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

linear regression
(cost를 최소화 하는 법)

- normal equation → 선형대수
- gradient descent → 미분

input vector $\longrightarrow$ linear regression $\rightarrow$ output scalar

Linear regression

Linear regression

output : y
label

prediction : $\hat{y}$

Linear model

$$\hat{y} = \underbrace{\theta_0}_{\text{bias}} + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{1}$$

In this equation,

- $\hat{y}$ is the predicted value (for true y)
- n is the number of features
- x_i is the i-th feature value
- θ_j is the j-th model parameter / weight
  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,

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

$$= \boldsymbol{\theta}^T \mathbf{x} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta \cdot x \tag{3}$$

In this equation,

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

Linear regression

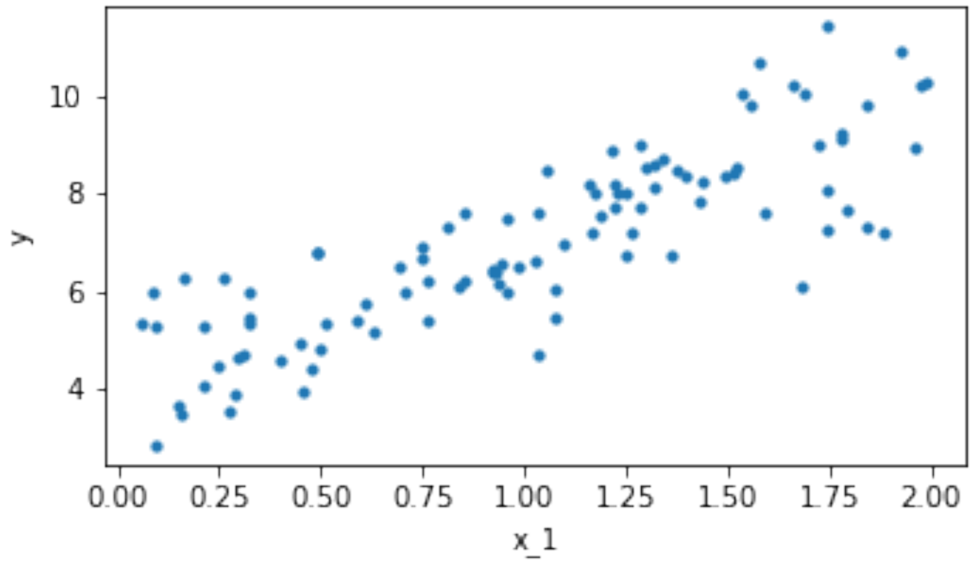


Figure 1: Linear regression: training dataset

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

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.

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

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

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.

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

route mean..

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

Mean of Squared Error

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

Sum of squared Error

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

bias 포함

- 훈련 데이터(sample, example)이 m개
- feature vector $\mathbf{x}$의 차원이 n일 때,

m×n or m×(n+1)

- m개의 sample을 row vector로 한 design matrix $\mathbf{X}$로 표시할 수 있음

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

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

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

$\mathbf{X}$ $m \times (n+1)$ $\quad$ $\boldsymbol{\theta}$ $n+1$

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

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

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

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

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

Normal equation → 해를 대수적으로 직접 구함

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

design vector를 열로 본 것

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

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

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

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

design matrix의 column들

내적

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

이 성립해야 함

전체 m개의 sample에 대해서

$$\mathbf{X}^T(\overbrace{\mathbf{y} - \mathbf{X}\boldsymbol{\theta}}^{= y - \hat{y}}) = 0 \quad \Longrightarrow \quad \mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y} \tag{18}$$

