linear regression

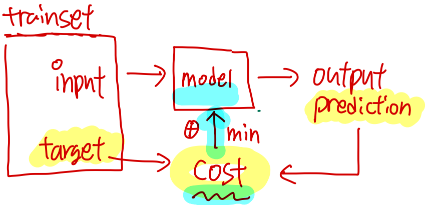
- normal equation ← 선형대수
- gradient descent ← 미분 근사적

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



$\underline{x}$
input
vector

$\rightarrow$

model
linear
regression

$\rightarrow$

y
output
real number
scalar

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

Linear regression

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

$= \theta^T x = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$

Linear model

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \quad (1)$$

In this equation,

- $\hat{y}$ is the predicted value (for true y)
- n is the number of features, input에 들어있는
- 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 $\theta_1, \theta_2, ..., \theta_n$

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

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

$$= \boldsymbol{\theta}^T \mathbf{x} \tag{3}$$

input → model → output

x $\boxed{\theta}$ $\hat{y}$

In this equation,

- $\boxed{\boldsymbol{\theta}}$ is the model's *parameter vector*, containing the bias term θ_0 and the feature weights θ_1 to θ_n
- $\boxed{\mathbf{x}}$ is the instance's *feature vector*, containing x_0 to x_n always equal to 1

  input $\quad$ $x_0 = 1$ bias
- $\boldsymbol{\theta} \cdot \mathbf{x}$ is the dot product of the vectors $\boldsymbol{\theta}$ and $\mathbf{x}$, which is of course equal to $\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$

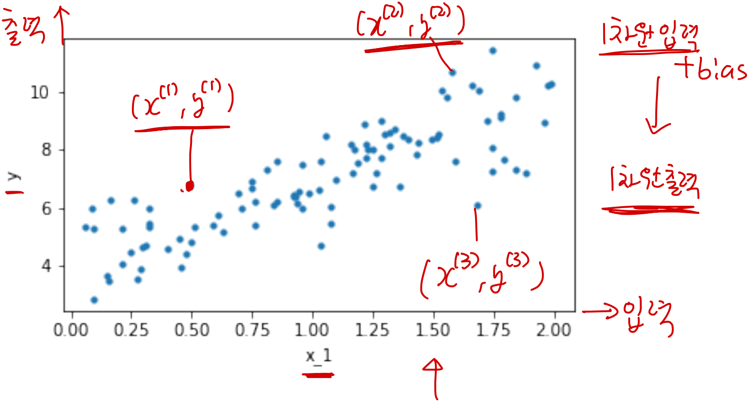


Figure 1: Linear regression: training dataset

Generating training dataset

$$y \approx \theta_0 + \theta_1 x \tag{4}$$

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

Cost function

θ

b

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)}) \ldots$ $(x^{(n)}, y^{(n)})$ $\Rightarrow \theta?$

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

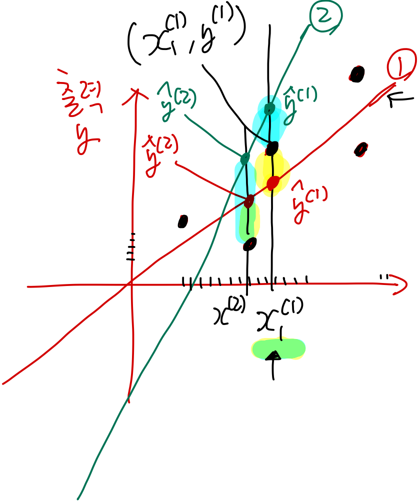
Recall that training a model means setting its parameters so that the model best fits the training set.

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

현재 주어진 데이터 $\leftarrow \theta_1, \theta_2$

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

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



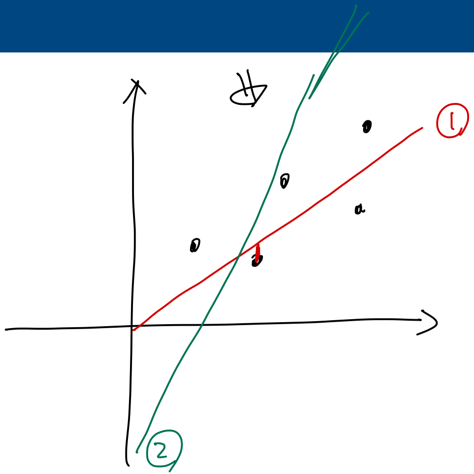
$$\boxed{\hat{y} = \theta_0 + \theta_1 x_1}$$

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

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

참값 예측값

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



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

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

① $\sum 1^2 = 20$

② $\sum 1^2 = 100$

직관적 vs. ① ≥ ② $\Longrightarrow$ $\sum \text{error}^2$ ① ≤ ②

regression problem에 사용되는 measure, error index
cost function.

