

4장 모델 훈련 (1부)

주요 내용

- 선형 회귀
- 경사하강법
- 다항 회귀
- 학습 곡선
- 모델 규제
- 로지스틱 회귀

4.1 선형 회귀

선형 회귀 예제: 1인당 GDP와 삶의 만족도

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

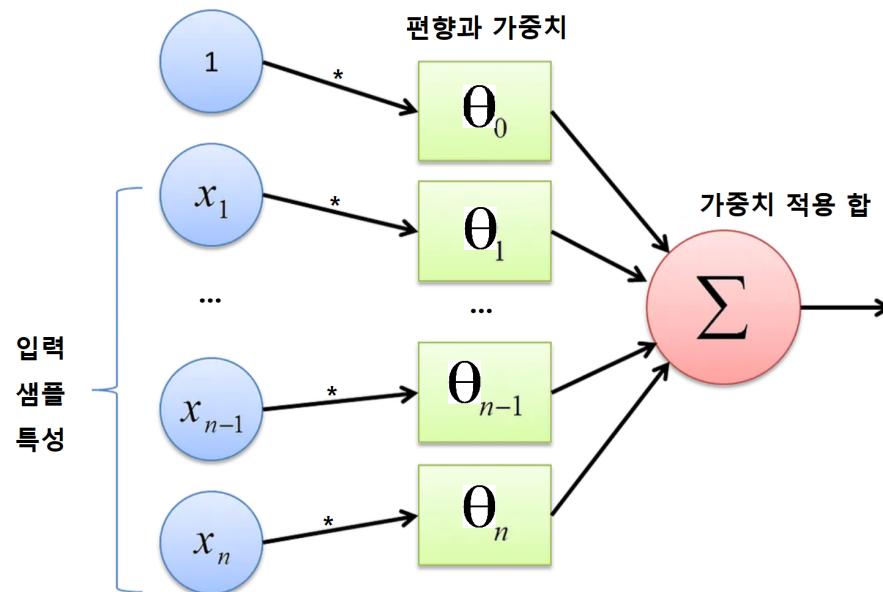
선형 회귀 예제: 캘리포니아 주택 가격 예측

- \hat{y} : 예측값
- x_i : 구역의 i 번째 특성값(위도, 경도, 중간소득, 가구당 인원 등)
- θ_0 : 편향
- $\theta_i: i (1 \leq i \leq 24)$ 번째 특성에 대한 (가중치) 파라미터

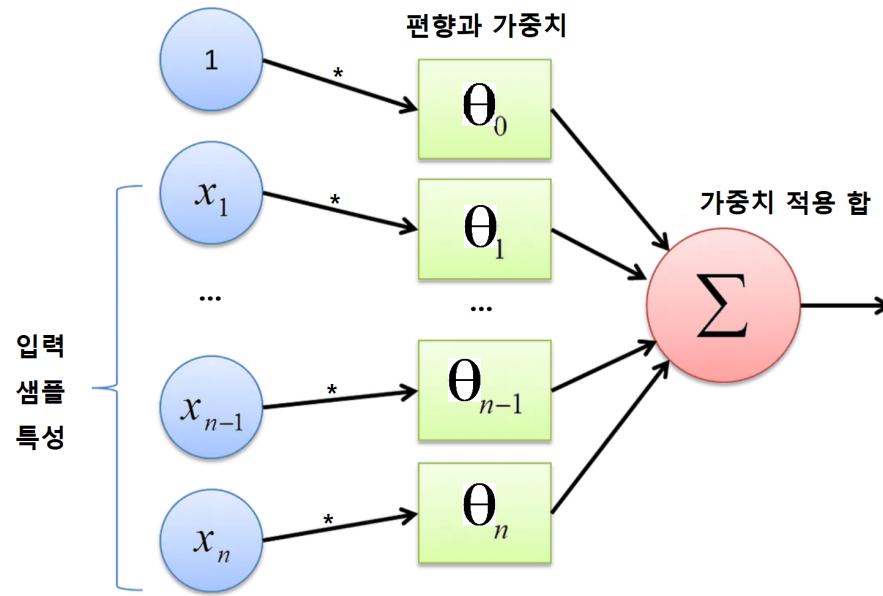
$$\hat{y} = \theta_0 + x_1 \cdot \theta_1 + \cdots + x_{24} \cdot \theta_{24}$$

선형 회귀 모델 함수

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



2D 어레이 표기법



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

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

선형 회귀 모델의 행렬 연산 표기법

\mathbf{X} 가 전체 입력 데이터셋, 즉 전체 훈련셋을 가리키는 $(m, 1+n)$ 모양의 2D 어레이, 즉 행렬이라 하자.

