MYONGJI UNIVERSITY

모델 훈련
- Part 1 -

Outline

- 지금까지는 머신러닝 모델과 훈련 알고리즘을 블랙박스로 취급했음
- 하지만 훈련 세부 진행 과정 및 원리를 이해하고 있으면 적절한 모델, 올바른 훈련 알고리즘, 작업에 적합한 하이퍼파라미터를 찾는데 도움이 됨

- **선형 회귀 모델을 훈련시키는 2가지 방법**
  - **정해진 공식을 사용**하여 훈련셋에 가장 잘 맞는 모델 파라미터(즉, 훈련셋에 대해 비용 함수를 최소화 하는 (최적의) 모델 파라미터)를 계산해내는 방법
  - **경사 하강법(gradient descent)**이라 불리는 iterative optimization 방식을 사용하여 모델 파라미터를 조금씩 조정해가면서 최적의 모델 파라미터를 찾아가는 방법

- **비선형 데이터셋을 학습**하는데 활용 가능한 (조금 더 복잡한 모델인) **다항 회귀**
  - 선형 회귀보다 모델 파라미터가 많아서 훈련 데이터에 과대적합되기 더 쉬움
- **학습 곡선(learning curve)**을 사용하여 모델 과대적합 여부를 감지하는 방법
- **과대적합을 완화하기 위한 규제 기법**
- 분류(classification) 작업에 사용 가능한 회귀 모델인 **로지스틱 회귀**와 **소프트맥스 회귀**

MYONGJI UNIVERSITY

선형 회귀(Linear Regression)

선형 회귀 모델 훈련 과정의 내부 작동 원리를 살펴보자.

MYONGJI UNIVERSITY

선형 회귀 모델

- 앞서 봤던 예: 어떤 국가의 1인당 GDP를 입력으로 삶의 만족도를 예측하는 간단한 선형 회귀 모델
  - $life\_sat = \theta_0 + \theta_1 \times GDP\_per\_capita$
  - ➔ $\hat{y}^{(i)} = \theta_0 + \theta_1 x_1^{(i)}$ i번째 샘플에 대한 입력 특성
    예측값을 보는 것이라 ^으로 구현함.

선형 회귀 모델

- n개의 입력 특성을 사용하여 주어진 샘플의 타깃/레이블을 예측하는 선형 회귀 모델

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

  - **입력 특성의 가중치 합(weighted sum)**과 **편향(bias term)**이라는 상수를 더해 예측을 만듦
  - $\hat{y}$: 주어진 샘플에 대한 (결과) 예측 값
  - n : 각 샘플의 입력 특성 수
  - x_i $(1 \le i \le n)$: 샘플의 i번째 특성 값
  - θ_j $(0 \le j \le n)$: j번째 모델 파라미터
    - $\theta_1 \cdots \theta_n$: 대응되는 입력 특성 x_i에 곱해지는 가중치(weight)
    - θ_0 : bias term(편향)

선형 회귀 모델: 벡터 형태 표현

- 벡터 형태로 표현하면 $\hat{y} = h_\theta(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$
  - $h_\theta()$: 선형 회귀 모델의 예측 함수
  - $\boldsymbol{\theta}$: $\theta_0 \sim \theta_n$으로 이루어진 모델 파라미터 벡터(**(n+1)x1의 column 벡터**)
  - $\mathbf{x}$: 주어진 샘플의 입력 특성 벡터(x_0부터 x_n으로 구성되며, **x_0=1이 추가**됨. **(n+1)x1의 column 벡터**)
  - $\boldsymbol{\theta} \cdot \mathbf{x} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n = \boldsymbol{\theta}^T \boldsymbol{x}$ ($\boldsymbol{\theta}^T$와 $\boldsymbol{x}$의 행렬 곱셈)
    - $\boldsymbol{\theta}^T$: $\boldsymbol{\theta}$의 transpose ➔ 1x(n+1)의 row 벡터가 됨
    - $\mathbf{x}$는 (n+1)x1의 column 벡터

세타와 형태를 맞추기 위해 1로 추가해서 넣었음.

세타.shape = ((n+1), 1)이므로 세타^T.shape = (1,(n+1))
x.shape = ((n+1), 1)

선형 회귀 모델: 벡터 형태 표현

- 넘파이 2차원 배열 표현
  - **m**: 훈련 샘플들의 총 개수, **n**: 샘플 당 특성 수(타깃 특성은 제외)

	notation	shape
타깃/레이블, 예측값	$\boldsymbol{y}$, $\hat{\boldsymbol{y}}$	(m, 1)
모델 파라미터 (벡터)	$\boldsymbol{\theta}$	(n+1, 1)
전체 훈련셋 (행렬)	$\boldsymbol{X}$	(m, n+1)
i번째 훈련 샘플	$\boldsymbol{x}^{(i)}$	(n+1, 1)

세타0부터 세타n까지

전체는 각 훈련 샘플 총수, 범위 특성 n+1까지

x0=1로 넣었기 때문에 x0~xn까지

  - 참고: $x_k^{(i)}\ (0 \le k \le n)$

선형 회귀 모델 훈련

- 모델을 훈련시킨다는 것은 모델이 훈련셋에 가장 잘 맞도록 모델 파라미터를 설정하는 것
- 이를 위해 모델이 훈련 데이터에 얼마나 잘 맞는지를 측정할 수 있는 지표가 필요
- 회귀에 가장 널리 사용되는 성능 측정 지표는 평균 제곱근 오차임(RMSE)
- 따라서 **선형 회귀 모델 훈련 과정의 최종 목표는 훈련 데이터에 대한 RMSE를 최소화 하는 모델 파라미터 벡터 θ를 찾는 것임**

예측오차에 관한

- 실제로는 RMSE 보다 MSE를 최소화 하는 모델 파라미터 θ를 찾는 것이 계산 과정이 더 간단하기 때문에 훈련 과정에서는 MSE가 많이 사용됨 루트는 계산이 어려우니까 뭐
- MSE를 최소화 하는 동일한 모델 파라미터를 사용하면 RMSE 또한 최소라는 것이 보장됨

선형 회귀 모델의 MSE 비용 함수

- 선형 회귀 모델의 훈련셋 전체에 대한 MSE 비용 함수

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

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

  - m : 훈련셋에 샘플들의 총 개수
  - $\boldsymbol{\theta}^{T}$: 모델 파라미터 벡터 $\boldsymbol{\theta}$의 transpose ➔ 1x(n+1)의 row 벡터
  - $\boldsymbol{x}^{(i)}$: 훈련셋 내 i번째 샘플의 특성 벡터. (n+1)x1의 column 벡터
  - $\boldsymbol{y}^{(i)}$: i번째 훈련 샘플의 타깃/레이블

  - m개 훈련 샘플들에 대한 모델의 예측이 타깃(정답)과 평균적으로 얼마나 차이가 나는지를 보여주는 지표임

선형 회귀 모델을 훈련시키는 2가지 방식

- 선형 회귀 모델을 훈련시키는 2가지 방법
  - **정해진 공식을 사용**하여 훈련셋에 가장 잘 맞는 모델 파라미터(즉, 훈련셋에 대해 비용 함수를 최소화 하는 (최적의) 모델 파라미터)를 계산해내는 방법
  - **경사 하강법(gradient descent)**이라 불리는 iterative optimization 방식을 사용하여 모델 파라미터를 조금씩 조정해가면서 최적의 모델 파라미터를 찾아가는 방법

- **방식 1: 정규방정식 또는 특이값 분해(SVD) 활용**
  - 드물지만 수학적으로 비용함수를 최소화하는 $\boldsymbol{\theta}$ 값을 직접 계산할 수 있는 경우 활용
  - **계산복잡도가 $O(n^2)$ 이상인 행렬 연산을 수행**해야 함.
  - 따라서 **특성 수(n)이 많은 경우 메모리 관리 및 시간복잡도 문제 때문에 비효율적임**
- **방식 2: 경사하강법**
  - 특성 수가 매우 많거나 훈련 샘플이 너무 많아서 메모리에 한꺼번에 저장할 수 없을 때 적합
  - 선형 회귀 모델 훈련 시 일반적으로 적용되는 기법

방법1: 정규 방정식 사용

Colab 확인

- 정규 방정식을 이용하여 비용함수를 최소화 하는 $\boldsymbol{\theta}$ 를 다음과 같이 계산할 수 있음:
  입력X, 타겟 Y

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^{\intercal}\mathbf{X})^{-1}\,\mathbf{X}^{\intercal}\,\mathbf{y}$$

  - 문제점: **행렬 곱셈과 역행렬 계산은 계산 복잡도가 $O(n^{2.4})$ 이상**이고 **항상 역행렬 계산이 가능한 것도 아님**. 정규방정식 사용하면 계산 복잡도가 높다. + 역행렬 계산해야되는데 계산 불가할수도 있다.

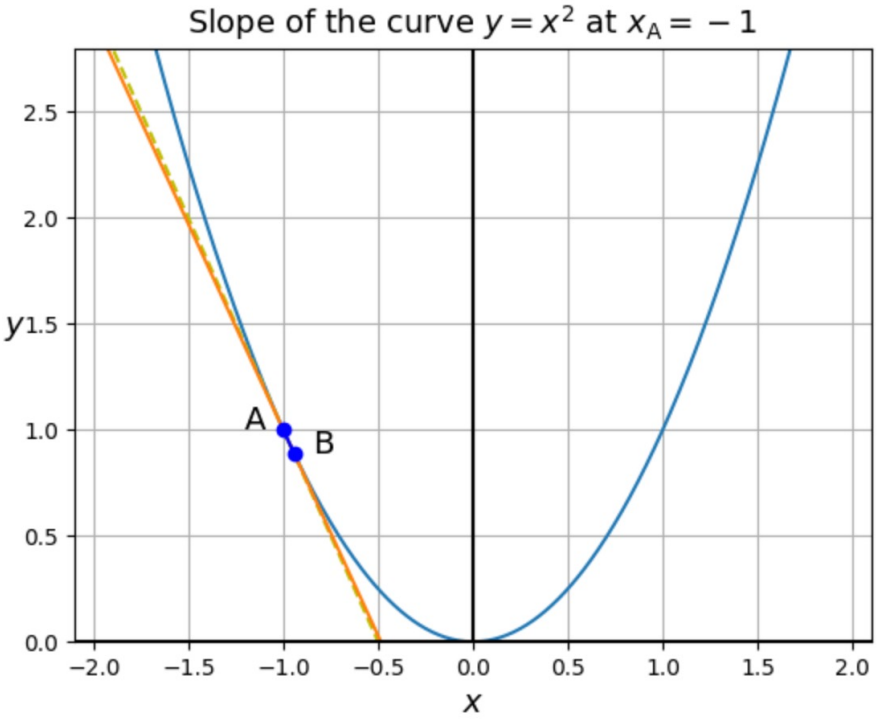
- SVD(특이값 분해) 활용 유사 역행렬을 구해서 정규방정식보단 계산이 간단해짐.
  - 특이값 분해를 활용하여 얻어지는 유사 역행렬(e.g., 무어-펜로즈(Moore-Penrose) 역행렬) $\boldsymbol{X}^{+}$ 계산이 보다 효율적임. 계산 복잡도는 $O(n^2)$.

