

# 머신러닝 모델 훈련 기 법

## 주요 내용

- 최적화, 일반화, 과대적합
- 다양체 가설
- 모델 평가
- 모델 훈련 최적화
- 모델 일반화 성능 향상 기법

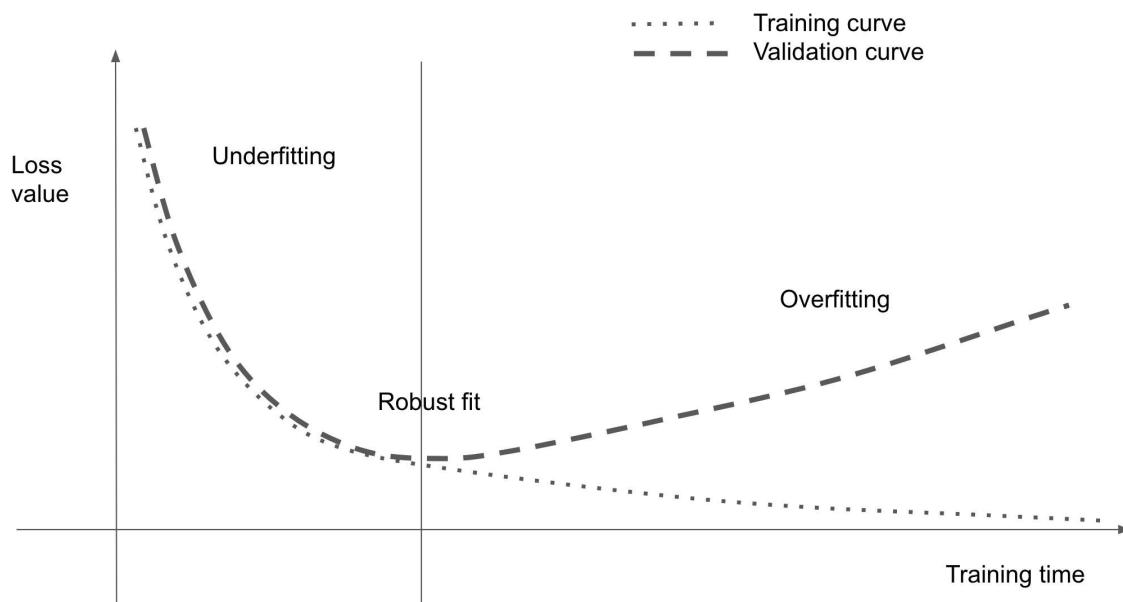
최적화, 일반화, 과대적합

## 머신러닝 모델 훈련의 핵심: 최적화대 일반화

머신러닝 모델 훈련의 주요 과제는 모델 훈련의 **최적화** optimization과 모델 **일반화** generalization 사이의 적절한 관계 찾기이다.