- m : 훈련셋의 크기.
- n : 특성 수

$$\mathbf{X} = \begin{bmatrix} [1, x_1^{(1)}, \dots, x_n^{(1)}] \\ \vdots \\ [1, x_1^{(m)}, \dots, x_n^{(m)}] \end{bmatrix}$$

예측값 일괄 계산

$$\begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_m \end{bmatrix} = \begin{bmatrix} [1, x_1^{(1)}, \dots, x_n^{(1)}] \\ \vdots \\ [1, x_1^{(m)}, \dots, x_n^{(m)}] \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

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

비용 함수: 평균 제곱 오차(MSE)

- MSE를 활용한 선형 회귀 모델 성능 평가

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

- 최종 목표: $\text{MSE}(\theta)$ 가 최소가 되도록 하는 θ 찾기

MSE(θ) 최소화

- 방식 1: 정규방정식 또는 특이값 분해(SVD) 활용
 - 드물지만 수학적으로 비용 함수를 최소화하는 θ 값을 직접 계산할 수 있는 경우 활용.
 - 계산복잡도가 $O(n^2)$ 이상인 행렬 연산을 수행해야 하기에 특성의 개수 n 이 매우 큰 경우 메모리 관리 및 시간 복잡도 문제로 인해 매우 비효율적임.
- 방식 2: 경사하강법
 - 특성이 매우 많거나 훈련 샘플이 너무 많아 메모리에 한꺼번에 담을 수 없을 때 적합
 - 선형 회귀 모델 훈련에 일반적으로 적용되는 기법

정규 방정식

- 비용 함수를 최소화 하는 θ 를 아래 정규 방정식 이용하여 계산 가능

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

SVD(특잇값 분해) 활용

- 행렬 연산과 역행렬 계산은 계산 복잡도가 $O(n^{2.4})$ 이상이고 항상 역행렬 계산이 가능한 것도 아님.
- 특잇값 분해를 활용하여 얻어지는 무어-펜로즈(Moore-Penrose) 유사 역행렬 \mathbf{X}^+ 계산이 보다 효율적임. 계산 복잡도는 $O(n^2)$.

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

4.2 경사 하강법

기본 아이디어

- 훈련셋이 크거나 특성 수가 많은 경우 정규 방정식과 무어-펜로즈 유사 역행렬 실전 활용 어려움.
- 훈련을 통해 절편과 편향 파라미터를 반복적으로 조금씩 조정
- 조정할 때 판단 기준: 비용 함수 크기 줄이기

경사 하강법 관련 주요 개념

하이퍼파라미터 hyperparameter

- 훈련시킬 모델을 지정할 때 사용되는 옵션: 학습률, 에포크, 허용오차, 배치 크기 등

파라미터

- 모델 훈련 중에 학습되는 파라미터
- 예제: 선형 회귀 모델의 편향과 가중치 파라미터

$$\theta = [\theta_0, \theta_1, \dots, \theta_n]$$

배치 크기

- 파라미터를 업데이트하기 위해 필요한 훈련 샘플의 개수

스텝과 스텝 크기

- 스텝_{step}: 배치 크기 만큼의 샘플에 대해 예측값을 계산한 후에 경사하강법을 적용하여 파라미터를 조정하는 단계
- 스텝 크기: 하나의 에포크 동안 실행되는 스텝 횟수
$$\text{스텝 크기} = (\text{훈련 샘플 수}) / (\text{배치 크기})$$
- 예제: 훈련셋의 크기가 1,000이고 배치 크기가 10이면, 에포크 당 100번의 스텝 실행

비용 함수

- 모델이 얼마나 나쁜지를 계산해주는 함수
- 예제: 선형 회귀 모델의 평균 제곱 오차(MSE). 단, m_b 는 배치 크기.