We need to find the value of $\boldsymbol{\theta}$ that minimizes the RMSE. In practice, it is simpler to minimize the sum of squared error (SSE) than the MSE or the RMSE.

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

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

sum of squared error

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

θ

(X, y, θ)

$\hat{y}^{(i)}$

Design matrix

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

- 훈련 데이터(sample, example)이 m개
- feature vector $\mathbf{x}$의 차원이 n일 때,
- m개의 sample을 row vector로 한 design matrix $\mathbf{X}$로 표시할 수 있음

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

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

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

training dataset $\quad (x^{(1)}, y^{(1)}) \quad (x^{(2)}, y^{(2)}) \quad (x^{(3)}, y^{(3)})$

$\quad \cdots\cdots \quad (x^{(m)}, y^{(m)})$

$\downarrow$

model prediction

$$\left\{\begin{aligned}
\hat{y}^{(1)} &= \theta^T x^{(1)} \\
\hat{y}^{(2)} &= \theta^T x^{(2)} \\
\hat{y}^{(3)} &= \theta^T x^{(3)} \\
&\vdots \\
\hat{y}^{(m)} &= \theta^T x^{(m)}
\end{aligned}\right.$$

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

$$\mathbb{X}_{m\times n} = \begin{bmatrix} — & \mathbb{x}^{(1)T} & — \\ — & \mathbb{x}^{(2),T} & — \\ — & \mathbb{x}^{(3),T} & — \\ & \vdots & \\ — & \mathbb{x}^{(m),T} & — \end{bmatrix}$$

design matrix ↑

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

$$\hat{\mathbb{y}}_{m\times 1} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix} = \underset{m\times n}{\mathbb{X}}\ \underset{n\times 1}{\theta} = \begin{bmatrix} — \mathbb{x}^{(1)T} — \\ — \mathbb{x}^{(2)T} — \\ \vdots \\ — \mathbb{x}^{(m)T} — \end{bmatrix} \Big| \theta \Big|$$

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

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

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

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

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

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

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta} \tag{12}$$

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

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

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

① Normal equation ← direct solution
closed form
그 해를 대수적으로 직접 구함.

② Gradient descent ← 미분, 근사적으로 계산

Geometric approach

$X = \begin{bmatrix} -x^{(1)T}- \\ \vdots \end{bmatrix} = \begin{bmatrix} | & | & | & | & \cdots & | \end{bmatrix}$ $\quad x \in \mathbb{R}^n$

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

모든 입력의, 첫번째 값들

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta} = \begin{bmatrix} | & | & & | \\ \mathbf{1} & \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \tag{15}$$

$$= \theta_0 \mathbf{1} + \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + \cdots + \theta_n \mathbf{x}_n \tag{16}$$

즉, $\hat{\mathbf{y}}$는 $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$이 생성하는 hyperplane, i.e., $\mathrm{span}\{\mathbf{1}, \mathbf{x}_1, \cdots, \mathbf{x}_n\}$ 상에 존재함

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

기하학적 해석

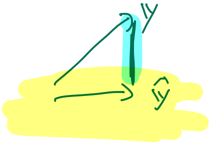
Residual 또는 error의 크기 $\|\mathbf{y} - \hat{\mathbf{y}}\|$를 최소화 하려면? 위의 hyperplane과 error $\mathbf{y} - \hat{\mathbf{y}}$가 서로 수직(orthogonal)해야함

즉, 모든 column vector에 대해서

X의 column들 = $x_1, x_2, \cdots, x_n$

error

$$\mathbf{x}_j^T(\mathbf{y} - \hat{\mathbf{y}}) = 0 \tag{17}$$



이 성립해야 함

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

전체 m개의 sample에 대해서

$y - \hat{y}$

$(X^TX)^{-1}$ $(X^TX)^{-1}$

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = \mathbb{0} \implies \mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y} \tag{18}$$

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{19}$$

⇐ normal equation.

