머신러닝 톺아보기 세션

4주차. 모델 훈련

유선호

0 CONTEXT

1. 선형 회귀
2. 경사 하강법
3. 다항 회귀
4. 학습 곡선
5. 모델 규제
6. 로지스틱 회귀
7. 소프트맥스 회귀

1 선형 회귀

선형 회귀의 중요성

선형 회귀 모델의 훈련 과정이 매우 단순하여 머신러닝의 기초 개념을 설명하는데 매우 유용하다.

딥러닝 심층 신경망 모델 등 대다수의 머신러닝 모델이 훈련 과정에서 선형 회귀 모델의 훈련 방식을 활용하면서 보다 복잡한 문제를 해결한다.

1 선형 회귀

1. 머신러닝 모델이란?

예제: 1인당 GDP와 삶의 만족도

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

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

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, $1 \leq i \leq 24$.

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

WeTIE

1 선형 회귀

1. 머신러닝 모델이란?

선형 회귀 모델 일반화

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

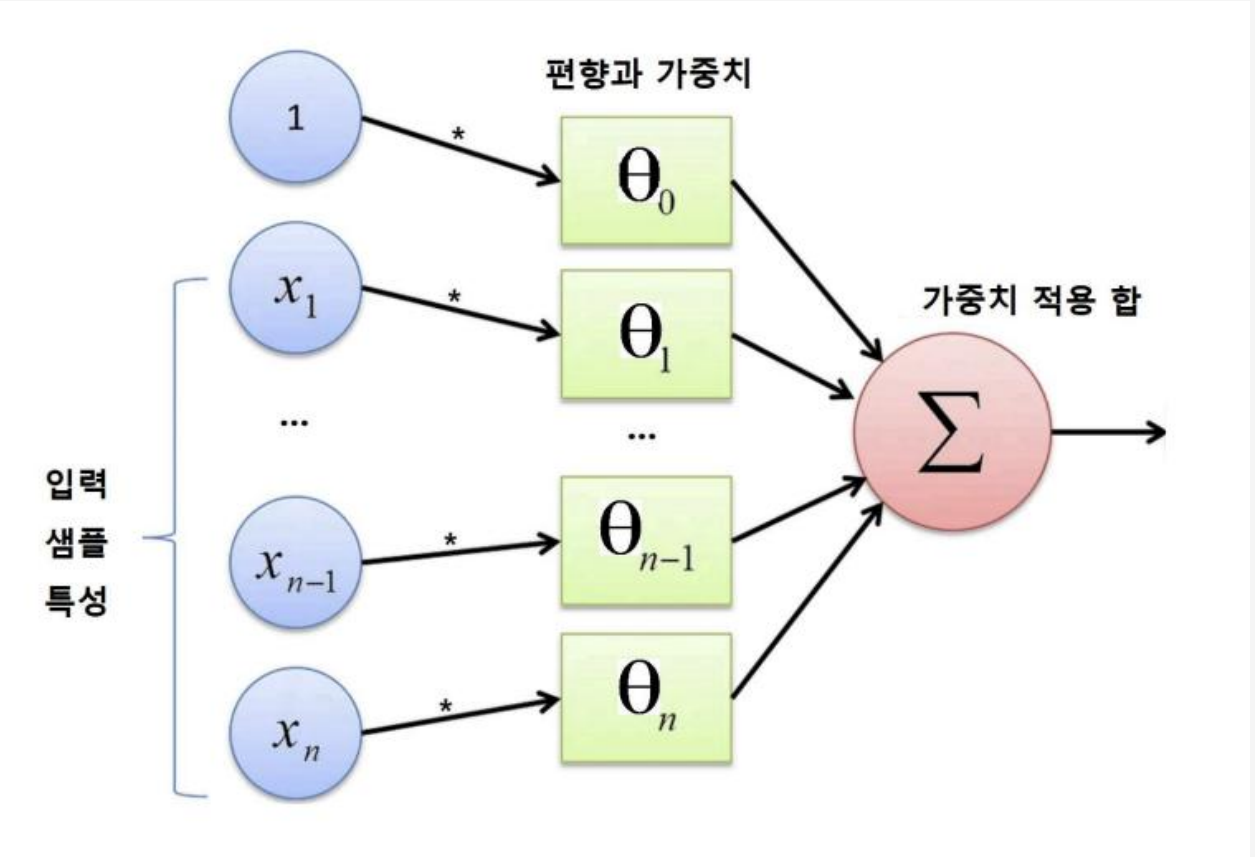
파라미터, 편향, 가중치

파라미터(parameter) : $\theta_0, \theta_1, \theta_2, \ldots, \theta_n$

편향(bias) : θ_0

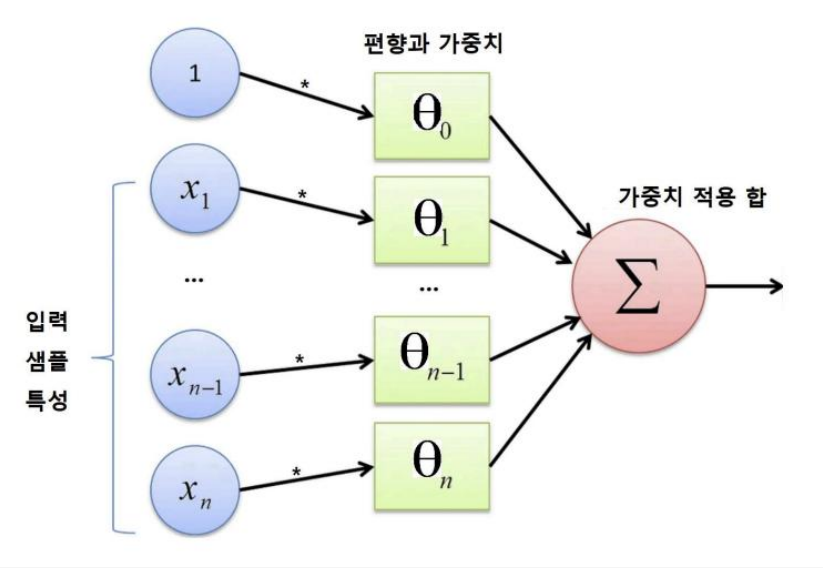
가중치(weight) : 편향을 제외한 나머지 파라미터

파라미터는 모델이 훈련을 통해 학습



2. 행렬 연산 표기법

선형 회귀 모델 : (1, 1+n) 모양의 행렬과 (1+n, 1) 모양의 행렬의 곱으로 표현



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

WeTIE

1 선형 회귀

2. 행렬 연산 표기법

일반적인 머신러닝 모델은 여러 개의 입력값에 대해 동시에 예측값을 계산한다.

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

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

$$\mathbf{x}^{(i)} = [1, x_1^{(i)}, \ldots, x_n^{(i)}]$$

$$\mathbf{X} = \begin{bmatrix} 1, x_1^{(0)}, & \cdots & , x_n^{(0)} \\ & \vdots & \\ 1, x_1^{(m-1)}, & \ldots & , x_n^{(m-1)} \end{bmatrix}$$

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

데이터	어레이 기호	어레이 모양(shape)
모든 예측값	$\hat{\mathbf{y}}$	$(m, 1)$
훈련셋	$\mathbf{X}$	$(m, 1+n)$
모델 파라미터	θ	$(1+n, 1)$

I번째 입력 샘플 : x^i

I번째 입력 샘플의 j번째 특성값 : x_j^i

WeTIE

3. 머신러닝 모델 훈련의 목표

타겟에 최대한 가까운 예측값 계산을 목표로 한다.

모델의 예측 성능을 최대화 해야한다.

모델 성능은 일반적으로 모델의 비용 함수를 이용하여 계산한다.

비용 함수

모델의 성능이 얼마나 나쁘지 평가

모델의 종류와 목표에 따라 다른 비용 함수 선택

회귀 모델 : 일반적으로 평균 제곱 오차(MSE)를 비용 함수로 사용

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

3. 머신러닝 모델 훈련의 목표

모델 훈련의 최종 목표

$MSE(\theta)$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 : 선형 회귀를 활용하는 극히 일부 모델에서, 훈련셋의 크기와 입력 특성 수 모두 작을 때 활용함.

경사하강법 : 딥러닝 모델에서도 기본으로 활용되는 훈련 기법

2 경사하강법

경사하강법 관련 주요 개념

하이퍼파라미터

파라미터

배치 크기

비용 함수

전역/지역 최소값

스텝

학습률

에포크

스텝 크기

최적 학습 모델

비용 함수의 gradient vector

허용오차

2 경사하강법

하이퍼파라미터

훈련시킬 모델을 지정할 때 사용되는 설정 옵션, 대표적으로 학습률, 에포크, 허용 오차, 배치 크기

(클래스의 객체를 생성할 때 클래스의 생성자 함수에 전달되는 인자들)

파라미터

선형 회귀 모델에 사용되는 편향과 가중치 파라미터처럼 모델 훈련 중에 학습되는 값들

(모델 훈련을 통해 학습된 파라미터는 모델 객체의 속성으로 저장됨)

배치 크기

보다 좋은 파라미터 값으로 업데이트하기 위해 필요한 훈련 샘플의 수

전체 데이터셋의 크기 m과 구분하기 위해 m_b개로 표기

파라미터 업데이트는 m_b개의 훈련셋을 학습할 때마다 이뤄짐

사이킷런 모델 : 배치 크기 선택 옵션 지원 없으며 경사하강법을 적용하는 모델의 배치 크기는 1임.

딥러닝 심층 신경망 모델 : 배치 크기 선택 옵션 제공, 일반적으로 8, 16, 32, 64, 128, 256 중에 하나 선택

`LinearRegression` : 경사하강법 사용하지 않음.