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

전역 최솟값

- 비용 함수가 가질 수 있는 최솟값
- 예제: 선형 회귀 모델의 평균 제곱 오차(MSE) 함수가 갖는 최솟값

최적 학습 모델

- 비용 함수를 최소화하는 파라미터를 사용하는 모델
- 최종적으로 훈련을 통해 얻고자 하는 훈련된 모델

비용 함수의 그레이디언트 벡터

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

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

학습률

- 훈련 과정에서 비용 함수의 파라미터(θ)를 얼만큼씩 조정할 것인지를 정하는 비율

허용오차 tolerance

- 비용 함수의 값이 허용오차보다 작아지면 훈련 종료

에포크 epoch

- 훈련셋에 포함된 모든 데이터를 대상으로 예측값을 한 번 계산하는 과정
- 이 과정동안 실행된 스텝 회수만큼 파라미터 업데이트

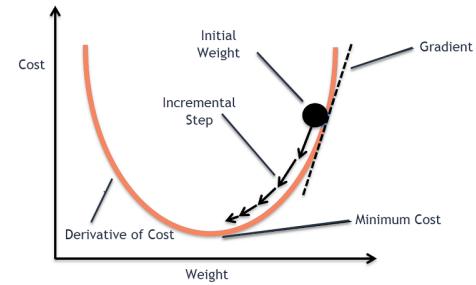
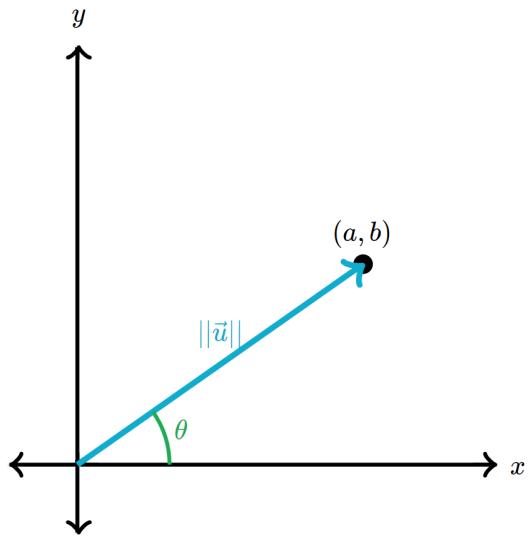
선형회귀 모델과 경사하강법

MSE를 비용 함수로 사용하는 경우 경사하강법은 다음 과정으로 이루어진다.

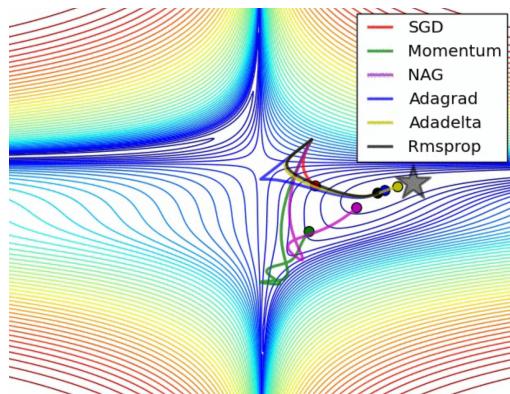
- (1) θ 를 임의의 값으로 지정한 후 훈련을 시작한다.
- (2) 아래 단계를 $\text{MSE}(\theta)$ 가 허용오차보다 적게 작아지는 단계까지 반복한다.
 - 배치 크기 m_b 만큼의 훈련 샘플을 이용하여 예측값 생성 후 $\text{MSE}(\theta)$ 계산.
 - 아래 점화식을 이용한 θ 업데이트