$$\hat{\boldsymbol{\theta}} = \mathbf{X}^{+}\mathbf{y}$$

경사 하강법(Gradient Descent)

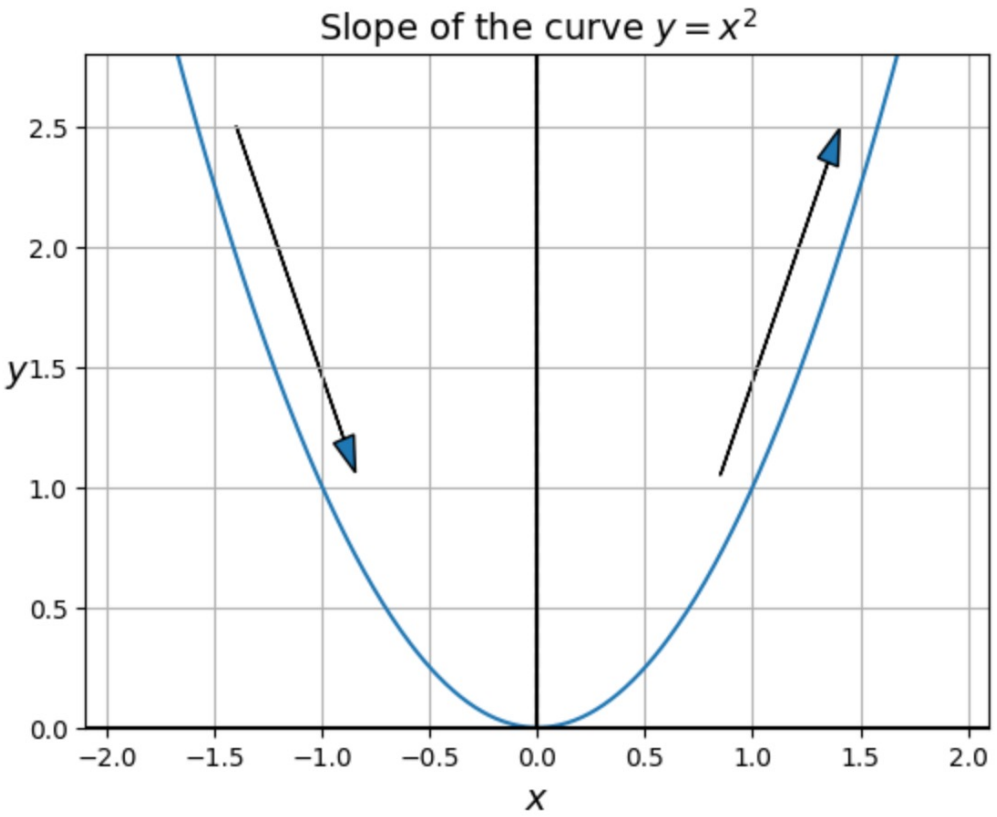
Review: Differential Calculus

- f(x)=x^2을 미분한 결과 도함수($f'(x) = 2x$)를 이용하여 y=x^2 곡선 상의 각 포인트에서 접선의 기울기를 계산할 수 있다.



$$
\begin{aligned}
f'(x_A) &= \lim_{x_B \to x_A} \frac{f(x_B) - f(x_A)}{x_B - x_A} \\
&= \lim_{x_B \to x_A} \frac{{x_B}^2 - {x_A}^2}{x_B - x_A} \\
&= \lim_{x_B \to x_A} \frac{(x_B - x_A)(x_B + x_A)}{x_B - x_A} \\
&= \lim_{x_B \to x_A} (x_B + x_A) \\
&= \lim_{x_B \to x_A} x_B + \lim_{x_B \to x_A} x_A \\
&= x_A + \lim_{x_B \to x_A} x_A \\
&= x_A + x_A \\
&= 2x_A
\end{aligned}
$$

Review: Differential Calculus



- When x < 0, 접선의 기울기는 음수
- When x > 0, 접선의 기울기는 양수
- 기울기가 0 일 때가 $y=x^2$의 minimum에 해당함 그래프의 모양마다 다를꺼지만 보통 기울기가 0일때가 최소치

경사 하강법의 기본 아이디어

- 훈련셋을 이용한 모델 훈련 과정 중에 모델의 비용 함수(MSE)의 크기가 줄어드는 방향으로 모델 파라미터를 조금씩 반복적으로 조정해가는 기법 루트는 너무 복잡해지므로 mse로 계산
  - **훈련 시 비용함수(MSE)에 입력으로 주어지는 훈련 데이터는 고정값(constant)으로 간주**되며, 반면에 **비용함수의 모델 파라미터가 변수**에 해당함

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

- **최종 목표: 비용 함수의 minimum에 도달할 수 있는 모델 파라미터 값 조합 찾기**

- Analogy: 짙은 안개 때문에 산속에서 길을 잃었다고 생각해보세요. 오직 발 밑 지면의 기울기만 느낄 수 있음. 가능한한 빨리 골짜기 bottom으로 내려가는 좋은 방법은 가장 가파른 길을 따라 아래로 내려가는 것임. 이것이 경사 하강법의 원리임!

  매순간마다 가장 최솟값을 찾아 내려가는 것

경사 하강법 관련 주요 개념

- 최적 학습 모델
  - 주어진 훈련 데이터에 대해 비용함수(cost function, e.g., MSE)를 최소화 하는 모델 파라미터로 설정된 모델

- 모델 파라미터
  - 주어진 인스턴스에 대한 예측값을 리턴하는 함수로 구현되는 머신러닝 모델에 포함된 파라미터
  - 예제: 다음과 같은 선형 회귀 모델에 사용되는 편향과 가중치 파라미터

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \boldsymbol{\theta} \cdot \mathbf{x}$$

경사 하강법 관련 주요 개념

- 비용함수
  - 머신러닝 모델의 예측값이 실제 타깃값과 얼마나 차이가 나는지를 계산해주는 함수
  - 예제: 선형 회귀 모델의 평균 제곱 오차(MSE, mean squared error)

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

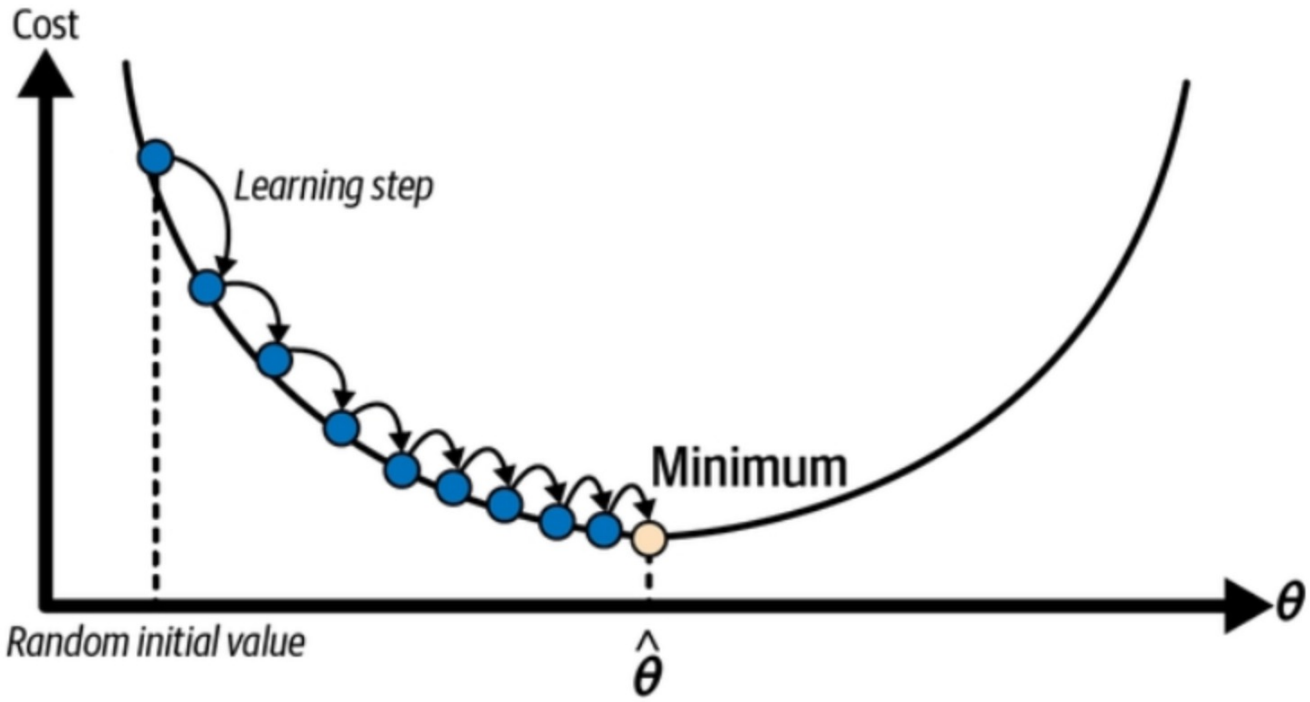
    - **훈련 시 비용함수에 입력으로 주어지는 훈련 데이터는 고정값(constant)으로 간주**되며, 반면에 **비용함수의 모델 파라미터가 변수**에 해당함

      즉 세타값이 결과를 좌지우지함.