- **최적화**: 훈련셋에 대해 가장 좋은 성능 이끌어내기
- **일반화**: 훈련 과정에서 보지 못한 데이터를 처리하는 능력 향상시키기

## 과대적합 대 과소적합

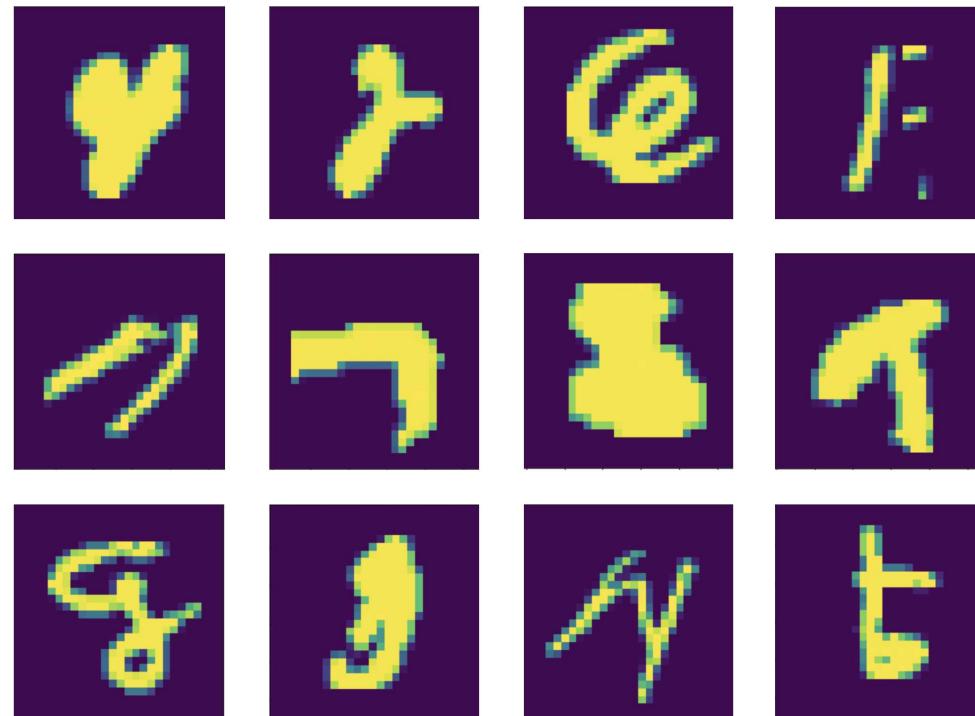


과대적합 발생 주요 요인

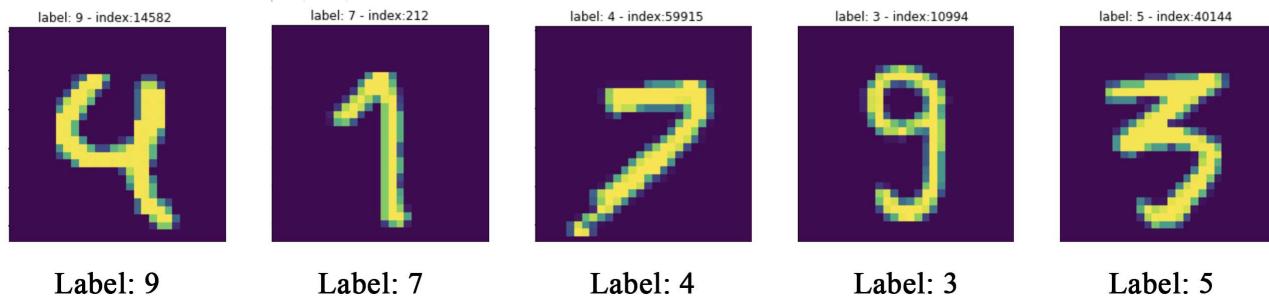
## 첫째, 훈련셋에 포함된 노이즈

적절하지 않은 데이터 또는 잘못된 라벨을 갖는 데이터 등을 노이즈noise라 부른다.

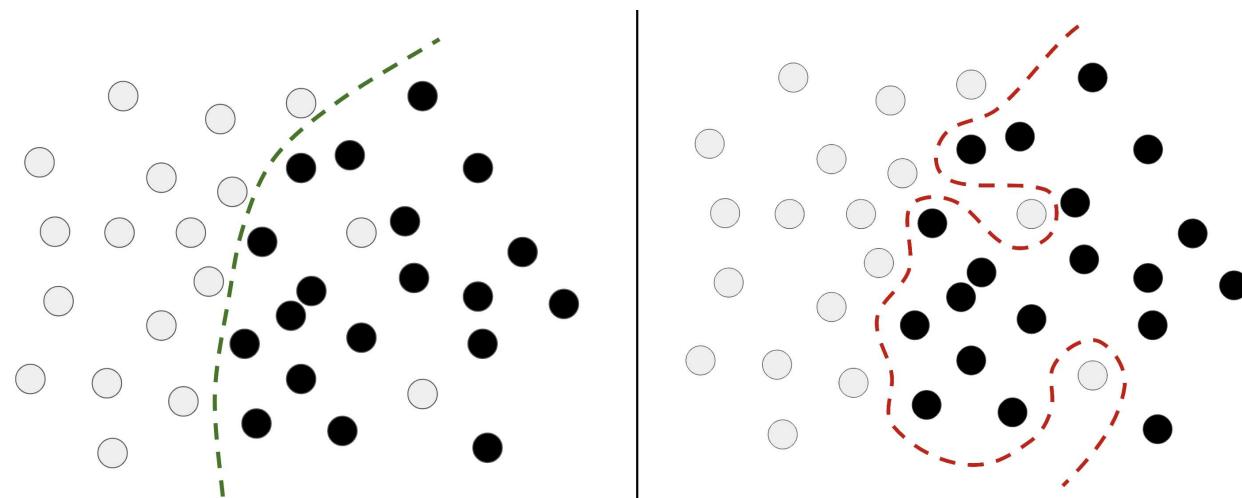
- 적절하지 않은 데이터: 다음 MNIST 이미지들처럼 불분명하면 특성 파악이 어렵다.



