\ \text{--- } X^{(m)T} \text{ ---} \end{bmatrix}$$

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$m \times 1$

$$\hat{y} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix} = \begin{bmatrix} \text{--- } X^{(1)T} \text{ ---} \\ \text{--- } X^{(2)T} \text{ ---} \\ \vdots \\ \text{--- } X^{(m)T} \text{ ---} \end{bmatrix} \oplus \begin{bmatrix} | \\ | \\ \vdots \\ | \end{bmatrix} \oplus$$

$m \times n$ $n \times 1$

↑

$$SSE(\theta) = \sum (y^{(i)} - \theta^T x^{(i)})^2 \Leftarrow$$

$$= \left\| \mathbf{y} - \hat{\mathbf{y}} \right\|_2^2$$

$$= \left\| \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} - \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix} \right\|_2^2$$

$$= \left\| \begin{bmatrix} y^{(1)} - \hat{y}^{(1)} \\ \vdots \\ y^{(m)} - \hat{y}^{(m)} \end{bmatrix} \right\|_2^2$$

$$SSE(\theta) = \|y - \hat{y}\|_2^2 = \|y - X\theta\|_2^2$$

그러면, $\hat{y} = \boldsymbol{\theta}^T \mathbf{x}$ 의 m 개의 sample에 대해 다음과 같이 표현할 수 있음

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta} \quad (12)$$

$$\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad (13)$$

$$\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix} = \begin{bmatrix} \text{---} & \mathbf{x}^{(1),T} & \text{---} \\ \text{---} & \mathbf{x}^{(2),T} & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{x}^{(m),T} & \text{---} \end{bmatrix} \boldsymbol{\theta} \quad (14)$$

$$\hat{\Theta} = \arg \min_{\Theta} SSE(\Theta)$$

① Normal equation $\leftarrow$ direct solution
closed form
그 해를 대입해서 직접 구함.

② Gradient descent $\leftarrow$ 미분, 근사적으로 계산

Geometric approach

$$X = \begin{bmatrix} -x^{(1)\top} \\ \vdots \end{bmatrix} = [1 \mid 1 \mid 1 \mid \dots] \quad x \in \mathbb{R}^n$$

Design matrix X 의 각 열을 $\vec{x}_j$ 라고 하면

$$\hat{y} = X\theta = \begin{bmatrix} 1 & x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad (15)$$

↓ 모든 입력의, 몇 번째 값들

$$= \theta_0 1 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n \quad (16)$$

즉, $\hat{y}$ 는 $x_1, x_2, \dots, x_n$ 이 생성하는 hyperplane, i.e., $\text{span}\{1, x_1, \dots, x_n\}$ 상에 존재함

Residual 또는 error의 크기 $\|y - \hat{y}\|$ 를 최소화 하려면? 위의 hyperplane과 error $y - \hat{y}$ 가 서로 수직(orthogonal)해야함

기하학적
생각

Geometric approach

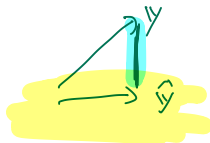
(Residual 또는 error의 크기 $\|y - \hat{y}\|$ 를 최소화 하려면? 위의 hyperplane과 error $y - \hat{y}$ 가 서로 수직(orthogonal)해야함

즉, 모든 column vector에 대해서

X 의 column = $x_1, x_2, \dots, x_n$

$$x_j^T (y - \hat{y}) = 0$$

error



(17)

이 성립해야 함

전체 m개의 sample에 대해서

$$X^T (y - \hat{y}) = 0 \implies X^T X \theta = X^T y$$

$j=1, 2, \dots, n$

(18)

$$\hat{\theta} = (X^T X)^{-1} X^T y \Leftarrow \text{normal equation.}$$

(19)

$$\hat{\theta} = (X^T X)^{-1} X^T y \iff \hat{\theta} = \underset{\theta}{\operatorname{argmin}} \operatorname{SSE}(\theta)$$

$$\min \underbrace{\|y - \hat{y}\|_2^2} \iff \underbrace{(y - \hat{y})} \perp \text{span}\{x_1, \dots, x_n\}$$

Analytic approach

Sum of squared error (SSE)

$$\underline{\text{SSE}} = \sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2 \quad (22)$$

$$= \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \underline{(\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}})} \quad (23)$$

