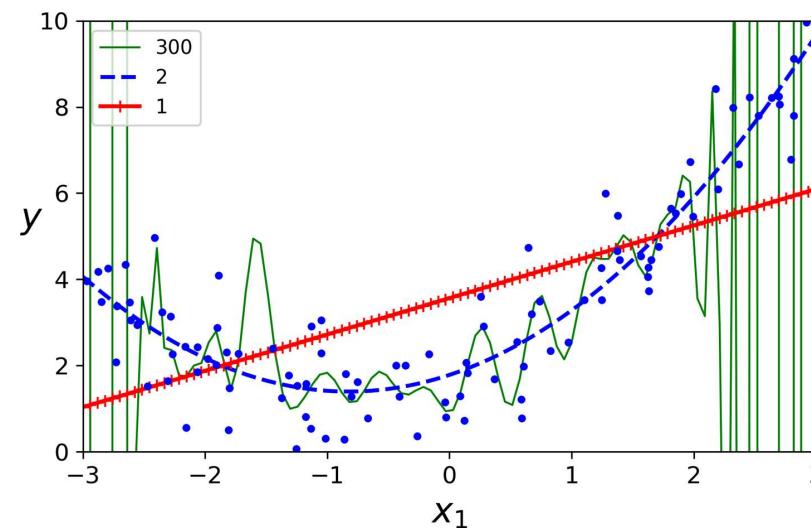


# 4장 모델 훈련 (2부)

## 4.4. 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.



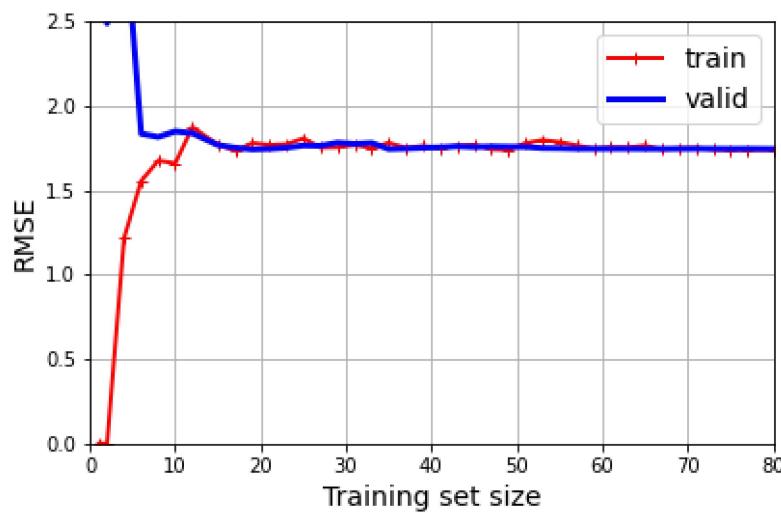
## 모델 성능 평가: 교차 검증 vs. 학습 곡선

모델 성능 평가는 보통 다음 두 가지 방식을 따른다.

- 교차 검증(2장 참고)
  - 과소적합: 훈련 점수와 교차 검증 점수 모두 낮은 경우
  - 과대적합: 훈련 점수는 높지만 교차 검증 점수가 상대적으로 많이 낮은 경우
- 학습 곡선 learning curve
  - 훈련셋과 검증셋에 대한 모델 성능을 비교하는 그래프
    - x-축: 훈련셋 크기. 훈련셋의 크기를 1%에서부터 출발해서 점차 키워 나가면서 모델 성능 평가
    - y-축: 훈련셋 크기에 따른 모델 성능. 훈련 점수와 검증 점수 사용
  - 학습 곡선의 모양에 따라 과소적합/과대적합 판정 가능