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

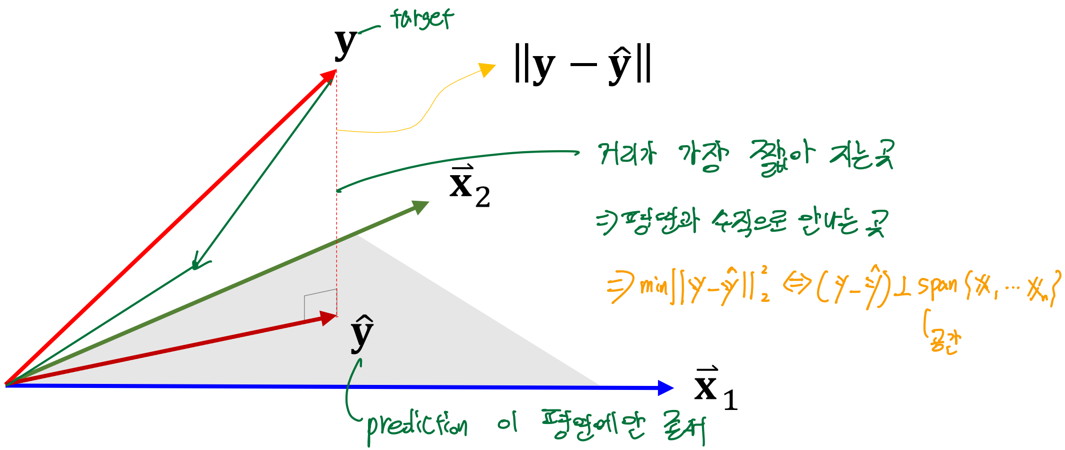


Figure 2: Normal equation: geometric interpretation

Normal equation

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

Projection matrix

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

Analytic approach

Sum of squared error (SSE)

$$\begin{aligned} \mathrm{SSE} &= \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2 && (22) \\ &= \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) && (23) \end{aligned}$$

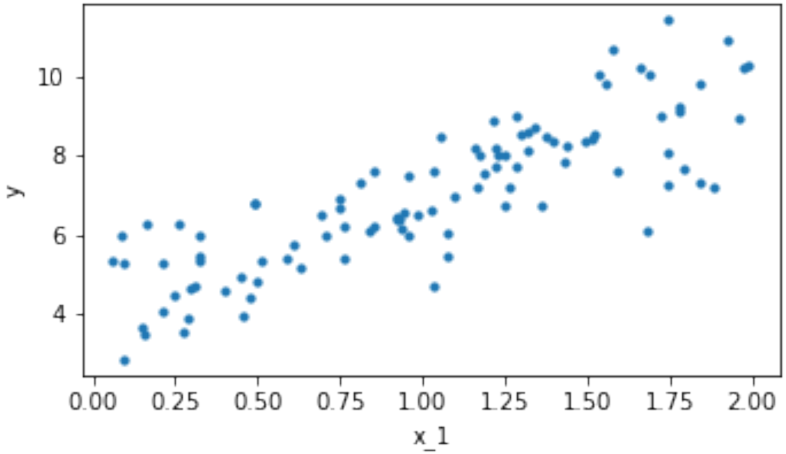
Using $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$

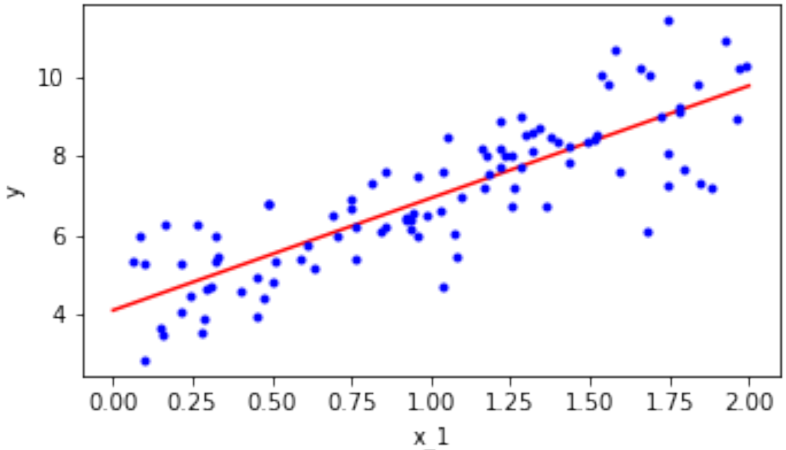
$$\begin{aligned} \mathrm{SSE}(\boldsymbol{\theta}) &= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\boldsymbol{\theta} + (\mathbf{X}\boldsymbol{\theta})^T(\mathbf{X}\boldsymbol{\theta}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} && (24) \\ \frac{\partial \mathrm{SSE}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= 2(\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - \mathbf{X}\mathbf{y}) = 0 \quad \Longrightarrow \quad \mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y} && (25) \end{aligned}$$