Using $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$

$$\text{SSE}(\boldsymbol{\theta}) = \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\boldsymbol{\theta} + (\mathbf{X}\boldsymbol{\theta})^T (\mathbf{X}\boldsymbol{\theta}) = \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} \quad (24)$$

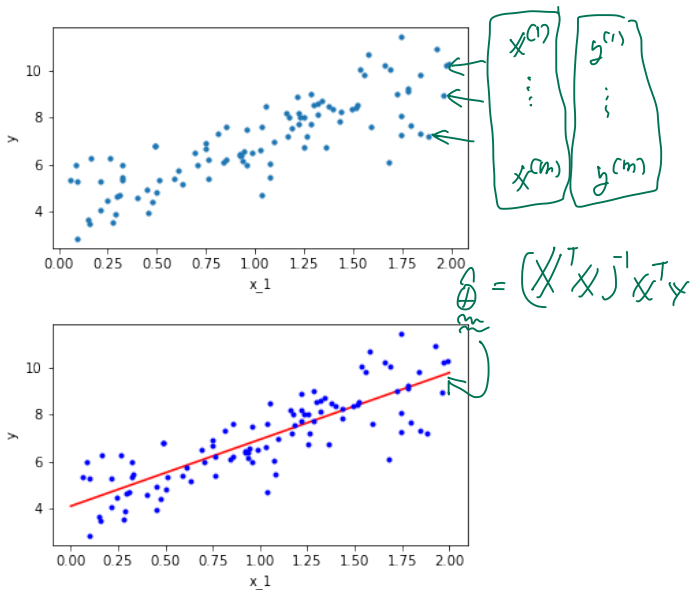
$$\boxed{\frac{\partial \text{SSE}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}} = 2(\mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - \mathbf{X}^T \mathbf{y}) \underset{n}{=} 0 \quad \Rightarrow \quad \underline{\mathbf{X}^T \mathbf{X} \boldsymbol{\theta} = \mathbf{X}^T \mathbf{y}} \quad (25)$$

Hence, we have

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad \text{normal equation} \quad (26)$$

$$\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (27)$$

Closed from solution



Normal equation의 계산

- Normal equation에 의한 예측치 $\hat{\mathbf{y}}$

$$\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \quad (28)$$

- $\mathbf{X} \in \mathcal{R}^{m \times n}$
- $\mathbf{X}^T\mathbf{X} \in \mathcal{R}^{n \times n}$
- $(\mathbf{X}^T\mathbf{X})^{-1}$ 의 계산 복잡도 = $O(n^{2.4}) \sim O(n^3)$
- Feature 수의 약 세제곱으로 계산 시간이 증가

Gradient descent

- ① 알파 값
- ② linear regression.

gradient

$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$

$$\nabla f: \mathbb{R}^n \rightarrow \mathbb{R}^n$$

$$x \in \mathbb{R}^2$$

$$f(x) = x^T x$$

$$f(x_1, x_2) = x_1^2 + x_2^2$$

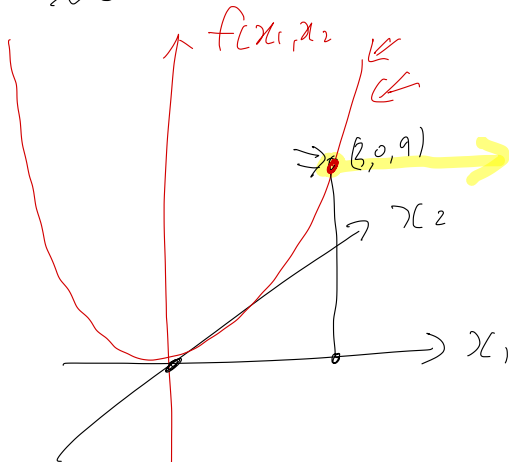
$$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \end{bmatrix}$$

$$\nabla f(x).$$

$\Rightarrow$ 가장 가파르게 증가하는 방향.
 $f(x)$

$$f(x_1, x_2) = x_1^2 + x_2^2$$

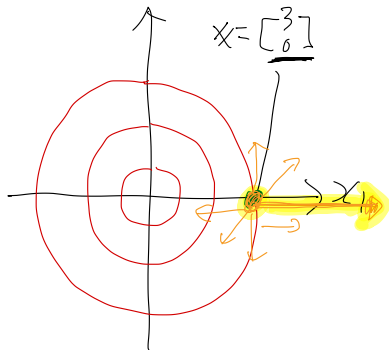
$$\nabla f(x_1, x_2) = [2x_1 \quad 2x_2]^T$$



$$\mathbb{R}^2 \rightarrow \mathbb{R}$$

$$\mathbb{R}^2 \rightarrow \mathbb{R}^2 \quad \nabla f(x) = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$$

