# chapter 4
# 모델 훈련

## 4.1 선형 회귀

선형 회귀 모델의 훈련 방법 두 가지

- 직접 계산할 수 잇는 공식을 사용하여 비용 함수를 최소화하는 모델 파라미터를 해석적으로 구한다.
- 경사하강법(Gradient Descent, GD)이라 불리는 반복적인 최적화 방식을 사용하여 모델 파라미터를 조금씩 바꾸면서 비용 함수를 훈련 세트에 대해 최소화시킨다.

선형 모델은 입력 특성의 가중치 합과 **편향bias** (또는 **절편intercept**)이라는 상수를 더해 예측을 만든다.

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

$\hat{y}$ : 예측값
$n$ : 특성의 수
$x_i$ : $i$번째 특성값
$\theta_j$ : $j$번째 모델 파라미터(편향 $\theta_0$과 특성의 가중치 $\theta_1, \theta_2, \cdots \theta_n$을 포함한다.)

$$\hat{y} = h_\theta(\boldsymbol{x}) = \boldsymbol{\theta} \cdot \boldsymbol{x} = \boldsymbol{\theta}^{\boldsymbol{T}} \boldsymbol{x}$$

$$\boldsymbol{\theta} \cdot \boldsymbol{x} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$\boldsymbol{\theta}$ : 편향 $\theta_0$과 $\theta_1$에서 $\theta_n$까지의 특성 가중치를 담은 모델의 파라미터 벡터
$\boldsymbol{x}$ : $x_i$에서 $x_i$까지 담은 샘플의 특성 벡터. $x_0$는 항상 1이다.
$h_\theta$ : 모델 파라미터 $\boldsymbol{\theta}$를 사용한 가설hypothesis 함수

머신러닝에서는 한줄의 벡터의 경우 기본적으로 **열 벡터column vector**로 취급한다.

모델을 훈련시킨다는 것 : 모델이 훈려 세트에 가장 잘 맞도록 모델 파라미터를 설정하는 것.

### 선형 회귀 모델의 MSE 비용 함수

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

### 4.1.1 정규방정식 normal eqution

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{y}$$

$\widehat{\boldsymbol{\theta}}$ : 비용함수를 최소화하는 $\boldsymbol{\theta}$값
$\boldsymbol{y}$ : $y^{(1)}$부터 $y^{(m)}$까지 포함하는 타깃 벡터

해석적으로 최적의 해를 찾는 것을 보장한다.
매우 큰 데이터셋을 다룬다면 역행렬을 구할 때 계산 비용이 너무 많이 든다. 또는 샘플 데이터 행렬이 특이 행렬(singular matrix, 비가역 행렬)일 수 있다. 이것이 반복적인 방법이 선호되는 이유이다.

LinearRegression 클래스는 scipy.linalg.lstsq() 함수(**최소제곱 least square**에서 이름을 따옴)를 기반으로 한다.
이 함수는 $\hat{\theta} = X^{+}y$(**유사역행렬pseudoinverse** (**무어-펜로즈Moore-Penrose 역행렬))** 를 계산한다.

유사 역행렬 자체는 **특잇값 분해singular value decomposition(SVD)** 라 부르는 표준 행렬 분해 기법을 사용해 계산함
훈련세트 행렬 $X$를 3개의 행렬 곱셈$U\Sigma V^{T}$로 분해함 → 유사역행렬은 $X^{+} = U\Sigma^{+}V^{T}$로 계산됨. → $\Sigma^{+}$를 계산하기 위해 알고리즘이 $\Sigma$를 먼저 구하고 어떤 낮은 임곗값보다 작은 수를 모두 0으로 바꿈 → 0이 아닌 모든 값을 역수로 치환함 → 만들어진 행렬을 전치

정규방정식을 계산하는 것보다 효율적, $m < n$이거나 어떤 특성이 중복되어 $X^{T}X$의 역행렬이 없어도(=특이행렬) 유사 역행렬은 항상 구할 수 있다.

### 4.1.2 계산복잡도 computational complexity

정규방정식은 $(n+1) \times (n+1)$크기가 되는 $X^{T}X$의 역행렬을 계산한다.($n$은 특성 수)
역행렬을 계산하는 **계산 복잡도computational complexity**는 일반적으로 $O(n^{2.4})$에서 $O(n^{3})$사이이다.
특성 수가 두 배로 늘어나면 계산시간이 $2^{2.4} = 5.3$에서 $2^{3} = 8$배로 증가한다.

scikit-learn의 LinearRegression 클래스가 사용하는 **SVD(특잇값 분해 singular value decomposition)** 방법은 약 $O(n^{2})$
특성의 개수가 두 배로 늘어나면 계산 시간이 약 4배로 증가한다.

정규방정식과 SVD 방법 모두 특성 수가 많아지면 매우 느려짐. 하지만 훈련 세트의 샘플 수에 대해서는 선형적으로 증가.(둘다 $O(m)$)

→$X$를 (샘플 수, 특성 수) = $m \times n$행렬이라 할 때 $X^{T}X$는 $(n \times m)(m \times n) = (n \times n)$크기의 행렬이 되므로 샘플 수($m$)는 계산의 복잡도를 증가시키지 않고 점곱의 양만 선형적으로 증가시킨다.

## 경사 하강법 gradient descent(GD)

**경사하강법 gradient descent(GD) (gradient : 비용 함수의 미분값)**

비용 함수를 최소화하기 위해 방복해서 파라미터를 조정해가는 것.
파라미터 벡터 $\theta$에 대해 비용함수의 현재 **그래디언트gradient**를 계산 → 그래디언트가 감소하는 방향으로 진행
→ 그래디언트가 0이 되면 최솟값에 도달한 것.
$\theta$를 임의의 값으로 시작해서 **(무작위 초기화 random initialization)** 한 번에 조금씩 비용 함수(ex. MSE)가 감소되는 방향으로 진행하여 알고리즘이 최솟값에 수렴할 때까지 점진적으로 향상

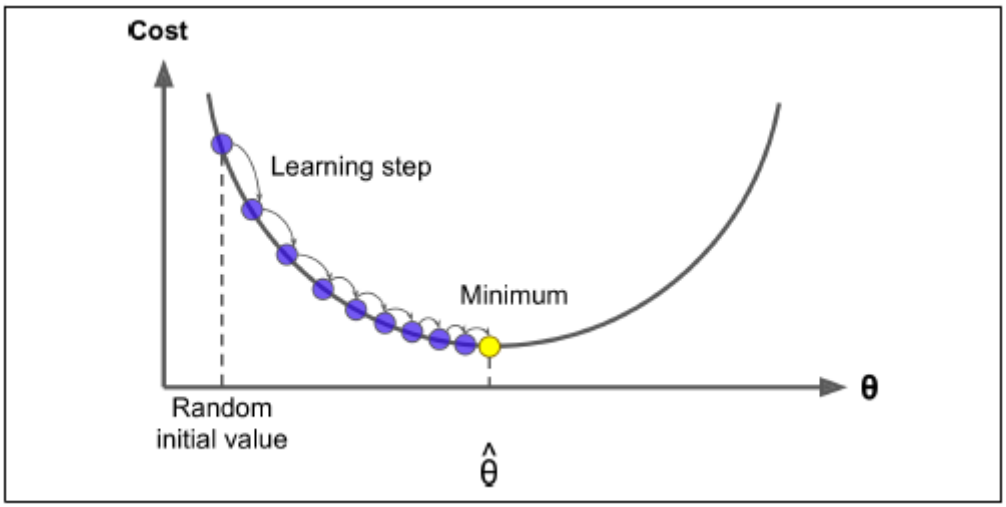
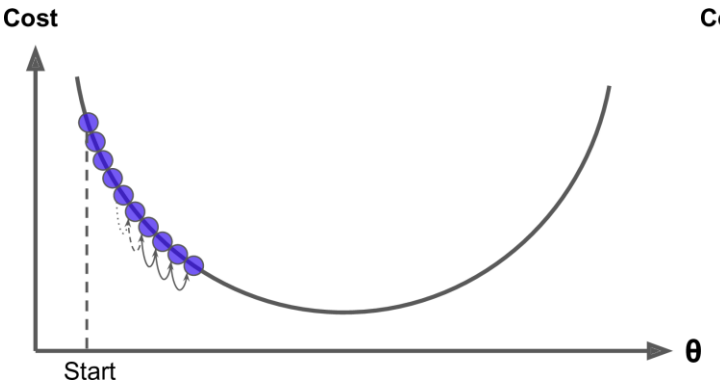
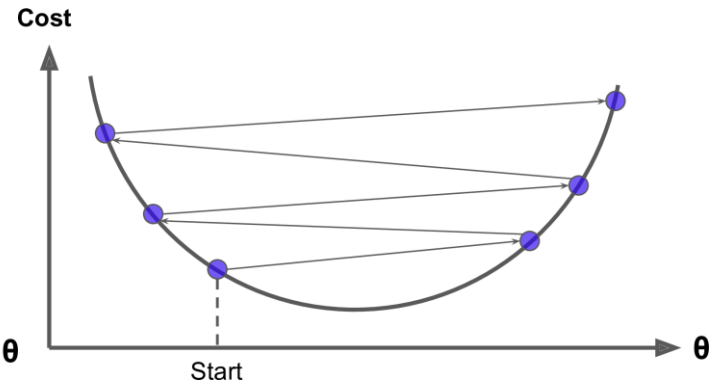