- 잘못된 라벨: 예를 들어, 잘못 분류된 1처럼 생긴 이미지를 7로 잘못 분류할 가능성이 높아진다.



노이즈 등의 이상치 outlier를 학습하면 아래 오른편 그림의 경우처럼 모델이 이상치의 특별한 특성을 학습하게 되어 새로운 데이터에 대한 예측 성능이 불안정해지게 된다.

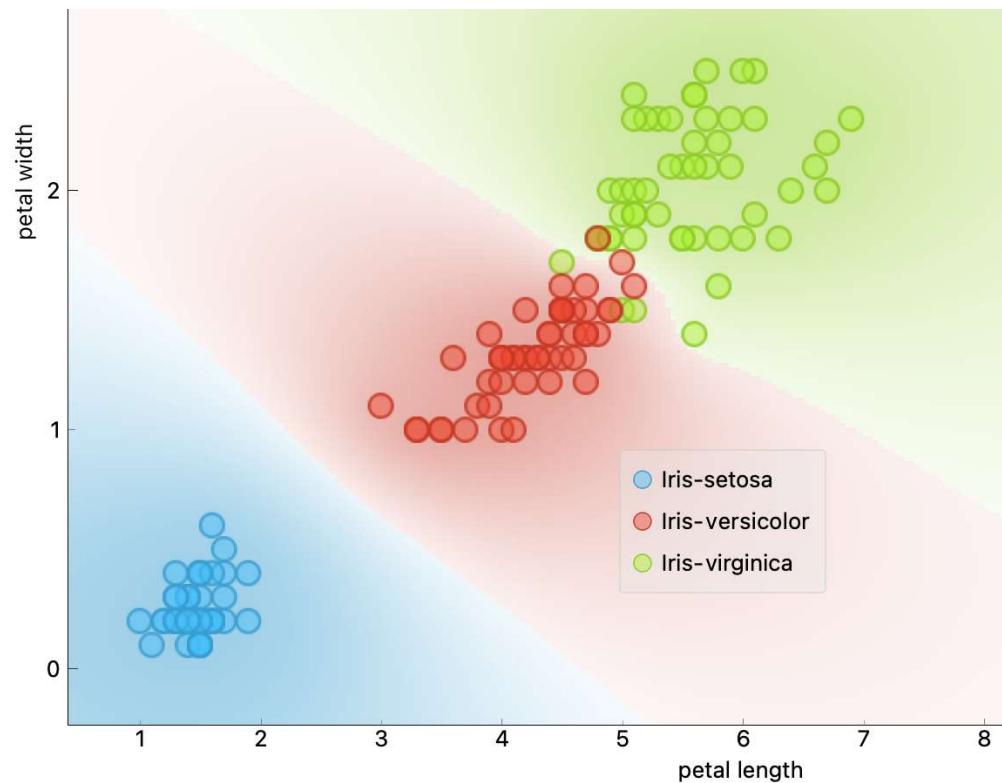


## 둘째, 애매한 특성

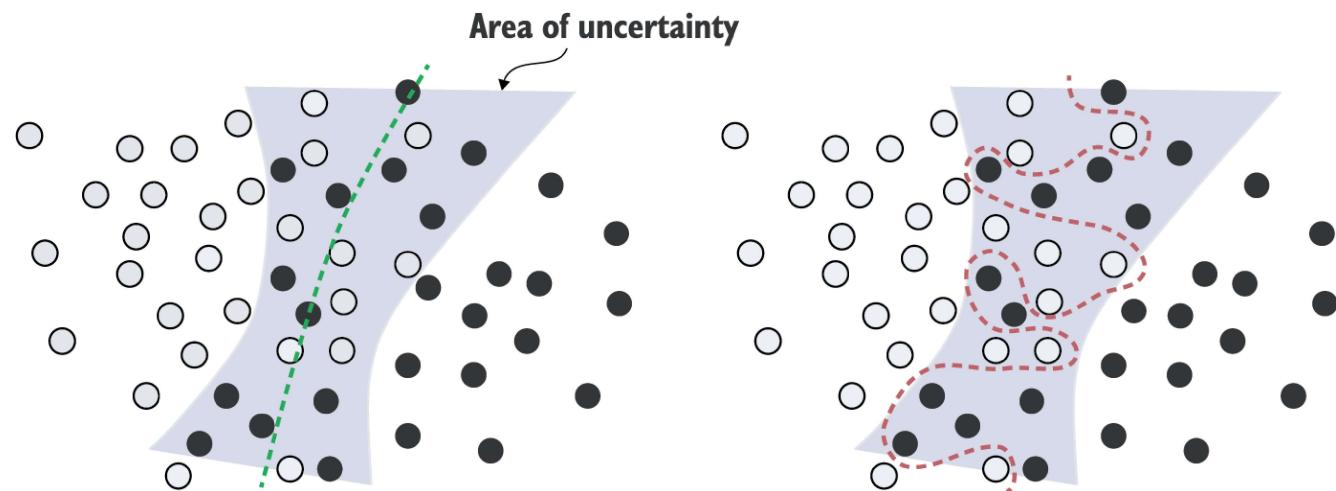
노이즈 등의 이상치가 전혀 없다 하더라도 특정 특성 영역에 대한 예측값이 여러 개의 값을 가질 수 있다.



예를 들어, 붓꽃 데이터셋의 경우 꽃잎의 길이와 너비만을 활용해서는 버시컬러*versicolor* 품종과 버지니카*virginica* 품종의 완벽한 구분이 애초에 불가능하다.