  - 선형 회귀 모델의 성능 평가를 위해서는 RMSE가 일반적으로 많이 사용됨
  - 하지만 **RMSE 보다 MSE를 최소화 하는 모델 파라미터 θ를 찾는 것이 계산 과정이 더 간단**하기 때문에 **훈련 과정에서는 비용함수로 MSE가 많이 사용됨. 모델 성능 평가에는 RMSE가 사용됨.**
  - **MSE를 최소화 하는 동일한 모델 파라미터를 사용하면 RMSE 또한 최소화 된다는 것이 보장됨**

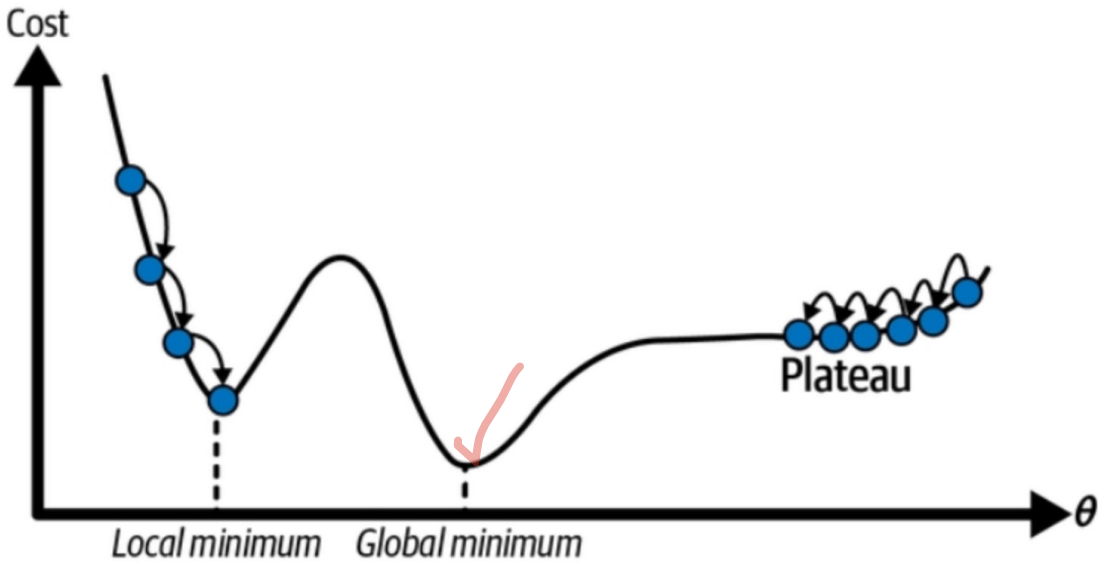
경사 하강법 관련 주요 개념

- 전역 최소값
  - 모델 파라미터 조정을 통해 도달할 수 있는 비용함수의 가능한 최소값



경사 하강법 관련 주요 개념

- 전역 최소값 vs 지역 최소값
  글로벌 미니멈 로컬 미니멈



너무 일찍 마치게 되면
글로벌 미니멈까지 가지 못하고 로컬미니멈에서 멈출 수

그레디언트 벡터(Gradient Vector)

- **비용함수(MSE)를 $\theta_0 \sim \theta_n$ 각 모델 파라미터로 편미분해서 구해진 (n+1)개 편도함수(partial derivative)들로 이루어진 벡터**이며, **다음 식을 통해 주어진 (고정값) 훈련 데이터(X, y)와 특정 모델 파라미터 값 θ 일 때 그레디언트 벡터**를 계산할 수 있다.

$$\nabla_{\boldsymbol{\theta}} \operatorname{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \operatorname{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \operatorname{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \operatorname{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \mathbf{X}^{\intercal} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

m : 훈련데이터의 샘플 수
X^T :훈련용 샘플들의 입력특성값
X: 훈련과정 고정값
y : 훈련과정 타겟특성 고정값

그레디언트 벡터(Gradient Vector)

- 그레디언트 벡터는 (n+1)-차원 공간 상의 한 point를 나타내며, **방향과 크기에 대한 정보를 제공**
  - 그레디언트 벡터는 **현재 모델 파라미터 θ 를 어느 방향으로 얼만큼 움직였을 때 비용함수가 가장 많이 증가하게 되는지에 대한 정보**를 담고 있다.

$$\nabla_{\boldsymbol{\theta}} \operatorname{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \operatorname{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \operatorname{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \operatorname{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \mathbf{X}^{\intercal} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

- 그레디언트 벡터가 가리키는 방향의 **반대 방향**으로 모델 파라미터 벡터 $\boldsymbol{\theta}$를 조정함으로써 비용함수의 최소값에 접근해갈 수 있음
  - η(eta)는 학습률(learning rate)

  즉 경사하강법은 이 식을 여러번 반복하는 것!

Equation 4-7. Gradient descent step

$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \operatorname{MSE}(\boldsymbol{\theta})$$

경사 하강법

- 학습률(learning rate, η)
  - 매 gradient descent step 마다 모델 파라미터 $\boldsymbol{\theta}$ 조정 폭을 결정

결국 경사하강법은 세타에 달려있고 이 세타는 eta에 또 달려있다.

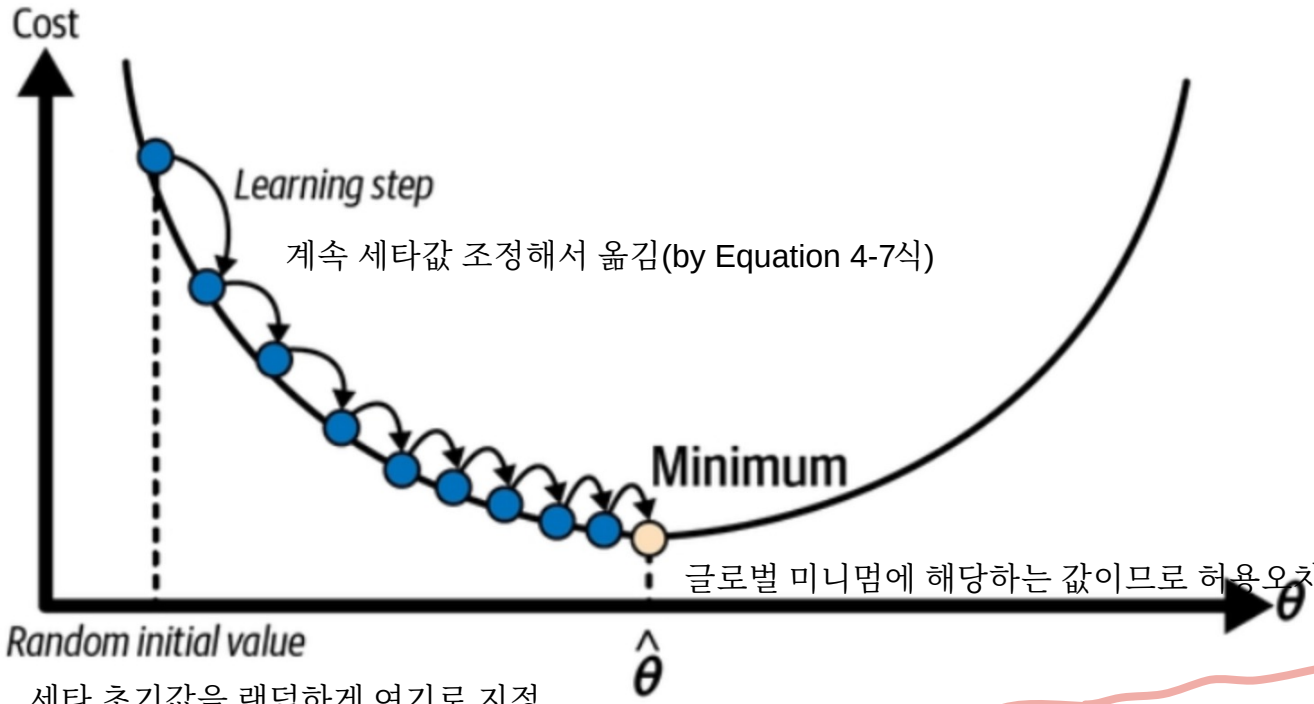
- E.g., 경사 하강법에 의한 선형 회귀 모델 파라미터 조정 과정
  - 모델 파라미터 벡터 $\boldsymbol{\theta}$를 랜덤하게 초기화 하여 훈련 시작
  - 현재 설정된 모델 파라미터 벡터 $\boldsymbol{\theta}$와 (batch size로 지정된 수의) 훈련 샘플들을 이용하여 그레디언트 벡터를 계산
  - 그레디언트 벡터의 크기(norm)이 허용오차(tolerance) 보다 작은지 확인
    - If yes, 비용함수의 최소값에 근접했음을 의미하며 최적의 모델 파라미터를 찾은 것이므로 훈련 과정 멈춤
    - If not,

Equation 4-7. Gradient descent step

$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \text{MSE}(\boldsymbol{\theta})$$

아직 최솟값이 안됐으므로 계속 세타값을 조정해야됨.

경사 하강법



계속 세타값 조정해서 옮김(by Equation 4-7식)

글로벌 미니멈에 해당하는 값이므로 허용오차보다 작아지므로 이제 종료함.

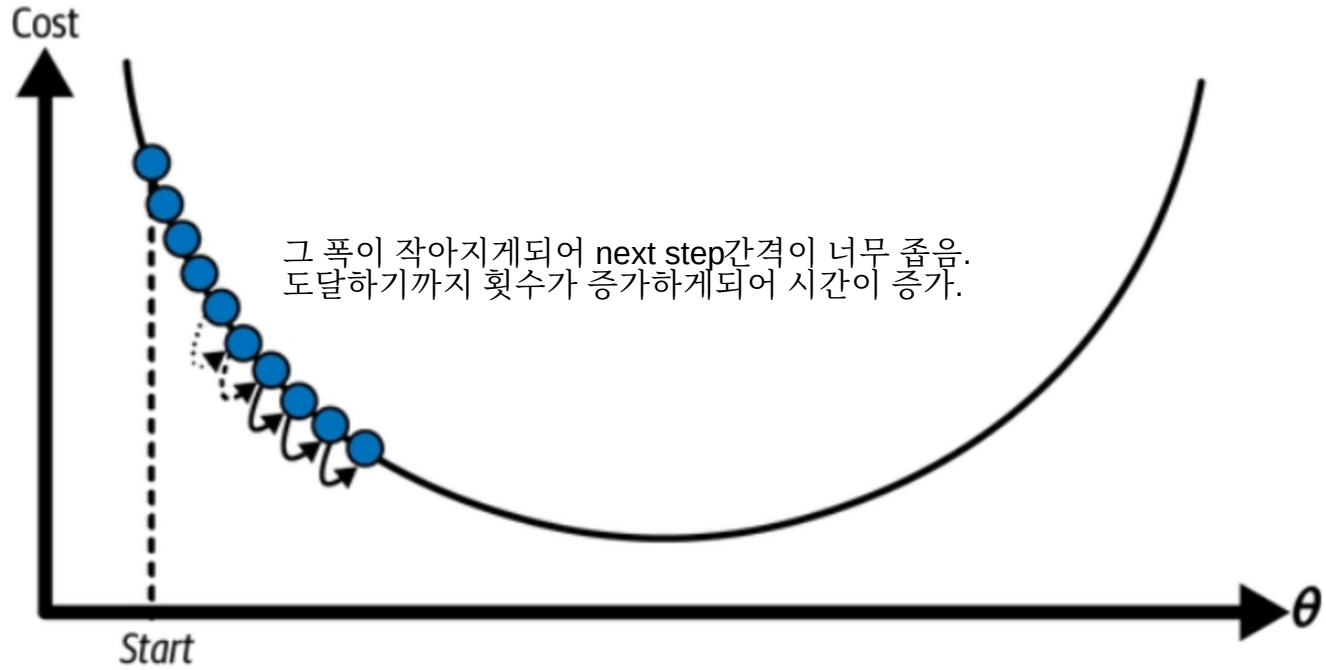
세타 초기값을 랜덤하게 여기로 지정

Equation 4-7. Gradient descent step

$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \text{MSE}(\boldsymbol{\theta})$$