*Figure 4-3. Gradient Descent*

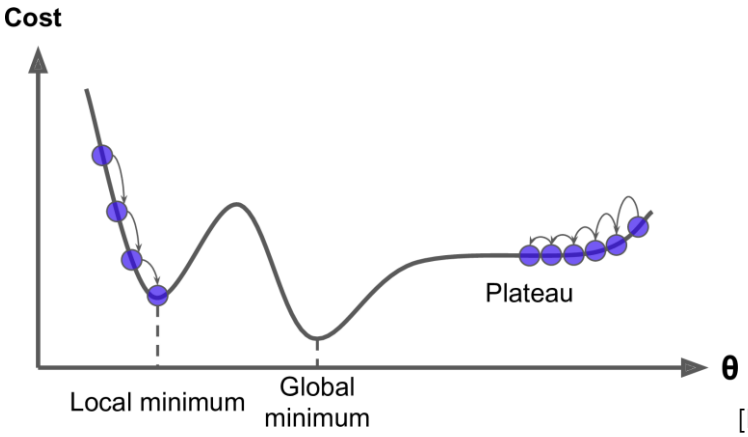
경사 하강법에서 중요한 파라미터는 스텝step의 크기로, 학습률learning rate 하이퍼 파라미터로 결정된다.
학습률이 작다 → 반복을 많이 해야 함 → 시간이 오래 걸림.
학습률이 크다 → 골짜기를 가로질러 반대편으로 건너뛰게됨 → 더 높은 곳으로 올라갈 수도 있음 →적절한 해법을 찾지 못하게 함




학습률이 너무 작을 때




학습률이 너무 클 때




[Figure 4-6]

**경사 하강법의 문제점(위 그림을 예시로 들어)**

알고리즘이 왼쪽에서 시작하면 **전역 최솟값**global minimum보다 덜 좋은 **지역 최솟값**local minimum에 수렴한다.
알고리즘이 오른쪽에서 시작하면 평탄한 지형을 지나기 위해 시간이 오래 걸리고 일찍 멈추게 되어 전역 최솟값에 도달하지 못한다.

선형 회귀를 위한 MSE 비용 함수는 **볼록 함수**convex function이다. 이는 다음을 뜻한다.
1. 지역 최솟값이 없고 하나의 전역 최솟값만 있다는 뜻이다.

2. 연속된 함수이고 기울기가 갑자기 변하지 않는다.

위의 두 사실로부터 경사 하강법이 전역 최솟값에 가깝게 접근할 수 있다는 것을 보장한다.

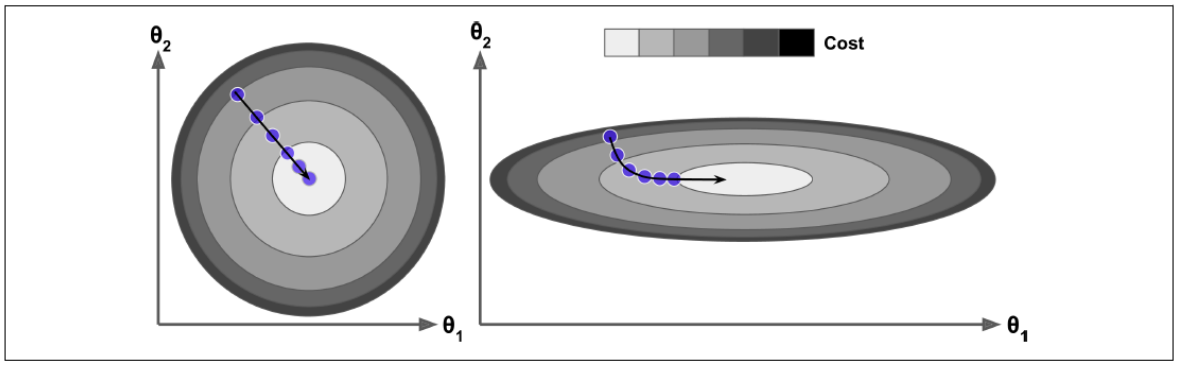



*Figure 4-7. Gradient Descent with and without feature scaling*

왼쪽 : 특성 1과 특성 2의 스케일이 같은 훈련세트 (특성 스케일 적용)
오른쪽 : 특성 1이 특성 2보다 더 작은 훈련 세트
(특성 1이 더 작기 때문에 비용 함수에 영향을 주기 위해서는 $\theta_1$이 더 크게 바뀌어야 한다.
→ $\theta_1$축을 따라서 길쭉한 모양이 된다.
왼쪽의 경사하강법 : 최솟값으로 곧잘 진행하고 있어 빠르게 도달한다.
오른쪽의 경사하강법 : 처음엔 전역 최솟값의 방향에 거의 직각으로 향하다가 평편한 골짜기를 길게 돌아서 나간다. 결국 최솟값에 도달하겠지만 시간이 오래 걸림
→경사 하강법을 사용할 때는 모든 특성이 같은 스케일을 갖도록 만들어야 한다. 그렇지 않으면 수렴하는 데 훨씬 오래 걸린다.

[Figure 4-7] 모델 훈련이 (훈련 세트에서) 비용 함수를 최소화하는 모델 파라미터의 조합을 찾는 일임을 설명해준다. 이를 모델의 **파라미터 공간**parameter space에서 찾는다고 말한다.
모델의 파라미터개수 증가 → 공간의 차원이 커진다 → 검색이 더 어려워짐
→ 선형 회귀의 경우 비용 함수가 볼록 함수이기 때문에 전역 최솟값은 맨 아래에 있을 것이다.

### 4.2.1 배치 경사 하강법 batch gradient descent

경사하강법 구현 시 각 모델 파라미터 $\theta_j$에 대해 비용 함수의 그래디언트를 계산해야한다.
→이를 **편도함수**partial derivative라고 한다.

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

→파라미터 $\theta_j$에 대한 비용 함수의 편도함수 $\frac{\partial}{\partial\theta_j}MSE(\theta)$

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

→비용함수의 그래디언트 벡터 $\nabla_\theta MSE(\boldsymbol{\theta})$는 비용 함수의 편도함수를 모두 담고있다.

이 공식은 매 경사하강법 스텝에서 전체 훈련 세트 $\boldsymbol{X}$에 대해 계산한다.
그래서 이 알고리즘을 **배치 경사 하강법**batch gradient descent이라고 한다. 즉 매 스텝에서 훈련 데이터 전체를 사용한다.
→ 훈련 세트가 매우 클 경우 매우 느리다.
그러나 경사 하강법은 특성 수에 민감하지 않다. 수십만개의 특성에서 선형 회귀를 훈련시키려면 정규방정식, SVD분해보다 경사 하강법을 사용하는 것이 훨씬 빠름

그래디언트 벡터가 구해지면 반대 방향으로 가야함 (= $\theta$에서 $\nabla_\theta MSE(\boldsymbol{\theta})$를 빼야 한다.)
→내려가는 스텝의 크기를 결정하기 위해 그래디언트 벡터에 학습률$\eta$를 곱한다.

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

적절한 학습률 $\eta$ 찾기 : 그리드 탐색을 사용. 반복 횟수를 제한하여 수렴까지 너무 오래 걸리지 않게 한다.
반복횟수 지정(n_iterations) :
너무 작음 → 최적점에 도달 전에 알고리즘이 멈춤
너무 큼 → 모델 파라미터가 더는 변하지 않는 동안 시간을 낭비
해결책 : 반복 횟수를 아주 크게 지정하고 그래디언트 벡터가 아주 작아지면(=벡터의 노름이 어떤 값 $\varepsilon$ (**허용오차**tolerance)보다 작아지면) 경사 하강법이 (거의) 최솟값에 도달한 것이므로 알고리즘을 중지한다.