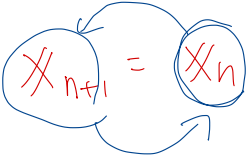
$$x_2 \quad f(x) = 9$$

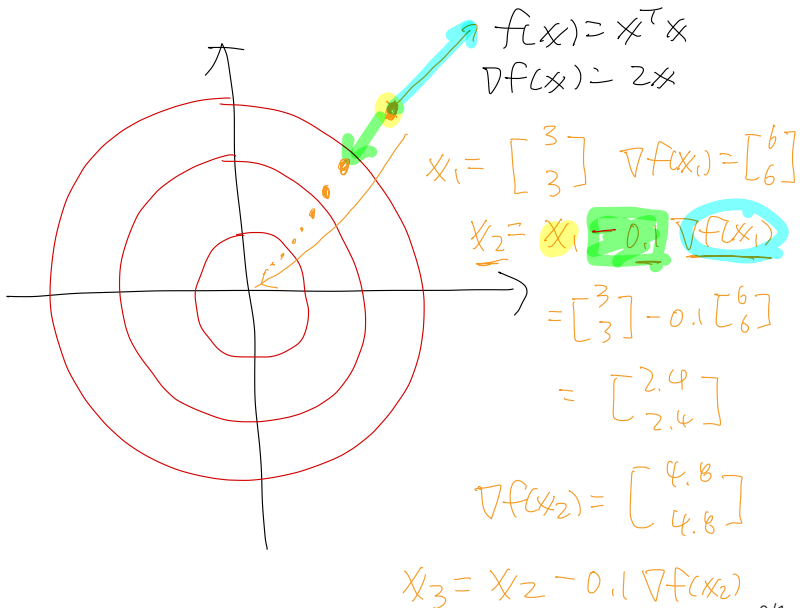


Gradient descent:

$f(x)$ 를 $\min$ 으로 하는 x 를 찾는 문제.

반복적 방법, iterative method.

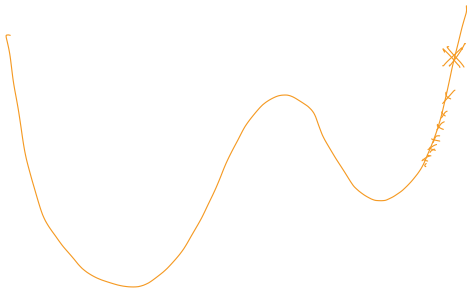

$$x_{n+1} = x_n - \eta \nabla f(x_n)$$



• 미분가능

• global minimum 보장 X

local minimum 여러개 있을 수 있음



Batch gradient descent

Linear regression model $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$ ← 모든 sample에 대한 예측

$J: \mathbb{R}^n \rightarrow \mathbb{R}$

$$J(\boldsymbol{\theta}) = \text{SSE}(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 = \sum (y^{(n)} - \boldsymbol{\theta}^T \mathbf{x}^{(n)})^2 \quad (29)$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 2\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \quad (30)$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial}{\partial \theta_1} J(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} 2 \sum_{i=1}^m x_1^{(i)} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \\ 2 \sum_{i=1}^m x_2^{(i)} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \\ \vdots \\ 2 \sum_{i=1}^m x_n^{(i)} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \end{bmatrix} \quad (31)$$