`SGDRegressor` : $m_b = 1$

`LogisticRegressor` : $m_b = 1$

2 경사하강법

비용 함수

모델의 성능이 얼마나 나쁜가를 측정하는 함수

배치 단위로 비용 함숫값 계산

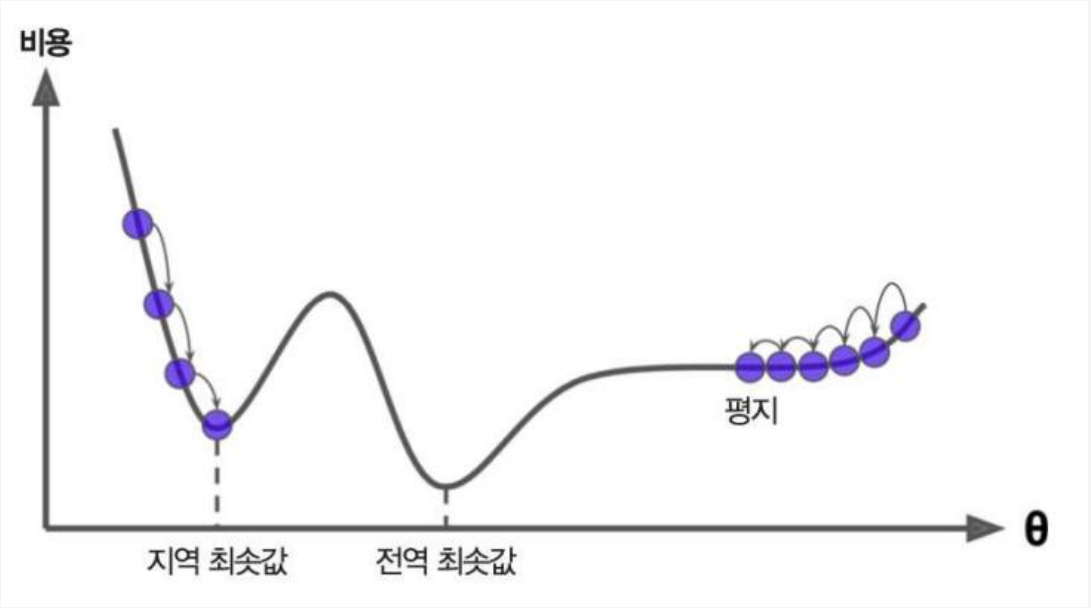
회귀 모델의 배치 단위로 계산되는 MSE

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

전역/지역 최소값

비용 함수가 가질 수 있는 전역/지역 최소값

예제: 선형 회귀 모델의 평균 제곱 오차(MSE) 함수가 갖는 전역/지역 최소값



학습률

훈련 스텝마다 비용 함숫값 계산에 사용되는 파라미터 θ를 얼만큼씩

조정할 것인지를 정하는 비율

WeTIE

2 경사하강법

스텝

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

에포크

훈련셋에 포함된 모든 데이터를 대상으로 예측값을 한 번 계산하는 과정

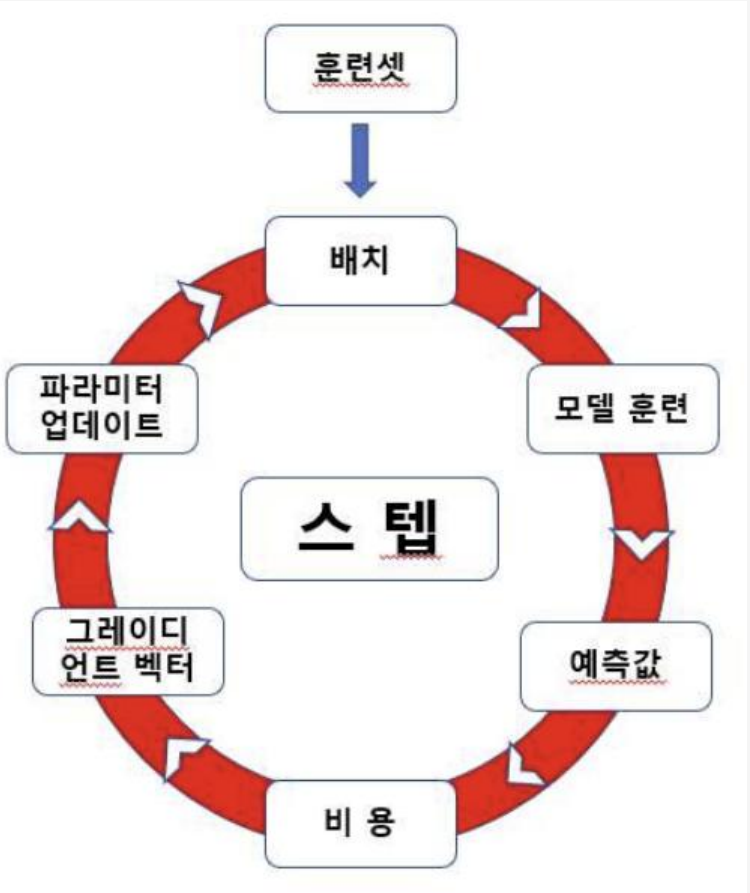
이 과정 동안 실행된 스텝 횟수만큼 파라미터의 업데이트가 이루어짐

스텝 크기

에포크 동안 실행된 스텝의 횟수, 즉 파라미터를 조정한 횟수

스텝 크기 = 훈련셋 크기 / 배치 크기

훈련셋 크기가 1000이고, 배치 크기가 10이면 에포크마다 100번의 스텝 실행



2 경사하강법

최적 학습 모델

비용 함수를 최소화하는 파라미터를 학습한 모델

최종적으로 훈련을 통해 얻고자 하는 모델

비용 함수의 그레디언트 벡터

함수의 그레디언트 벡터는 방향과 크기에 대한 정보 제공

그레디언트가 가리키는 방향의 반대 방향으로 움직여야 가장 빠르게 전역 최소값에 접근

$MSE(\theta)$ 함수의 θ에 대한 그레디언트 벡터

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

허용 오차

비용 함수의 그레디언트 벡터의 크기가 허용 오차보다 작아질 때 훈련 종료

그레디언트의 크기가 0에 가까우면 비용 함수의 전역 또는 지역 최소값에 거의 다다랐음을 의미하기 때문임

2 경사하강법

1. 선형 회귀 모델 훈련과 경사하강법

MSE를 비용 함수로 사용하는 경우 경사하강법은 다음 과정으로 이루어짐

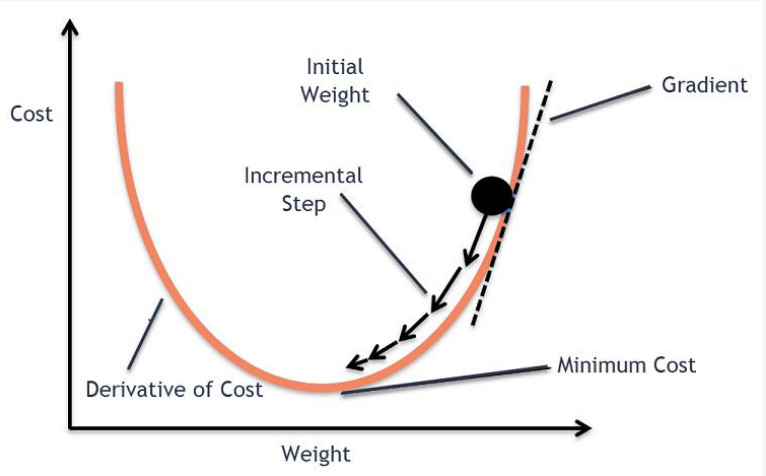
θ 를 임의의 값으로 지정한 후 훈련 시작

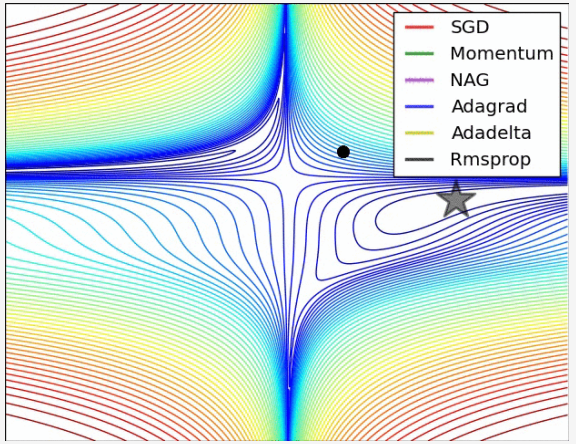
$MSE(\theta)$ 가 허용 오차보다 적게 작아질 때까지 아래 과정 반복

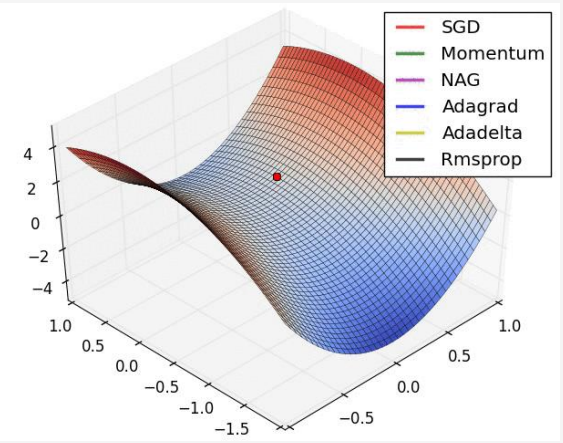
배치 크기 m_b만큼의 훈련 샘플을 이용하여 예측값 생성 후 $MSE(\theta)$ 계산.