$\hat{\theta} = (X^TX)^{-1}X^Ty \iff \hat{\theta} = \operatorname{argmin} SSE(\theta)$

Geometric approach

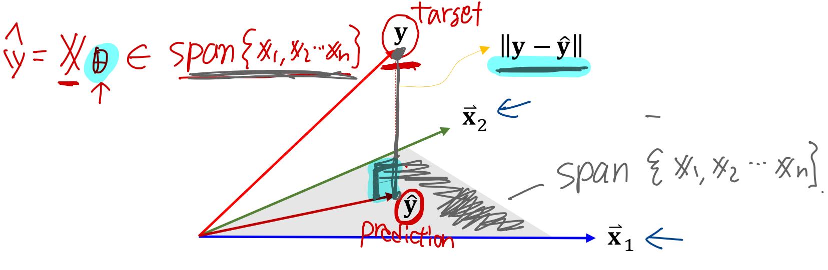


Figure 2: Normal equation: geometric interpretation

Normal equation

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{20}$$

Projection matrix

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\theta}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{21}$$

$$\min \underline{\|\mathbb{y}-\hat{\mathbb{y}}\|_2^2} \quad \underset{\sim\sim\sim}{\Longleftrightarrow} \quad (\mathbb{y}-\hat{\mathbb{y}}) \underset{\sim}{\perp} \operatorname{span}\{\mathbb{x}_1 \cdots \mathbb{x}_n\}$$

Analytic approach

Sum of squared error (SSE)

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

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

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

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

$$\boxed{\frac{\partial \text{SSE}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}} = 2(\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - \mathbf{X}\mathbf{y}) = 0 \implies \mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y} \tag{25}$$

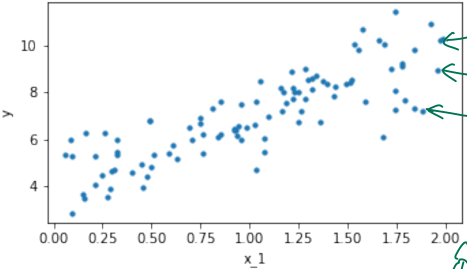
Hence, we have

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{26}$$

normal equation

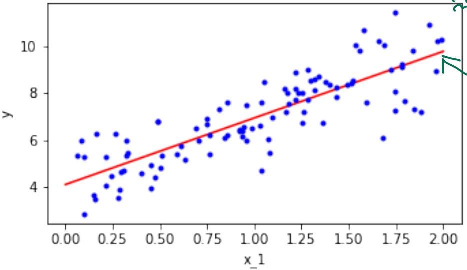
$$\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{27}$$

Closed from solution



$$\begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(m)} \end{bmatrix} \quad \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

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



Computational complexity

Normal equation의 계산

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

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

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

linear regression : linear model + $\overset{\text{min}}{\text{cost}}$

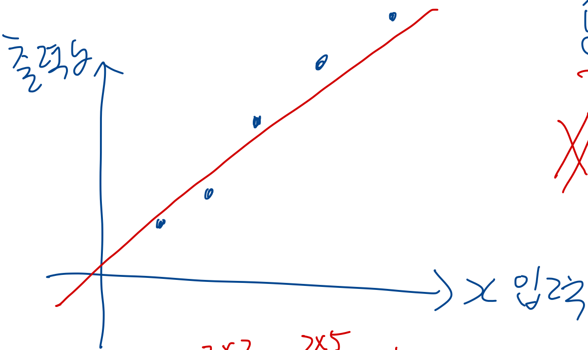
① 해석적, normal equation

② 근사적, Gradient descent → 경사하강법.

Gradient descent

$$D = \{(x_1, y_1), (x_2, y_2), \cdots (x_5, y_5)\}$$

$$= \{(1,1), (2,2), (3,4), (4,5), (5,7)\}$$



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

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \quad 5\times2$$

$$y = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 5 \\ 7 \end{bmatrix} \quad 5\times1$$

$$\hat{\Theta} = \overbrace{(\underbrace{X^T}_{2\times5}\underbrace{X}_{5\times2})^{-1}}^{2\times2}\overbrace{X^T}^{2\times5}\overbrace{y}^{5\times1} \rightarrow 2\times1$$

$$D = \{(1,1,2),\ (2,3,5),\ (3,4,8)\}$$

$$X \in \mathbb{R}^2 \longrightarrow y \in \mathbb{R}$$

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 4 \end{bmatrix}$$

$$\overset{3\times 1}{\hat{y}} = \overset{3\times 3}{X}\,\overset{3\times 1}{\Theta}$$

