

4장 모델 훈련 (1부)

주요 내용

- 선형 회귀
- 경사하강법
- 비선형 데이터 학습: 다항 회귀
- 학습 곡선
- 모델 규제
- 로지스틱 회귀

4.1. 선형 회귀

4.1.1. 머신러닝 모델이란?

예제: 1인당 GDP와 삶의 만족도

$$(\text{삶의 만족도}) = \theta_0 + (\text{1인당GDP}) \cdot \theta_1$$

또는

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

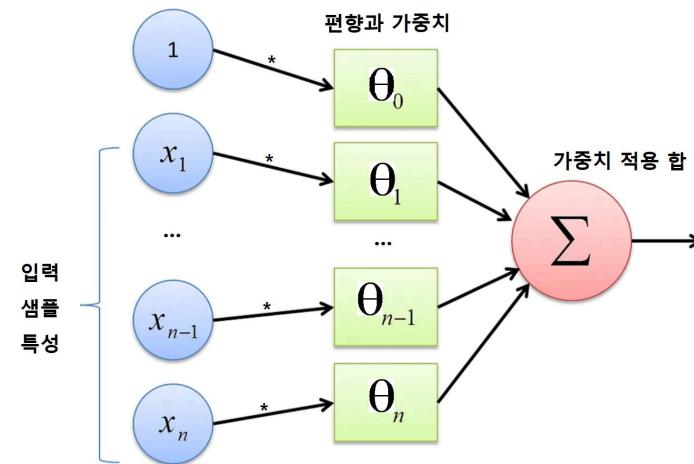
예제: 캘리포니아 주택 가격 예측

- \hat{y} : 예측된 주택 중위 가격
- x_i : 구역의 i 번째 특성값(위도, 경도, 중간소득, 가구당 인원 등)
- θ_0 : 편향
- θ_i : i 번째 특성에 대한 가중치. 단, $1 \leq i \leq 24$.

$$\hat{y} = \theta_0 + x_1 \cdot \theta_1 + \cdots + x_{24} \cdot \theta_{24}$$

선형 회귀 모델

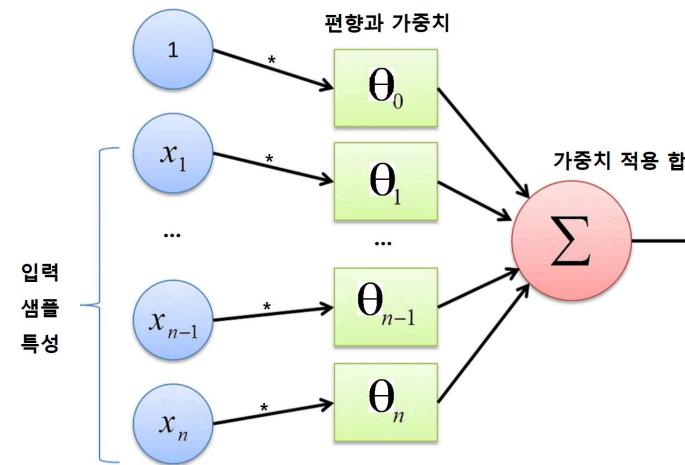
$$\hat{y} = \theta_0 + x_1 \cdot \theta_1 + \cdots + x_n \cdot \theta_n$$



파라미터, 편향, 가중치

- **파라미터**_{parameter}: $\theta_0, \theta_1, \dots, \theta_n$
- 파라미터는 모델이 훈련을 통해 학습
- θ_0 : **편향**_{bias}
- 나머지 파라미터: **가중치**_{weight}

4.1.2. 행렬 연산 표기법



$$\hat{y} = 1 \cdot \theta_0 + x_1 \cdot \theta_1 + \cdots + x_n \cdot \theta_n = [1, x_1, \dots, x_n] \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

Linear Regression 모델의 예측값 계산

- 머신러닝 모델은 일반적으로 여러 개의 입력값에 대해 동시에 예측값 계산

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 \\ \vdots \\ \hat{y}_{m-1} \end{bmatrix} = \begin{bmatrix} 1, x_1^{(0)}, \dots, x_n^{(0)} \\ \vdots \\ 1, x_1^{(m-1)}, \dots, x_n^{(m-1)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

- 간략하게 줄이면:

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

4.1.3. 머신러닝 모델 훈련의 목표

- 타깃에 최대한 가까운 예측값 계산
- 즉, 모델의 예측 성능 최대화
- 모델 훈련중에는 모델의 성능을 일반적으로 모델의 비용 함수를 이용하여 계산

비용 함수

- 모델의 성능이 얼마나 나쁜지 평가
- 모델의 종류와 목표에 따라 다른 비용 함수 선택
- 회귀 모델: 일반적으로 평균 제곱 오차(MSE)를 비용 함수로 사용

$$\text{MSE}(\theta) = \frac{1}{m} \sum_{i=1}^m (\mathbf{x}^{(i)} \theta - y^{(i)})^2$$

모델 훈련의 최종 목표

- $\text{MSE}(\theta)$ 가 최소가 되도록 하는 θ 찾기
- 선형 회귀의 경우 모델에 따라 다음 두 가지 방식 중 하나 이용
 - 방식 1: 정규방정식 또는 특이값 분해(SVD)
 - 방식 2: 경사하강법
- 정규 방정식: LinearRegression 등 선형 회귀를 활용하는 극히 일부 모델에서, 그것도 훈련셋의 크기와 입력 특성수 모두 작을 때만 활용됨.
- 경사하강법: 딥러닝 모델에서도 기본으로 활용되는 훈련 기법임.