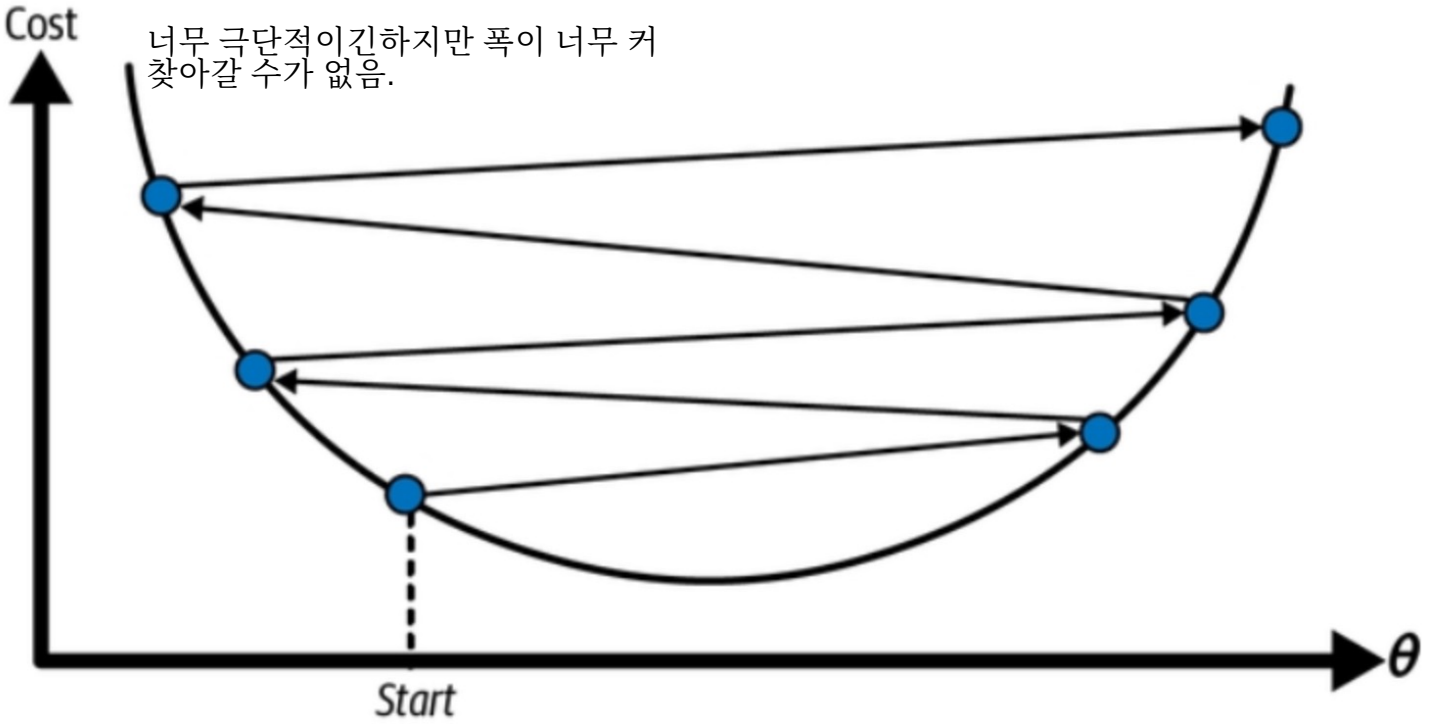
경사 하강법

- 학습률(η, eta)이 너무 작으면 비용함수의 전역 최소값에 도달하기까지 시간이 오래 걸리게 됨



경사 하강법

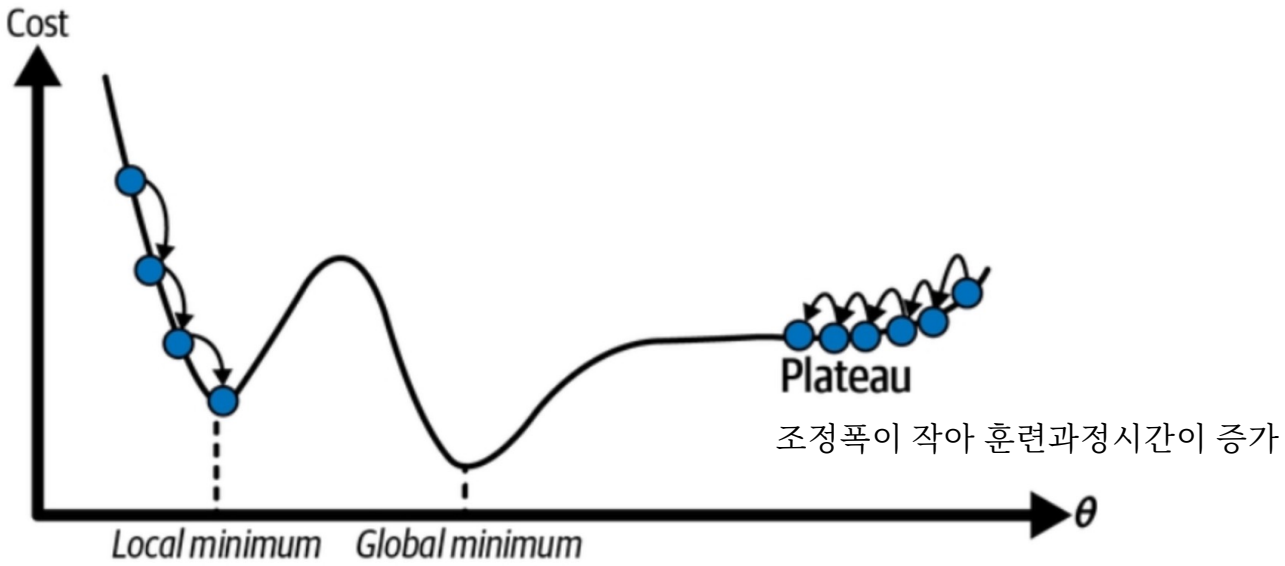
- 반대로 학습률이 너무 크면 비용함수의 전역 최소값에 도달하지 못 할 수 있음



경사 하강법

- 모델 비용함수 곡선이 아래와 같은 경우 경사 하강법 알고리즘 실행 결과로 전역 최소값이 아닌 지역 최소값으로 귀결될 수 있음

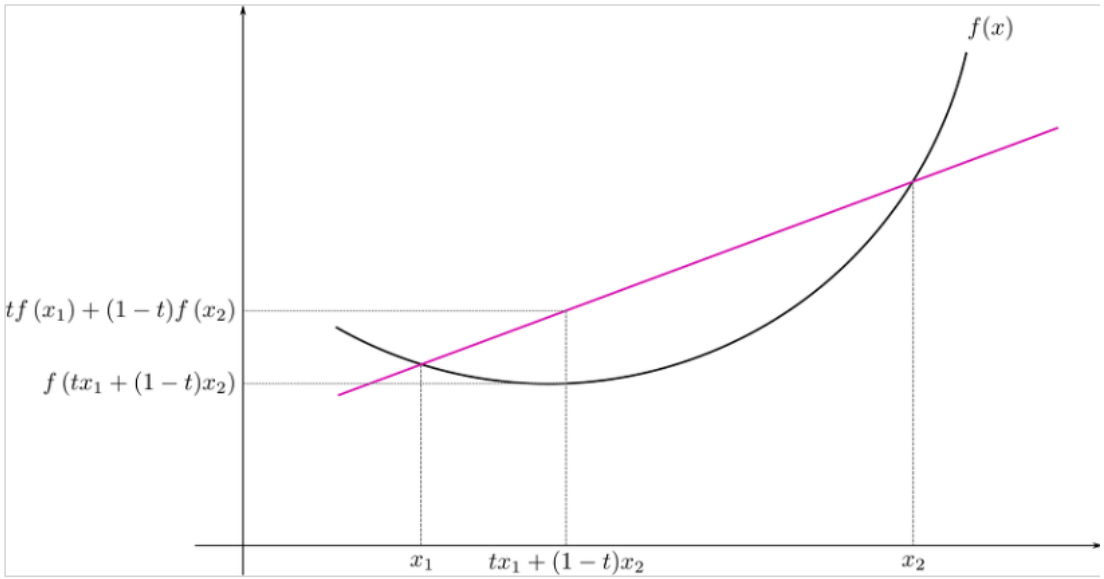
그래프의 모양,초기랜덤 위치에 따라 다름.(이차함수꼴인 경우 전역 최솟값으로 찾아감)



조정폭이 작아 훈련과정시간이 증가

- 선형 회귀 모델의 비용함수는 convex function에 해당함
  - 오직 전역 최소값만 존재(**지역 최소값 없음**)
  - 학습률이 너무 크지 않다면 **결국엔 전역 최소값에 수렴 가능**함

    이차함수꼴은 극소구간이 최솟값이므로



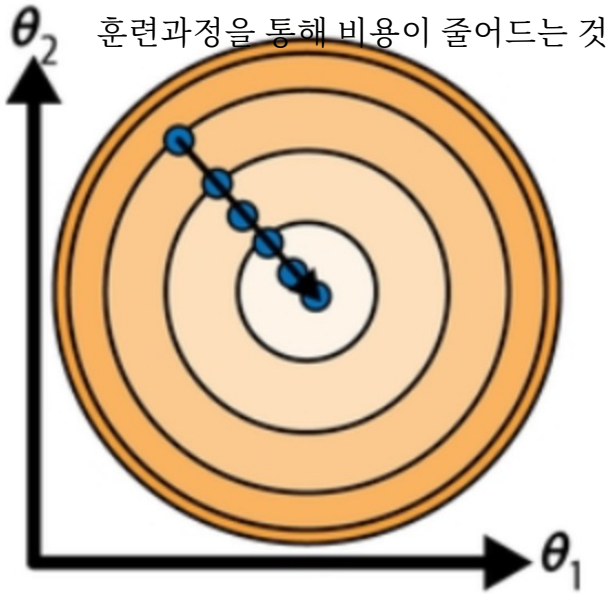
이미지 출처: https://en.wikipedia.org/wiki/Convex_function

위 그림처럼 구간 [x_1, x_2] 에 대해 함수값 f(x_1), f(x_2) 를 연결한 직선(보라색)보다 구간 안의 함수 값이 더 작은 함수일 때, 구간 [x_1, x_2]에 대하여 f(x) 는 convex function 입니다. 이를 더 일반화하여 수식으로 표현하면 다음과 같습니다.