하지만 훈련을 오래 시키면 각 샘플의 특성을 해당 라벨의 고유의 특성으로 간주하는 정도 까지 모델이 훈련되어 아래 오른편 그림과 같이 샘플의 특성에 너무 민감하게 작동한다.



<그림 출처: [Deep Learning with Python\(2판\)](#)>

## **셋째: 특성과 타깃 사이의 거짓 상관관계**

특성과 타깃 사이의 거짓 상관관계를 유발하는 여러 상황이 있다.

- 매우 드문 특성을 사용하는 데이터셋으로 훈련하는 경우



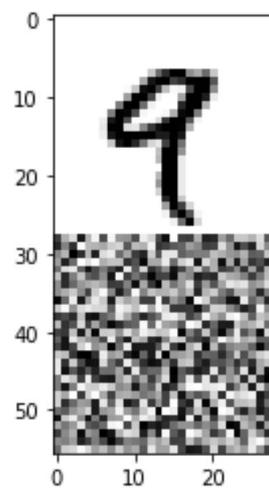
체리모야 열매

- 우연에 의한 경우

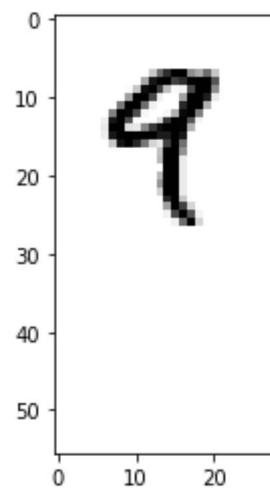
자주 발생하는 특성이라 하더라도 우연히 잘못된 편견을 훈련중인 모델에 심어줄 수 있다. 예를 들어, "너무"라는 단어를 포함한 100개의 영화 후기 중에서 54%는 긍정, 나머지 46%는 부정이었다면 훈련중인 모델은 "너무"라는 단어를 긍정적으로 평가할 가능성을 높힌다.