4.2. 경사하강법

경사하강법 관련 주요 개념

하이퍼파라미터

- 훈련시킬 모델을 지정할 때 사용되는 설정 옵션
- 즉 해당 클래스의 객체를 생성할 때 클래스의 생성자 함수에 전달되는 인자들.
- 대표적으로 학습률, 에포크, 허용 오차, 배치 크기 등등

파라미터

- 선형 회귀 모델에 사용되는 편향과 가중치 파라미터처럼 모델 훈련중에 학습되는 값들
- 모델 훈련을 통해 학습된 파라미터는 훈련된 모델 객체의 속성으로 저장됨.

배치 크기

- 보다 좋은 파라미터 값으로 업데이트하기 위해 필요한 훈련 샘플의 수
- 전체 데이터셋의 크기 m 과 구분하기 위해 m_b 로 표기
- 파라미터 업데이트는 따라서 m_b 개의 훈련셋을 학습할 때마다 이뤄짐.

비용 함수

- 모델의 성능이 얼마나 나쁜가를 측정하는 함수
- 배치 단위로 비용 함수값 계산
- 회귀 모델의 배치 단위로 계산되는 MSE

$$\text{MSE}(\theta) = \frac{1}{m_b} \sum_{i=0}^{m_b-1} (\mathbf{x}^{(i)} \theta - y^{(i)})^2$$

전역/지역 최소값

- 비용 함수가 가질 수 있는 전역/지역 최소값
- 예제: 선형 회귀 모델의 평균 제곱 오차(MSE) 함수가 갖는 전역/지역 최소값

스텝

- 모델이 파라미터를 한 번 업데이트 하는 과정
- 즉, 배치 크기 m_b 만큼의 샘플에 대해 예측값을 계산한 후에 비용 함수를 이용하여 성능을 평가한 후에 비용 함수를 줄이는 방향으로 파라미터를 한 번 업데이트 하는 과정

학습률

- 훈련 스텝마다 비용 함수값 계산에 사용되는 파라미터 θ 를 얼만큼씩 조정할 것인지
를 정하는 비율

에포크

- 훈련셋에 포함된 모든 데이터를 대상으로 예측값을 한 번 계산하는 과정
- 이 과정동안 실행된 스텝 회수만큼 파라미터의 업데이트가 이루어짐.

스텝 크기

- 에포크 동안 실행된 스텝의 횟수, 즉 파라미터를 조정한 횟수

$$\text{스텝 크기} = (\text{훈련셋 크기}) / (\text{배치 크기})$$

- 예를 들어, 훈련셋 크기가 1,000이고 배치 크기가 10이면, 에포크 당 100번의 스텝이 실행됨.

최적 학습 모델

- 비용 함수를 최소화하는 파라미터를 학습한 모델
- 최종적으로 훈련을 통해 얻고자 하는 모델

비용 함수의 그레이디언트 벡터

- 함수의 그레이디언트 벡터는 방향과 크기에 대한 정보 제공
- 그레이디언트가 가리키는 방향의 **반대 방향**으로 움직여야 가장 빠르게 전역 최솟값에 접근
- MSE(θ) 함수의 θ 에 대한 그레이디언트 벡터

$$\nabla_{\theta} \text{MSE}(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \text{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1} \text{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \text{MSE}(\theta) \end{bmatrix}$$

허용 오차

- 비용 함수의 그레이디언트 벡터의 크기가 허용 오차보다 작아질 때 훈련 종료
- 그레이디언트 벡터의 크기가 0에 가까우면 비용 함수의 전역 또는 지역 최소값에 거의 다다랐음을 의미하기 때문임.

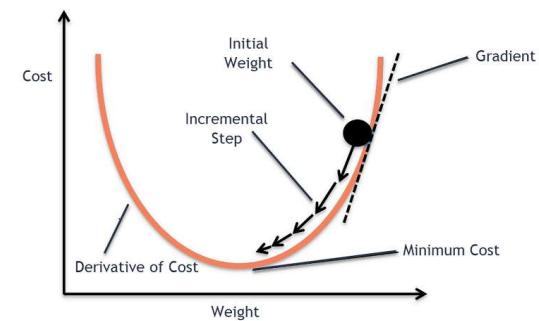
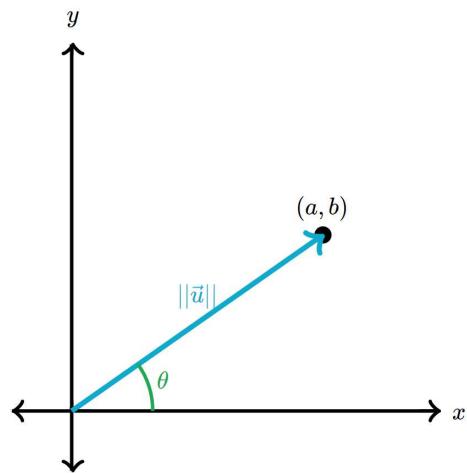
4.2.1. 선형 회귀 모델 훈련과 경사하강법

MSE를 비용 함수로 사용하는 경우 경사하강법은 다음 과정으로 이루어짐.