아래 점화식을 이용한 θ 업데이트

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





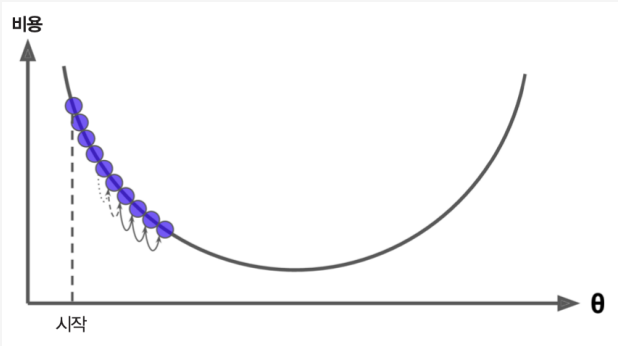


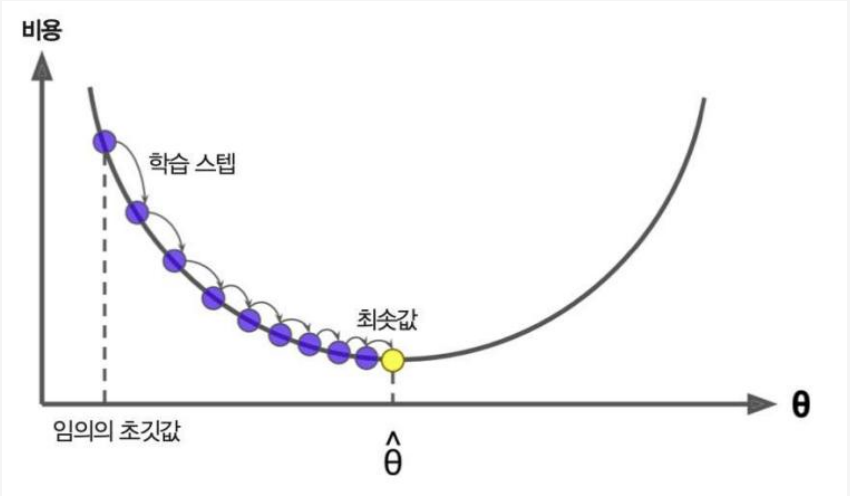
WeTIE

2 경사하강법

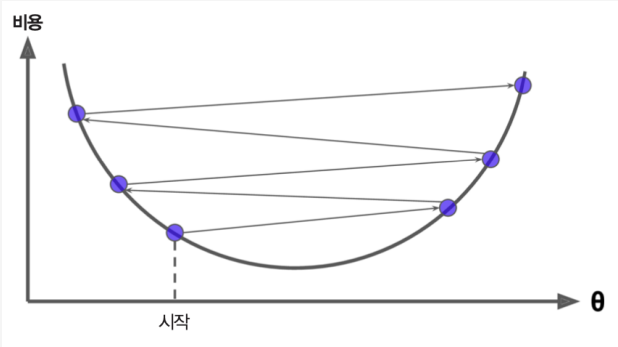
2. 학습률의 중요성

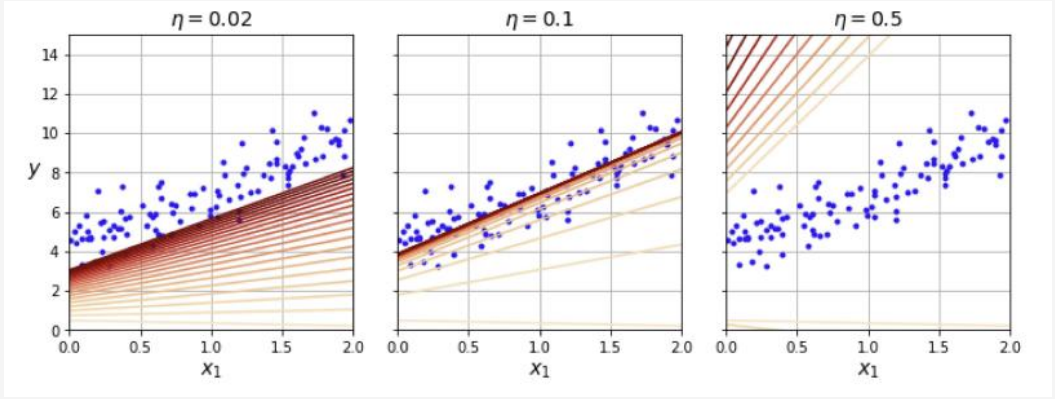
학습률이 너무 적은 경우 : 비용 함수가 전역 최소값을 갖도록 하는 $\hat{\theta}$에 너무 느리게 수렴





학습률이 너무 큰 경우 : 비용 함수가 전역 최소값을 갖도록 하는 $\hat{\theta}$에 수렴하지 않고 발산함





WeTIE

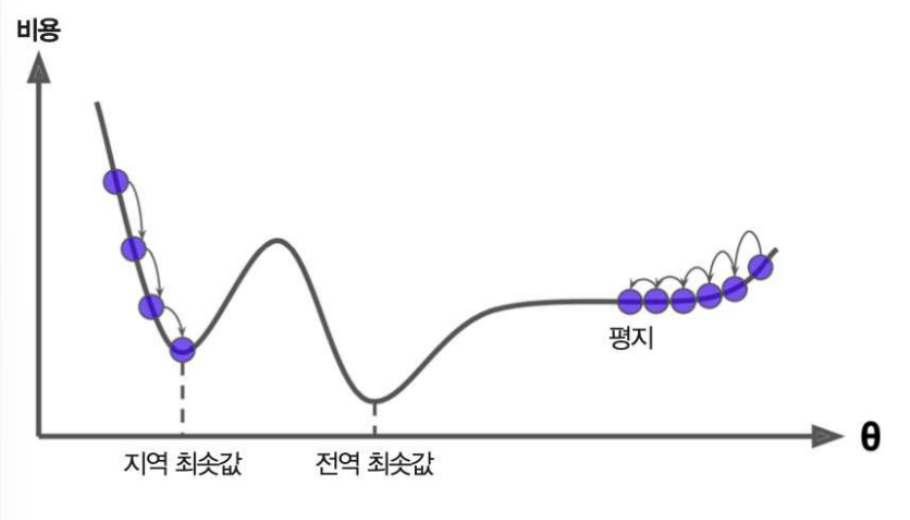
2. 학습률의 중요성

비선형 모델 훈련의 어려움

선형 회귀 모델은 학습률을 적절하게 잡으면 언제나 최적의 모델로 수렴함

반면에 비선형 모델은 학습률과 상관없이 파라미터를 초기화하는 방식에 따라 지역 최소값에 수렴하거나 수렴하지 못하고 정체할 수도 있음

대다수의 머신러닝 모델은 비선형 문제를 다룸



2. 학습률의 중요성

특성 스케일링의 중요성

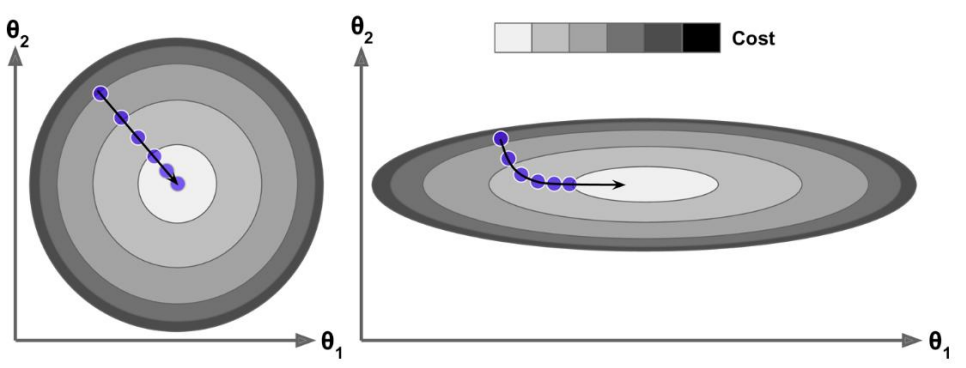
특성들의 스케일을 통일시키면 보다 빠른 학습이 이루어진다.

- 왼쪽 그림 : 두 특성의 스케일이 동일하게 조정된 경우, 비용 함수의 최소값으로 최단 거리로 수렴함.

(비용 등고선이 원 모양으로 그려지는 경우)

- 오른쪽 그림 : 두 특성의 스케일이 다른 경우 비용 함수의 최소값으로 보다 먼 거리를 지나감.

(비용 등고선이 타원 모양 또는 찌그러진 모양으로 그려지게 됨)