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

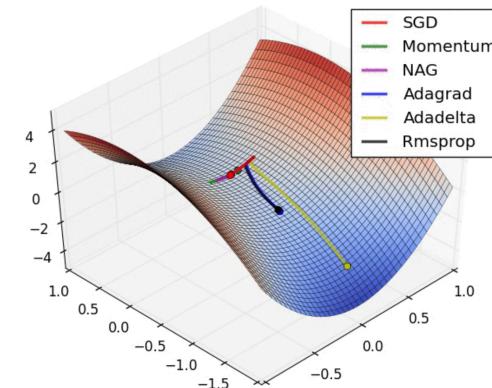
기울기 벡터의 방향과 크기



그레이디언트 벡터의 방향과 크기

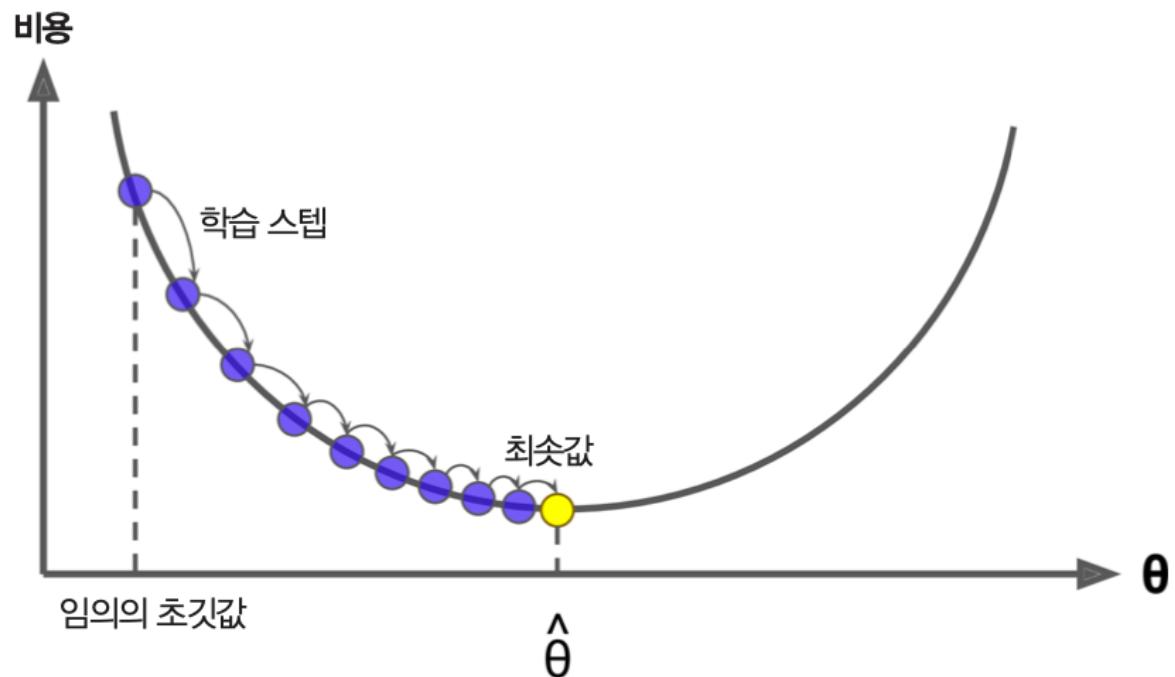


SGD optimization on loss surface contours

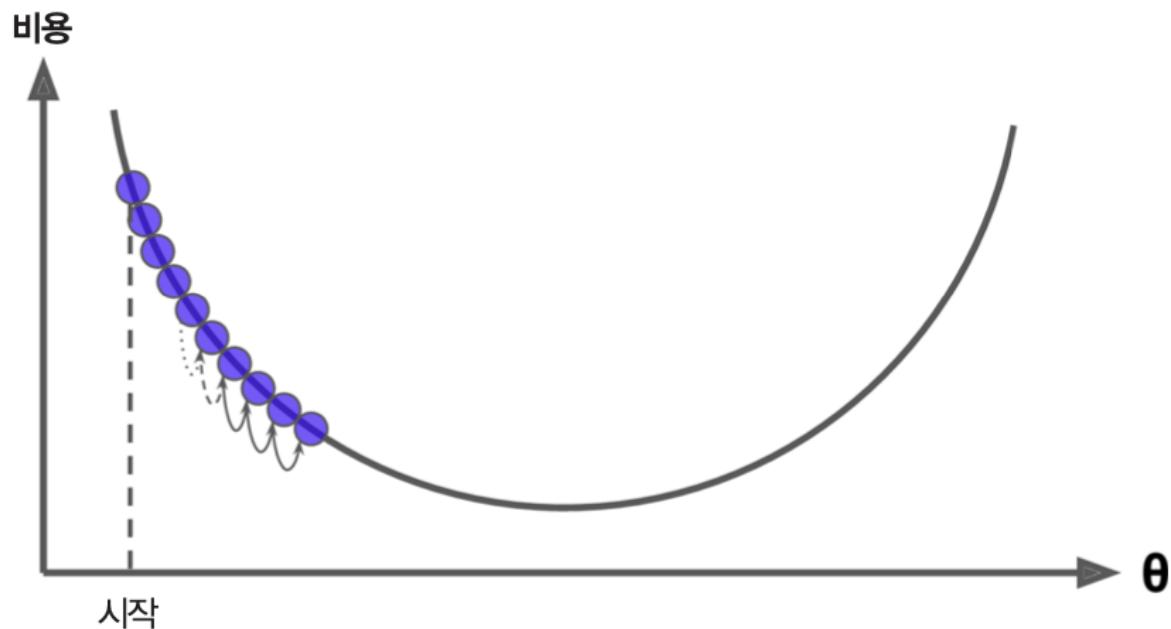