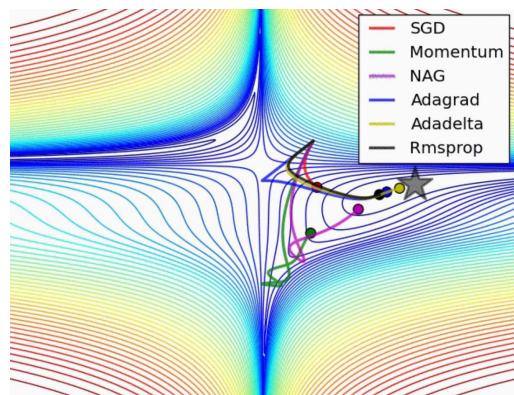
- θ 를 임의의 값으로 지정한 후 훈련 시작
- $\text{MSE}(\theta)$ 가 허용 오차보다 적게 작아질 때까지 아래 과정 반복
 - 배치 크기 m_b 만큼의 훈련 샘플을 이용하여 예측값 생성 후 $\text{MSE}(\theta)$ 계산.
 - 아래 점화식을 이용한 θ 업데이트

$$\theta^{(\text{new})} = \theta^{(\text{old})} - \eta \cdot \nabla_{\theta} \text{MSE}(\theta^{(\text{old})})$$