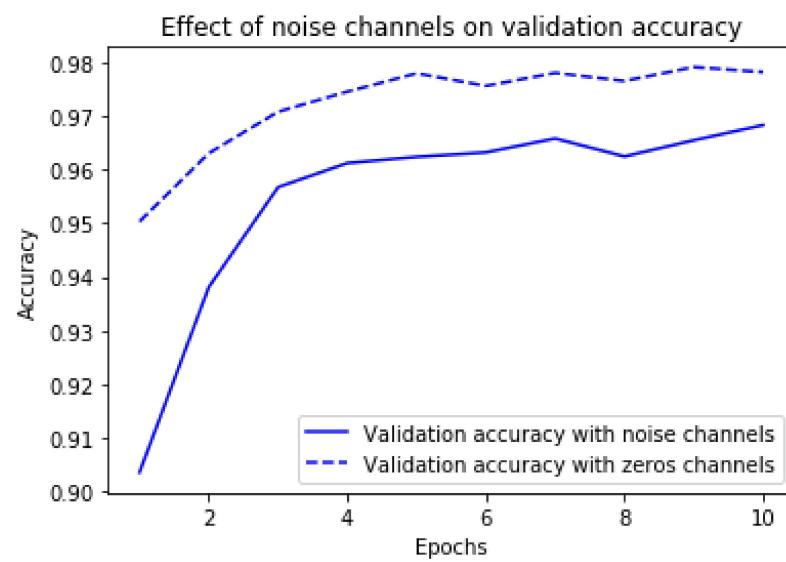
- 의미 없는 특성에 의한 경우

백색 잡음 추가 손글씨



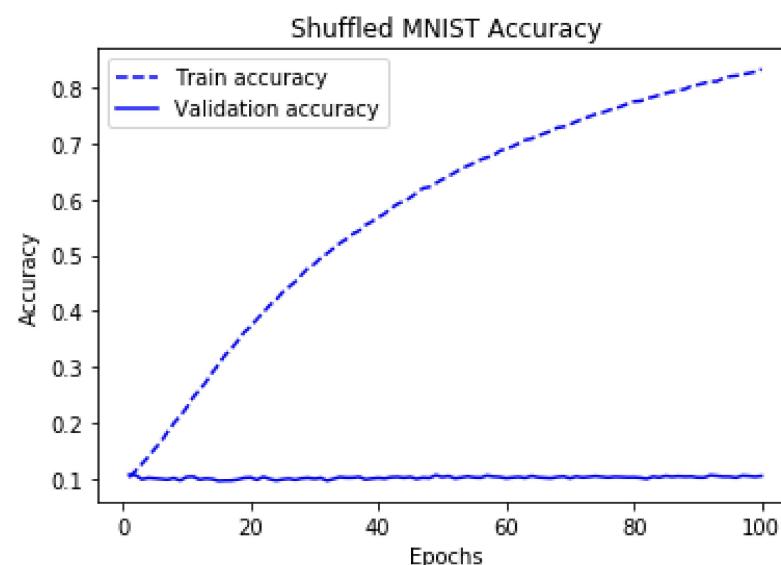
여백 추가 손글씨





# 다양체 가설과 일반화

딥러닝 모델은 어떤 무엇도 학습할 수 있다.



이는 다음 네 가지를 의미한다.

첫째, 일반화는 모델 훈련 과정 중에 제어할 수 있는 대상이 아니다.

둘째, 모델 훈련을 통해 할 수 있는 것은 주어진 훈련 데이터셋에 모델이 잘 적응하도록 하는 것 뿐이다.