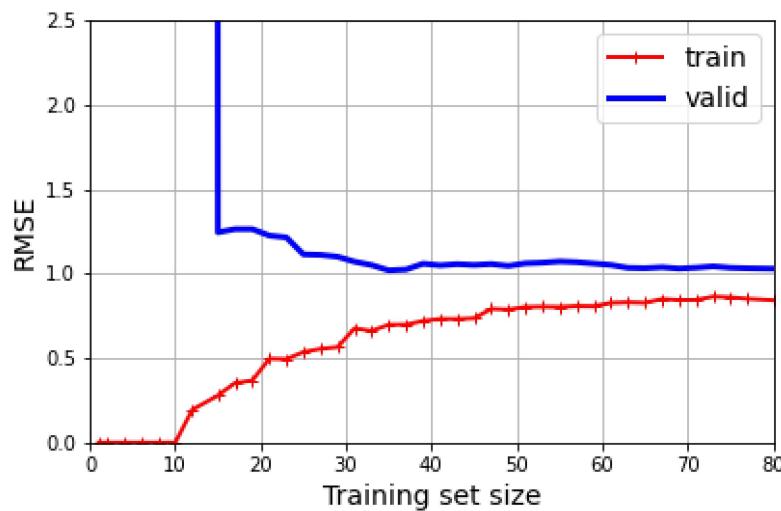
## 과소적합 모델의 학습 곡선 특징

- 2차 다항 함수의 분포를 따르는 데이터셋에 LinearRegression 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



## 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. 하지만 일반적으로 매우 어렵거나 불가능.



## 모델의 일반화 성능

- 훈련 과정에서 다루지 않은 새로운 데이터 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

## 편향

- 데이터셋에 대한 모델링이 틀린 경우
- 예를 들어 실제로는 2차원 모델인데 1차원 모델을 사용하는 경우 발생
- 과소적합 발생 가능성 높음.

## 분산

- 모델이 훈련 데이터에 민감하게 반응하는 정도
- 고차 다항 회귀처럼 자유도<sup>degree of freedom</sup>가 높은 모델일 수록 분산이 커짐
- 모델의 자유도: 모델이 찾아야 하는 파라미터의 개수
- 과대적합 발생 가능성 높음.

## 편향-분산 트레이드 오프

- 복잡한 모델일 수록 편향을 줄고 분산은 커짐.
- 단순한 모델일 수록 편향은 커지고 분산은 줄어듦

## 4.5. 모델 규제와 조기 종료

## 모델 규제

- 릿지 회귀: 가중치의 절대값을 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

## 릿지 회귀

- 비용함수

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

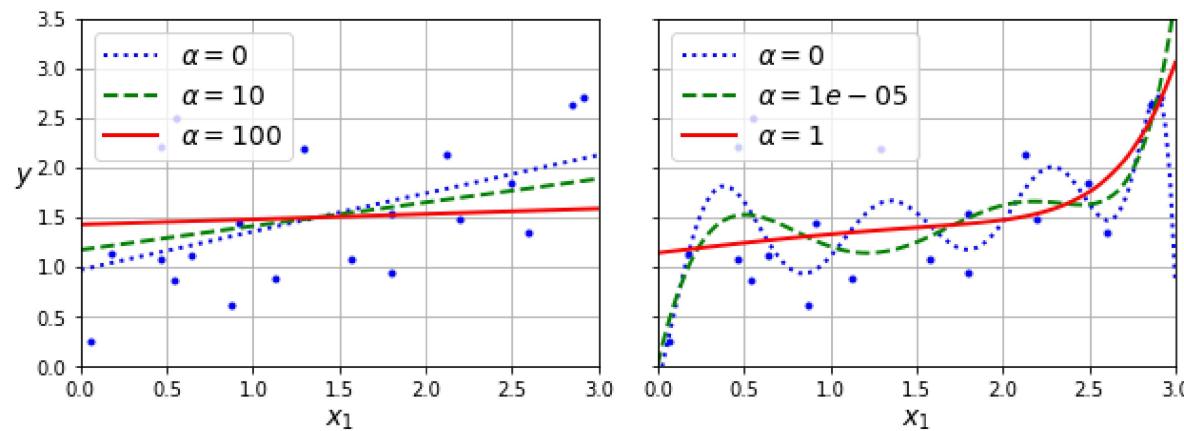
- 편향( $\theta_0$ ): 제하지 않음. 규제해선 않됨.
- $m_b$ : 배치 크기(하이퍼파라미터)
- $\alpha$ (알파): 규제 강도(하이퍼파라미터).
  - $\alpha = 0$ : 규제 없음
  - $\alpha$  값을 크게 잡으면 가중치( $\theta_i$ )의 절대값은 보다 0에 가깝워지도록 유도됨.  
즉, 가중치의 역할이 줄어듦.
- 주의사항: 특성 스케일링 전처리를 해야 규제 모델의 성능이 좋아짐.