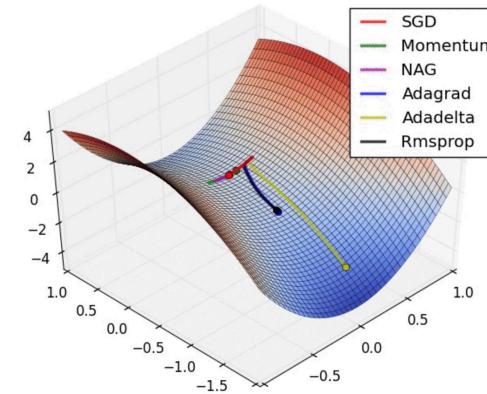
기울기 벡터의 방향과 크기



그레이디언트 벡터의 방향과 크기

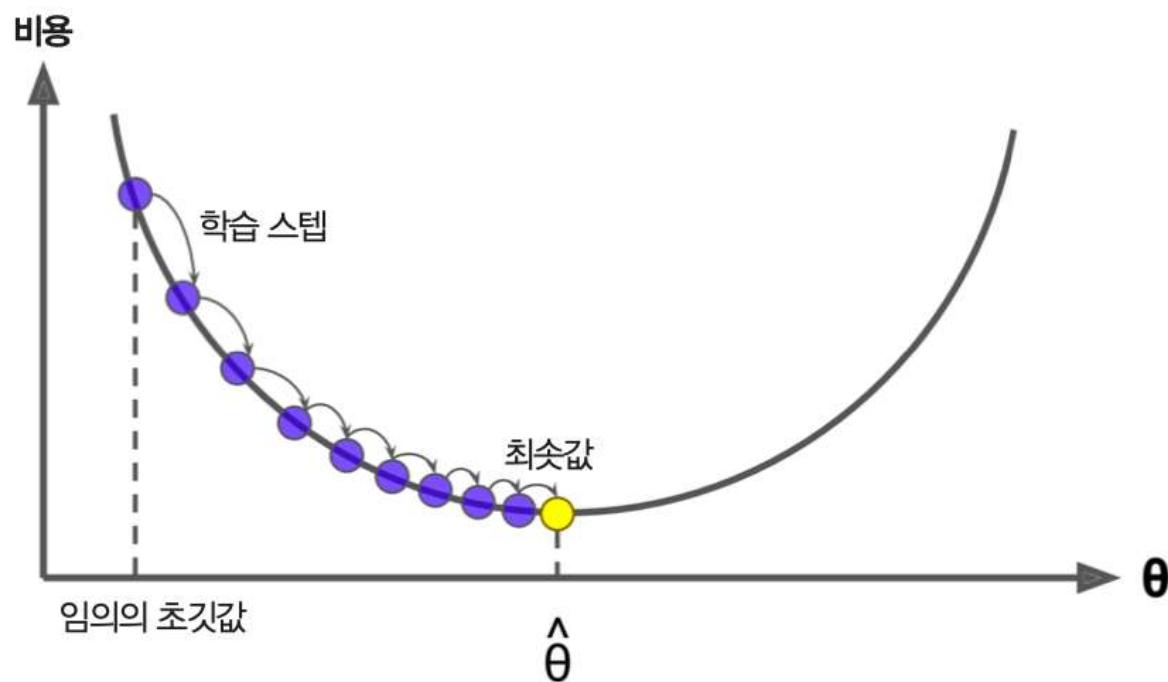


SGD optimization on loss surface contours

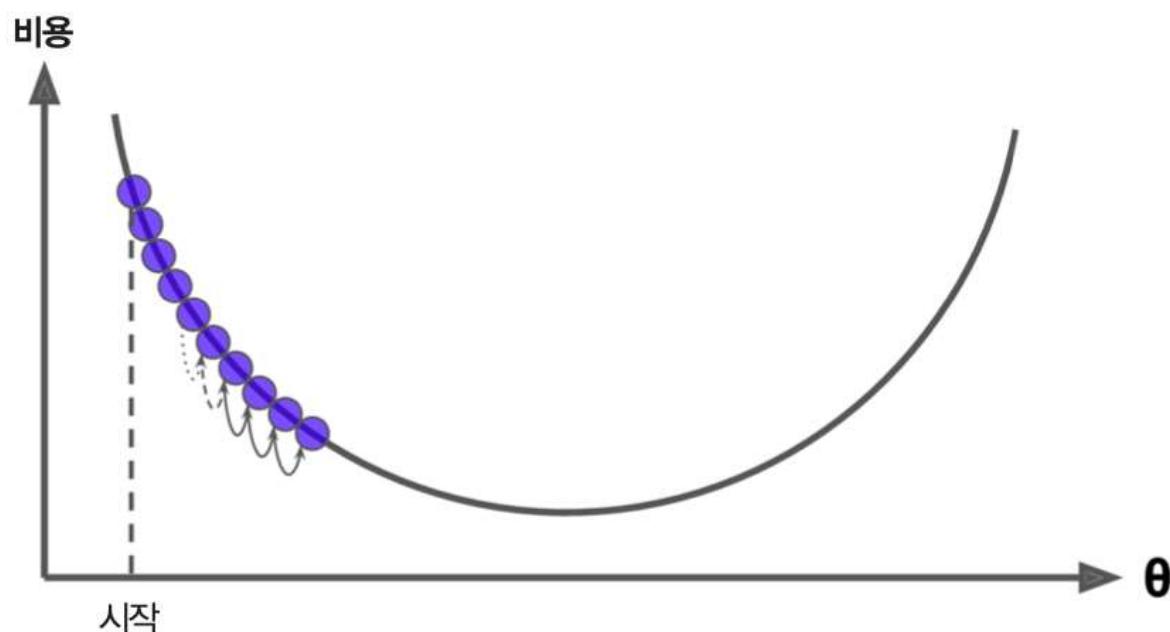


SGD optimization on saddle point

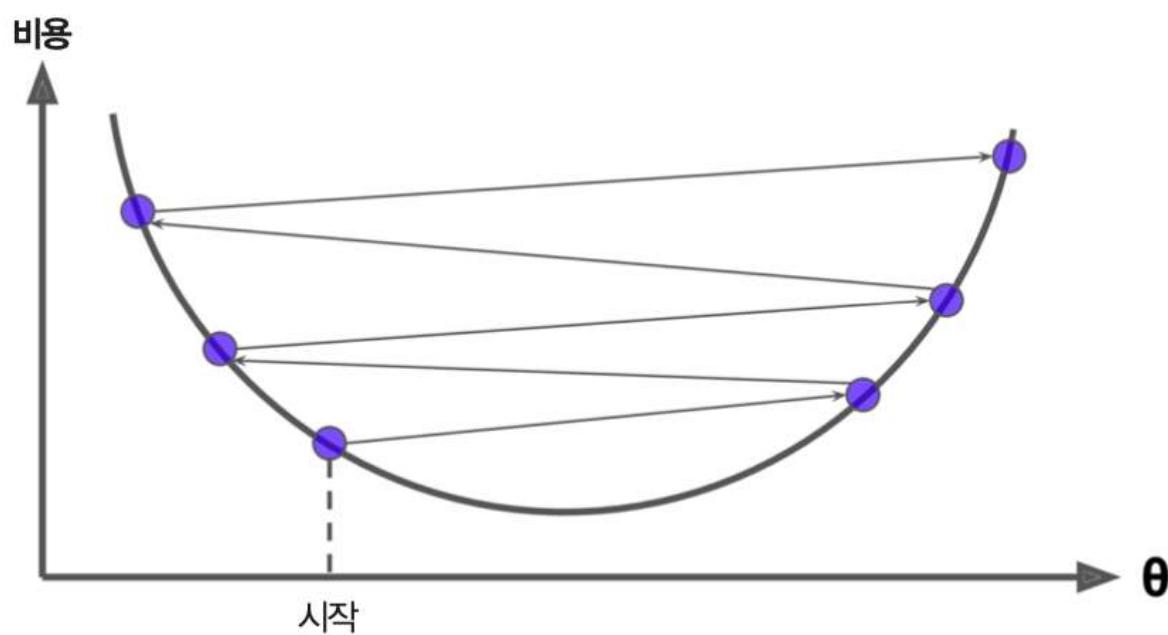