Gradient descent step

learning rate

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^t) \quad (32)$$

where iteration number t and $\boldsymbol{\theta}$ arbitrary initial value

$$f(x) = x^T x$$

$$\nabla f(x) = 2x$$

$$\nabla_{\theta} J(\theta) = \|y - X\theta\|^2$$

$$= (y - X\theta)^T (y - X\theta)$$

$$= \cancel{y^T y} - 2\theta^T \cancel{x^T y} + \theta^T \cancel{x^T x} \theta \quad \text{with } 2\theta^T x^2 \text{ above the last term}$$

$$= -2x^T y + 2x^T x \theta$$

$$= 2x^T (x\theta - y) \Leftarrow$$

Batch gradient descent

Batch gradient descent에서

$$\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 2\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \quad (33)$$

$$= 2 \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(m)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}^T \mathbf{x}^{(1)} - y^{(1)} \\ \boldsymbol{\theta}^T \mathbf{x}^{(2)} - y^{(2)} \\ \vdots \\ \boldsymbol{\theta}^T \mathbf{x}^{(m)} - y^{(m)} \end{bmatrix} \quad (34)$$

$$= 2 \sum_{i=1}^m \mathbf{x}^{(i)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right) \quad (35)$$

그러므로 이 gradient vector의 j 번째 component는

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = 2 \sum_{i=1}^m x_j^{(i)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right) \quad (36)$$

model

$$\hat{y} = \Theta^T X = f(X; \Theta)$$

sample

$$(x^{(1)}, y^{(1)}) \quad (x^{(2)}, y^{(2)}) \quad \dots \quad (x^{(n)}, y^{(n)})$$

cost

$$J(\Theta) = \sum (\underbrace{\Theta^T}_{\text{red}} \underbrace{x^{(n)}}_{\text{red}} - \underbrace{y^{(n)}}_{\text{red}})^2$$

design
matrix

$$X = \begin{bmatrix} -x^{(1)T} - \\ -x^{(2)T} - \\ \vdots \\ -x^{(n)T} - \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$$J(\Theta) = \|X\Theta - y\|^2$$

Normal

$$\hat{\Theta} = (X^T X)^{-1} X^T y$$
$$= \operatorname{argmin} J(\Theta)$$

g.d.

batch

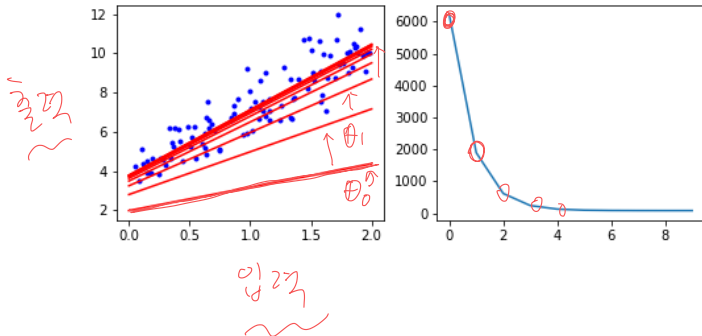
$$\Theta \leftarrow \Theta - \eta \left\{ \frac{\nabla J(\Theta)}{2 X^T (X \Theta - y)} \right\}$$

Stochastic

$$\Theta \leftarrow \Theta - \eta \left\{ 2 x^{(t)T} (\Theta x^{(t)} - y^{(t)}) \right\}$$

Batch gradient descent

$1D \rightarrow 1D$.
+ bias



Batch gradient descent: computational complexity

Batch gradient descent algorithm

- 매 스텝마다 batch 전체에 대한 계산 필요
- 데이터셋이 커지면 속도가 느려짐
- Normal equation: feature 수에 따라 계산 속도가 지수적으로 느려짐
- Gradient descent: feature 수가 늘어도 크게 변하지 않음

Learning rate

- Hyperparameter인 학습률(learning rate) η 가 너무 작은 경우 시간이 오래 걸림 $\rightarrow$ 만가짜
- 너무 큰 경우 최적해를 지나쳐 해를 찾지 못할 수 있음 $\rightarrow$ 뽕그대가기