$$\forall x_1,\ x_2 \in X,\ \forall t \in [0,1]:\ f(tx_1+(1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

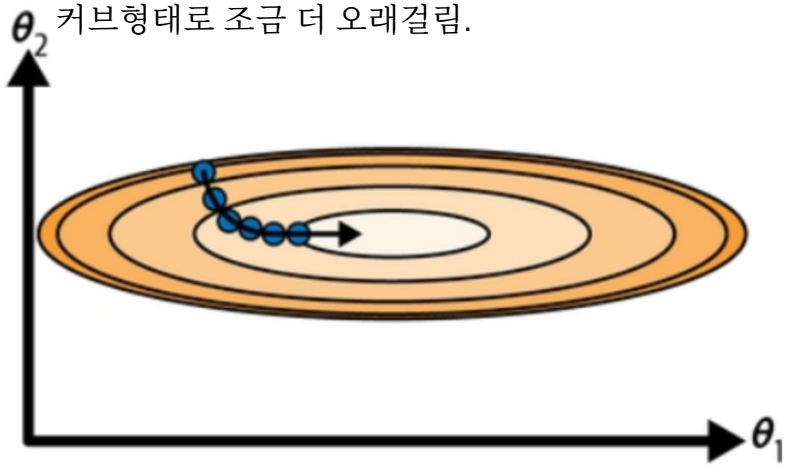
특성 스케일링의 중요성

- 데이터셋에 포함된 특성들의 스케일을 통일시키면 학습에 걸리는 시간이 단축됨
  - 특성 1과 2의 스케일을 통일시킨 경우



훈련과정을 통해 비용이 줄어드는 것으로 보임.(적

  - 특성 1의 값들의 스케일이 특성 2보다 더 큰 경우



커브형태로 조금 더 오래걸림.

경사 하강법의 하이퍼파라미터

- **경사 하강법을 통한 모델 훈련 과정이 어떻게 진행되도록 할 것인지를 설정**하기 위한 파라미터.
- **배치(batch) 사이즈:** 현재 모델 파라미터 벡터 조정을 위한 **그레디언트 벡터를 계산에 사용되는 훈련 샘플의 수**
  - **배치 경사 하강법:** 훈련셋 전체를 사용하여 그레디언트 벡터를 계산
  - **확률적 경사 하강법:** 랜덤하게 선택된 훈련 샘플 하나를 사용하여 그레디언트 벡터를 계산
  - **미니 배치 경사 하강법:** 훈련셋의 랜덤 서브셋을 사용하여 그레디언트 벡터를 계산

$$\nabla_{\boldsymbol{\theta}} \operatorname{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \operatorname{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \operatorname{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \operatorname{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \mathbf{X}^{\intercal} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

최종형태 n+1,1쉐입

만약 확률적이나 미니배치일경우 m이 아닌 I로 사용한다보자 (I < m이라는 가정)
계산을 해보면 최종 쉐입 형태는 (n+1,1)쉐입

m : 전체 훈련샘플 수
즉 훈련셋 전체를 사용해 계산한 배치 경사 하강법

이때 X의 쉐입은 (m, n+1)형태 (n+1은 X0 = 1까지포함해서)
세타의 쉐입 : (n+1,1), y의 쉐입 : (m, 1)

즉 최종형태는 3가지 모두 (n+1,1)쉐입으로 같다.

경사 하강법의 하이퍼파라미터

- **에포크(epoch):**
  - 총 훈련 샘플 수 m개만큼의 훈련 데이터에 대한 학습이 이루어지는 주기
    - **m개만큼의 훈련 샘플들을 이용하여 모델 파라미터 벡터 조정이 이루어지는 주기**
  - **1 에포크 동안 1번 이상의 gradient descent step 들이 수행**된다.

- **스텝(gradient descent step):** 지정된 **배치 사이즈만큼의 훈련 샘플들을 이용하여 그레디언트 벡터를 계산하고 모델 파라미터 벡터 조정을 수행하는 주기**
  - **1 에포크 당 총 스텝 수 = (훈련 샘플 수) / (배치 사이즈)**
  - 예: 훈련셋의 크기가 2,000이고 배치 사이즈가 10이면, 1 에포크 동안 총 200번의 스텝이 실행됨
    m= 2000, 미니배치 경사하강법으로 I = 10인 경우

- **허용오차(tolerance):** **그레디언트 벡터의 크기(norm)가 허용오차 보다 작아지면 비용함수의 최소값에 근접했음을 의미**하므로 경사 하강법 알고리즘의 반복 과정을 종료
  종료판단 기준

경사 하강법의 종류

- **배치 경사 하강법(Batch gradient descent)**
  - **배치 크기: m(= total number of training samples)** 전체 훈련수와 동일
  - 매 gradient descent step 마다 훈련셋 전체를 사용하여 그레디언트 벡터를 계산

    가장 많은 정보량으로 세타 조정
- **확률적 경사 하강법(stochastic gradient descent)**
  - **배치 크기: 1**
  - 매 gradient descent step 마다 랜덤하게 선택된 훈련 샘플 하나를 사용하여 그레디언트 벡터를 계산

    가장 적은 정보량으로 세타 조정
- **미니배치 경사 하강법(Mini-batch gradient descent)**
  - **2 ≤ (배치 크기) < m**
  - 매 gradient descent step 마다 훈련셋의 랜덤 서브셋을 사용하여 그레디언트 벡터를 계산

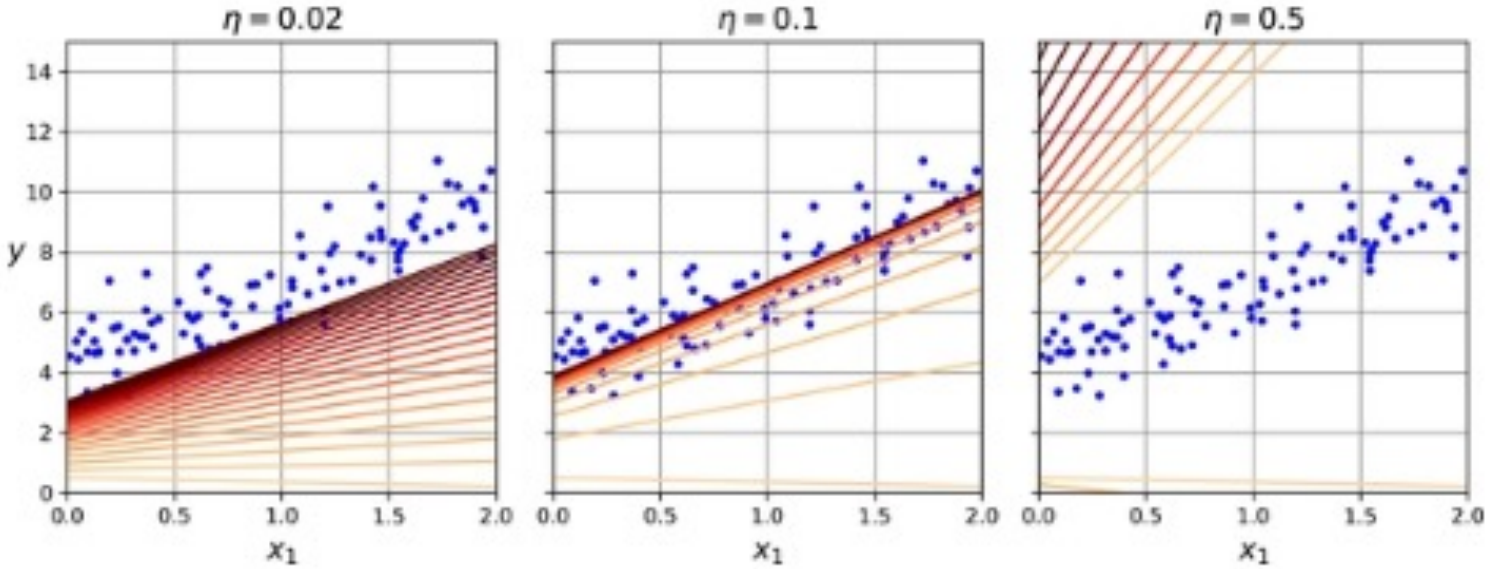
배치 경사 하강법

- 진행 과정
  - 초기화: 모델 파라미터 벡터 $\boldsymbol{\theta}$ 를 랜덤하게 초기화 하여 훈련 시작
  - 매 에포크/스텝 마다 전체 훈련셋을 사용하여 그레이디언트 벡터를 계산하여 모델 파라미터를 조정한다.
    - 배치 사이즈가 전체 훈련셋의 크기와 같고, 따라서 에포크 당 스텝 수((훈련 샘플 수) / (배치 사이즈))는 1이 된다. 배치 경사 하강법의 경우 무조건 1로 고정(m/m = 1이므로)
  - 계산된 그레디언트 벡터의 크기가 허용오차(ϵ) 이내이면 반복을 종료한다.
- 허용오차(ϵ)와 에포크 수 간 관계
  - 요구되는 허용오차가 작을수록 비용함수의 최소값에 더 근접함을 의미
  - 허용오차와 (그에 도달하기 위해) 필요한 에포크 수는 상호 반비례 관계임.
    - E.g., 허용오차를 1/10로 줄이려면 수행 에포크 수를 10배 늘려야 함
- 단점
  - 훈련셋이 클 경우 **전체 훈련셋을 사용하여 반복적으로 그레디언트 벡터를 계산하는데 많은 시간과 메모리가 필요함** ➔ 이런 이유로 사이킷런은 배치 경사 하강법을 지원하지 않음

배치 경사 하강법

- 학습율(η)과 경사 하강법의 관계
  - 학습률에 따라 선형 회귀 모델이 최적의 모델로 수렴하는지 여부와 수렴 속도가 달라진다.
  - 최적의 학습률은 그리드 탐색 등을 통해 찾아볼 수 있다.
  - 20 에포크 동안 배치 경사 하강법 진행 과정

학습률이 작을수록 예측함수그래프에 촘촘하게 나타난다.



20에포크로 고정을 했기 때문에 이러한 결과가 나옴.