## 릿지 회귀의 비용 함수 해석

- 비용 함수의  $J(\theta)$  전역 최솟값:  $\sum_{i=1}^n \theta_i^2$  값이 최대한 작은 지점
- $\theta_i$ 의 절대값을 가급적 0에 가깝게하는 방향으로 경사하강법이 작동
- 이렇게 하면 자연스럽게 모델의 분산도 작아짐.
- 즉, 입력값이 조금 변하더라도 예측값이 변하는 정도가 약해짐.
- 예측값  $\hat{y}$ 를 계산할 때  $\theta_i$ 가 미치는 영향이 작아지기 때문임.

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

- 릿지 규제를 적용한 6 가지 경우: 분산 줄고 편향 늘어남.
  - 왼편: 선형 회귀 모델에 세 개의  $\alpha$  값 적용.
  - 오른편: 10차 다항 회귀 모델에 세 개의  $\alpha$  값 적용.



## 라쏘 회귀

- 비용함수

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

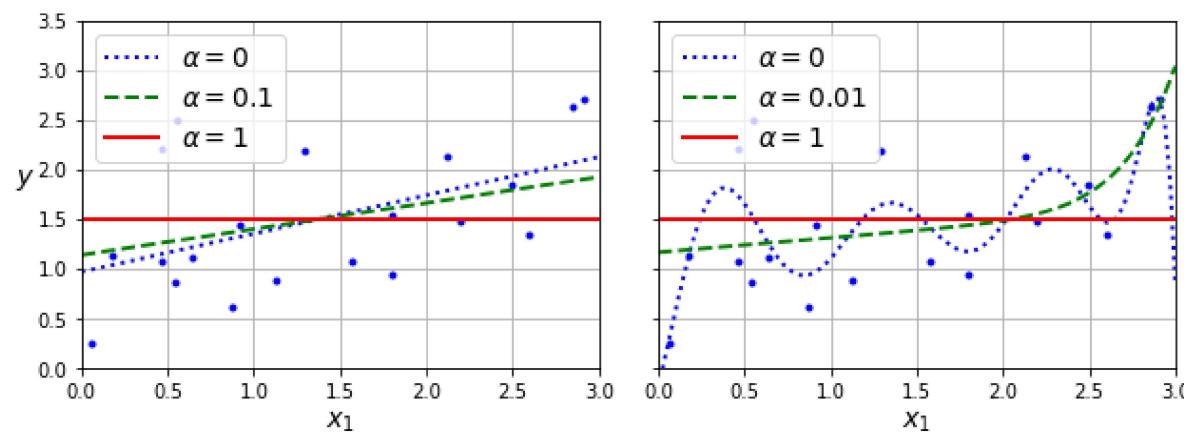
- 덜 중요한 특성을 무시하기 위해 해당 특성의 가중치  $|\theta_i|$ 를 보다 빠르게 0에 수렴하도록 유도. 또한 기본적으로  $|\theta_i|$ 가 가능하면 작게 움직이도록 유도.