SGD optimization on saddle point

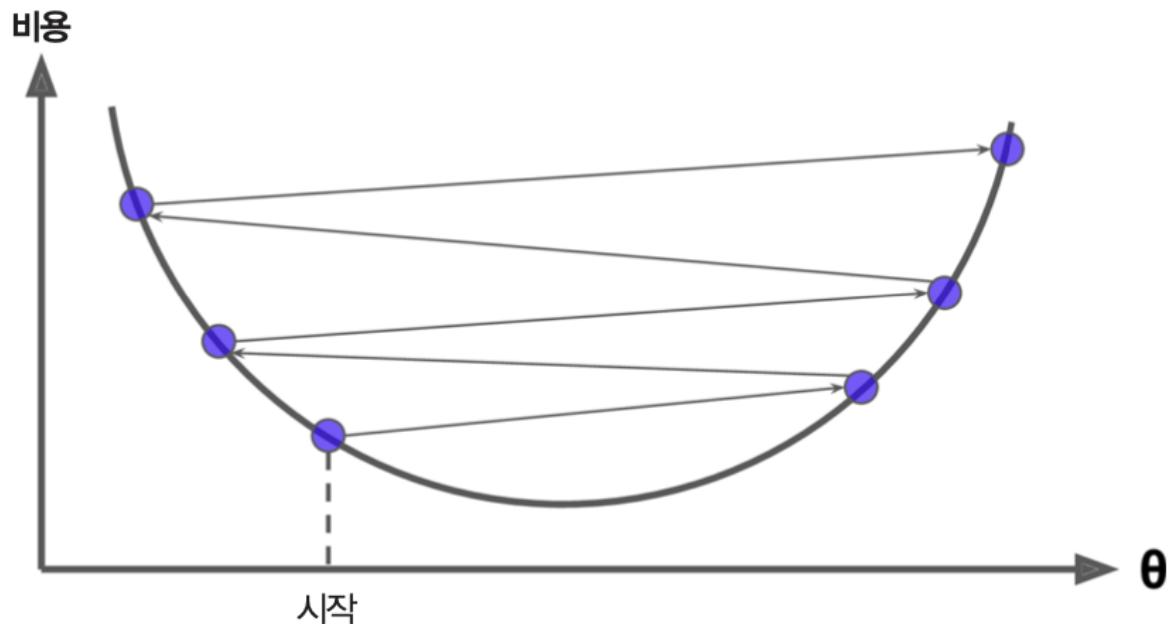
학습률의 중요성



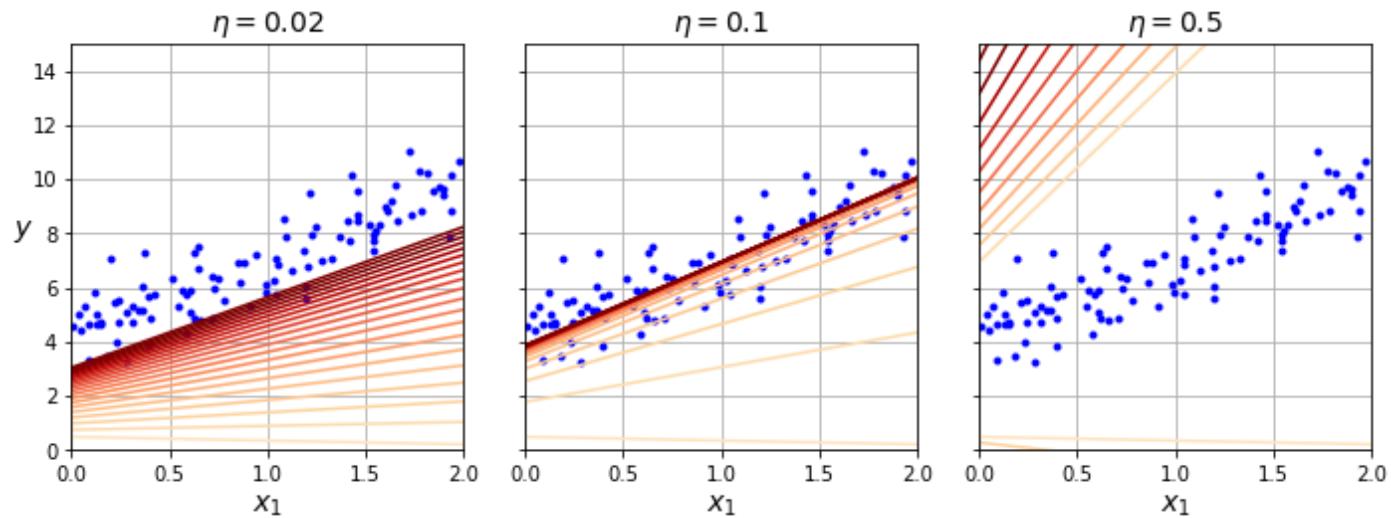
- 학습률이 너무 작은 경우: 비용 함수가 전역 최소값에 너무 느리게 수렴.



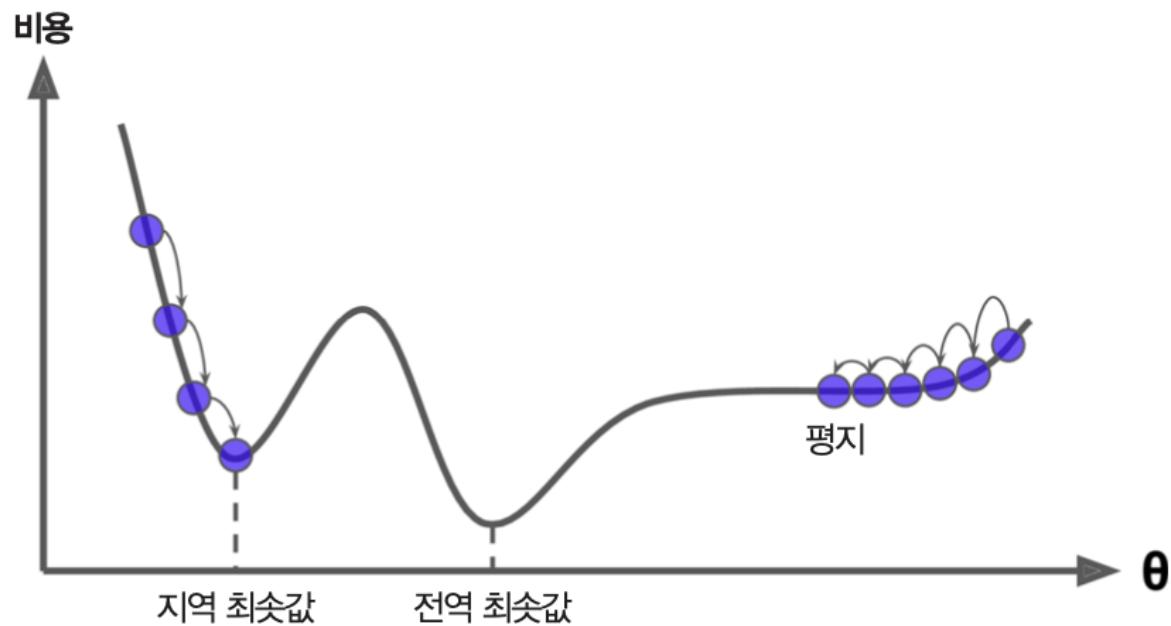
- 학습률이 너무 큰 경우: 비용 함수가 수렴하지 않음.