3. 경사하강법 종류

배치 경사하강법

$m_b = m$

확률적 경사하강법

$m_b = 1$

하나의 훈련 샘플을 학습할 때마다 그레디언트를 계산하여 파라미터 조정

미니배치 경사하강법

$m_b \in \{2, 4, 6, 8, 16, 32, 64, 128, 256, 512, \dots, \}$

사이킷런 모델은 미지원

딥러닝 심층 신경망 모델은 배치 크기 선택 옵션 지원

3. 경사하강법 종류

배치 경사하강법

에포크마다 한 번 그레디언트를 계산하여 파라미터 조정

사이킷런 모델을 포함하여 일반적으로 사용되지 않음

단점

훈련 세트가 크면 그레디언트를 계산하는데 많은 시간이 필요

아주 많은 데이터를 저장해야 하는 메모리 문제도 발생 가능

3. 경사하강법 종류

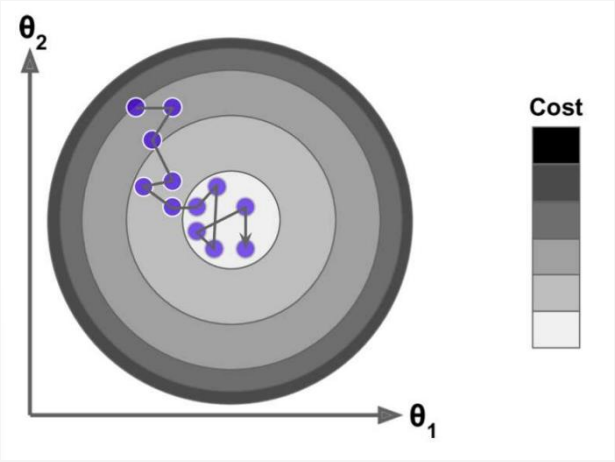
확률적 경사하강법

매우 큰 훈련 세트를 다룰 수 있음

학습 과정이 매우 빠르며 파라미터 조정이 불안정 할 수 있기 때문에 지역 최소값에 상대적으로 덜 민감함

단점

학습 과정에서 파라미터의 동요가 심해서 경우에 따라 전역 최소값에 수렴하지 못하고 계속해서 발산할 가능성도 높음



3. 경사하강법 종류

학습 스케줄(learning schedule) : 훈련이 지속될수록 학습률을 조금씩 줄이는 기법

일반적으로 훈련 에포크가 진행될수록 학습률이 조금씩 작아지도록 설정

요동치는 파라미터를 제어하기 위해 학습률을 학습 과정 동안 천천히 줄어들게 만들 수 있음

주의사항

학습률이 너무 빨리 줄어들면, 지역 최소값에 갇힐 수 있음

학습률이 너무 느리게 줄어들면 전역 최소값에 제대로 수렴하지 못하고 맴돌 수 있음

3. 경사하강법 종류

미니배치 경사하강법

장점

배치 크기를 어느 정도 크게 하면 확률적 경사하강법보다 파라미터의 움직임이 덜 불규칙적이 됨

반면에 배치 경사하강법보다 빠르게 학습

학습 스케줄 잘 활용하면 최소값에 수렴함

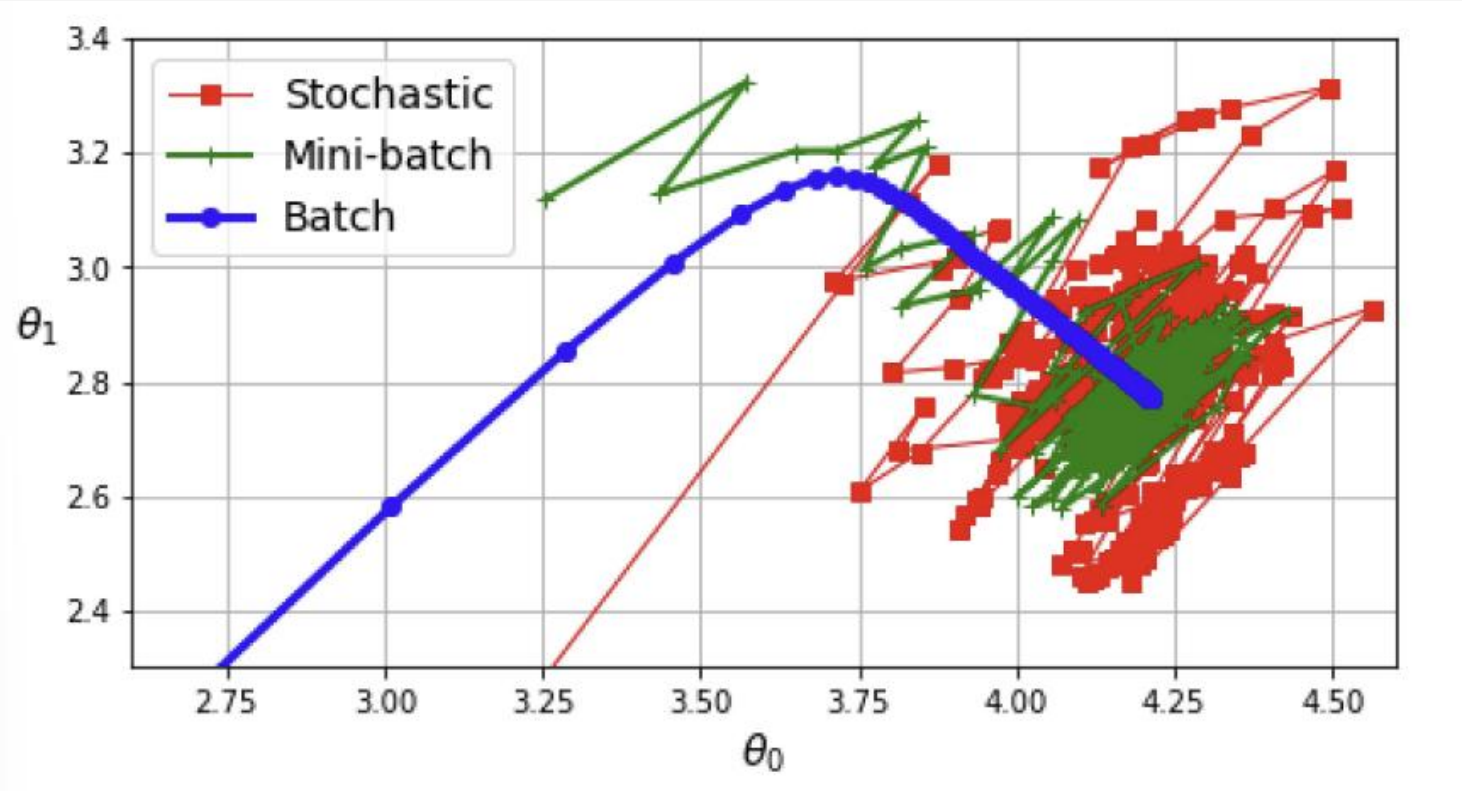
단점

SGD에 비해 지역 최소값에 수렴할 위험도가 보다 큼

대부분의 딥러닝 심층 신경망 모델에서 지원됨

3. 경사하강법 종류

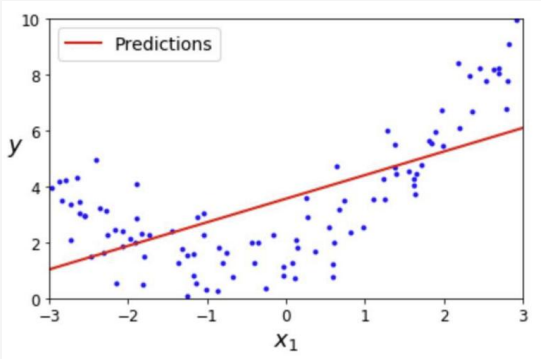
3가지 경사하강법 비교



3 비선형 데이터 학습: 다항 회귀

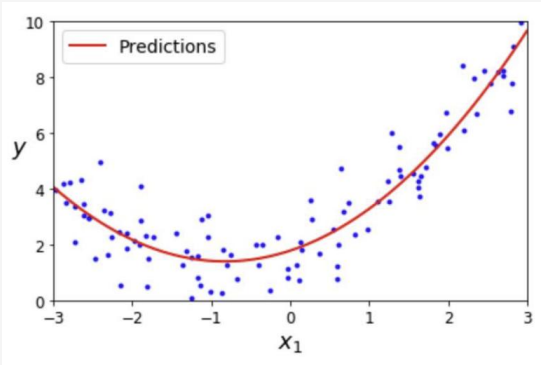
비선형 데이터를 선형 회귀를 이용하여 학습하는 기법을 다항 회귀(polynomial regression)라 한다.

예제 : 2차 함수 모델을 따르는 데이터셋에 선형 회귀(1차 선형) 모델 적용



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

예제 : 2차 함수 모델을 따르는 데이터셋에 2차 다항식 모델 적용



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

WeTIE

3 비선형 데이터 학습: 다항 회귀

2차 다항 회귀

두 개의 특성을 갖는 데이터셋에 대해 2차 다항식 모델을 훈련시키고자 하면

다음 3개의 특성을 추가한 후

$$x_1^2,\ x_2^2,\ x_1 x_2$$

해당 특성들을 선형 회귀 모델로 훈련시키면 예측값은 아래와 같이 계산된다.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2$$

다항 회귀의 단점

몇 차 다항 회귀를 사용해야 할지 일반적으로 알 수 없음

심층 신경망처럼 비선형 데이터를 분석하는 보다 좋은 모델이 개발되어 다항 회귀를 사용할 필요 없음

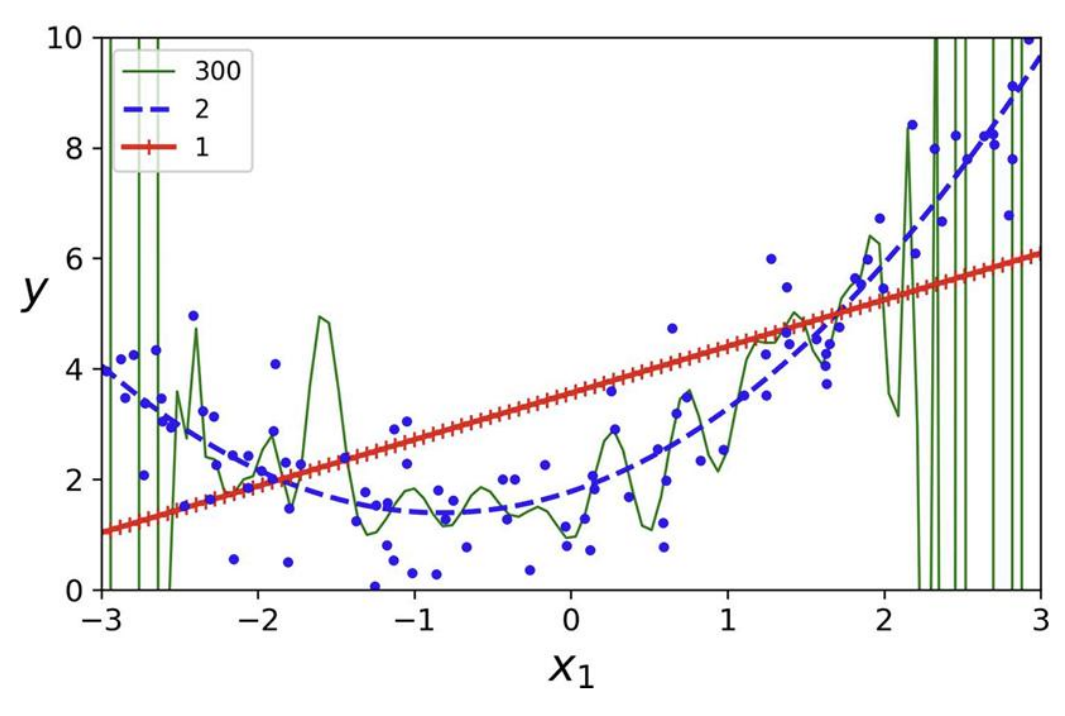
WeTIE

4 학습 곡선

과소적합/과대적합 판정

예제 : 선형 모델, 2차 다항 회귀 모델, 300차 다항 회귀 모델 비교

차수에 따라 모델이 훈련셋에 과소 또는 과대 적합할 수 있음



4 학습 곡선

모델 성능 평가: 검증 데이터셋 활용 vs. 학습 곡선

모델 성능 평가는 보통 다음 두 가지 방식을 따른다.