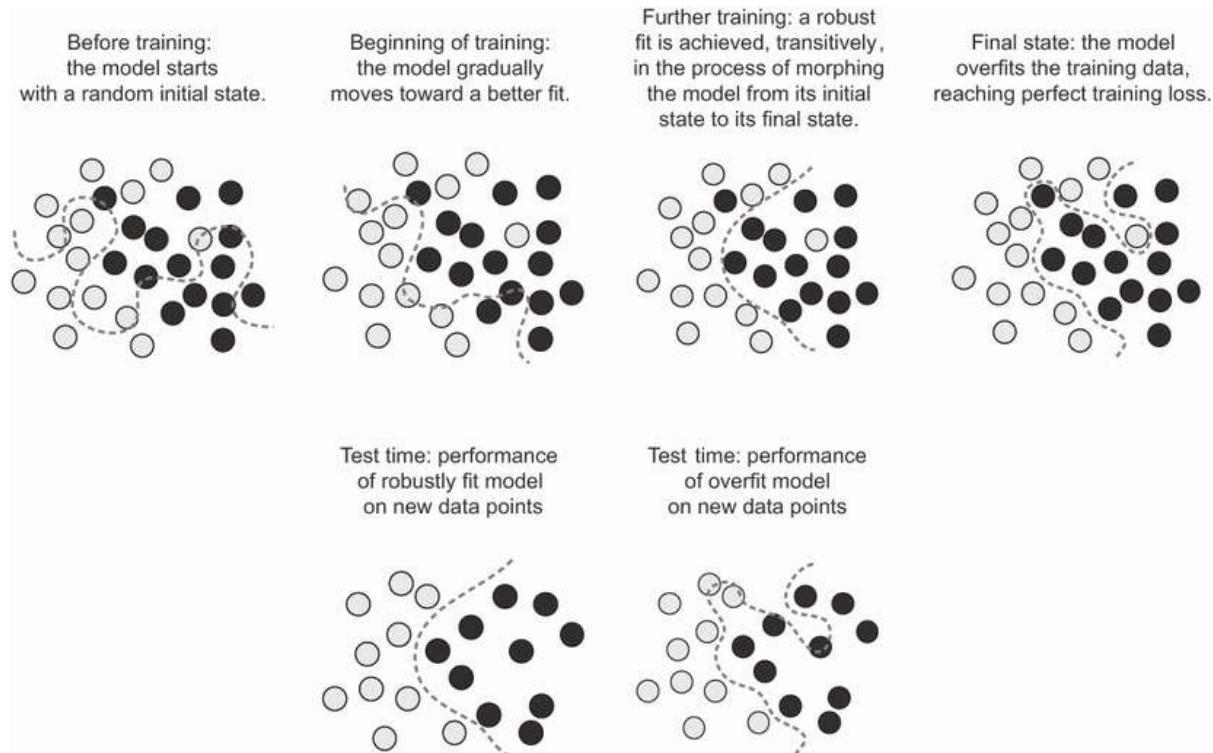
셋째, 딥러닝 모델은 어떤 데이터셋에도 적응할 수 있기에 너무 오래 훈련시키면 과대적합은 반드시 발생하고 일반화는 어려워진다.

넷째, 모델의 **일반화** 능력은 모델 자체보다는 **훈련셋에 내재하는 정보의 구조**와 보다 밀접히 관련된다.