### 수렴율
비용 함수의 모양에 따라 달라지겠지만 $\varepsilon$ 범위 안에서 최적의 솔루션에 도달하기 위해서는 $O(1/\varepsilon)$의 반복이 걸릴 수 있다. (= 더 정확한 최솟값을 얻기 위해 허용 오차 $\varepsilon$을 1/10로 줄이면 알고리즘의 반복은 10배 늘어남)

## 확률적 경사 하강법 Stochastic Gradient Descent
확률적 경사 하강법은 매 스텝에서 한 개의 샘플을 무작위로 선택하고 그 하나의 샘플에 대한 그래디언트를 계산한다.
→한 번에 하나의 샘플을 처리하면 알고리즘이 훨씬 빠름
→매우 큰 세트도 훈련시킬 수 있다(외부 메모리 학습 알고리즘으로 구현할 수 있다.)

반면 확률적이므로 배치 경사 하강법보다 훨씬 불안정
비용 함수가 최솟값에 다다를 때까지 부드럽게 감소하지 않고 위아래로 요동치며 평균적으로 감소
알고리즘이 멈출 때 좋은 파라미터가 구해지겠지만 최적치는 아니다.

비용 함수가 매우 불규칙할 때 ([Figure 4-6]처럼) 알고리즘이 지역 최솟값을 건너뛰도록 도와주므로 확률적 경사 하강법이 배치 경사하강법보다 전역 최솟값을 찾을 가능성이 더 높다.

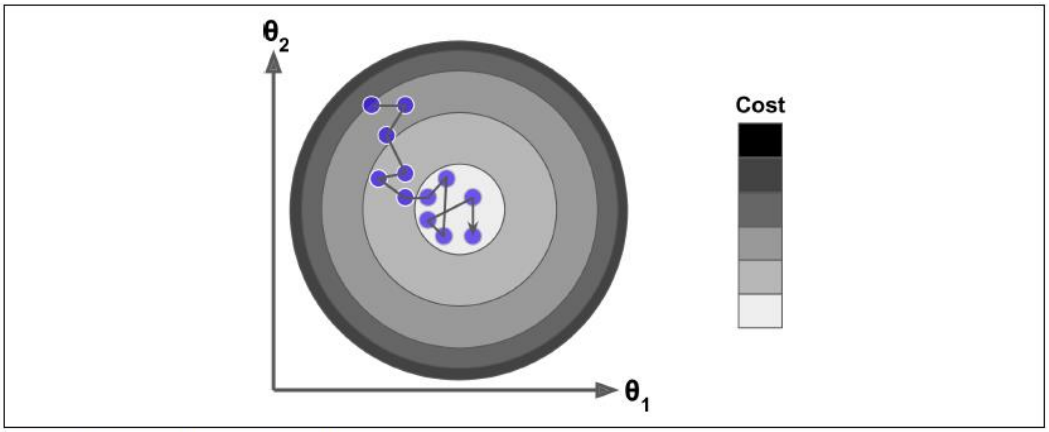


*Figure 4-9. Stochastic Gradient Descent*

무작위성은 지역 최솟값에서 탈출시켜줘서 좋지만 알고리즘은 전역 최솟값에 다다르지 못하게 한다는 점에서 좋지 않음.
딜레마 해결방법 : 학습률을 점진적으로 감소시킴, 시작할 때 학습률을 크게 하고(지역 최솟값에 빠지지 않게 한다), 점차 작게 줄여서 알고리즘이 전역 최솟값에 도달하게 한다.→**담금질 기법simulated annealing** 알고리즘과 유사
매 반복에서 학습률을 결정하는 함수를 **학습 스케줄learning schedule**(=**학습률 스케줄learning rate schedule**)이라고 부른다.

SGD에서 종종 고정된 학습률 $\eta$를 시간이 지남에 따라 적응적인 학습률로 대체한다.
ex)

$$\frac{c_1}{[number\ of\ iterations] + c_2}$$

→ $c_1$과 $c_2$는 상수이다. SGD는 전역 최솟값에 도달하지 못하지만 매우 가까운 지역에 근접한다. 적응적 학습률을 사용하면 최솟값에 더욱 가깝게 다가갈 수 있다.

학습률이 너무 빨리 줄어듬 : 지역 최솟값에 갇히거나 최솟값까지 가는 중간에 멈춰버릴 수 있음
학습률이 너무 천천히 줄어듬 : 오랫동안 최솟값 주변을 맴돌거나 훈련을 너무 일찍 중지해서 지역 최솟값에 머무를 수 있음.

확률적 경사 하강법 구현시
일반적으로 한 반복에서 m(훈련 세트에 있는 샘플 수)번 되풀이되고 각 반복을 **에포크epoch**라고 한다.

확률적 경사 하강법을 사용할 때 훈련 샘플이 **IID independent and identically distributed** (독립항등분포)를 만족해야 평균적으로 파라미터가 전역 최적점을 향해 진행한다고 보장할 수 있다.
훈련하는 동안 샘플을 무작위로 섞음으로 IID를 만족시킬 수 있다.
만약 샘플을 섞지 않은 채로 사용하면 확률적 경사 하강법이 먼저 한 레이블에 최적화하고 그 다음 두 번째 레이블을 최적화하는 식으로 진행된다. 결국 이 모델은 최적점에 가깝게 도달하지 못할 것이다.

## 미니배치 경사 하강법 mini-batch gradient descent

미니배치라 부르는 임의의 작은 샘플 세트에 대해 그래디언트를 계산한다.
→SGD에 비해 mini-batch GD의 주요 장점은 행렬연산에 최적화된 하드웨어, 특히 GPU를 사용해서 얻는 성능 향상이다.

미니배치를 어느 정도 크게 한다 → 파라미터 공간에서 SGD보다 덜 불규칙하게 움직인다. → 결국 mini-batch GD이 SGD보다 최솟값에 더 가까이 도달하게 될 것이다.
하지만 지역 최솟값에서 빠져나오기는 더 힘들지도 모른다.(선형 회귀와 같지 않고 지역 최솟값이 문제가 되는 경우)

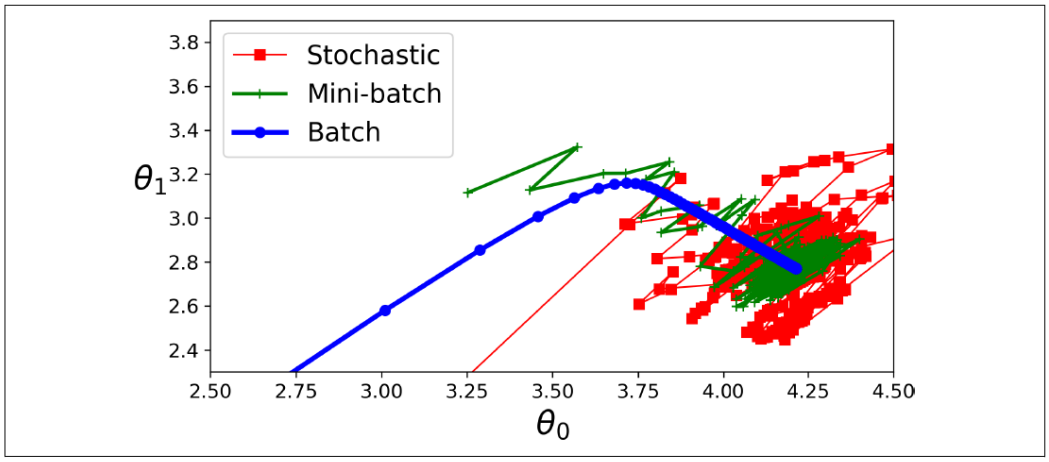
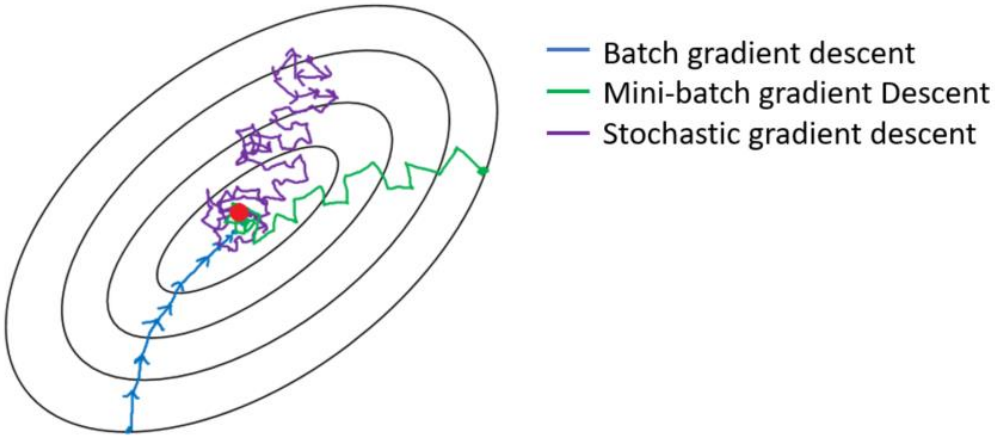



*Figure 4-11. Gradient Descent paths in parameter space*

BGD : 경로가 실제로 최솟값에서 멈춤 & 스텝에서 많은 시간이 소요됨

Mini-batch GD, SGD : 최솟값의 근처에서 멤돌고 있다 & 적절한 학습 스케줄을 사용하면 최솟값에 도달한다.