검증 데이터셋 활용

과소적합 : 훈련 점수와 검증 데이터셋 점수 모두 낮은 경우

과대적합 : 훈련 점수는 높지만 검증 데이터셋 점수가 상대적으로 많이 낮은 경우

학습 곡선(learning curve)

훈련셋과 검증셋에 대한 모델 성능을 비교하는 그래프

x축 : 훈련셋 크기, 훈련셋의 크기를 1%에서부터 출발해서 점차 키워 나가면서 모델 성능 평가

y축 : 훈련셋 크기에 따른 모델 성능, 훈련 점수와 검증 점수 사용

학습 곡선의 모양에 따라 과소적합/과대적합 판정 가능

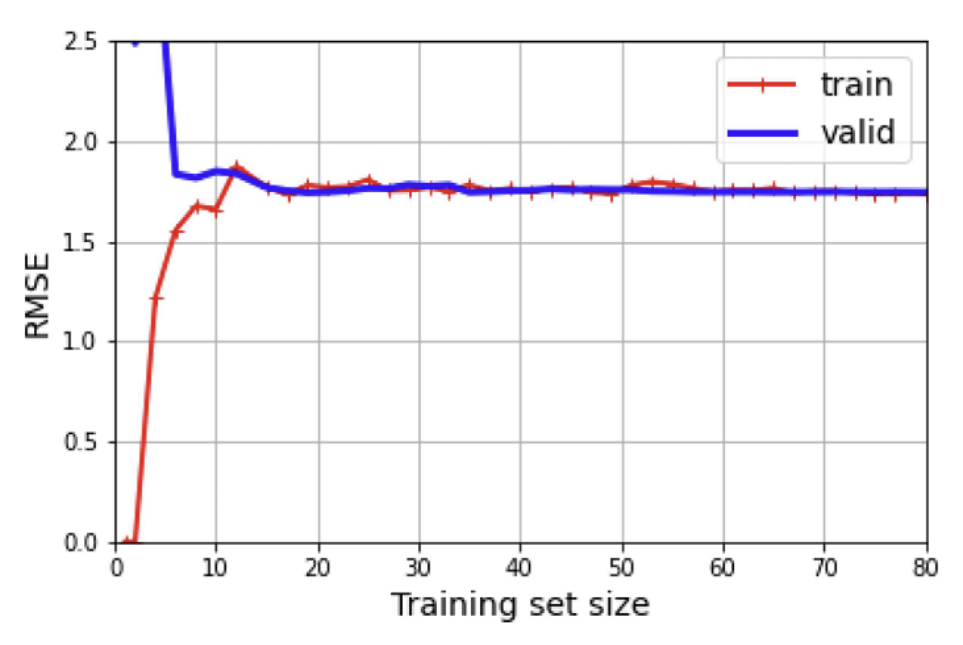
4 학습 곡선

과소적합 모델의 학습 곡선 특징

2차 다항 함수의 분포를 따르는 데이터셋에 선형 회귀 모델을 적용한 학습 곡선

훈련셋에 대한 성능(빨강) : 훈련셋이 커지면서 RMSE(평균 제곱근 오차)가 커지면서 어느 순간 변화 없음

검증셋에 대한 성능(파랑) : 검증셋에 대한 성능이 훈련셋에 대한 성능과 거의 비슷해짐



WeTIE

4 학습 곡선

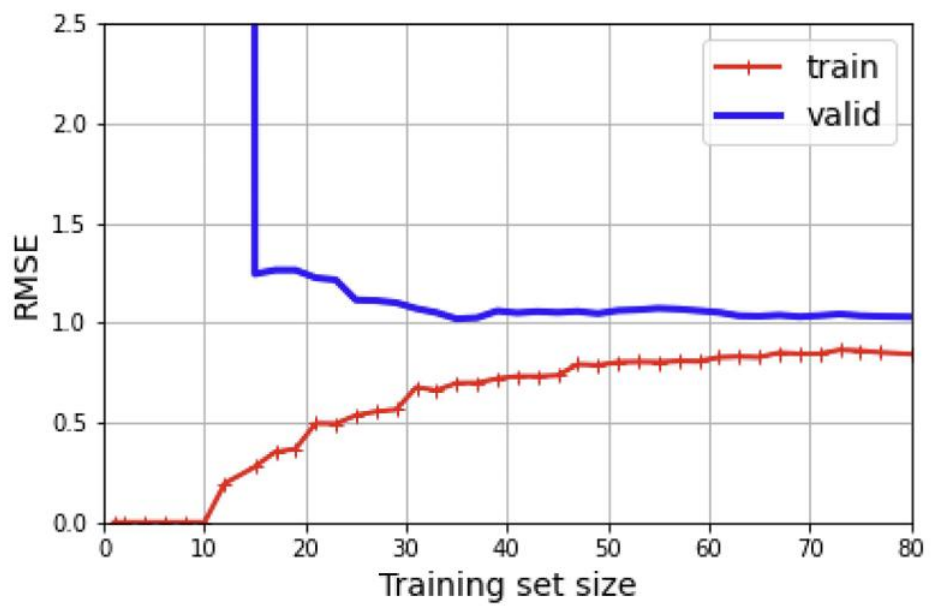
과대적합 모델의 학습 곡선 특징

2차 다항 함수의 분포를 따르는 데이터셋에 10차 다항회귀 모델을 적용한 학습 곡선

훈련셋에 대한 성능(빨강) : 훈련셋에 대한 평균 제곱근 오차가 매우 낮음

검증셋에 대한 성능(파랑) : 검증셋에 대한 평균 제곱근 오차가 보다 높음

과대적합 모델 개선법 : 훈련 데이터 추가, 일반적으로 매우 어려움



WeTIE

4 학습 곡선

모델의 일반화 성능(테스트 데이터셋에 대한 성능)

훈련 과정에서 다루지 않은 새로운 데이터에 대한 예측 능력

일반화 성능을 떨어뜨리는 요소

편향

분산

줄일 수 없는 오차 : 데이터 자체가 갖고 있는 잡음(noise) 때문에 발생하는 어쩔 수 없는 오차

4 학습 곡선

편향

데이터셋에 대한 모델링이 틀린 경우

실제로는 2차원 모델인데 1차원 모델을 사용하는 경우 발생

과소적합 발생 가능성 높음

분산

모델이 훈련 데이터에 민감하게 반응하는 정도

고차 다항 회귀처럼 자유도(degree of freedom)가 높은 모델일수록 분산이 커짐

모델의 자유도 : 모델이 찾아야 하는 파라미터의 개수

과대적합 발생 가능성 높음

편향-분산 트레이드 오프

복잡한 모델일수록 편향을 줄고 분산은 커짐

단순한 모델일수록 편향은 커지고 분산은 줄어듦

5 모델 규제와 조기 종료

모델 규제

릿지 회귀 : 가중치의 절댓값을 최대한 작게 유지, 모델의 분산을 줄임. 단 편향은 커짐.

라쏘 회귀 : 중요하지 않은 특성의 가중치를 0으로 만듦. 자유도가 줄어들어 모델의 분산이 줄어듦. 단, 편향은 커짐

엘라스틱 넷 : 릿지 회귀와 라쏘 회귀 혼합

5 모델 규제와 조기 종료

릿지 회귀

비용함수

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

편향(θ_0) : 제하지 않음. 규제하지 않는다.

m_b : 배치 크기(하이퍼파라미터)

α : 규제 강도(하이퍼파라미터)

$\alpha = 0$: 규제 없음

α값을 크게 잡으면 가중치(θ_i)의 절댓값은 보다 0에 가까워지도록 유도된다. 즉, 가중치의 역할이 줄어든다.