학습율과 경사 하강법의 관계

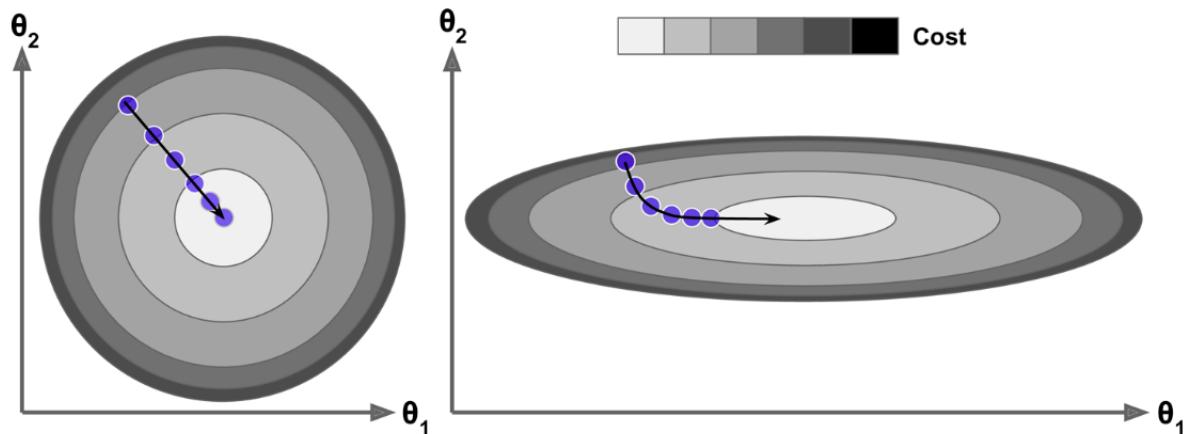


- (선형 회귀가 아닌 경우에) 시작점에 따라 지역 최솟값에 수렴하지 못할 수도 있음.



특성 스케일링의 중요성

- 왼편 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.



경사 하강법 종류

배치 경사 하강법

- 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 경사 하강법

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

미니배치 경사 하강법

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

배치 경사 하강법

- 에포크와 허용오차

- 에포크 수는 크게 설정한 후 허용오차를 지정하여 학습 시간 제한 필요. 이유는 포물선의 최솟점에 가까워질 수록 그레이디언트 벡터의 크기가 0에 수렴하기 때문임.
- 허용오차와 에포크 수는 서로 반비례의 관계임. 즉, 오차를 1/10로 줄이려면 에포크 수를 10배 늘려야함.

- 단점

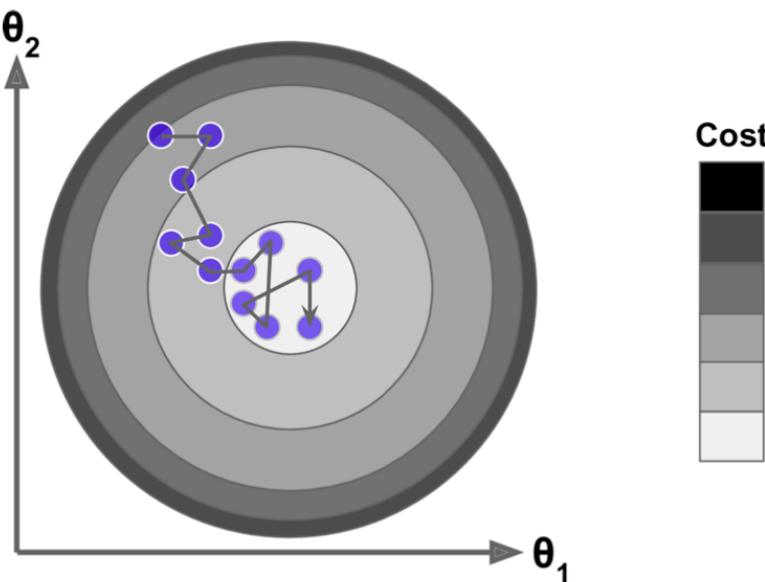
- 훈련 세트가 크면 그레이디언트를 계산하는 데에 많은 시간 필요
- 아주 많은 데이터를 저장해야 하는 메모리 문제도 발생 가능

- 주의사항

- 사이킷런은 배치 경사 하강법을 활용한 선형 회귀를 지원하지 않음.