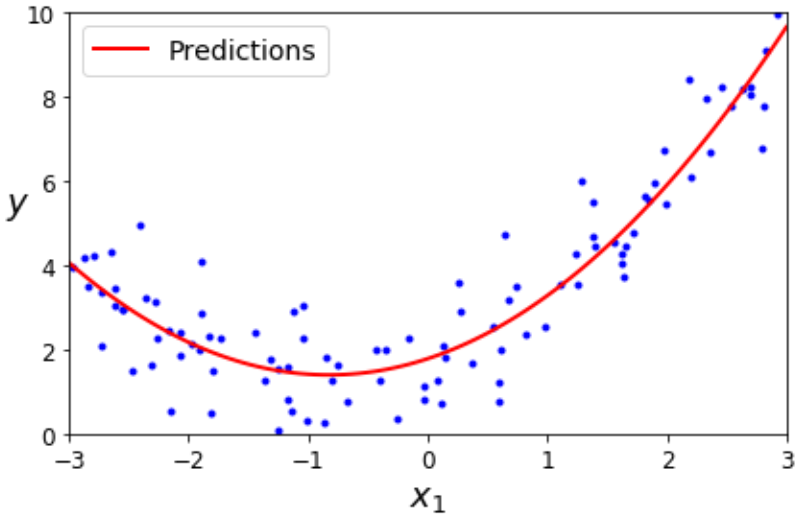

선형 회귀를 사용한 알고리즘 비교(m : 훈련 샘플 수, n : 특성 수)

| 알고리즘 | m이 클 때 | 외부 메모리 학습 지원 | n이 클 때 | 하이퍼 파라미터수 | 스케일 조정 필요 | 사이킷런 |
|---|---|---|---|---|---|---|
| **정규방정식** | 빠름 | No | 느림 | 0 | No | N/A |
| **SVD** | 빠름 | No | 느림 | 0 | No | LinearRegression |
| **BGD** | 느림 | No | 빠름 | 2 | Yes | SGDRegressor |
| **SGD** | 빠름 | Yes | 빠름 | ≥2 | Yes | SGDRegressor |
| **mini-batch GD** | 빠름 | Yes | 빠름 | ≥2 | Yes | SGDRegressor |

### 다항 회귀 polynomial regression

선형 회귀보다 파라미터가 많아서 훈련 데이터에 과대적합되기 더 쉽다.

각 특성의 거듭제곱을 새로운 특성으로 추가하고, 이 확장된 특성을 포함한 데이터셋에 선형 모델을 훈련시키는 기법. 비선형 데이터를 학습하는 데 선형 모델을 사용한다.




특성이 여러 개일 때 다항 회귀는 이 특성 사이의 관계를 찾을 수 있다.

PolynomialFeatures가 주어진 차수까지 특성간의 모든 교차항을 추가하기 때문

ex) 두 개의 특성 a, b가 있을 때 degree=3으로 PolynomialFeatures를 적용하면 $a^2, a^3, b^2, b^3$ 뿐만 아니라 $ab, a^2b, ab^2$도 특성으로 추가한다.

→PolynomialFeatures(degree=d)는 특성이 n개인 배열을 특성이 $\frac{(n+d)!}{d!n!}$개인 배열로 변환한다.

## 4.4 학습곡선

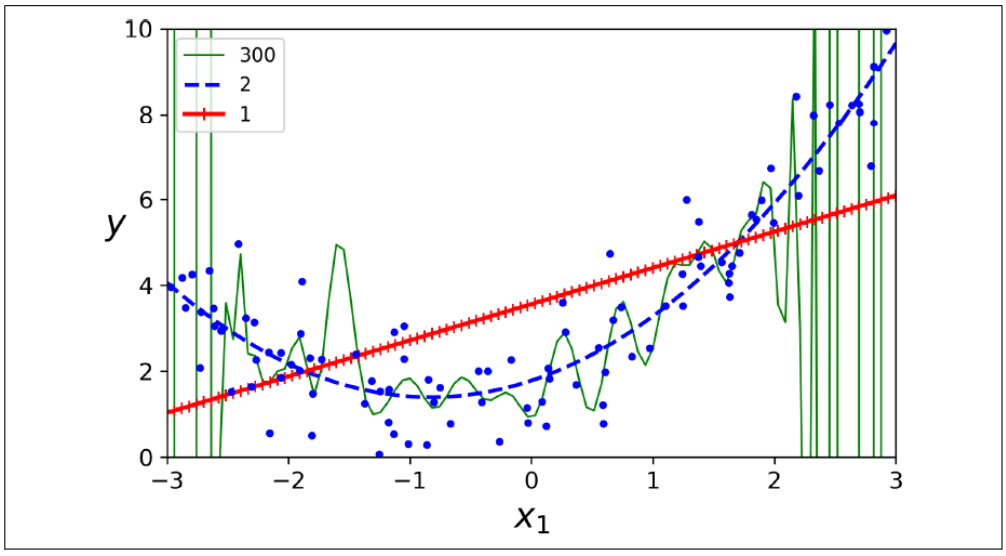



*Figure 4-14. High-degree Polynomial Regression*

주어진 훈련 데이터셋에 비해 모델이 너무 복잡하면, 즉 모델의 자유도나 모델 파라미터가 너무 많으면 모델이 훈련 데이터에 과대적합되고 처음 본 데이터에 잘 일반화되지 못하는 경향이 있다.

어떻게 모델이 데이터에 과대적합 또는 과소적합되었는지 알 수 있을까?

**학습곡선**을 살펴본다.

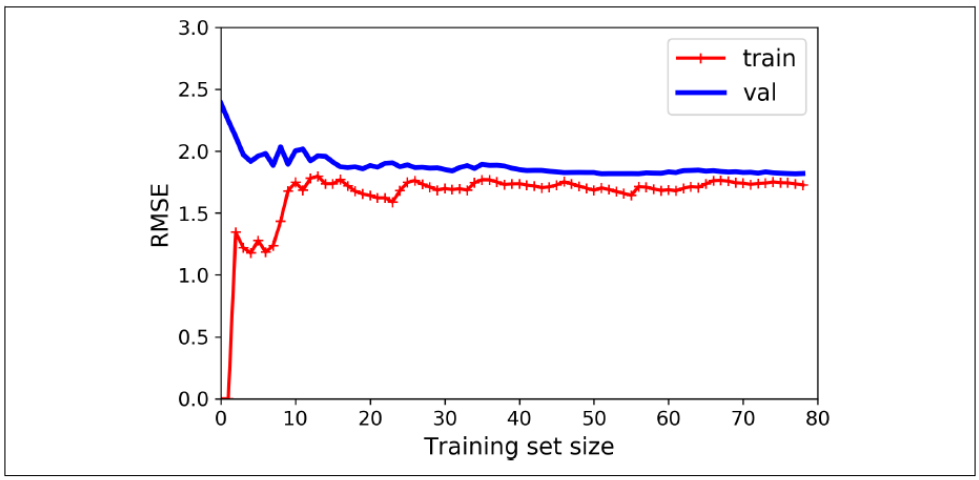



*Figure 4-15. Learning curves*

단순 선형 회귀 모델(직선)의 학습곡선

훈련 데이터의 성능 : 그래프가 0에서 시작하므로 훈련 세트에 한 개 혹은 두 개의 샘플이 있을 땐 모델이 완벽하게 작동한다. 하지만 샘플이 추가되면서 잡음도 있고 비선형이기 때문에 모델이 훈련 데이터를 완벽히 학습하는 것이 불가능해짐. 그래서 곡선이 어느 정도 평편해질 때까지 오차가 계속 상승한다. 이 위치에서는 훈련 세트에 샘플이 추가되어도 평균오차가 크게 나아지거나 나빠지지 않는다.

검증 데이터의 성능 : 모델이 적은 수의 훈련 샘플로 훈련될 때는 제대로 일반화될 수 없어서 검증 오차가 초기에 매우 크다. 모델에 훈련 샘플이 추가됨에 따라 학습이 되고 검증 오차가 서서히 감소한다. 하지만 선형 회귀의 직선은 데이터를 잘 모델링할 수 없으므로 오차의 감소가 완만해져서 훈련 세트의 그래프와 가까워진다.

전형적인 과소적합 모델의 모습, 두 곡선이 수평한 구간을 만들고 꽤 높은 오차에서 매우 가까이 근접해 있다.
→훈련 샘플을 더 추가해도 효과가 없다, 더 복잡한 모델을 사용하거나 더 나은 특성을 선택해야 한다.