## 라쏘 회귀의 비용 함수 해석

- 비용 함수의  $J(\theta)$  전역 최솟값:  $\sum_{i=1}^n |\theta_i|$  값이 최대한 작은 지점
- 별로 중요하지 않은 특성에 대해  $\theta_i$ 가 0에 빠르게 수렴하도록 훈련 중에 유도됨
- 이유:  $|\theta_i|$ 의 미분값이 1또는 -1 이기에  $\theta_i$ 가 0이 아닌 이상 비용 함수의 그레이디언트의 크기가 0에 가까워 지기가 상대적으로 어렵기 때문
- 경사하강법을 반복적으로 적용하는 훈련 과정동안 중요하지 않은 특성의 가중치를 빠르게 0으로 수렴하게 만듦.
- 대신 중요한 특성의 가중치를 결정하는 데에 보다 집중

라쏘 규제를 적용한 6 가지의 경우: 분산 줄고 편향 늘어남.

- 왼편: 선형 회귀 모델에 세 개의  $\alpha$  값 적용.
- 오른편: 10차 다항 회귀 모델에 세 개의  $\alpha$  값 적용.



## 엘라스틱 넷 회귀

- 비용함수

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

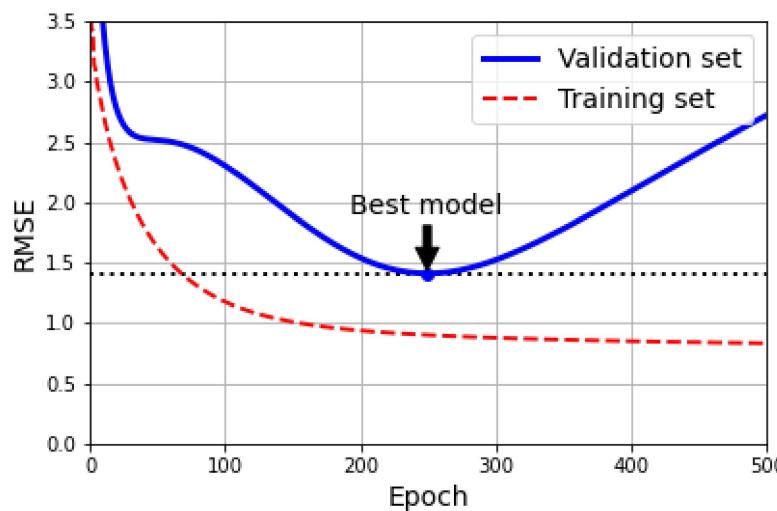
- 릿지 회귀와 라쏘 회귀를 절충한 모델
- 혼합 비율  $r$ 을 이용하여 릿지 규제와 라쏘 규제를 적절하게 조절

## 규제 선택

- 대부분의 경우 약간이라도 규제 사용 추천
- 유용한 속성이 많지 않다고 판단되는 경우
  - 라쏘 규제나 엘라스틱 넷 활용 추천
  - 불필요한 속성의 가중치를 0으로 만들기 때문
- 일반적으로 엘라스틱 넷을 보다 많이 추천

## 조기 종료

- 모델이 훈련 중에 훈련셋에 너무 과하게 적응하지 못하도록 하는 가장 일반적인 규제 기법
- 에포크가 남아있다 하더라도 검증셋 대한 비용함수의 값이 줄어 들다가 다시 커지는 순간 훈련 종료
- 검증셋에 대한 비용 함수의 곡선이 진동이 발생할 있기에 검증 손실이 한동안 최솟값보다 높게 유지될 때 훈련 멈추고 기억해둔 최적의 모델 사용
- 아래 그래프: 250 에포크 정도에서 훈련 조기 종료



## 확률적 경사하강법과 조기 종료

아래 코드는 `SGDRegressor` 모델에 조기 종료를 지정한다.