확률적 경사 하강법(Stochastic Gradient Descent)

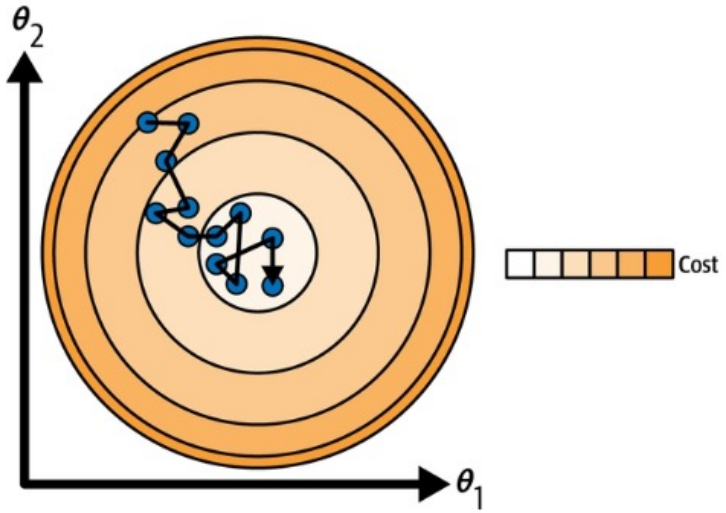
- 배치 사이즈: 1

- 매 gradient descent step 마다 **랜덤하게 선택된 훈련 샘플 하나를 사용하여 그 레디언트 벡터를 계산**하고 모델 파라미터 벡터를 조정한다.
  - 1 에포크 동안 위 과정을 m 번 반복해서 수행(m: 총 훈련 샘플 수) m/1 = m이므로
  - 1 에포크 동안 중복 선택되거나 한번도 선택되지 않는 훈련 샘플이 존재할 수 있음

확률적 경사 하강법(Stochastic Gradient Descent)

- 장점:
  - 매 gradient descent step 당 계산량이 상대적으로 적어 매우 큰 훈련셋을 다룰 수 있으며, 외부 메모리(out-of-core) 학습을 활용할 수 있음
  - 모델 파라미터 조정이 불규칙한 패턴으로 이루어질 수 있으며, 이런 이유로 지역 최소값에는 상대적으로 덜 민감함

- 단점:
  - 전역 최소값에 수렴하지 못하고 주변을 맴돌 수 있음



학습 스케줄(Learning Schedule)

- 확률적 경사 하강법의 randomness는 지역 최소값으로부터 벗어나는데는 좋으나 전역 최소값에 수렴하지 못하고 주변을 맴돌 수 있다는 단점 또한 있음
- 이러한 딜레마에 대한 해결책으로 **훈련이 진행되어감에 따라 학습률을 점진적으로 줄여나가는 방법**을 취할 수 있음

  학습률 (eta)를 고정하지 않고 계속 변화주기
- 주의사항
  - **학습률이 너무 빨리 줄어들면, 지역 최소값에 갇힐 수 있음**
  - 반대로 **학습률이 너무 천천히 줄어들면 전역 최소값에 제대로 수렴하지 못할 수 있음**

    그래프를 생각해보자.
- **학습 스케줄(learning schedule)**
  - 모델 훈련이 진행되는 동안 학습률을 조금씩 줄여나가는 기법
  - 일반적으로 훈련 과정이 얼마나 진행되었는지에 대한 에포크 및 스텝 카운트 값을 이용하여 매 스템마다 적용할 학습률을 계산

확률적 경사 하강법 적용 예: SGDRegressor

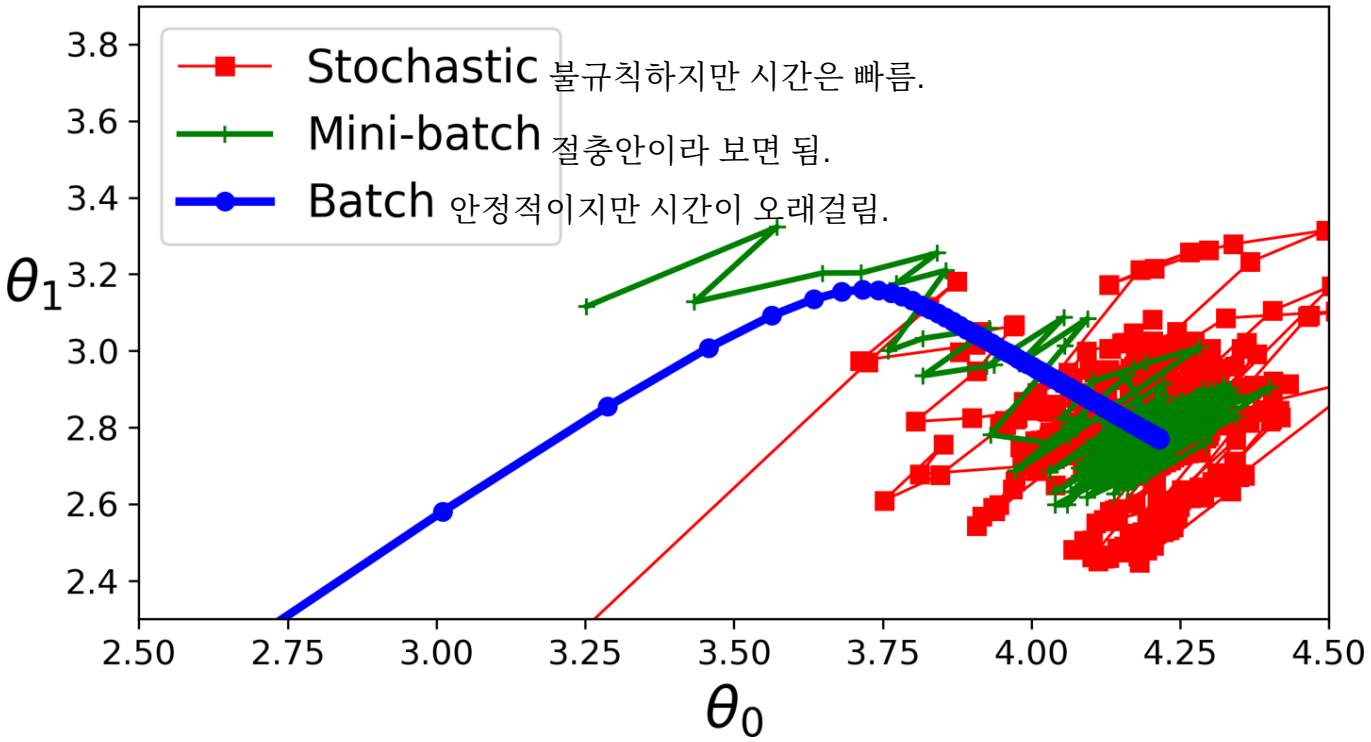
```
from sklearn.linear_model import
SGDRegressor

sgd_reg = SGDRegressor(max_iter=1000,
tol=1e-5, penalty=None, eta0=0.01,

n_iter_no_change=100, random_state=42)
sgd_reg.fit(X, y.ravel())  # y.ravel()
because fit() expects 1D targets
```

- 사이킷런의 SGDRegressor 클래스를 이용하여 확률적 경사 하강법 적용 가능
  - 모델 파라미터 조정 과정을 언제까지 반복할 것인지를 설정하는 파라미터
    - max_iter=1000 : 최대 1000 에포크 동안 실행
    - 또는 n_iter_no_change=100: 100 에포크 동안 향상되는 정도가 10^{-5}(=tol) 보다도 적으면 종료
  - eta0=0.01: 학습률 0.01로 시작되며 default learning schedule 사용됨

미니배치 경사 하강법(Mini-Batch Gradient Descent)

Colab 실습

- 장점
  - 배치 사이즈를 어느 정도 크게 하면 확률적 경사 하강법(SGD) 보다 모델 파라미터의 움직임의 불규칙성을 완화할 수 있음
  - 반면에 배치 경사 하강법보다 빠르게 학습 전체 m개보단 작으므로
  - 학습 스케줄을 잘 설정하면 최소값에 수렴 가능함



선형 회귀 모델 훈련 알고리즘 비교

- 선형 회귀 모델 훈련을 위한 각 알고리즘 별 주요 특징 및 관련 사이킷런 클래스

Table 4-1. Comparison of algorithms for linear regression

Algorithm	Large *m* 훈련샘플수	외부메모리 사용 Out-of-core support	Large *n* 입력특성	Hyperparams	Scaling required 특성 스케일링	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	N/A
Stochastic GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`
Mini-batch GD	Fast	Yes	Fast	≥2	Yes	N/A

즉 입력특성이 많아질수록 시간 오래걸림.

훈련샘플수에대해선 아무래도 시간이 제약적.

입력특성에 대해선 민감하지 않아보임.

적용할수록 빠르다.

MYONGJI UNIVERSITY

다항 회귀(POLYNOMIAL REGRESSION)

MYONGJI UNIVERSITY

다항 회귀(Polynomial Regression)

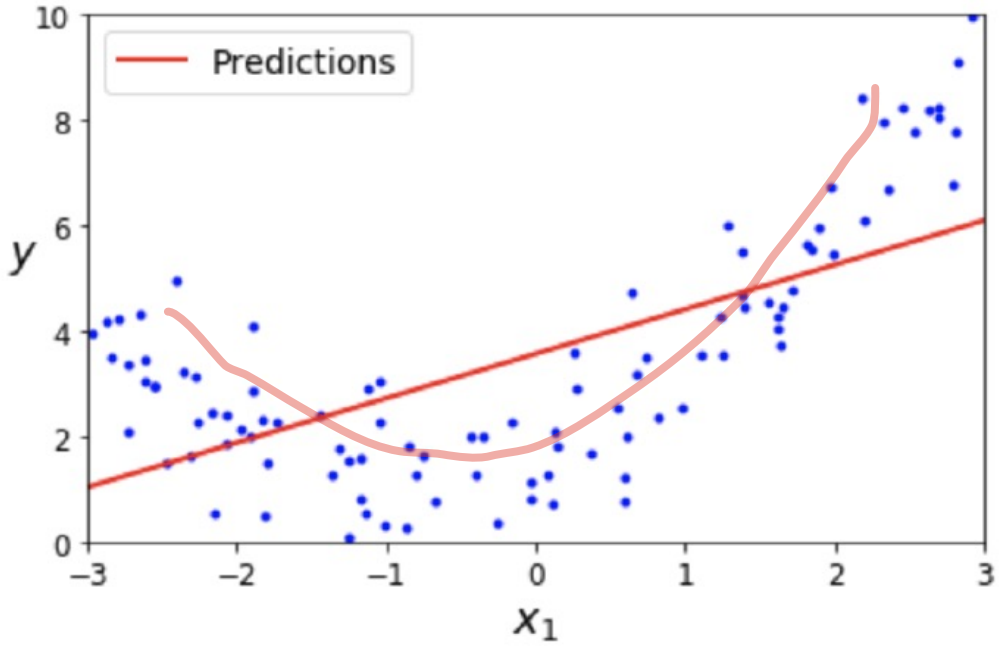
- 다항 회귀(polynomial regression)이란?
  - 선형 회귀 모델을 이용하여 비선형 데이터를 학습하는 기법
    - 즉, 비선형 데이터를 학습하는데 선형 회귀 모델 사용을 가능하게 함.