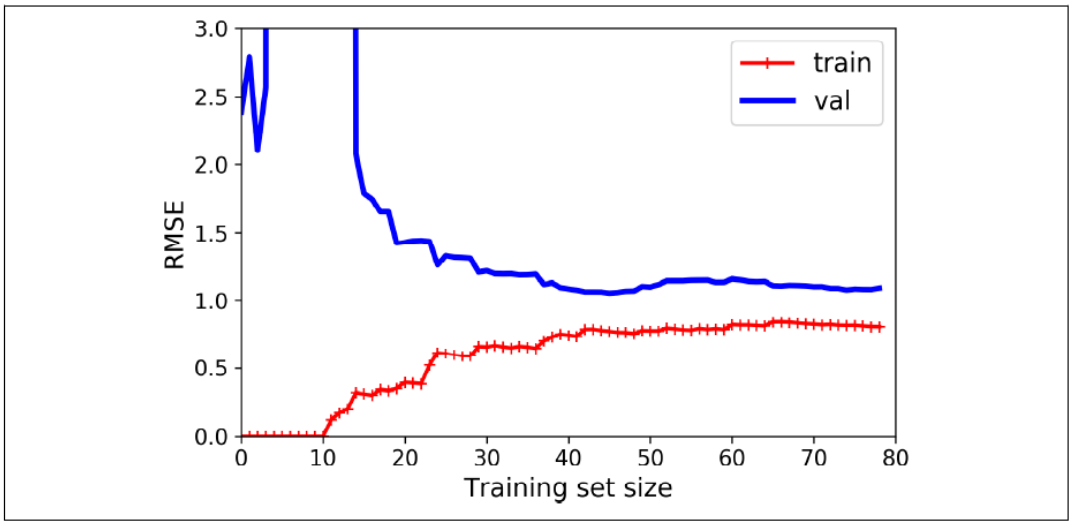



*Figure 4-16. Learning curves for the polynomial model*

다항 회귀 모델의 학습 곡선

훈련 데이터의 오차가 선형 회귀 모델보다 훨씬 낮다.
두 곡선 사이에 공간이 있다 → 훈련 데이터에서의 모델 성능이 검증 데이터에서보다 훨씬 낫다 → 과대적합 모델의 특징 → 더 큰 훈련 세트를 사용하면 두 곡선이 점점 가까워진다(과대적합 모델을 개선하는 한가지 방법)

## 편향/분산 트레이드 오프

모델의 일반화 오차는 세 가지 다른 종류의 오차의 합으로 표현할 수 있다.

**편향 bias**

일반화 오차 중에서 편향은 잘못된 가정으로 인한 것이다.

ex) 데이터가 실제로는 2차인데 선형으로 가정하는 경우

→편향이 큰 모델은 훈련 데이터에 과소적합되기 쉽다.

**분산 variance**

분산은 훈련 데이터에 있는 작은 변동에 모델이 과도하게 민감하기 때문에 나타남.

자유도가 높은 모델(ex. 고차 다항 회귀 모델)이 높은 분산을 가지기 쉬워 훈련 데이터에 과대적합되는 경향이 있다.

**줄일 수 없는 오차 irreducible error**

줄일 수 없는 오차는 데이터 자체에 있는 잡음 때문에 발생한다. 데이터에서 잡음을 제거하는 것이 오차를 줄이는 유일한 방법이다(ex. 데이터 소스를 고치거나 이상치를 감지해 제거한다)

모델의 복잡도가 커짐 → 분산이 늘어나고 편향이 줄어듬

모델의 복잡도가 줄어듬 → 분산이 작아지고 편향이 커짐

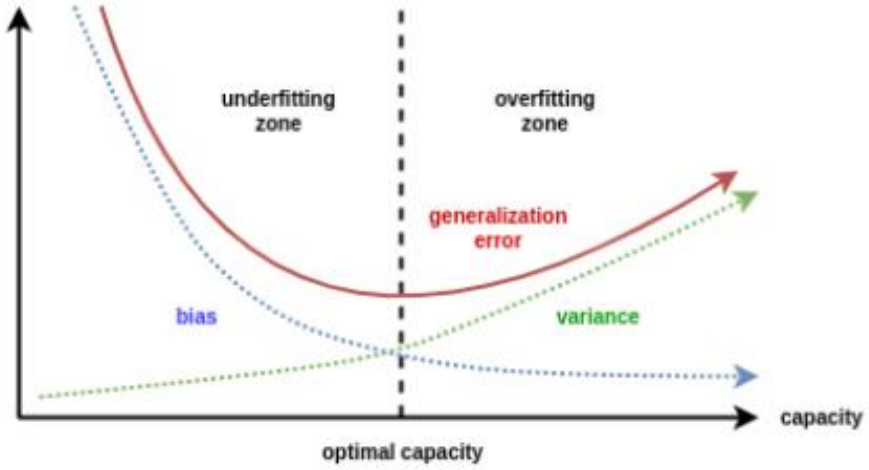
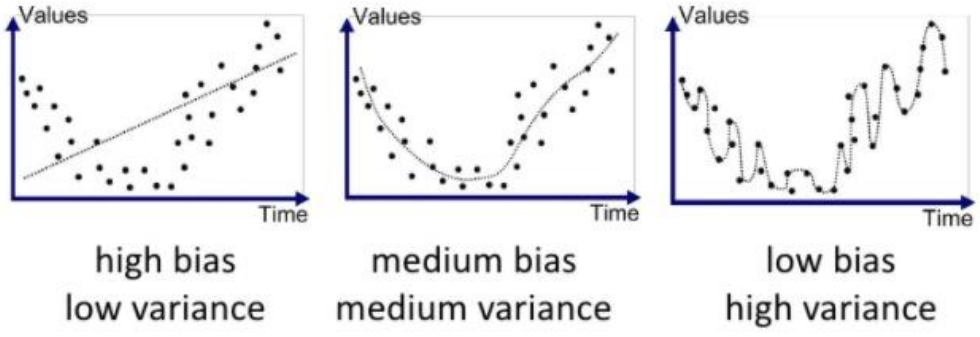



*Figure 1: Bias-Variance Tradeoff as a Function of Model Capacity*




## 4.5 규제가 있는 선형 모델

과대적합을 감소시키는 좋은 방법은 모델을 규제하는 것이다.(즉 모델을 제한한다) 자유도를 줄이면 데이터에 과대적합되기 더 어려워진다.

다항 회귀 모델을 규제하는 간단한 방법은 다항식의 차수를 감소시키는 것이다.

선형 회귀 모델에서는 보통 모델의 가중치를 제한함으로써 규제를 가한다.

→'릿지' 회귀, '라쏘' 회귀, 엘라스틱넷

### 4.5.1 릿지 회귀 ridge regression

**릿지 회귀**(또는 **티호노프Tikhonov 규제**)는 규제가 추가된 선형 회귀 버전이다.
규제항 $\alpha \sum_{i=1}^{n} \theta_i^2$이 비용 함수에 추가된다. 이는 학습 알고리즘을 데이터에 맞추는 것 뿐만 아니라 모델의 가중치가 가능한 한 작게 유지되도록 노력한다.
규제항은 훈련하는 동안에만 비용함수에 추가된다. 모델의 훈련이 끝나면 모델의 성능을 규제가 없는 성능 지표로 평가한다.

일반적으로 훈련시 사용되는 비용 함수와 테스트에서 사용되는 성능 지표는 다르다.
훈련시 사용되는 비용함수는 최적화를 위해 미분 가능해야 하기 때문이다.
반면 테스트에 사용되는 성능 지표는 최종 목표에 가능한 한 가까워야 한다.

릿지 회귀의 비용 함수

$$J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \frac{\alpha}{2}\sum_{i=1}^{n} \theta_i^2 = MSE(\boldsymbol{\theta}) + \frac{\alpha}{2}(\|\boldsymbol{w}\|_2)^2$$

하이퍼파라미터 $\alpha$ : 모델을 얼마나 많이 규제할지 조절한다
$\alpha = 0$ : 릿지 회귀는 선형 회귀와 같아진다
$\alpha$가 아주 큼 : 모든 가중치가 거의 0에 가까워지고 결국 데이터의 평균을 지나는 수평선이 된다.
편향 $\theta_0$는 규제되지 않는다.($\Sigma$가 $i = 1$에서 시작한다.)
$\boldsymbol{w}$를 특성의 가중치 벡터($\theta_1$에서 $\theta_n$)이라고 정의하면 규제항은 $1/2\,(\|w\|_2)^2$와 같다.
경사 하강법에 적용하려면 $MSE$ 그래디언트 벡터에 $\alpha\boldsymbol{w}$를 더하면 된다.