Hence, we have

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} && (26) \\ \hat{\mathbf{y}} &= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} && (27) \end{aligned}$$

Closed from solution





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

Gradient descent

Batch gradient descent

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

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

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

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

Gradient descent step

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

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

Batch gradient descent에서

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

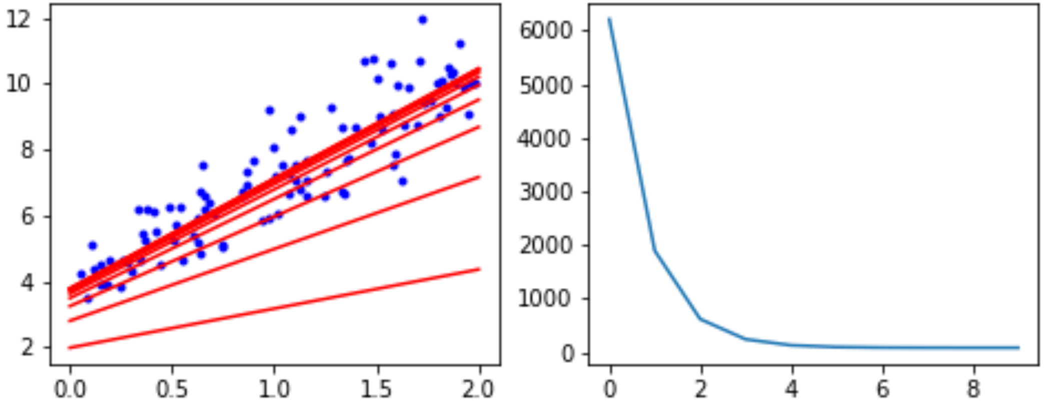
$$= 2 \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}2^{(m)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}^T\mathbf{x}^{(1)} - y^{(1)} \\ \boldsymbol{\theta}^T\mathbf{x}^{(2)} - y^{(2)} \\ \vdots \\ \boldsymbol{\theta}^T\mathbf{x}^{(m)} - y^{(m)} \end{bmatrix} \tag{34}$$

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

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 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

Learning schedule

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

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

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
- Exponential decay

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

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

Stochastic gradient descent

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

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

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

- 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

Mini-batch gradient descent

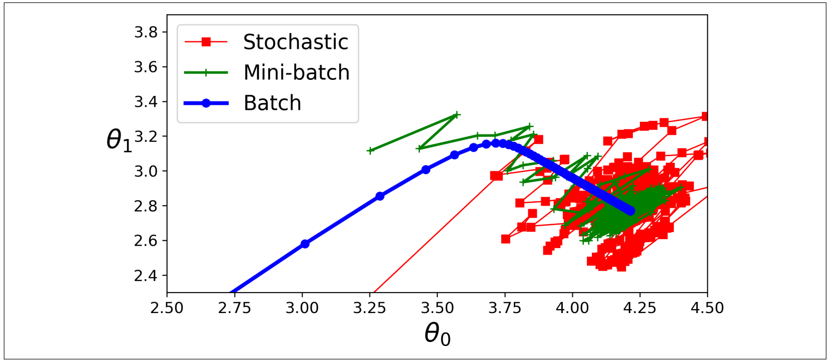


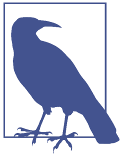
Figure 4-11. Gradient Descent paths in parameter space

- 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.