4.2.2. 학습률의 중요성



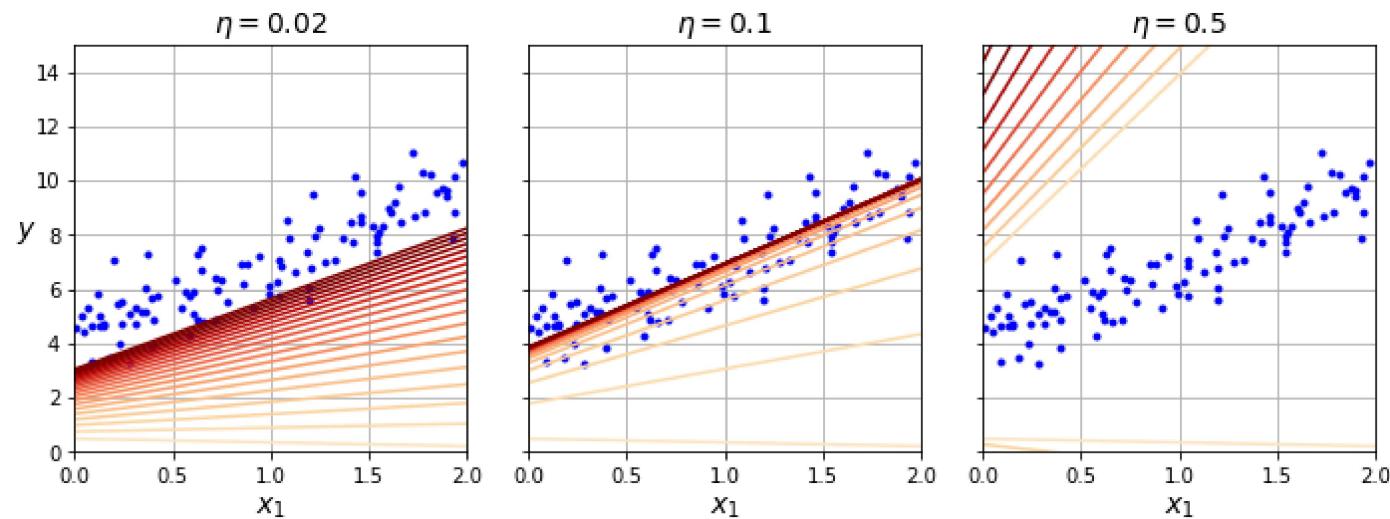
- 학습률이 너무 작은 경우: 비용 함수가 전역 최소값에 너무 느리게 수렴.



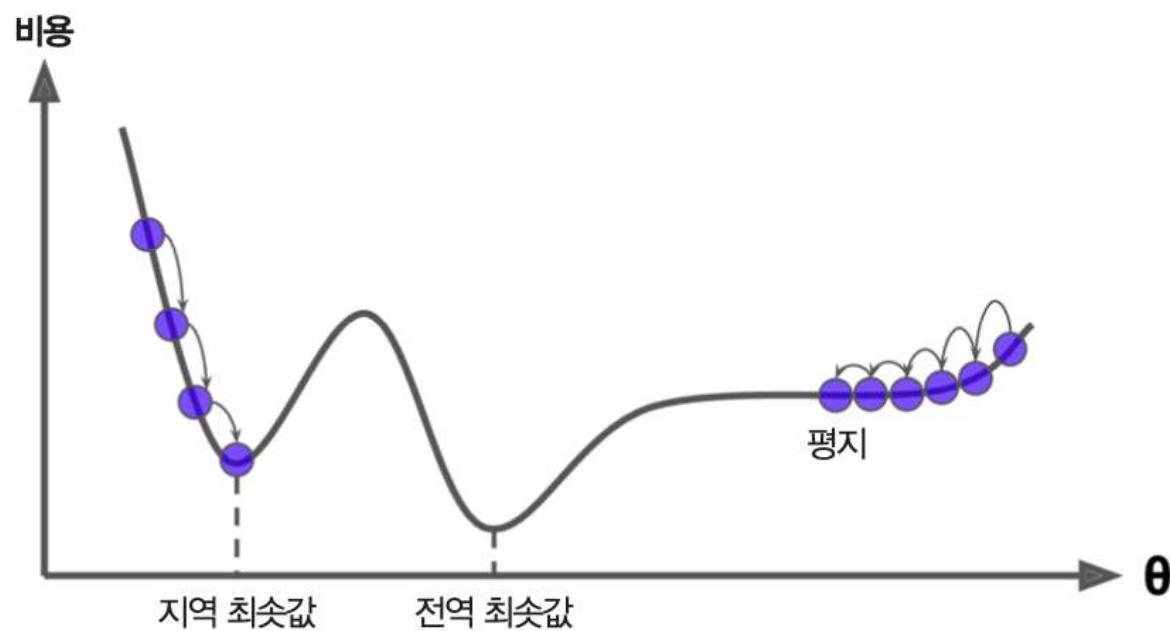
- 학습률이 너무 큰 경우: 비용 함수가 수렴하지 않음.



학습율과 경사하강법의 관계

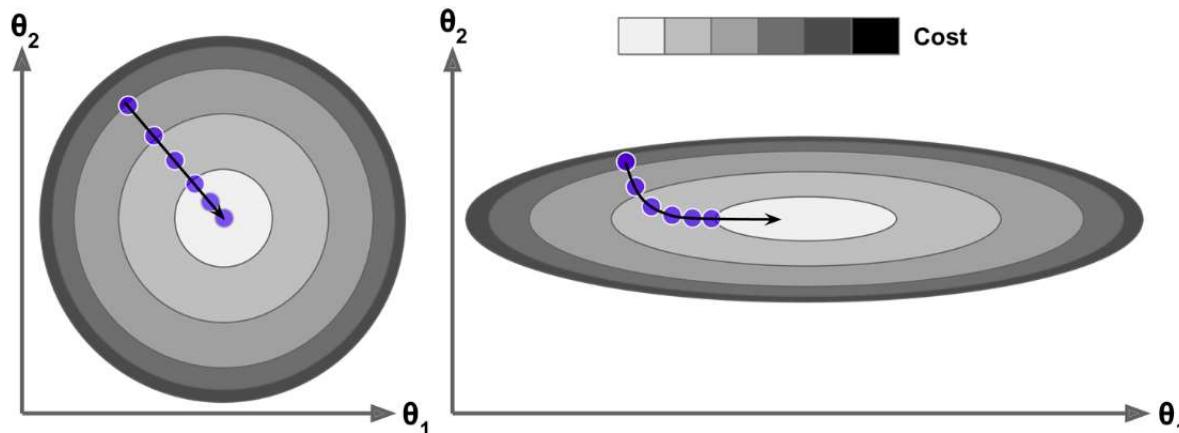


- (선형 회귀가 아닌 경우에) 시작점에 따라 지역 최솟값에 수렴하지 못할 수도 있음.