비용함수의 그래디언트 벡터

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

원소가 $n$개인 벡터$v$의 $l_k$노름은 $\|\boldsymbol{v}\|_k = (|v_0|^k + |v_1|^k + \cdots + |v_n|^k)^{\frac{1}{k}}$로 정의합니다.
RMSE계산은 $\boldsymbol{l_2}$노름에 해당하며 $\|\cdot\|_2$ 또는 그냥 $\|\cdot\|$로 표시한다. → 유클리디안 노름(Euclidean norm)

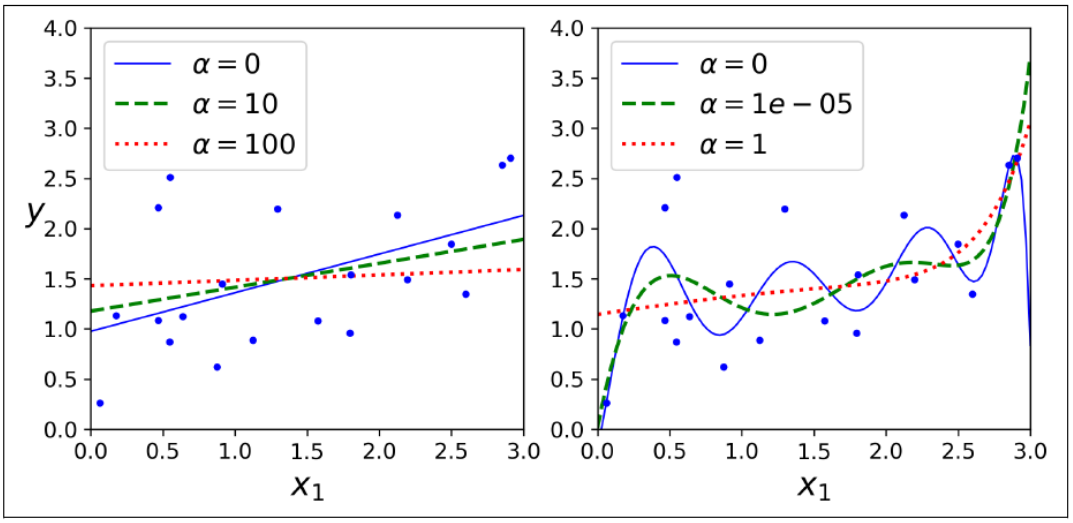



*Figure 4-17. Ridge Regression*

오른쪽 : 데이터를 확장하고(PolynomialFeatures) 스케일을 조정(StandardScaler)한 후 릿지 모델을 적용
→릿지 규제를 사용한 다항 회귀, $\alpha$를 증가시킬수록 직선에 가까워진다.
→모델의 분산은 줄고 편향은 커지게 된다.

릿지 회귀의 정규방정식

$$\hat{\theta} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + \alpha\boldsymbol{A}\right)^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

($\boldsymbol{A}$는 편향에 해당하는 맨 왼쪽 위의 원소가 0인 $(n+1)\times(n+1)$의 단위행렬이다.)
→편향에 해당하는 $\theta_0$은 규제에 포함되지 않으므로 단위행렬의 주대각선 첫 번째 원소가 0이 되어야 한다.

### 4.5.2 라쏘 least absolute shrinkage and selection operator(Lasso) 회귀

라쏘회귀는 선형 회귀의 또 다른 규제된 버전
릿지 회귀처럼 비용 함수에 규제항을 더하지만 가중치 벡터의 $l_1$노름을 사용한다.

라쏘 회귀의 비용 함수

$$J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha\sum_{i=1}^{n}|\theta_i|$$

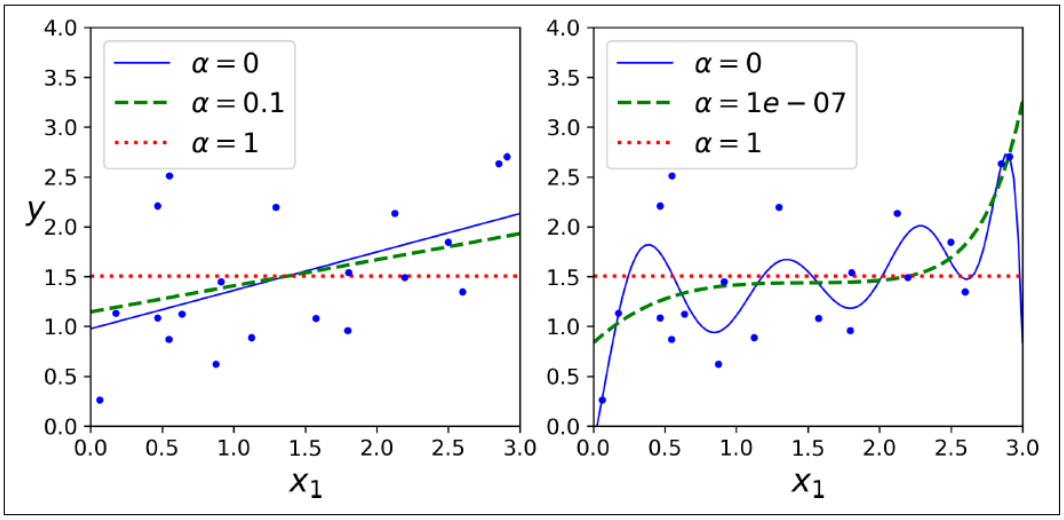
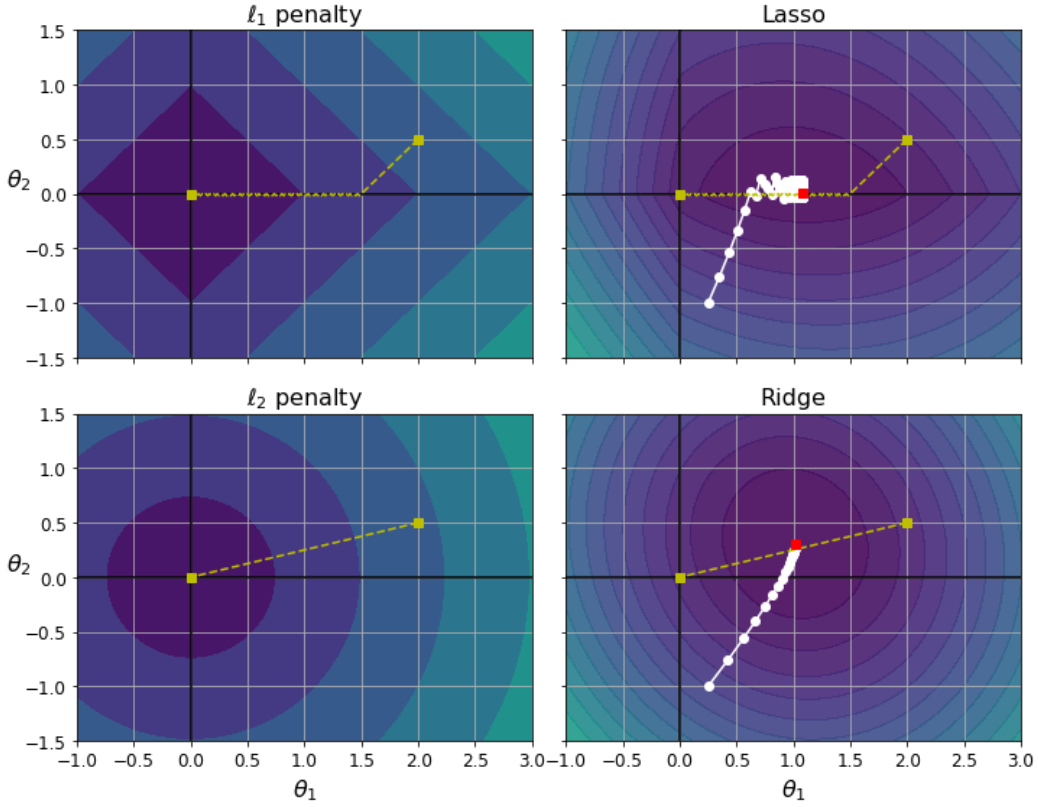



*Figure 4-18. Lasso Regression*

라쏘 회귀의 중요한 특징 : 덜 중요한 특성의 가중치를 제거하려고 한다는 점이다. (=가중치가 0이 된다.)
ex) [Figure 4-18]의 오른쪽 그래프에서 파선($\alpha = 10^{-7}$)은 3차방정식처럼 보인다. 차수가 높은 다항 특성의 가중치가 모두 0이 되었다.
→라쏘 회귀는 자동으로 특성 선택을 하고 **희소 모델**sparse model을 만든다.(=0이 아닌 특성의 가중치가 적다)