Learning schedule

- Constant learning rate
 - 보통 0.1, 0.01부터 시작하여 여러 가지 값으로 시험해보며 범위를 좁혀 나감
- Time-based decay

$$\eta = \frac{\eta_0}{(1 + kt)} \quad (37)$$



η_0 : 학습률 초기값, k : hyperparameter, t : iteration

- Step decay = 전체 데이터를 다 쓰면 1 epoch
 - 정해진 epoch마다 학습률을 줄이는 방법
 - 예: 5 epoch마다 반으로, 20 epoch마다 1/10로
 - Epoch: 훈련 데이터셋 전체를 모두 사용할 때 = 한 epoch
- Exponential decay

$$\eta = \eta_0 e^{-kt} \quad (38)$$

η_0 : 학습률 초기값, k : hyperparameter, t : iteration

data 1,000,000 $\Leftarrow$

mini-batch,

batch size 100 $\Rightarrow$ mini-batch 10,000
= 1 epoch

Stochastic gradient descent

For our linear regression model $\hat{y} = \boldsymbol{\theta}^T \mathbf{x}$

$$J(\boldsymbol{\theta}) = \text{SSE}(\boldsymbol{\theta}) = \sum_{i=1}^m \left(\mathbf{y}^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)} \right)^2 = \sum_{i=1}^m J_i(\boldsymbol{\theta}) \quad (39)$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - 2\eta \left((\boldsymbol{\theta}^t)^T \mathbf{x}^{(t)} - y^{(t)} \right) \mathbf{x}^{(t)} \quad (40)$$


- 무작위로 선택한 한 개의 sample에 대해서만 gradient를 계산하여 parameter를 update
- sequential learning or online learning
- 대규모 데이터셋을 처리하는데 유리
- 선택하는 사례의 무작위성으로 움직임이 불규칙
- BGD에 비해 local optimum에서 쉽게 빠져나올 수 있음
- 최적해에 도달하지만 지속적으로 요동

⦿ BGD와 마찬가지로 global optimum이라는 보장이 없음

$$\hat{y} = \underline{\Theta}^T x$$

$$\underline{\Theta} = \underline{\Theta} - 2\eta (\underline{\Theta}^T \underline{x}_1 - y_1) \underline{x}_1$$

$$\hat{y}_1 = \underline{\Theta}^T \underline{x}_1 \quad \text{and} \quad \underline{\Theta}^T \underline{x}_1$$

$$x_1^T x_1 \geq 0$$

$$\underline{\Theta}' = \underline{\Theta} + \underline{x}_1$$

$$\hat{y}_1 = \underline{\Theta}'^T x_1 = (\underline{\Theta} + x_1)^T x_1 \geq \underline{\Theta}^T x_1$$

$$\hat{y} = \Theta^T x$$

$$\Theta^T x_1 = \hat{y}_1 \uparrow$$

$$\begin{aligned} \Theta' &= \Theta - 2\eta (\Theta^T x_1 - y_1) x_1 = \Theta + x_1 \\ &\quad \underbrace{-0.5 x_1}_{-10 x_1} = \Theta + 20 x_1 \end{aligned}$$

Mini-batch gradient descent

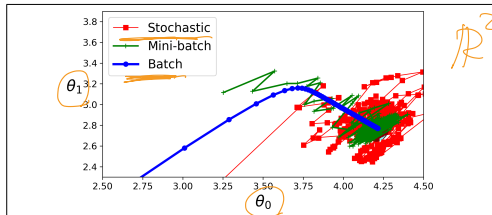


Figure 4-11. Gradient Descent paths in parameter space

- 훈련 데이터셋을 작은 크기의 무작위 부분 집합으로 나누어서 gradient를 구하는 방법
- 예 100,000개의 데이터 = (mini-batch size 100) $\times$ (1,000 mini-batches)
- Batch gradient descent와 stochastic gradient descent(SGD)의 절충
- SGD보다 불규칙한 움직임이 덜함
- SGD보다 local minimum에서 빠져나오기가 상대적으로 더 어려움
- GPU를 통한 매트릭스 연산의 속도를 높일 수 있음

Linear regression comparison

Table 4-1. Comparison of algorithms for Linear Regression

Algorithm	Large m	Out-of-core support	Large n	Hyperparams	Scaling required	Scikit-Learn
Normal Equation	Fast	No	Slow	0	No	n/a
SVD	Fast	No	Slow	0	No	LinearRegression
Batch GD	Slow	No	Fast	2	Yes	SGDRegressor
Stochastic GD	Fast	Yes	Fast	≥ 2	Yes	SGDRegressor
Mini-batch GD	Fast	Yes	Fast	≥ 2	Yes	SGDRegressor



There is almost no difference after training: all these algorithms end up with very similar models and make predictions in exactly the same way.