확률적 경사 하강법

- 매우 큰 훈련 세트를 다룰 수 있음. 예를 들어, 외부 메모리(out-of-core) 학습을 활용 할 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

- 경사 하강법 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 경사 하강법

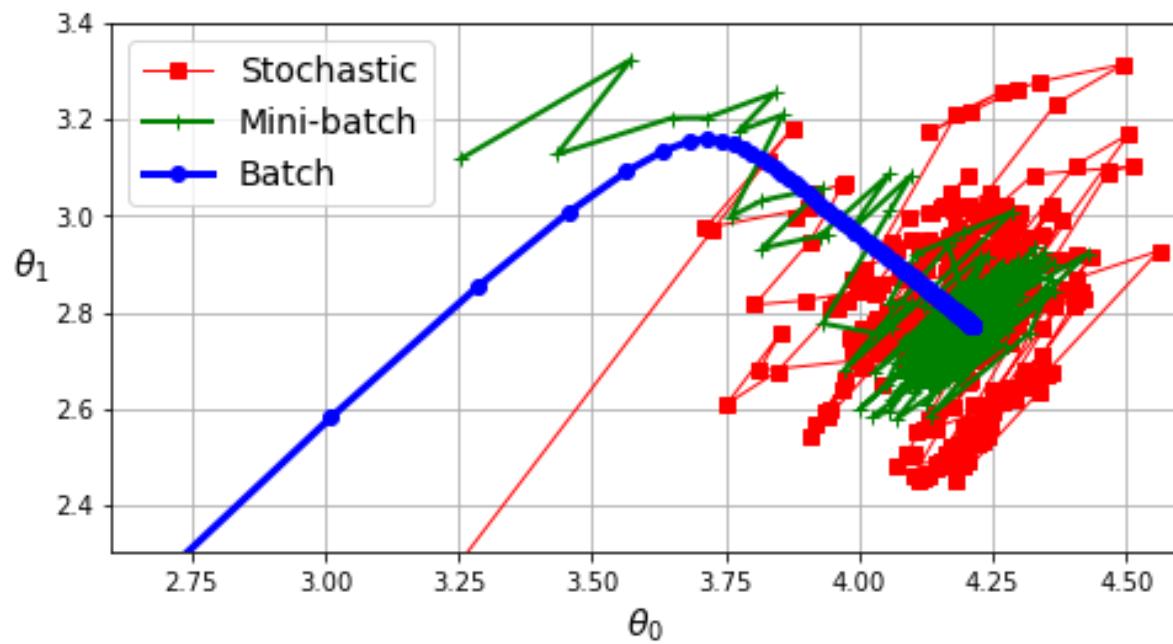
- 장점

- 배치 크기를 어느 정도 크게 하면 확률적 경사 하강법(SGD) 보다 파라미터의 움직임이 덜 불규칙적이 됨
- 반면에 배치 경사 하강법보다 빠르게 학습
- 학습 스케줄 잘 활용하면 최솟값에 수렴함.

- 단점

- SGD에 비해 지역 최솟값에 수렴할 위험도가 보다 커짐.

경사 하강법 비교



선형 회귀 알고리즘 비교

알고리즘	많은 샘플 수	외부 메모리 학습	많은 특성 수	하이퍼 파라미터 수	스케일 조정	사이킷런 지원
정규방정식	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