두 축 : 모델 파라미터 / 배경의 등고선 : 각기 다른 손실 함수

왼쪽 위 그래프의 등고선 :
$l_1$손실 ($|\theta_1| + |\theta_2|$)을 나타낸다. 축에 가까워지면서 선형적으로 줄어든다.
ex) $\theta_1 = 2$, $\theta_2 = 0.5$로 초기화 하고 경사 하강법을 실행 → 두 파라미터가 동일하게 감소(노란 점선)
→ $\theta_2$가 먼저 0에 도달 → 경사 하강법이 $\theta_1 = 0$에 도달할 때까지 축을 따라 내려감($l_1$의 그래디언트는 0에서 정의되지 않기 때문에 진동이 조금 있다, 이 지점에서 그래디언트는 -1 또는 1)

오른쪽 위 그래프의 등고선 :
라쏘 손실 함수를 나타낸다.($l_1$손실을 더한 MSE 손실 함수)
하얀 작은 원 : 경사 하강법이 $\theta_1 = 0.25$, $\theta_2 = -1$로 초기화된 모델 파라미터를 최적화 하는 과정을 보여준다.
$\theta_2 = 0$으로 빠르게 줄어들고 그 다음 축을 따라 진동하면서 전역 최적점(빨간 사각형)에 도달한다.
$\alpha$ 증가 : 전역 최적점이 노란 점선을 따라 왼쪽으로 이동 (가중치가 0이 되기 때문에)
$\alpha$ 감소 : 전역 최적점이 오른쪽으로 이동한다.
→아래 두개의 그래프도 동일하지만 $l_2$ 페널티를 사용한다.

왼쪽 아래 그래프 :
$l_2$ 손실은 원점에 가까울수록 줄어든다. → 경사하강법이 원점까지 직선 경로를 따라간다.

오른쪽 아래 그래프의 등고선 :

릿지 회귀의 비용 함수를 나타냄($l_2$손실을 더한 MSE 손실 함수).
라쏘와 다른 점
1. 파라미터가 전역 최적점에 가까워질수록 그래디언트가 작아진다. → 경사 하강법이 자동으로 느려지고 수렴이 도움이 됨(=진동이 없다)
2. $\alpha$를 증가시킬수록 최적의 파라미터(빨간 사각형)가 원점에 더 가까워진다.(하지만 완전히 0이 되지는 않는다.)
→라쏘를 사용할 때 경사 하강법이 최적점 근처에서 진동하는 것을 막으려면 훈련하는 동안 점진적으로 학습률을 감소시켜야 한다.(여전히 최적점 근처에서 진동하겠지만 스텝이 갈수록 작아지므로 수렴할 것이다.)

라쏘의 비용함수는 $\theta_i = 0(i = 1, 2, \cdots, n$일 때)에서 미분 가능하지 않다.
$\theta_i = 0$일 때 서브그래디언트 벡터subgradient vector g를 사용하면 경사하강법을 적용하는 데 문제가 없다.

경사 하강법을 위해 라쏘 비용 함수에 사용할 수 있는 서브그래디언트 벡터

$$g(\boldsymbol{\theta}, J) = \nabla_\theta MSE(\boldsymbol{\theta}) + \alpha \begin{pmatrix} sign(\theta_1) \\ sign(\theta_2) \\ \vdots \\ sign(\theta_n) \end{pmatrix} \quad \text{여기서 } sign(\theta_n) = \begin{cases} -1 & if\ \theta_1 < 0 \\ 0 & if\ \theta_1 = 0 \\ +1 & if\ \theta_1 > 0 \end{cases}$$

## 엘라스틱넷 elstic net

릿지 회귀와 라쏘 회귀를 절충한 모델
규제항은 릿지와 회귀의 규제항을 단순히 더해서 사용하며, 혼합 정도는 혼합 비율 $r$을 사용해 조절한다.
$r = 0$이면 릿지 회귀와 같고, $r = 1$이면 라쏘 회귀와 같다.

엘라스틱넷 비용 함수

$$J(\theta) = MSE(\theta) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2} \alpha \sum_{1=1}^{n} \theta_i^2$$

### 모델의 선택

적어도 규제가 약간 있는 것이 대부분의 경우에 좋음 → 평범한 선형 회귀는 피한다.
릿지가 기본이 되지만 특성이 몇 개뿐이라고 의심되면 라쏘나 엘라스틱넷이 낫다.(불필요 가중치를 0으로 만듬)
(특성 수 > 샘플 수) 또는 (특성 몇 개가 강하게 연관) → 라쏘보다는 엘라스틱넷 선호

엘라스틱넷 선호 이유 : 보통 라쏘가 문제를 일으키기 때문
라쏘는 특성 수가 샘플 수(n)보다 많으면 최대 n개의 특성을 선택
여러 특성이 강하게 연관되어 있으면 이들 중 임의의 특성 하나를 선택한다.

## 4.5.4 조기 종료 early stopping

검증 에러가 최솟값에 도달하면 바로 훈련을 중지시키는 것

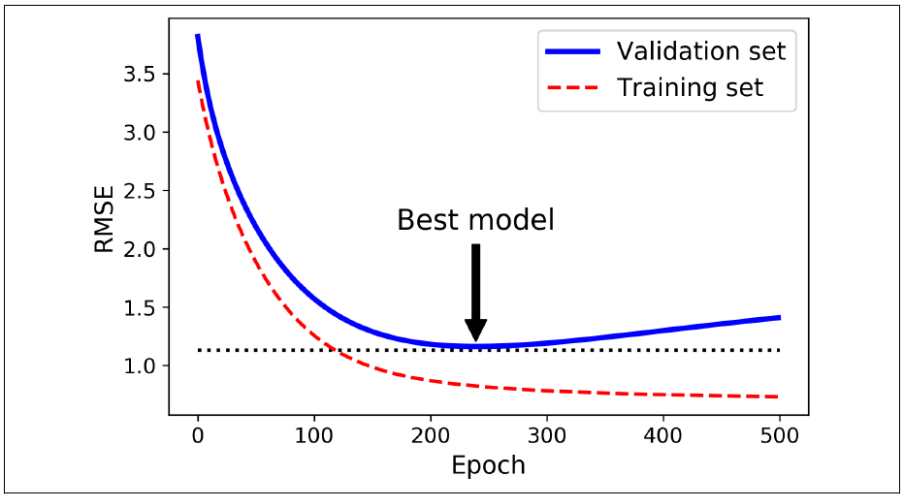



*Figure 4-20. Early stopping regularization*

에포크가 진행됨에 따라 알고리즘이 점차 학습되어 훈련 세트에 대한 RMSE와 검증 세트에 대한 RMSE가 줄어든다. 그러나 잠시 후 감소하던 검증 에러가 멈추었다가 다시 상승한다(=모델이 훈련 데이터에 과대적합되기 시작함)
→조기 종료는 검증에러가 최소에 도달하는 즉시 훈련을 멈추는 것이다.

SGD or mini-batch GD → 곡선이 그리 매끄럽지 않아 최솟값에 도달했는지 확인이 어려울 수 있다.
해결책 : 검증 에러가 일정 시간 동안 최솟값보다 클 때(모델이 더 나이지지 않는다고 확신이 들 때) 학습을 멈추고 검증 에러가 최소였을 때의 모델 파라미터로 되돌린다.

## 4.6 로지스틱 회귀 logistic regression 또는 로짓 회귀 logit regression

분류에서 사용되는 회귀 알고리즘, 샘플이 특정 클래스에 속할 확률을 추정하는 데 널리 사용.
추정확률이 50%가 넘으면 샘플이 해당 클래스에 속한다고 예측(=레이블이 '1'인 **양성클래스positive class**)
아니면 클래스에 속하지 않는다고 예측(=레이블이 '0'인 **음성클래스negative class**)
→이를 이진 분류기라고 한다.

### 4.6.1 확률 추정

로지스틱 회귀 모델은 입력 특성의 가중치 합을 계산하고 편향을 더한다.
→선형 회귀처럼 바로 결과를 출력하지 않고 결괏값의 로지스틱logistic을 출력한다.

로지스틱 회귀 모델의 확률 추정(벡터 표현식) / 로지스틱은 $\sigma(\cdot)$으로 표시한다

$$\hat{p} = h_\theta(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$$

로지스틱은 0과 1사이의 값을 출력하는 **시그모이드 함수sigmoid function**이다.