주의사항 : 특성 스케일링 전처리를 해야 규제 모델의 성능이 높아짐(θ_i는 특성의 값 크기에 의존하기에 모든 특성의 크기를 비슷하게 맞추면 일정하게 수렴)

5 모델 규제와 조기 종료

릿지 회귀의 비용 함수 해석

비용함수의 전역 최솟값: $\sum_{i=1}^{n} \theta_i^2$ 값이 최대한 작은 지점

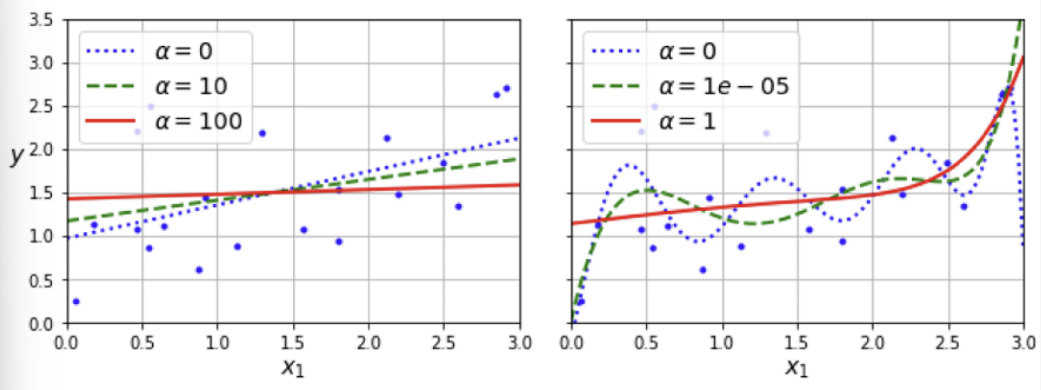
θ_i의 절댓값을 가급적 0에 가깝게 하는 방향으로 경사하강법이 작동

자연스럽게 모델의 분산이 작이짐, 입력값이 조금 변하더라도 예측값이 변하는 정도가 약해짐

예측값을 계산할 때, θ_i가 미치는 영향이 작아지기 때문이다.

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

릿지 규제를 적용한 경우 : 분산이 줄고, 편향이 늘어남



WeTIE

5 모델 규제와 조기 종료

라쏘 회귀

비용함수

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

덜 중요한 특성을 무시하기 위해 해당 특성의 가중치 $|\theta_i|$를 보다 빠르게 0에 수렴하도록 유도,

기본적으로 $|\theta_i|$가 가능하면 작게 움직이도록 유도

5 모델 규제와 조기 종료

라쏘 회귀

비용함수의 전역 최솟값: $\sum_{i=1}^{n} |\theta_i|$ 값이 최대한 작은 지점

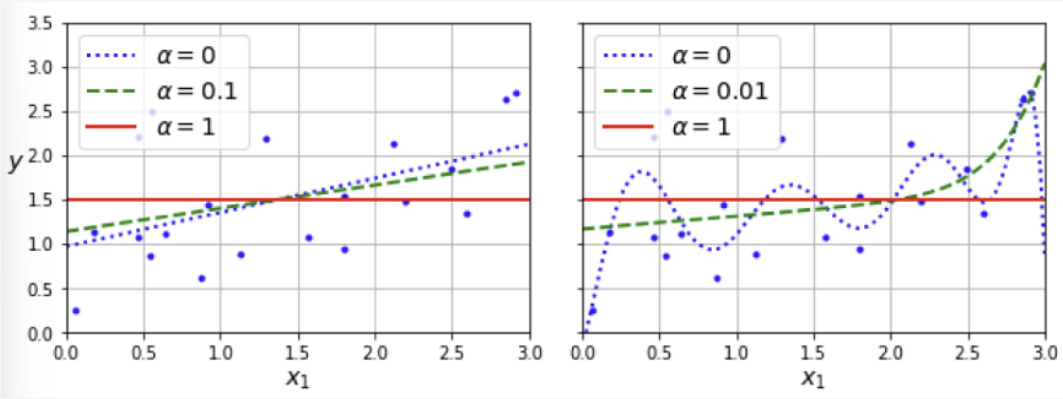
별로 중요하지 않은 특성에 대해 θ_i가 0에 빠르게 수렴하도록 훈련 중에 유도됨

이유 : $|\theta_i|$ 의 미분값이 1 또는 -1이기에 θ_i가 0이 아닌 이상 비용 함수의 그레디언트의 크기가 0에 가까워지기가 상대적으로 어렵기 때문

경사하강법을 반복적으로 적용하는 훈련 과정동안 중요하지 않은 특성의 가중치를 빠르게 0으로 수렴하게 만듦

대신 중요한 특성의 가중치를 결정하는데 집중

라쏘 규제를 적용한 경우 : 분산이 줄고, 편향이 늘어남



WeTIE

5 모델 규제와 조기 종료

엘라스틱 넷 회귀

비용함수

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

릿지 회귀와 라쏘 회귀를 절충한 모델

혼합 비율 τ을 이용하여 릿지 규제와 라쏘 규제를 적절하게 조절

5 모델 규제와 조기 종료

규제 선택

대부분의 경우 약간이라도 규제 사용 추천

유용한 속성이 많지 않다고 판단되는 경우

라쏘 규제나 엘라스틱넷 활용 추천

불필요한 속성의 가중치를 0으로 만들기 때문

일반적으로 엘라스틱 넷을 많이 추천

5 모델 규제와 조기 종료

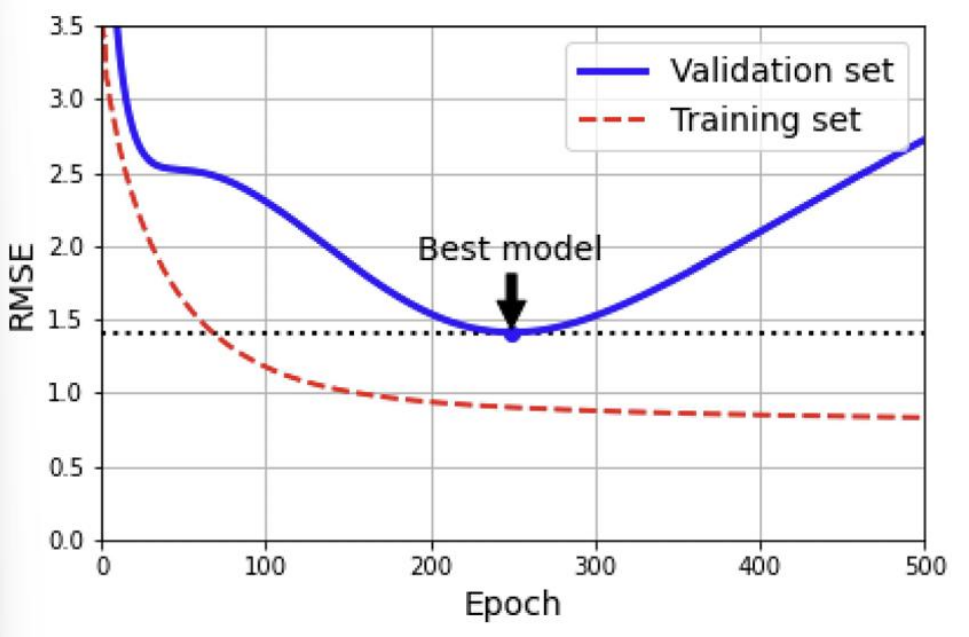
조기 종료

모델이 훈련 중에 훈련셋에 너무 과하게 적응하지 못하도록 하는 가장 일반적인 규제 기법

에포크가 남아있다 하더라도 검증셋에 대한 비용함수의 값이 줄어들다가 다시 커지는 순간 훈련 종료

검증셋에 대한 비용 함수의 곡선이 진동이 발생할 수 있기에 검증 손실이 한동안 최솟값보다 높게 유지될 때 훈련을 멈추고 기억해둔 최적의 모델 사용

아래 그래프 : 250 에포크 정도에서 훈련 조기 종료



WeTIE

확률적 경사하강법과 조기 종료

SGDRegressor 모델에 조기 종료를 지정하는 코드

```
sgd_reg = SGDRegressor(penalty='elasticnet', alpha=0.1, l1_ratio=0.5,
                       eta0=0.002, random_state=42,
                       early_stopping=True,
                       max_iter=1000, tol=1e-3, n_iter_no_change=5)
```

`penalty='elasticnet'`: 엘라스틱 넷 회귀 적용

`alpha=0.1`: 규제 강도

`l1_ratio=0.5`: 라쏘 규제 비율

`early_stopping=True`: 조기 종료 실행. 훈련셋의 일부를 검증셋으로 활용.

`max_iter=1000`: 최대 훈련 에포크

`tol=1e-3`: 훈련 점수 또는 검증 점수가 지정된 값 이하로 최대 `n_iter_no_change` 에포크 동안 변하지 않으면 조기 종료 실행

`n_iter_no_change=5`: 훈련 점수 또는 검증 점수가 지정된 에포크 동안 얼마나 변하는지 확인

6 로지스틱 회귀

로지스틱 회귀와 소프트맥스 회귀

회귀 모델을 분류 모델로 활용

이진 분류 : 로지스틱 회귀 사용, 분류 모델에서 가장 중요한 역할 수행

다중 클래스 분류 : 소프트맥스 회귀 사용

6 로지스틱 회귀

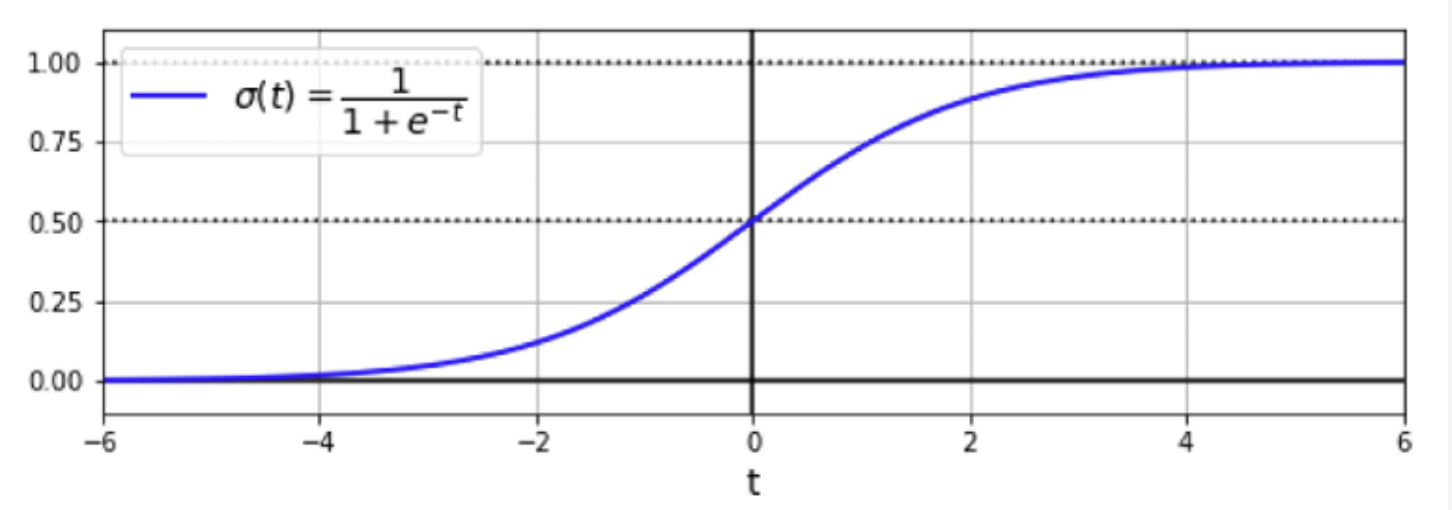
1. 확률 예측

로지스틱 회귀 모델에서 샘플 x가 양성 클래스에 속할 확률

$$\hat{p} = h_\theta(\mathbf{x}) = \sigma(\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n)$$