- 기본 아이디어
  - 훈련셋에 포함된 각 특성의 power(e.g., 제곱)를 새로운 특성으로 추가
  - 새로운 특성 추가를 통해 확장된 훈련셋을 이용하여 선형 회귀 모델을 훈련함

선형 회귀 vs. 다항 회귀

- 선형 회귀: 1차 선형 회귀 모델 입력특성이 x1하나인 모델

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



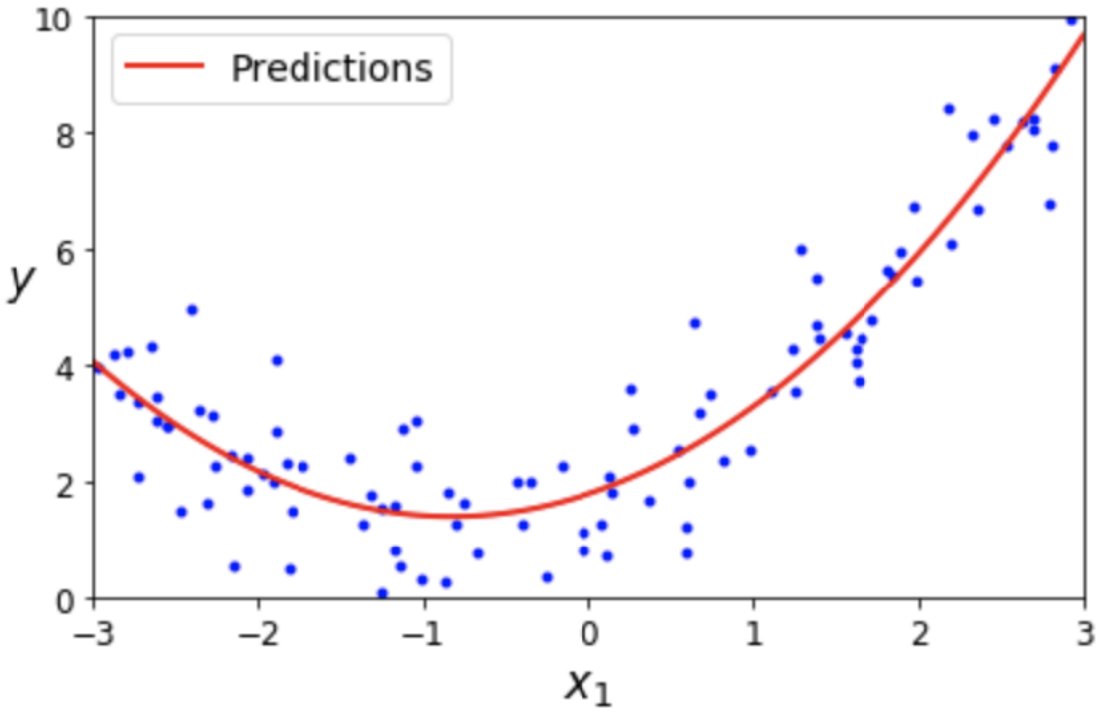
즉 2차식모양의 비선형적 모델을 띄기 때문에
1차선형회귀로 나타내기엔 무리

- 다항 회귀: 2차 다항식 모델

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

x1^2이라는 새로운 특성값 추가

따라서, 다항회귀로 만들어보자.



사이킷런의 PolynomialFeatures 변환기

- 주어진 훈련셋에 포함된 특성들 각각의 거듭제곱과 특성들 간의 곱셈을 실행하여 새로운 특성을 추가하는 기능 제공

- degree = d : 몇 차 다항식 모델에 해당하는 새로운 특성들을 추가 생성할지를 지정하는 하이퍼파라미터
  - 이전 예제: degree=2로 지정하면 x_1^2 이 새로운 특성으로 추가됨.

사이킷런의 PolynomialFeatures 변환기

- 예제1: (훈련셋 특성 수 n)=1, degree=2인 경우 기존 특성 x_1에 더불어서 x_1^2이 새로운 특성으로 추가됨

- 예제2: (훈련셋 특성 수 n)=2, degree=2인 경우 기존 특성 x_1, x_2에 더불어서 다음 3개 특성이 새로 추가됨
  - $x_1^2,\ x_1x_2,\ x_2^2$ 까먹지 말기

- 예제3: (훈련셋 특성 수)=2, degree=3인 경우에 다음 7개 특성 추가됨

즉 다항식의 형태 그대로 따라간다 생각하면 됨. 차수늘었을때 항들이 몇개생

$$x_1^2,\ x_1x_2,\ x_2^2,\ x_1^3,\ x_1^2x_2,\ x_1x_2^2,\ x_2^3$$

사이킷런의 PolynomialFeatures 변환기

- **PolynomialFeatures(degree=d, include_bias=True)** transforms an array con taining n features into an array containing $(n+d)!/d!\,n!$ features.
  - $(n+d)!/d!\,n!$ features include $x_0 = 1$. 일반화식

더미특성 포함.

사이킷런의 PolynomialFeatures 변환기

Colab 실습

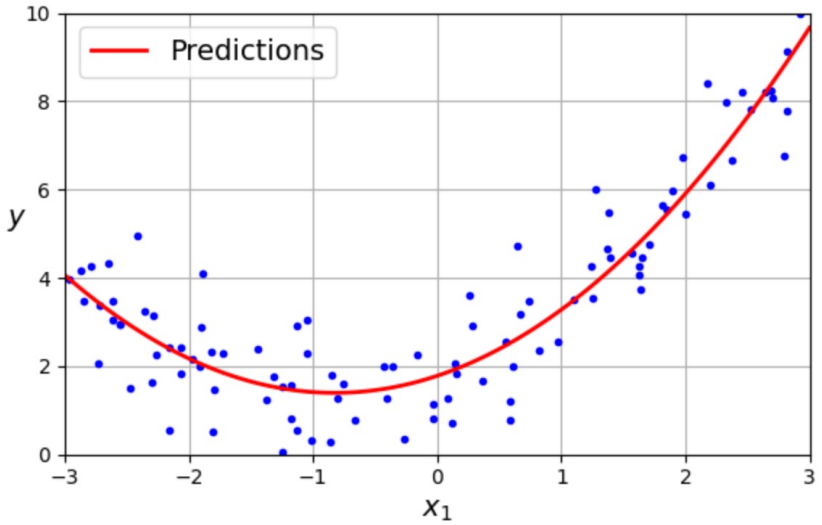
- PolynomialFeatures 변환기를 이용하여 새로운 다항 특성 추가
  - X_poly: 새로운 다항 특성 추가를 통해 확장된 훈련셋

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly_features = PolynomialFeatures(degree=2, include_bias=False)
>>> X_poly = poly_features.fit_transform(X)
>>> X[0]
array([-0.75275929])
>>> X_poly[0]
array([-0.75275929,  0.56664654])
```

- 확장된 훈련셋을 이용하여 선형 회귀 모델 훈련

다항특성이 포함된 선형회귀 모델