로지스틱 함수

$$\sigma(t) = \frac{1}{1 + exp(-t)} = \frac{1}{1 + e^{-t}}$$

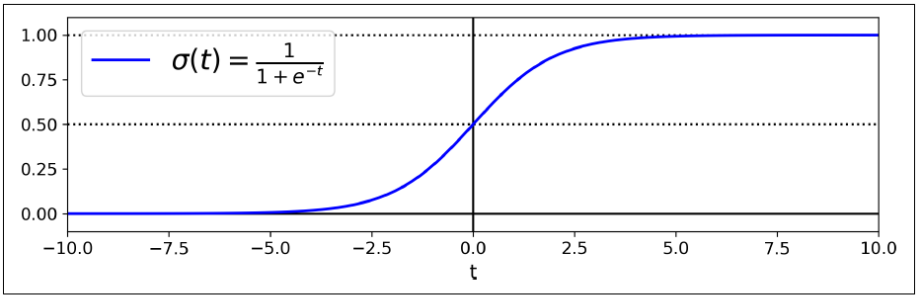



*Figure 4-21. Logistic function*

로지스틱 회귀 모델이 샘플 $\boldsymbol{x}$가 양성 클래스에 속할 확률 $\hat{p} = h_\theta(\boldsymbol{x})$을 추정하여 이에대한 예측 $\hat{y}$를 구한다.

로지스틱 회귀 모델 예측

$$\hat{y} = \begin{cases} 0 & if\ \hat{p} < 0.5 \\ 1 & if\ \hat{p} \geq 0.5 \end{cases}$$

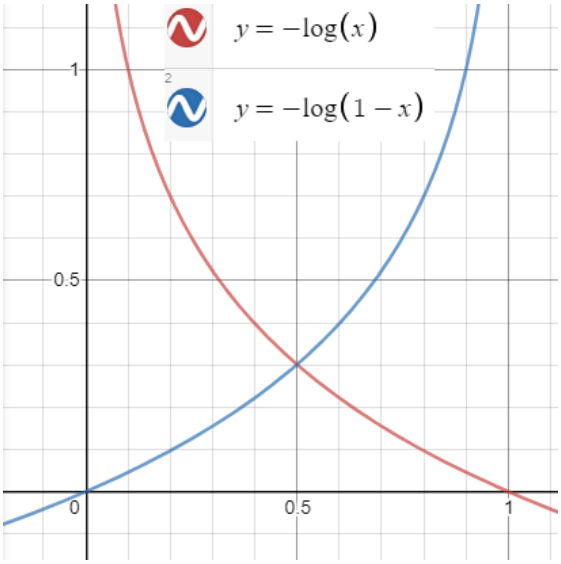
$\boldsymbol{\theta}^T\boldsymbol{x} > 0 \rightarrow t > 0 \rightarrow \sigma(t) > 0.5 \rightarrow$ 1(양성 클래스)으로 예측
$\boldsymbol{\theta}^T\boldsymbol{x} < 0 \rightarrow t < 0 \rightarrow \sigma(t) < 0.5 \rightarrow$ 0(음성 클래스)으로 예측

### 4.6.2 훈련과 비용 함수

로지스틱 회귀 모델의 훈련의 목적 : 양성 샘플$(y = 1)$에 대해서는 높은 확률을 추정하고, 음성 샘플$(y = 0)$에 대해서는 낮은 확률을 추정하는 모델의 파라미터 벡터 $\boldsymbol{\theta}$를 찾는 것이다.

하나의 훈련 샘플에 대한 비용 함수

$$c(\theta) = \begin{cases} -log(\hat{p}) & if\ y = 1 \\ -log(1-\hat{p}) & if\ y = 0 \end{cases}$$




양성 샘플 : 0에 가깝게 추정 → 비용증가 | 1에 가깝게 추정 → 비용이 0에 가까워짐
음성 샘플 : 1에 가깝게 추정 → 비용증가 | 0에 가깝게 추정 → 비용이 0에 가까워짐

로지스틱 회귀의 비용 함수(로그 손실), 전체 훈련 세트에 대한 비용 함수

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\,log\left(\hat{p}^{(i)}\right) + \left(1-y^{(i)}\right)log\left(1-\hat{p}^{(i)}\right)\right]$$

이 비용 함수는 볼록 함수이므로 경사 하강법(또는 다른 최적화 알고리즘)이 전역 최솟값을 찾는 것을 보장한다. (학습률이 너무 크지 않고 충분한 시간이 있다면)

로지스틱 비용 함수의 편도함수( $j$번째 모델 파라미터 $\theta_j$에 대해 편미분 )

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

→각 샘플에 대해 예측오차를 계산하고 $j$번째 특성값을 곱해서 모든 훈련 샘플에 대해 평균을 낸다.
모든 편도함수를 포함한 그래디언트 벡터를 만들면 BGD 알고리즘을 사용할 수 있다.

### 4.6.4 소프트맥스 회귀 softmax regression 또는 다항 로지스틱 회귀 multinomial logistic regression

로지스틱 회귀 모델을 여러 개의 이진 분류기를 훈련시켜 연결하지 않고 직접 다중 클래스를 지원하도록 일반화

샘플 $\boldsymbol{x}$가 주어짐 → 소프트맥스 회귀 모델이 각 클래스 $k$에 대한 점수 $s_k(\boldsymbol{x})$를 계산 → 그 점수에 **소프트맥스 함수 softmax function** (또는 **정규화된 지수 함수 normalized exponentail function**)를 적용하여 클래스와 확률 추정

클래스 k에 대한 소프트맥스 점수

$$s_k(\boldsymbol{x}) = \left(\boldsymbol{\theta}^{(k)}\right)^T\boldsymbol{x}$$

각 클래스는 자신만의 파라미터 벡터 $\boldsymbol{\theta}^{(k)}$가 있다. 이 벡터들은 **파라미터 행렬parameter matrix** $\boldsymbol{\Theta}$에 행으로 저장됨

샘플 $\boldsymbol{x}$에 대해 각 클래스의 점수가 계산되면 소프트맥스 함수를 통과시켜 클래스 $\boldsymbol{k}$에 속할 확률 $\hat{p}_k$을 추정할 수 있다. 이 함수는 각 점수에 지수 함수를 적용한 후 정규화한다.(모든 지수 함수 결과의 합으로 나눔)
→일반적으로 이 점수를 로짓 또는 로그-오즈 라고 부른다.

소프트맥스 함수

$$\hat{p}_k = \sigma(s(\boldsymbol{x}))_k = \frac{exp\ (s_k(\boldsymbol{x}))}{\sum_{j=1}^{K} exp\ (s_j(\boldsymbol{x}))}$$

- $K$ : 클래스 수
- $s(x)$ : 샘플 $x$에 대한 각 클래스의 점수를 담은 벡터
- $\sigma\left(s(\boldsymbol{x})\right)_k$ : 샘플 $x$에 대한 각 클래스의 점수가 주어졌을 때 이 샘플이 클래스 $k$에 속할 추정 확률

로지스틱 회귀 분류기와 마찬가지로 소프트 맥스 회귀 분류기는 추정 확률이 가장 높은 클래스(=가장 높은 점수를 가진 클래스)를 선택한다.

$$\hat{y} = \underset{k}{argmax}\ \sigma\left(\boldsymbol{s}(\boldsymbol{x})\right)_k = \underset{k}{argmax}\ s_k(\boldsymbol{x}) = \underset{k}{argmax}(\left(\boldsymbol{\theta}^{(k)}\right)^T\boldsymbol{x})$$

$argmax$연산 : 함수를 최대화하는 변수의 값을 반환한다.

소프트맥스 회귀 분류기는 한 번에 하나의 클래스만 예측한다.
ex) 종류가 다른 붓꽃 같이 상호 배타적인 클래스→사용가능
하나의 사진에서 여러 사람의 얼굴 인식 →사용불가

훈련방법 :
타깃 클래스 → 높은 확률 | 다른 클래스 → 낮은 확률을 추정하도록 하는 것이 목적
**크로스 엔트로피 cross entropy** 비용 함수를 최소화하는 것은 타깃 클래스에 대해 낮은 확률을 예측하는 모델을 억

제하므로 이 목적에 부합하다.
크로스 엔트로피는 추정된 클래스의 확률이 타깃 클래스에 얼마나 잘 맞는지 측정하는 용도로 종종 사용된다.

크로스 엔트로피 비용 함수

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log\ (\hat{p}_k^{(i)})$$

$y_k^{(i)}$ : $i$번째 샘플이 클래스 $k$에 속할 타깃 확률, 샘플이 클래스에 속하는지 아닌지에 따라 1또는 0이 된다.
딱 두개의 클래스가 있을 때 $(K = 2)$ 이 비용 함수는 로지스틱 회귀의 비용 함수와 같다.

클래스 k에 대한 크로스 엔트로피의 그래디언트 벡터

$$\nabla_{\theta^k} J(\Theta) = \frac{1}{m}\sum_{i=1}^{m} (\hat{p}_k^{(i)} - y_k^{(i)}) x^{(i)}$$

각 클래스에 대한 그래디언트 벡터를 계산할 수 있으므로 비용 함수를 최소화하기 위한 파라미터 행렬 $\boldsymbol{\Theta}$를 찾기 위해 경사 하강법(또는 다른 최적화 알고리즘)을 사용할 수 있다.