시그모이드 함수 활용

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



WeTIE

2. 비용함수

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

양성 클래스인 경우

$$\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \geq 0$$

음성 클래스인 경우

$$\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n < 0$$

2. 비용함수

비용함수 : 로그 손실 함수 사용

$$J(\theta) = -\frac{1}{m_b}\sum_{i=1}^{m_b}\left(y^{(i)} \cdot \log(\hat{p}^{(i)}) + (1-y^{(i)}) \cdot \log(1-\hat{p}^{(i)})\right)$$

모델 훈련 : 위 비용함수에 대해 경사하강법 적용

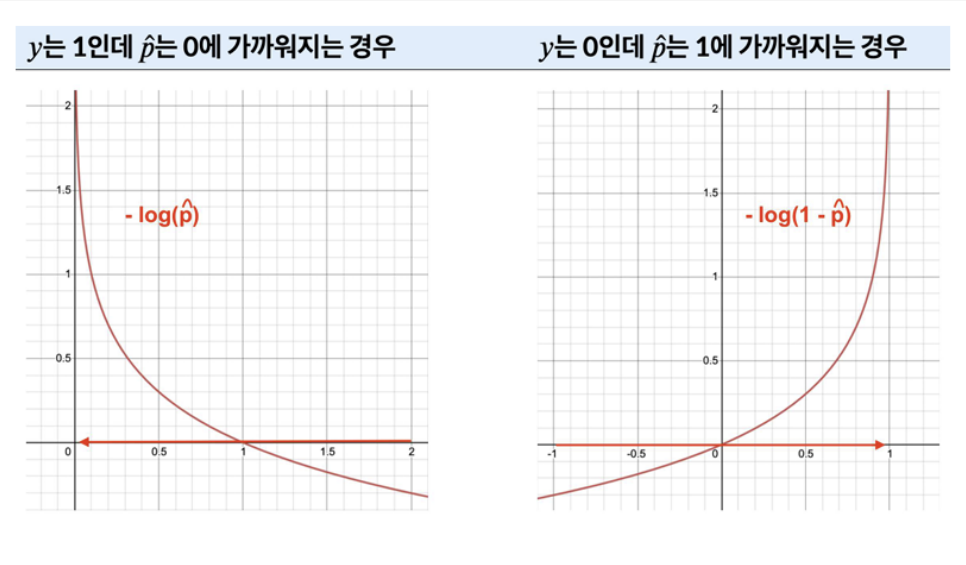
2. 비용함수

로그 손실 함수 이해

틀린 예측을 하면 손실값이 무한이 커짐

왼쪽 그림 : 샘플의 라벨이 1(양성)인데 예측 확률이 0에 가까운 경우, 로그 손실이 매우 크다

오른쪽 그림 : 샘플의 라벨이 0(음성)인데 예측 확률이 1에 가까운 경우, 로그 손실이 매우 크다



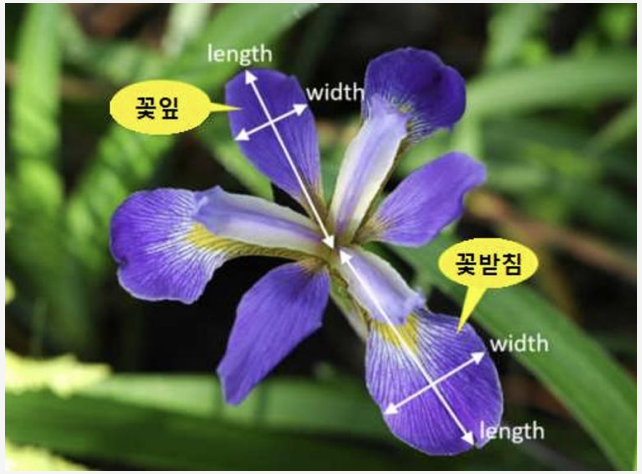
3. 붓꽃 데이터셋

붓꽃의 품종 분류를 로지스틱 회귀로 진행

붓꽃 데이터셋의 샘플의 특성 4개:

꽃받침(sepal)의 길이와 너비

꽃잎(petal)의 길이와 너비

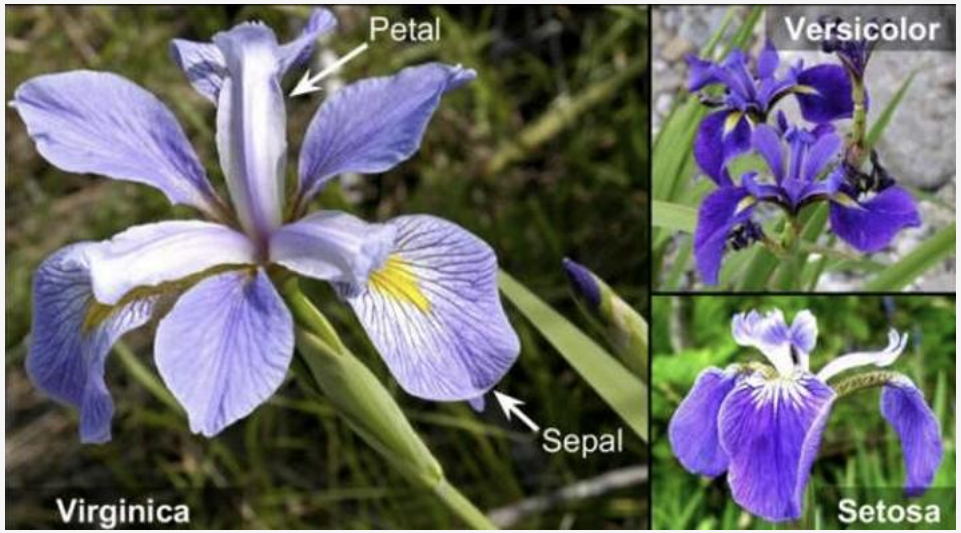


붓꽃 데이터셋의 라벨

0 : Iris-Setosa(세토사)

1 : Iris-Versicolor(버시컬러)

2 : Iris-Virginica(버지니카)



WeTIE

3. 붓꽃 데이터셋

```
from sklearn.datasets import load_iris
iris = load_iris(as_frame=True)
iris.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

```
iris.data.head(5)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
iris.target.head(5) # 세토사 품종 5개
```