```
>>> lin_reg = LinearRegression()
>>> lin_reg.fit(X_poly, y)
>>> lin_reg.intercept_, lin_reg.coef_
(array([1.78134581]), array([[0.93366893, 0.56456263]]))
```



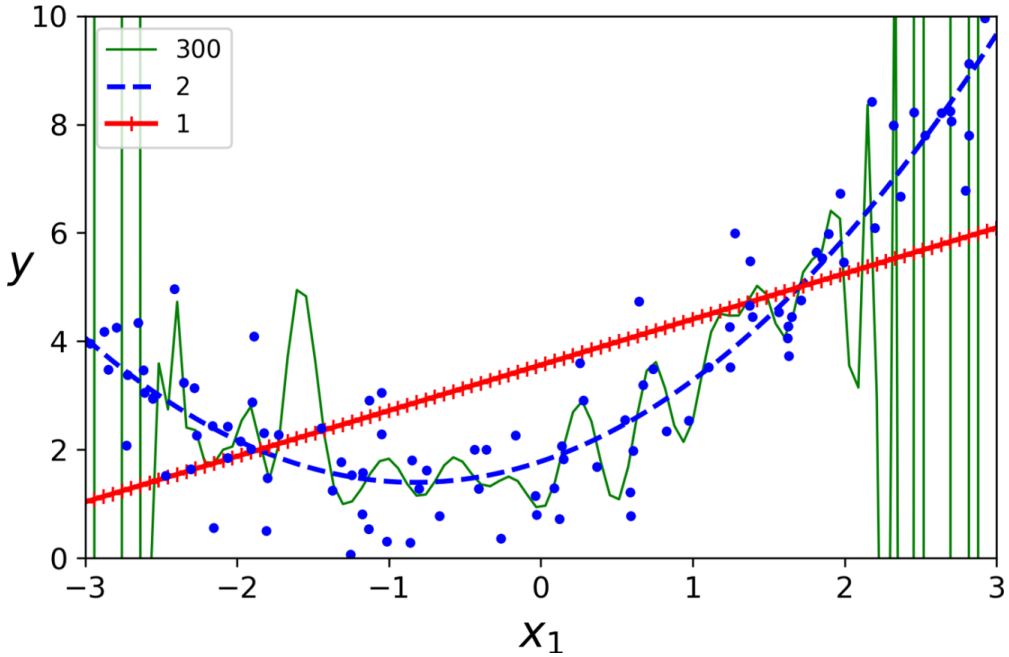
MYONGJI UNIVERSITY

여기까지 시험범위!

학습 곡선(Learning Curve)

모델 과소적합/과대적합 판정

- 예제: 선형 모델, 2차 다항 회귀 모델, 300차 다항 회귀 모델 비교
- 다항 회귀 모델의 차수에 따라 훈련된 모델이 훈련셋에 과소 또는 과대 적합될 수 있음.

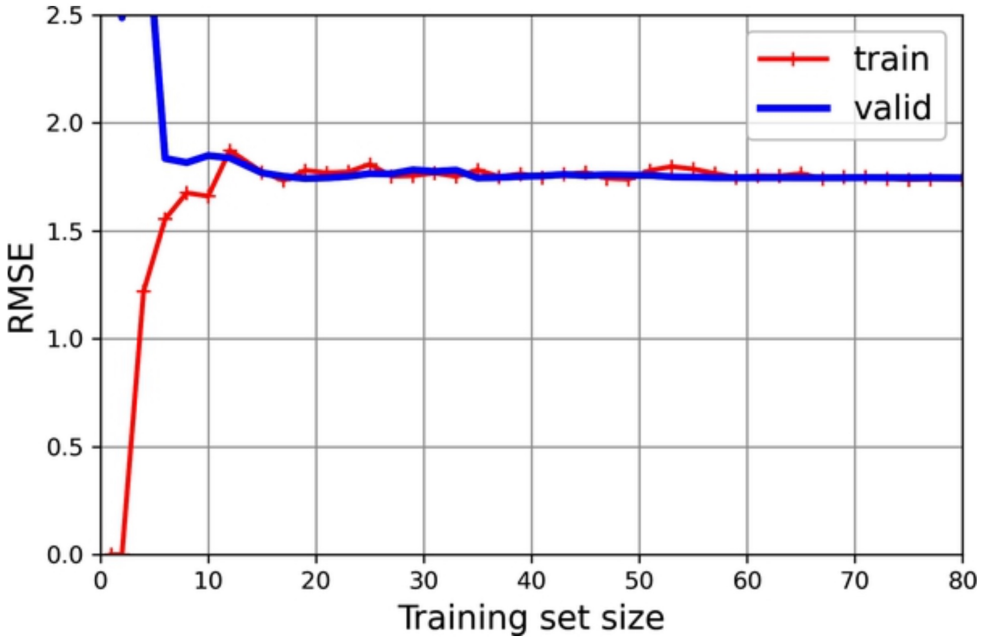


교차 검증 vs. 학습 곡선

- 교차 검증
  - 과소적합: 훈련 세트와 교차 검증 점수 모두 낮은 경우
  - 과대적합: 훈련 세트에 대한 성능은 우수하지만 교차 검증 점수가 낮은 경우

- 학습 곡선 살피기
  - 학습 곡선: 모델이 학습한 훈련 샘플 수가 조금씩 증가함에 따라 훈련 세트와 검증 세트에 대한 모델 성능을 비교하는 그래프
  - 훈련 샘플들에 대한 학습이 진행되는 동안 주기적으로 훈련셋과 검증셋에 대한 모델의 성능을 측정
  - 학습 곡선의 모양에 따라 과소적합/과대적합 판정 가능

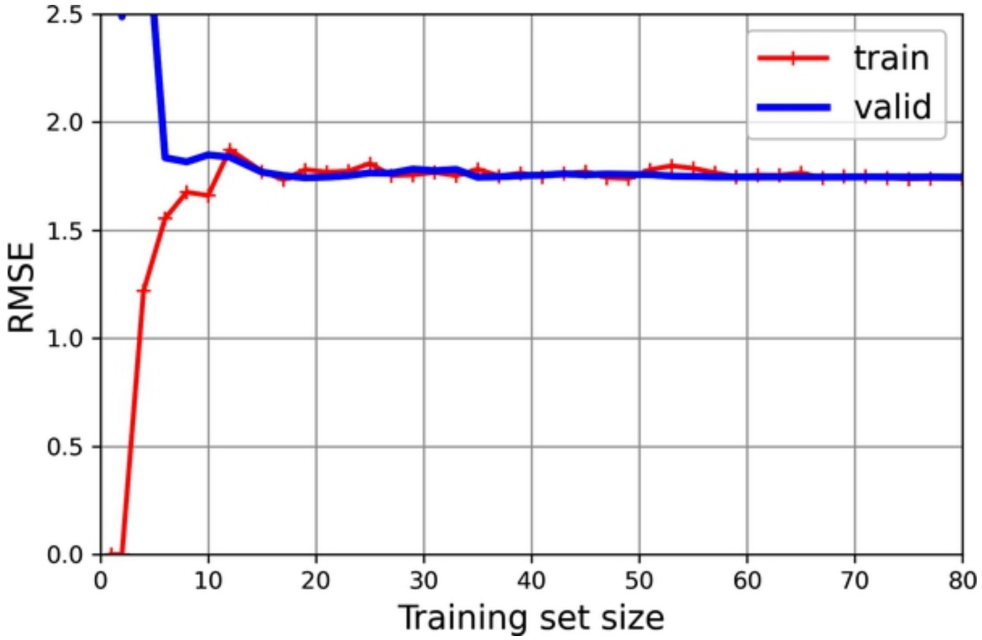
과소적합 모델의 학습 곡선 특징



훈련 샘플들에 대한 training error (red line)

- 모델이 1 or 2개 훈련 샘플만을 학습했을 때 training error (e.g., RMSE)는 0에서 출발
- 훈련 샘플들이 추가되면서 training error가 커짐
- 훈련 세트가 어느 정도 커지면 새로운 훈련 샘플이 추가되더라도 더 이상 training error가 향상되지 않음

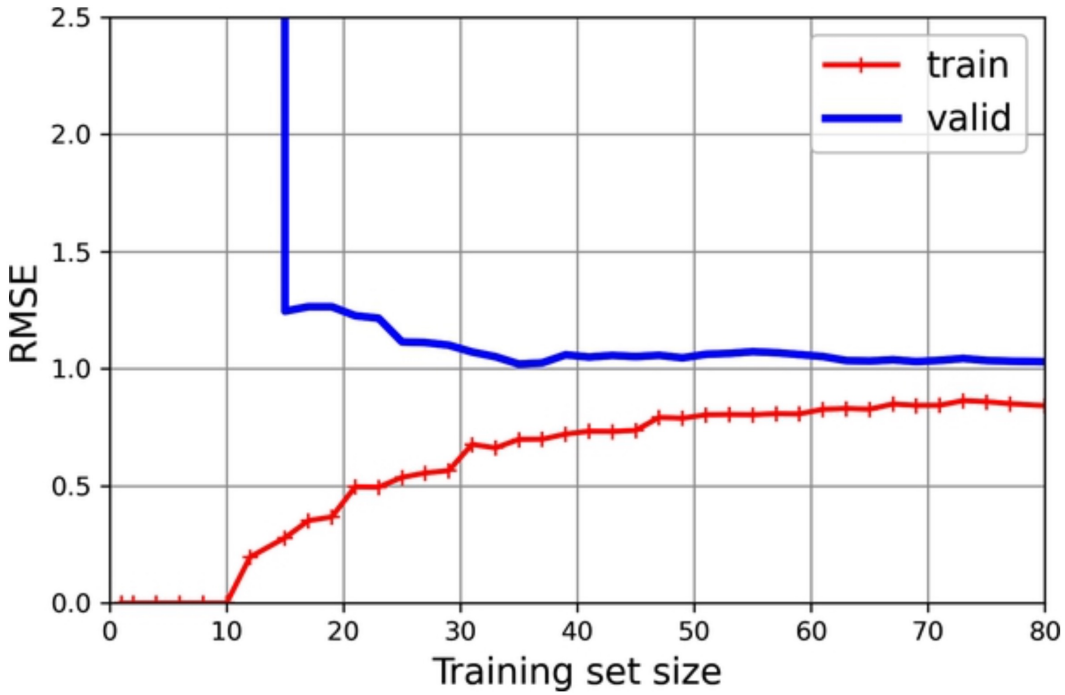
과소적합 모델의 학습 곡선 특징



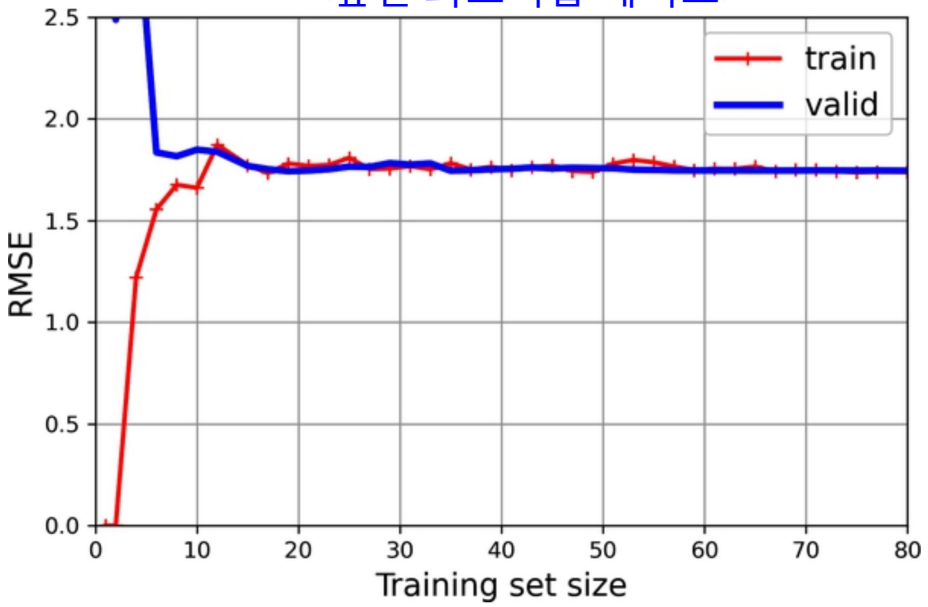
검증 데이터에 대한 validation error (blue line)

- 모델이 적은 수의 훈련 샘플들만을 학습한 상태일 때는 상당히 높은 valid. error가 관찰됨
- 모델이 학습한 훈련 샘플 수가 늘어나면 valid. error가 줄어드는 패턴
- 하지만 역시 훈련셋 사이즈가 일정 수준에 도달하면 valid. error가 더 이상 나아지지 않음
- 검증 세트에 대한 성능이 훈련 세트에 대한 성능과 거의 비슷해짐

과대적합 모델의 학습 곡선 특징



<앞선 과소적합 케이스>



- 훈련 세트(red)에 대한 성능: 훈련셋에 대한 RMSE가 과소적합 모델 경우보다 상대적으로 매우 낮음.
- 검증 세트(blue)에 대한 성능: 훈련셋에 대한 성능과 차이가 벌어짐.
- 과대적합 모델 개선법: 훈련셋 사이즈를 훨씬 더 크게 키우면 두 커브가 더 가까워 질 수 있어 보임

모델 일반화 오차: 편향/분산 간 트레이드오프

- 모델 일반화 오차(generalization error)는 다음 세가지 종류 오차들의 합으로 표현될 수 있음
  - 편향(bias), 분산(variance), 축소 불가능 오차(irreducible error)
- **편향**: 데이터에 대한 잘못된 가정에서 기인하는 오차
  - E.g., 데이터가 실제로는 2차원인데 선형으로 잘못 가정함으로 인해 발생하는 예측 오차
  - 일반적으로 편향이 큰 모델은 과소적합 되는 경향이 큼
- **분산**: 훈련 데이터에 존재하는 작은 variation에 모델이 과도하게 민감하게 반응함에서 기인하는 오차
  - 고차 다항 회귀 모델 같이 자유도가 높은 모델일 수록 분산 오차가 커지며, 과대적합 될 가능성이 높음
- **축소 불가능 오차**: 훈련 데이터 자체에 존재하는 노이즈 때문에 발생되는 오차
  - 훈련 데이터의 노이즈를 제거해야만 해결될 수 있는 오차

- **편향-분산 오차 간 트레이드오프**: **모델의 복잡도가 커지면 통상적으로 분산 오차가 늘어나고 편향은 줄어듬**. 반대로 **모델의 복잡도를 줄이면 편향이 늘어나고 분산은 줄어듬**. 따라서 트레이드오프 관계임.

모델 일반화 오차: 편향/분산 간 트레이드오프