특성 스케일링의 중요성

- 원편 그림: 두 특성의 스케일이 동일하게 조정된 경우 비용 함수의 최솟값으로 최단 거리로 수렴한다. 등고선이 원 모양으로 그려지는 경우를 생각하면 된다.
- 오른편 그림: 두 특성의 스케일이 다른 경우 비용 함수의 최솟값으로 보다 먼 거리를 지나간다. 이런 경우엔 등고선이 타원 모양 또는 찌그러진 모양으로 그려지게 된다.



4.2.3. 경사하강법 종류

배치 경사하강법

- 전체 훈련 샘플을 대상으로 훈련한 후에, 즉 에포크마다 그레이디언트를 계산하여 파라미터 조정
- $m_b = m$
- **주의:** 여기서 사용되는 '배치'의 의미가 '배치 크기'의 '배치'와 다른 의미

확률적 경사하강법

- 배치 크기가 1. 즉, $m_b = 1$
- 즉, 하나의 훈련 샘플을 학습할 때마다 그레이디언트를 계산해서 파라미터 조정

미니배치 경사하강법

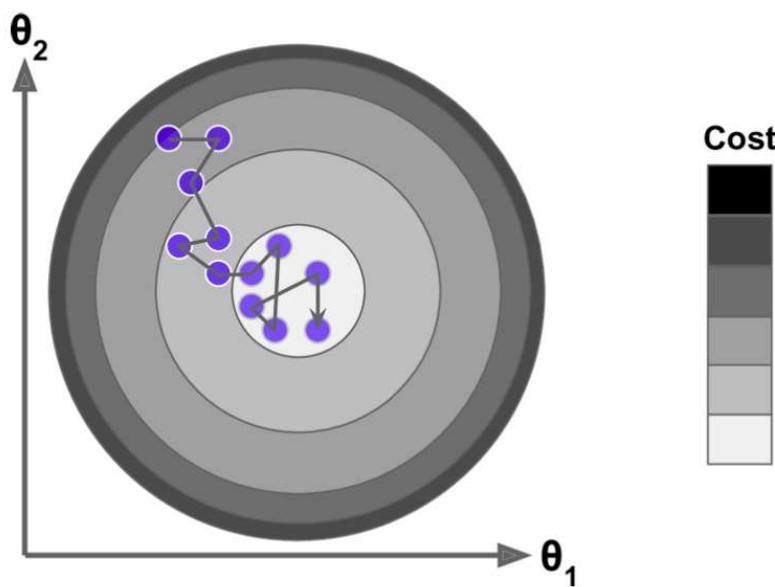
- 배치 크기: 2에서 수백 사이
- $2 \leq m_b = m < m$

배치 경사하강법

- 에포크와 허용 오차
 - 에포크 수는 크게 설정한 후 허용 오차를 지정하여 학습 시간 제한 필요. 이유는 포물선의 최솟점에 가까워질 수록 그레이디언트 벡터의 크기가 0에 수렴하기 때문임.
 - 허용 오차와 에포크 수는 서로 반비례의 관계임. 즉, 오차를 1/10로 줄이려면 에포크 수를 10배 늘려야함.
- 단점
 - 훈련 세트가 크면 그레이디언트를 계산하는 데에 많은 시간 필요
 - 아주 많은 데이터를 저장해야 하는 메모리 문제도 발생 가능
- 주의사항
 - 사이킷런은 배치 경사하강법을 활용한 선형 회귀를 지원하지 않음.

확률적 경사하강법

- 매우 큰 훈련 세트를 다룰 수 있음. 예를 들어, 외부 메모리(out-of-core) 학습을 활용할 수 있음
- 학습 과정이 매우 빠르며 파라미터 조정이 불안정 할 수 있기 때문에 지역 최솟값에 상대적으로 덜 민감
- 단점: 학습 과정에서 파라미터의 동요가 심해서 경우에 따라 전역 최솟값에 수렴하지 못하고 계속해서 발산할 가능성도 높음



학습 스케줄

- 요동치는 파라미터를 제어하기 위해 학습률을 학습 과정 동안 천천히 줄어들게 만들 수 있음
- 주의사항
 - 학습률이 너무 빨리 줄어들면, 지역 최솟값에 갇힐 수 있음
 - 학습률이 너무 느리게 줄어들면 전역 최솟값에 제대로 수렴하지 못하고 맴돌 수 있음
- 학습 스케줄(learning schedule)
 - 훈련이 지속될 수록 학습률을 조금씩 줄이는 기법
 - 에포크, 훈련 샘플 수, 학습되는 샘플의 인덱스에 따른 학습률 지정