- `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

```
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)
```

## 4.6 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

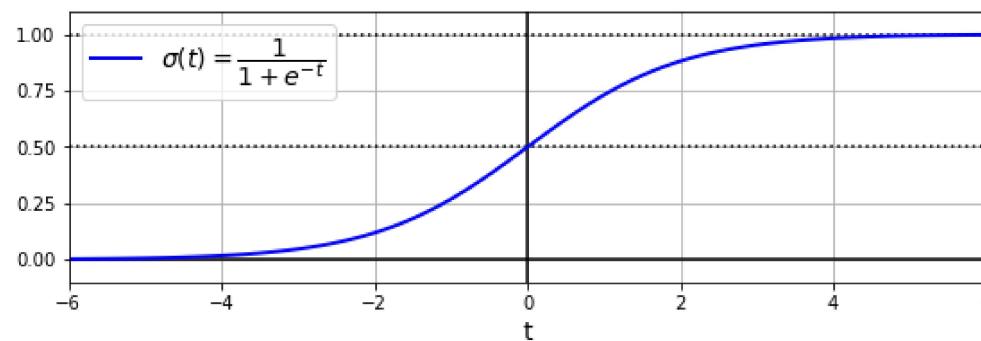
## 확률 예측

- 로지스틱 회귀 모델에서 샘플  $\mathbf{x}$ 가 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

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



## 예측값

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

- 양성 클래스인 경우:

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

- 음성 클래스인 경우:

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

## 비용함수

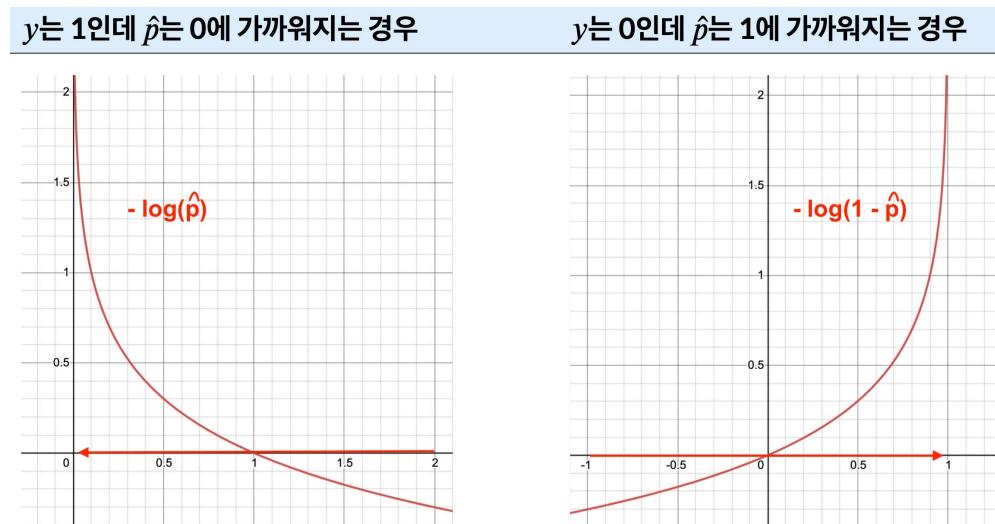
- 비용함수: 로그 손실 $\log \text{loss}$  함수 사용

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

- 모델 훈련: 위 비용함수에 대해 경사 하강법 적용

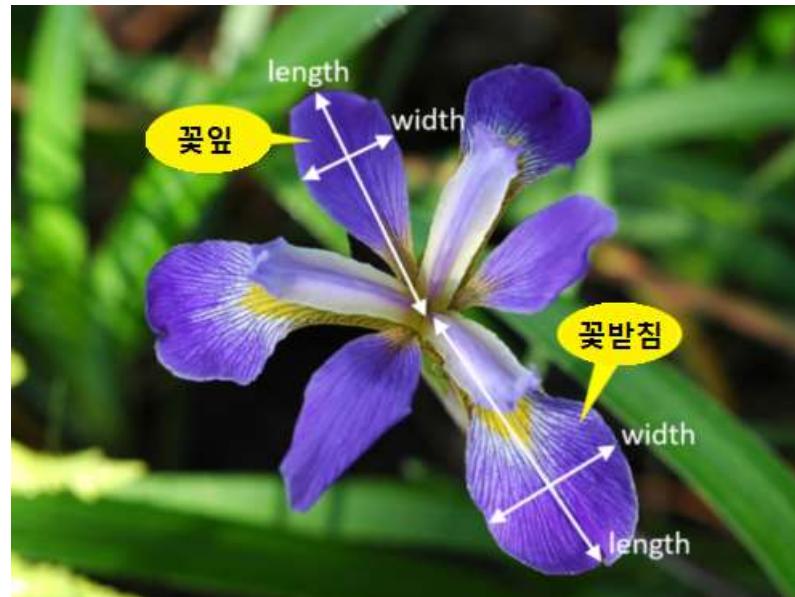
## 로그 손실 함수 이해

- 틀린 예측을 하면 손실값이 무한이 커짐
- 아래 왼쪽 그림: 샘플의 라벨이 1(양성)인데 예측 확률( $\hat{p}$ )이 0에 가까운 경우 로그 손실이 매우 클 수 있음
- 아래 오른쪽 그림: 샘플의 라벨이 0(음성)인데 예측 확률( $\hat{p}$ )이 1에 가까운 경우 로그 손실이 매우 클 수 있음



## 붓꽃 데이터셋

- 붓꽃의 품종 분류를 로지스틱 회귀로 진행
- 붓꽃 데이터셋의 샘플의 특성 4개:
  - 꽃받침<sub>sepal</sub>의 길이와 너비,
  - 꽃입<sub>petal</sub>의 길이와 너비



## 붓꽃 데이터셋의 라벨

- 0: Iris-Setosa(세토사)
- 1: Iris-Versicolor(버시컬러)
- 2: Iris-Virginica(버지니카)



## 붓꽃 데이터셋 불러오기

사이킷런 자체 제공

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

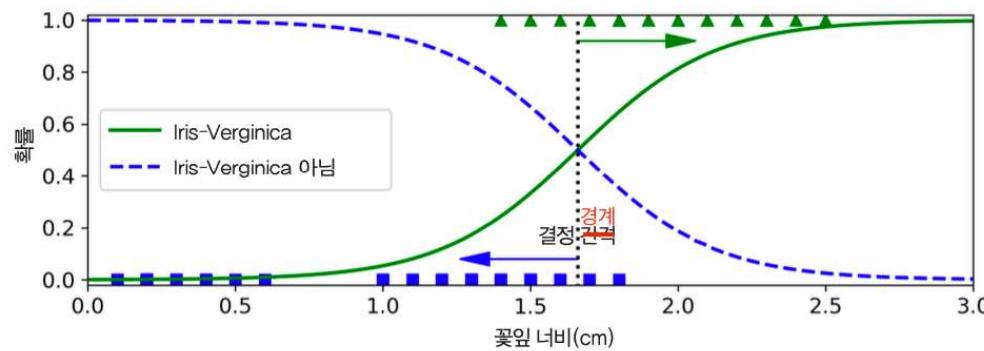
## Bunch 자료형

- `load_iris()` 함수는 데이터셋을 사전 자료형과 유사한 `Bunch` 자료형으로 불러온다.
- `Bunch` 자료형은 키를 사용한 인덱싱을 마치 클래스의 속성을 확인하는 방식으로 다룰 수 있음
  - 예제: `iris['data']` 대시 `iris.data` 사용 가능
- `data` 키: 4개의 특성으로 구성된 훈련셋 데이터프레임`DataFrame`
- `target` 키: 라벨셋 시리즈`Series`
- 첫 5개 데이터 샘플