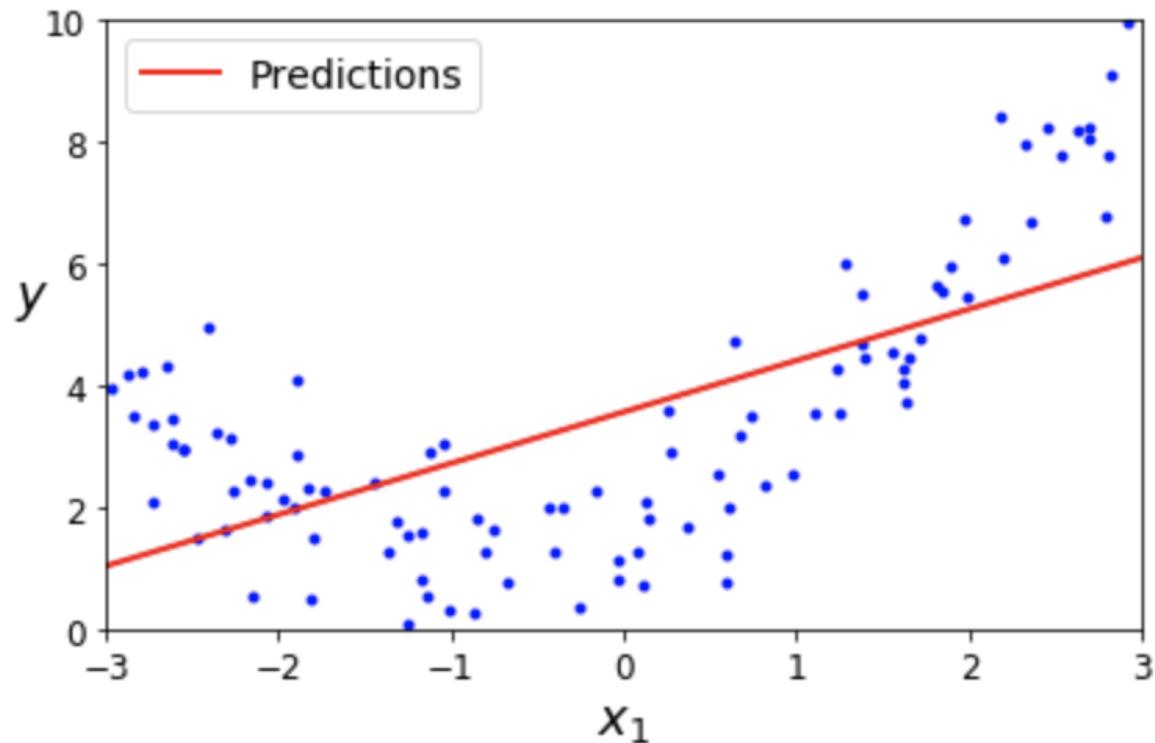
4.3 다행 회귀

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

선형 회귀 대 다항 회귀

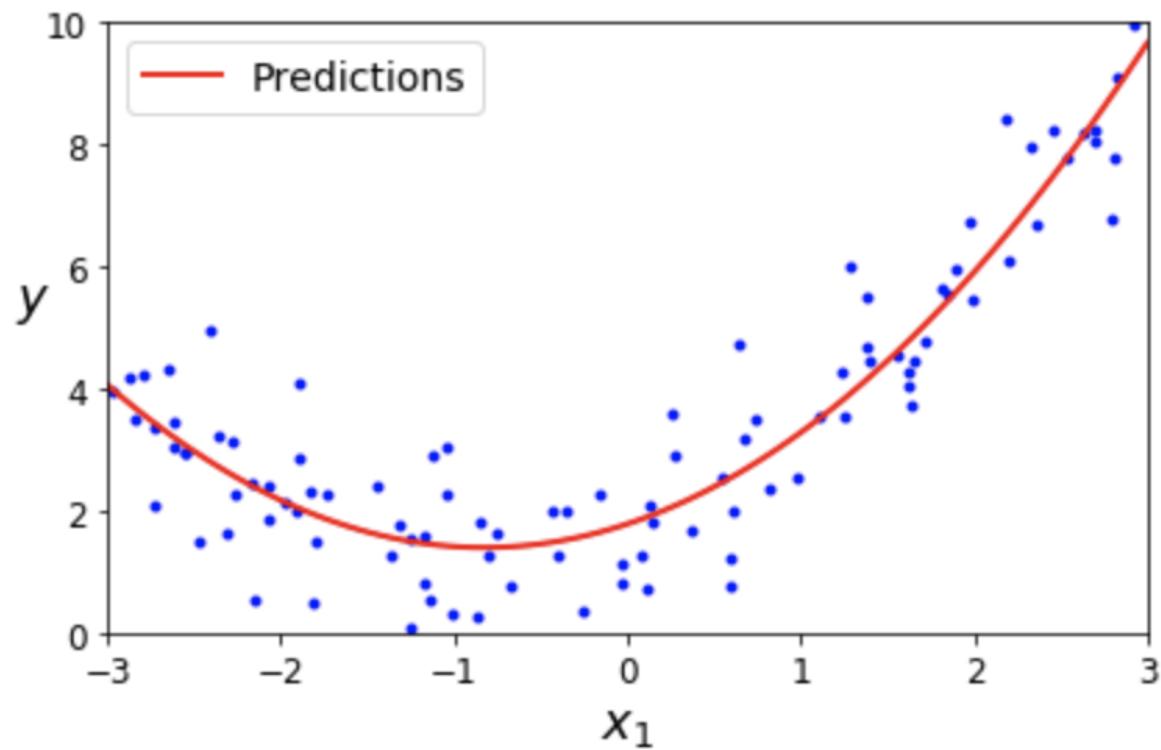
선형 회귀: 1차 선형 모델

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



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

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