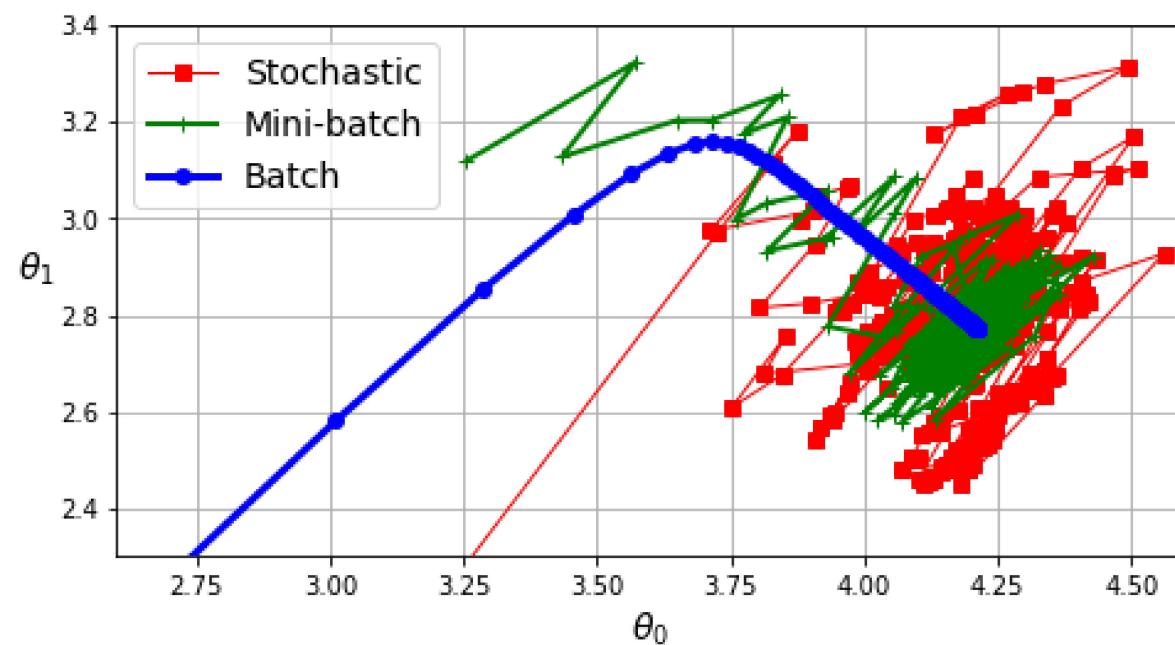
사이킷런의 `SGDRegressor`

- 경사하강법 사용
- 사용되는 하이퍼파라미터
 - `max_iter=1000`: 에포크 수 제한
 - `tol=1e-3`: 하나의 에포크가 지날 때마다 0.001보다 적게 손실이 줄어들 때 까지 훈련.
 - `eta0=0.1`: 학습 스케줄 함수에 사용되는 매개 변수. 일종의 학습률.
 - `penalty='l2'`: 규제 사용 여부 결정 (추후 설명)

미니배치 경사하강법

- 장점
 - 배치 크기를 어느 정도 크게 하면 확률적 경사하강법(SGD) 보다 파라미터의 움직임이 덜 불규칙적이 됨
 - 반면에 배치 경사하강법보다 빠르게 학습
 - 학습 스케줄 잘 활용하면 최솟값에 수렴함.
- 단점
 - SGD에 비해 지역 최솟값에 수렴할 위험도가 보다 커짐.

경사하강법 비교



선형 회귀 알고리즘 비교

알고리즘	많은 샘 플 수	외부 메모 리 학습	많은 특 성 수	하이퍼 파라 미터 수	스케일 조정	사이킷런 지원
정규방정 식	빠름	지원 안됨	느림	0	불필요	지원 없음
SVD	빠름	지원 안됨	느림	0	불필요	LinearRegression
배치 GD	느림	지원 안됨	빠름	2	필요	지원 없음
SGD	빠름	지원	빠름	$>= 2$	필요	SGDRegressor
미니배치 GD	빠름	지원	빠름	$>= 2$	필요	지원 없음