	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

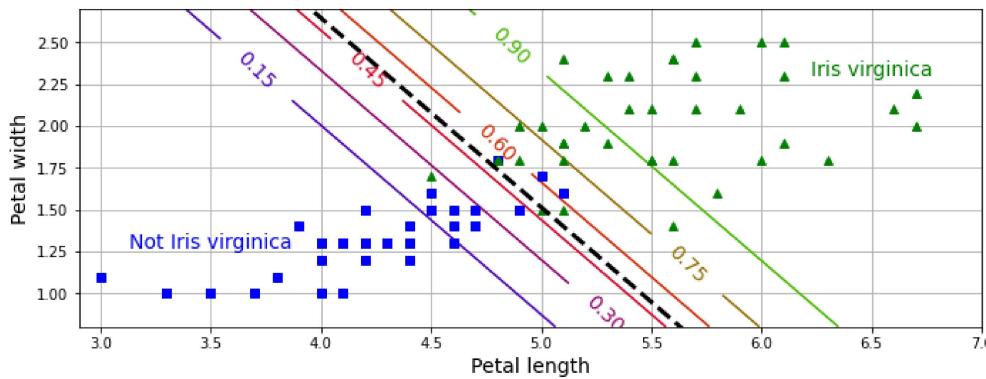
## 결정 경계: 꽃잎의 너비 기준 Iris-Virginica 여부 판정

```
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)
```



## 결정 경계: 꽃잎의 너비, 길이 기준 Iris-Virginica 여부 판정

```
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)  
  
log_reg = LogisticRegression(C=2, random_state=42)  
log_reg.fit(X_train, y_train)
```



## 로지스틱 회귀 규제

- 하이퍼파라미터 `penalty` 와 `C` 이용
- `penalty`
  - `l1`, `l2`, `elasticnet` 세 개중에 하나 사용.
  - 기본은 `l2`, 즉,  $\ell_2$  규제를 사용하는 릿지 규제.
  - `elasticnet` 을 선택한 경우 `l1_ratio` 옵션 값을 함께 지정.
- `C`
  - 릿지 또는 라쏘 규제 정도를 지정하는  $\alpha$ 의 역수에 해당.
  - 따라서 0에 가까울 수록 강한 규제 의미.

## 4.7. 소프트맥스(softmax) 회귀

- 로지스틱 회귀 모델을 일반화하여 다중 클래스 분류를 지원하도록 한 회귀 모델

## 소프트맥스 회귀 학습 아이디어

- 샘플  $\mathbf{x} = [x_1, \dots, x_n]$  가 주어졌을 때 각각의 분류 클래스  $k$  에 대한 점수  $s_k(\mathbf{x})$  계산. 즉,  $k*(n+1)$  개의 파라미터를 학습시켜야 함.

$$s_k(\mathbf{x}) = \theta_0^{(k)} + \theta_1^{(k)} x_1 + \cdots + \theta_n^{(k)} x_n = [\mathbf{1}, \mathbf{x}_1, \dots, \mathbf{x}_n] \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \\ \vdots \\ \theta_n^{(k)} \end{bmatrix}$$

## 가중치 어레이

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

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

## 소프트맥스 함수: $\sigma()$

- $K$ : 클래스(라벨)의 개수
- $\mathbf{s}(\mathbf{x}) = [s_1(\mathbf{x}), \dots, s_K(\mathbf{x})]$
- $\sigma()$  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  $\theta_k^{(i)}$  학습
  - $K = 2$ 이면 로지스틱 회귀의 로그 손실 함수와 정확하게 일치.
- 비용함수: 크로스 엔트로피 비용 함수 사용

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

## 다중 클래스 분류 예제

- 사이킷런의 LogisticRegression 예측기 활용
  - solver=lbfgs: 기본값이며 다중 클래스 분류에서 자동으로 소프트맥스 사용.

```
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) # 조금 약한 alpha 규제  
softmax_reg.fit(X_train, y_train)
```

- 붓꽃 꽃잎의 너비와 길이를 기준으로 품종 분류
  - 결정경계: 배경색으로 구분
  - 곡선: 버시컬러 품종에 속할 확률