```
0    0
1    0
2    0
3    0
4    0
Name: target, dtype: int32
```

```
iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

4. 결정 경계

버지니카 품종 감지기 : 꽃잎 너비 특성 활용

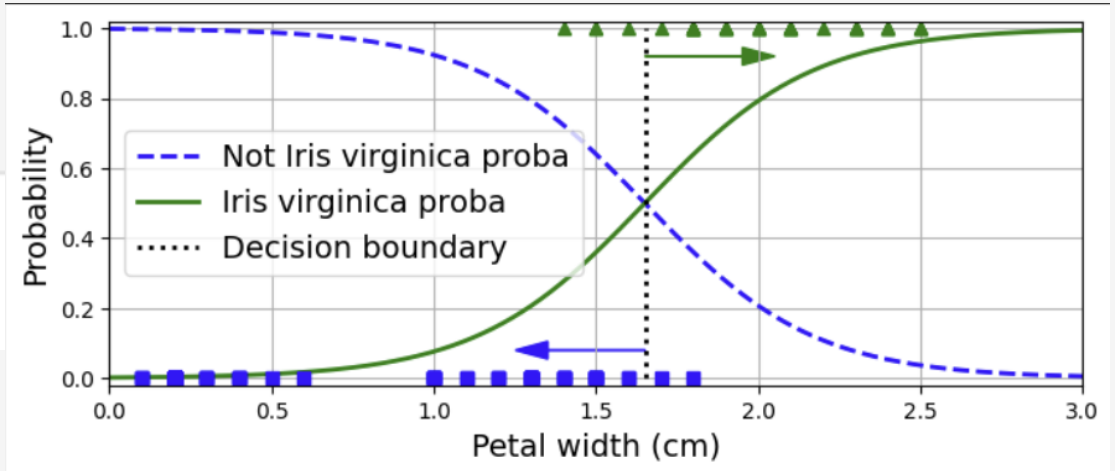
꽃잎의 너비 특성 하나만 이용하여 버지니카 여부 판별

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X = iris.data[["petal width (cm)"]].values
y = iris.target_names[iris.target] == 'virginica'
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(random_state=42)
```



4. 결정 경계

버지니카 품종 감지기 : 꽃잎 너비 특성 활용

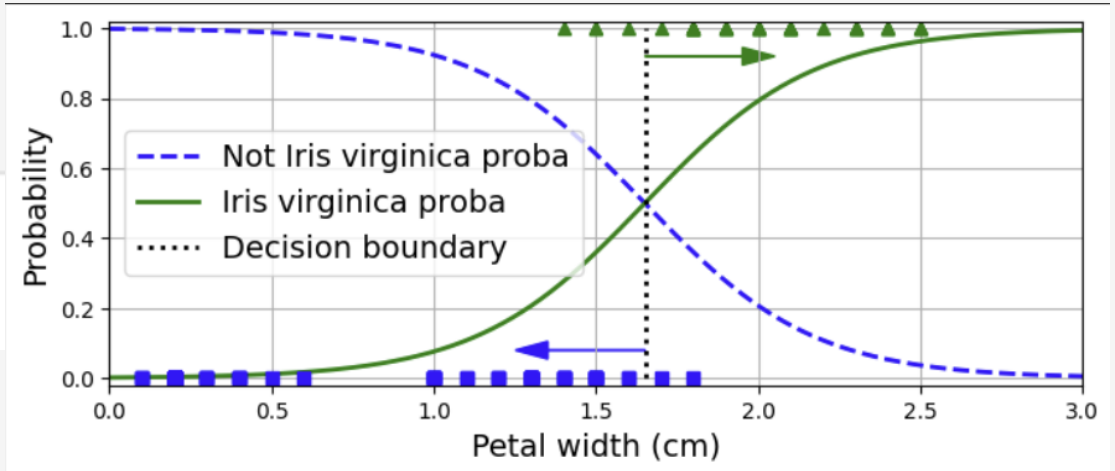
꽃잎의 너비 특성 하나만 이용하여 버지니카 여부 판별

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X = iris.data[["petal width (cm)"]].values
y = iris.target_names[iris.target] == 'virginica'
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(random_state=42)
```



4. 결정 경계

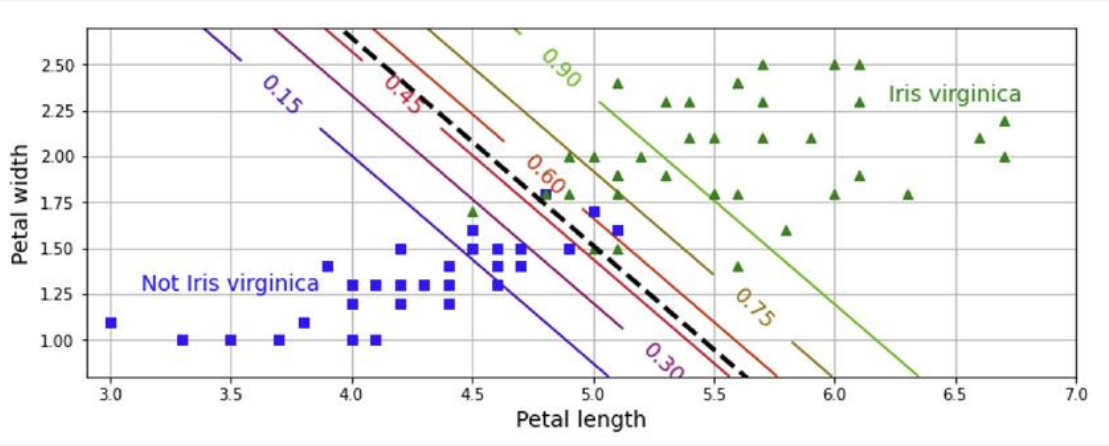
버지니카 품종 감지기 : 꽃잎 길이와 너비 특성 활용

꽃잎의 길이와 너비 두 특성을 이용하여 붓꽃의 품종을 판별한다.

```
# 두 특성만 사용하는 입력 데이터셋 준비
X = iris.data[["petal length (cm)", "petal width (cm)"]].values
y = iris.target_names[iris.target] == 'virginica'
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

# 로지스틱 회귀 모델 준비: `C`는 규제 강도. 기본값은 1. 작은 값일 수록 큰 규제.
log_reg = LogisticRegression(C=2, random_state=42)
log_reg.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(C=2, random_state=42)
```



WeTIE

7 소프트맥스 회귀

로지스틱 회귀 모델을 일반화하여 다중 클래스 분류를 지원하도록 한 회귀 모델

소프트맥스 회귀 학습 아이디어

샘플 x = $[x_1, \dots, x_n]$ 가 주어졌을 때, 각각의 분류 클래스 k에 대한 점수 $s_k(x)$ 계산. k*(n+1) 개의 파라미터를 학습시켜야 한다.

$$s_k(\mathbf{x}) = \theta_0^{(k)} + \theta_1^{(k)} x_1 + \cdots + \theta_n^{(k)} x_n = [1, x_1, \dots, x_n] \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \\ \vdots \\ \theta_n^{(k)} \end{bmatrix}$$

가중치 어레이

편향과 가중치들의 2차원 어레이 : n = 4, 클래스 종류는 3개인 경우

$$\Theta = \begin{bmatrix} \theta_0^{(0)} & \theta_0^{(1)} & \theta_0^{(2)} \\ \theta_1^{(0)} & \theta_1^{(1)} & \theta_1^{(2)} \\ \theta_2^{(0)} & \theta_2^{(1)} & \theta_2^{(2)} \\ \theta_3^{(0)} & \theta_3^{(1)} & \theta_3^{(2)} \\ \theta_4^{(0)} & \theta_4^{(1)} & \theta_4^{(2)} \end{bmatrix}$$

7 소프트맥스 회귀

소프트맥스 함수

K: 클래스(라벨)의 개수
$\mathbf{s}(\mathbf{x}) = [s_1(\mathbf{x}), \ldots, s_K(\mathbf{x})]$
$\sigma()$ 함수: 소프트맥스 확률값 계산 함수. 각 클래스별 확률 예측값으로 구성된 어레이 생성.

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

$$\hat{y} = \text{np.argmax}(\sigma(\mathbf{s}(\mathbf{x})))$$

소프트맥스 회귀의 비용함수

아래 비용 함수에 대해 경사 하강법을 적용하여 최적의 가중치 벡터 θ_k^i 학습

K = 2이면 로지스틱 회귀의 로그 손실 함수와 정확하게 일치

비용함수 : 크로스 엔트로피 비용 함수 사용

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

WeTIE

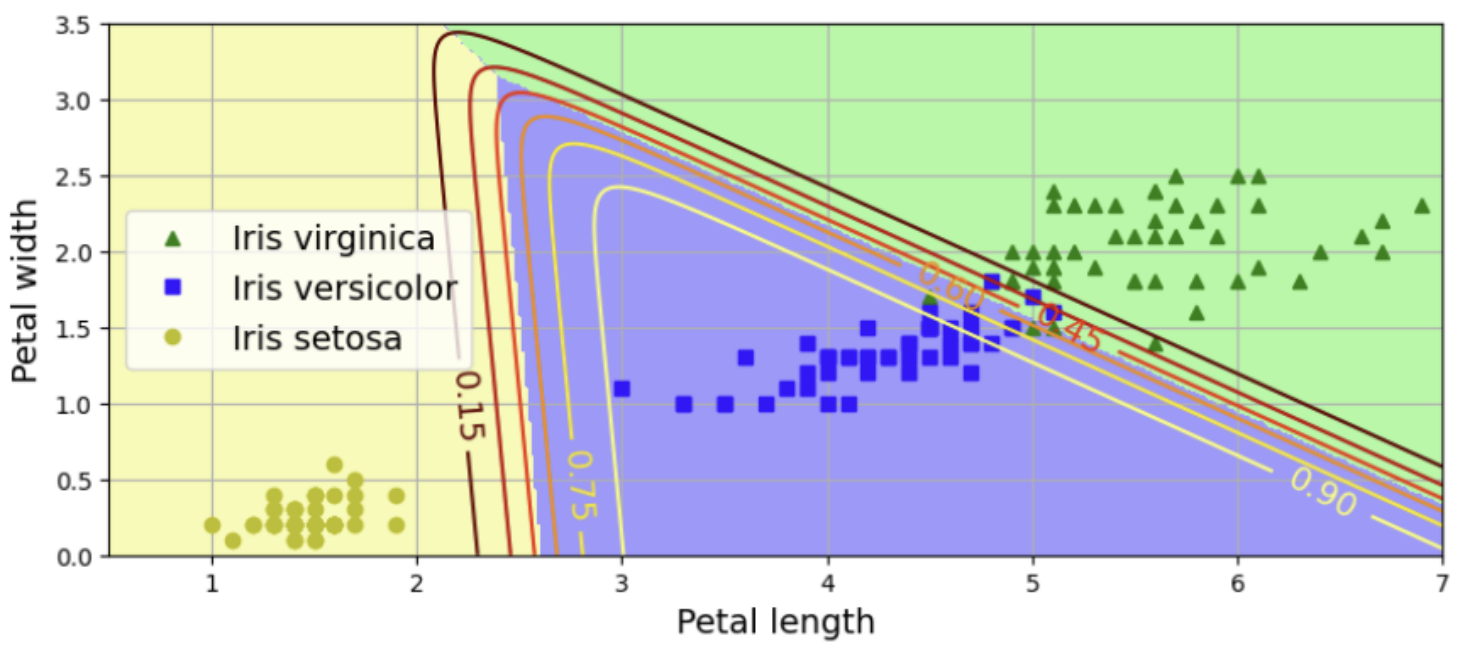
7 소프트맥스 회귀

다중 클래스 분류 예제

```
X = iris.data[["petal length (cm)", "petal width (cm)"]].values
y = iris["target"]
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

softmax_reg = LogisticRegression(C=30, random_state=42)
softmax_reg.fit(X_train, y_train)
```

```
LogisticRegression(C=30, random_state=42)
```



WeTIE

E.O.D

4주차. 모델 훈련