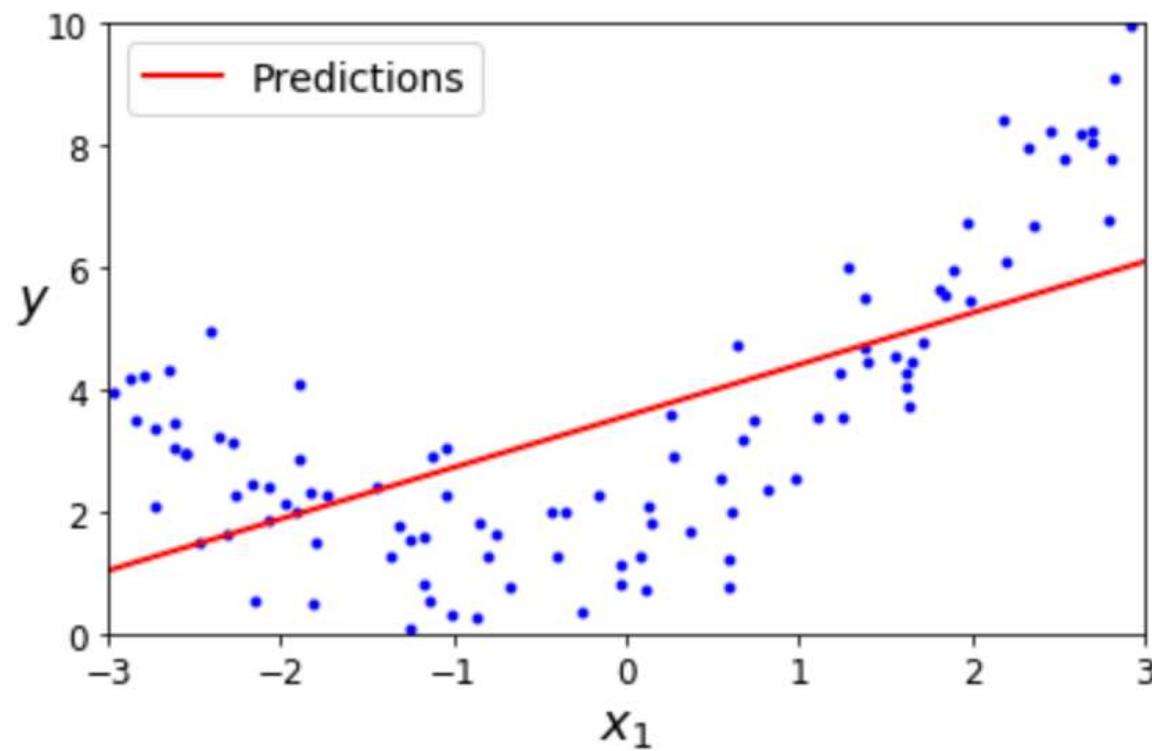
4.3. 비선형 데이터 학습: 다행 회귀

- 다항 회귀(polynomial regression)란?
 - 선형 회귀를 이용하여 비선형 데이터를 학습하는 기법
 - 즉, 비선형 데이터를 학습하는데 선형 모델 사용을 가능하게 함.
- 기본 아이디어
 - 특성들의 조합 활용
 - 특성 변수들의 다항식을 조합 특성으로 추가

선형 회귀 대 다행 회귀

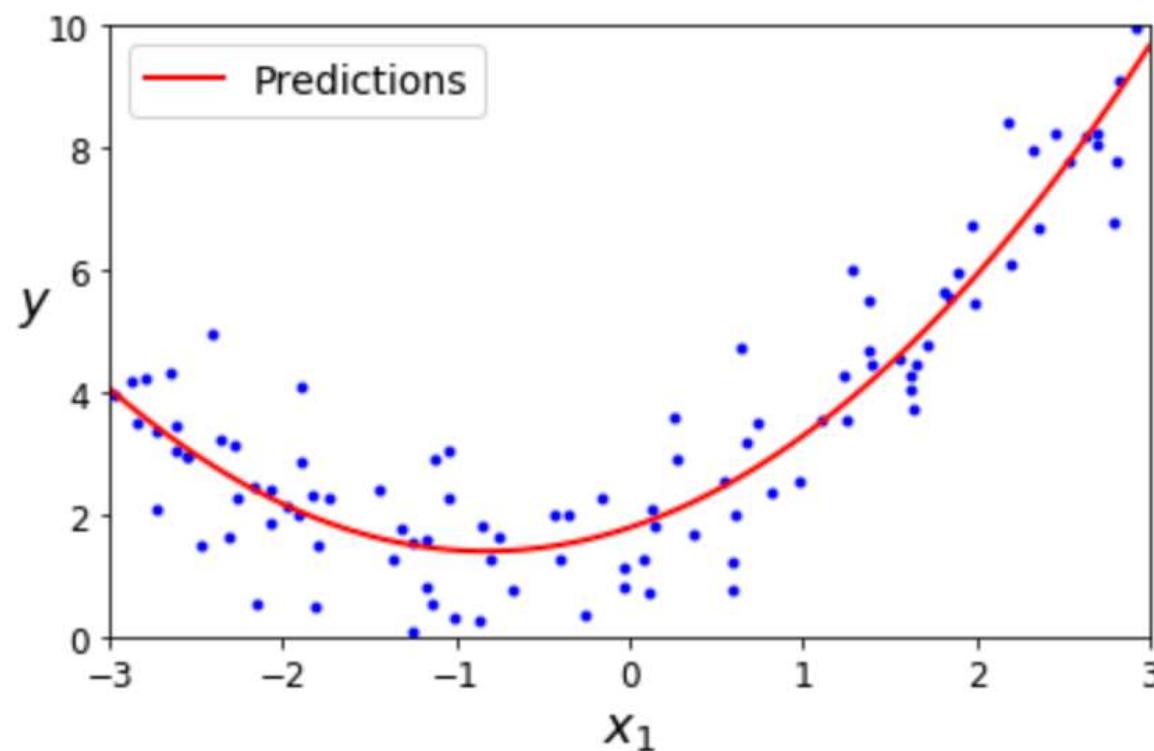
선형 회귀: 1차 선형 모델

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



다항 회귀: 2차 다항식 모델

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



사이킷런의 `PolynomialFeatures` 변환기

- 주어진 특성들의 거듭제곱과 특성들 사이의 곱셈을 실행하여 특성을 추가하는 기능 제공

`PolynomialFeatures(degree=d)`

- `degree=d`: 몇 차 다항식을 활용할지 지정하는 하이퍼파라미터
- 예제: $n = 3, d = 2$ 인 경우에 $(x_1 + x_2 + x_3)^2$ 의 항목에 해당하는 6 개 특성 추가

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_1 x_2 + \theta_5 x_1 x_3 + \theta_6 x_2 x_3 + \theta_7 x_1^2 + \theta_8 x_2^2 + \theta_9 x_3^2$$