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \qquad y = \begin{bmatrix} 2 \\ 5 \\ 8 \end{bmatrix}$$

$\boxed{\text{Gradient}}$

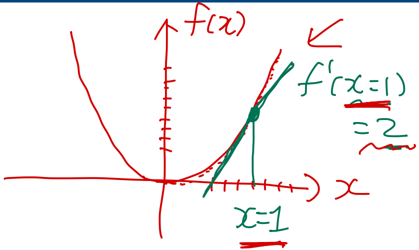
$$J(\theta) = SSE(\theta)$$

$$J : \underset{\text{벡터}\ \theta}{\mathbb{R}^n} \longrightarrow \underset{\text{스칼라}}{\mathbb{R}}$$

$f: \mathbb{R} \to \mathbb{R}$

$f(x) = x^2$

$f' = \dfrac{df}{dx} = 2x =$



$$f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{x} = \mathbf{x}^T \mathbf{x}$$

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

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

$$f : \mathbb{R}^2 \to \mathbb{R}$$

$$\nabla f(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}^2$$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f(\mathbf{x})}{\partial x_1} \\ \\ \dfrac{\partial f(\mathbf{x})}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ \\ 2x_2 \end{bmatrix}$$

$f : \mathbb{R}^3 \to \mathbb{R}$

$f(\mathbf{x}) = f(x_1, x_2, x_3) = 2\mathbf{x}^T\mathbf{x}$

$= 2(x_1^2 + x_2^2 + x_3^2)$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 4x_1 \\ 4x_2 \\ 4x_3 \end{bmatrix} \in \mathbb{R}^3$$

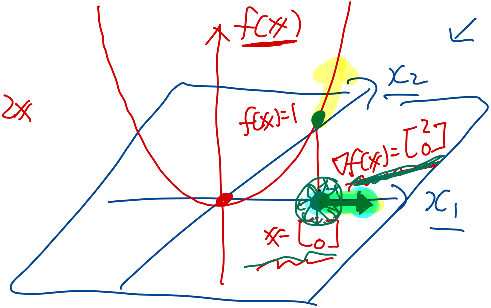
$$\mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \qquad \nabla f(\mathbf{x}) = \begin{bmatrix} 4 \\ 4 \\ 0 \end{bmatrix}$$

?

$f : \mathbb{R}^2 \to \mathbb{R}$

$f(x) = x^T x$

$\nabla f(x) = 2x$



f' :

스칼라→스칼라.

주어진 위치 기울기

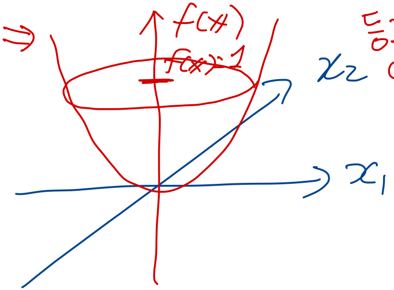
gradient :

벡터 → 벡터

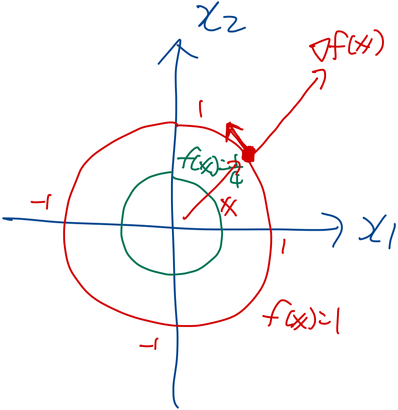
입력 출력

주어진 위치

원래 함수 f(x)가 가장 가파르게 증가하는 방향



등고선 Contour



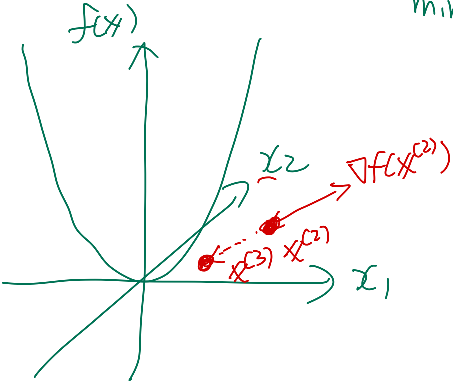
$$f(x) = x^T x = 1$$

$$\{x : x^T x = 1\}$$

gradient를 이용해서 : 목적함수의 최대·최소를 점진적으로 근사하여 찾는 방법.

목적함수가 가장 빠르게 증가하는 방향

$\min f(x)$



$x^{(1)} \in$ 랜덤

$x^{(2)} = x^{(1)} - \nabla f(x^{(1)})$

$x^{(3)} = x^{(2)} - \nabla f(x^{(2)})$

$\vdots$

스칼라.

$$x^{(k+1)} = x^{(k)} - \eta \nabla f(x^{(k)})$$

learning rate

Batch gradient descent

Linear regression model $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$

$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots$

$$\boxed{J(\boldsymbol{\theta})} = \boxed{\mathrm{SSE}(\boldsymbol{\theta})} = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 \tag{29}$$

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

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

Gradient descent step

$\theta_1^{(t+1)} = \theta_1^{(t)} - \eta\left(\quad\right)$

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

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

Batch gradient descent

Batch gradient descent에서

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

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

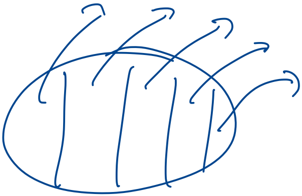
$$= 2 \sum_{i=1}^{m} \mathbf{x}^{(i)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right) \tag{35}$$

그러므로 이 gradient vector의 j번째 component는

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

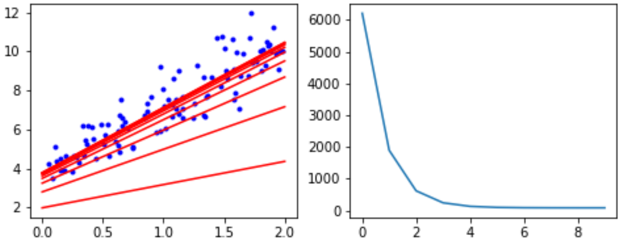
Batch learning : 모든 샘플을 한번에 다

mini-batch. :



Online, stochastic : 한번에 하나씩만

Batch gradient descent



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) η가 너무 작은 경우 시간이 오래 걸림
- 너무 큰 경우 최적해를 지나쳐 해를 찾지 못할 수 있음

최소지점

Learning schedule

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

초반: 큰값 → 후반: 작은값

- Time-based decay

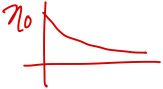
$$\eta = \frac{\eta_0}{(1 + kt)} \tag{37}$$

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

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

epoch: 전체데이터를 한번다쓰면 한 epoch.

- Exponential decay



$$\eta = \eta_0 e^{-kt} \tag{38}$$

η_0 : 학습률 초기값, k : hyperparameter, t : iteration mini-batch.

1	2	3

1 epoch

$$\theta^{(0)} \leftarrow \text{랜덤}$$

$$\theta^{(1)} \leftarrow \theta^{(0)} - \eta \nabla J(\theta^{(0)}) \quad 1.$$

$$\theta^{(2)} \leftarrow \theta^{(1)} - \eta \nabla J(\theta^{(1)}) \quad 2$$

$$\nabla J(\quad) \quad 3$$

2 epoch

1

2

3

Stochastic gradient descent

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

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

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

$-\eta \nabla J(\theta) = -\eta \cancel{\sum_{i=1}^{m}}$

- 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

초기위치 달라짐
→ 결과가 달라짐

Mini-batch gradient descent

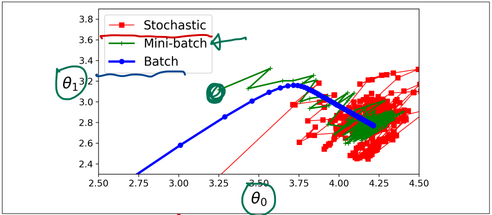


Figure 4-11. Gradient Descent paths in parameter space

- 훈련 데이터셋을 작은 크기의 무작위 부분 집합으로 나누어서 gradient를 구하는 방법
- 예 100,000개의 데이터 = (mini-batch size 100) × (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.