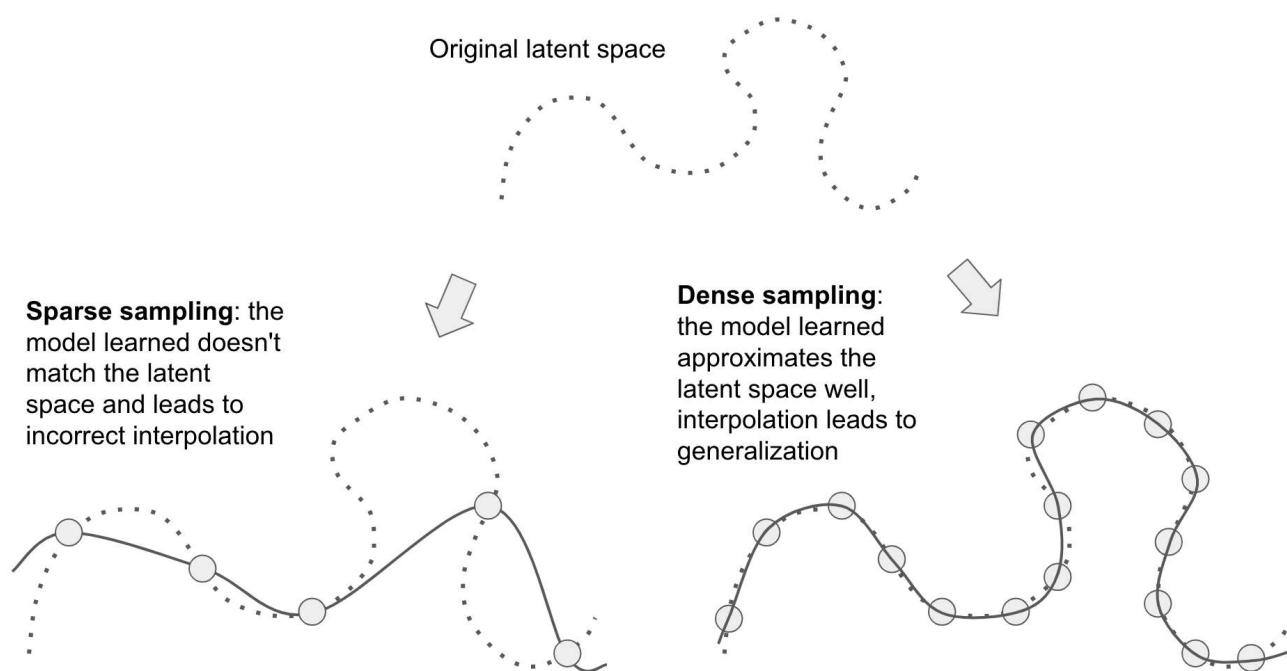
# 다양체 가설

다양체 가설 manifold hypothesis은 데이터셋에 내재하는 정보의 구조에 대한 다음과 같은 가설이다.

일반적인 데이터셋은 고차원상에 존재하는 (저차원의) 연속이며 미분 가능한 다양체를 구성한다.



# 보간법과 일반화



# 모델 평가

검증셋을 활용하여 모델의 일반화 능력을 평가하는 방법을 소개한다.

## 훈련셋, 검증셋, 테스트셋

- 테스트셋은 모델 구성과 훈련에 전혀 관여하지 않아야 한다.
- 구성된 모델의 성능을 평가하려면 테스트셋을 제외한 다른 데이터셋이 필요하다.
- 훈련셋의 일부를 검증셋으로 활용한다.

## 모델 튜닝

- 검증셋은 훈련 과정 중에 모델의 일반화 성능을 테스트하는 용도로 사용
- 모델 튜닝: 모델의 검증셋에 대한 성능 평가를 바탕으로 모델 구성과 모델의 **하이퍼파라미터**hyperparameter 설정 조정

## 정보 유출

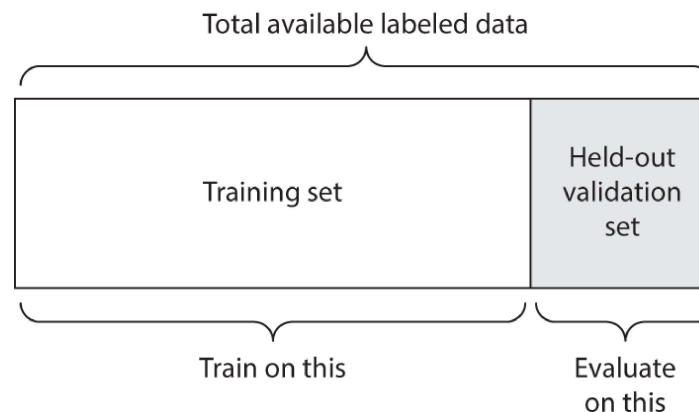
- 모델 튜닝도 모델의 좋은 하이퍼파라미터를 찾아가는 일종의 **학습**
- 튜닝을 많이 하게되면 검증셋에 특화된 모델이 얻어질 가능성이 커짐.
- 정보 유출: 검증셋에 과대적합된 모델이 훈련될 가능성이 높아지는 현상

## 모델 평가용 데이터셋 준비 관련 주의사항

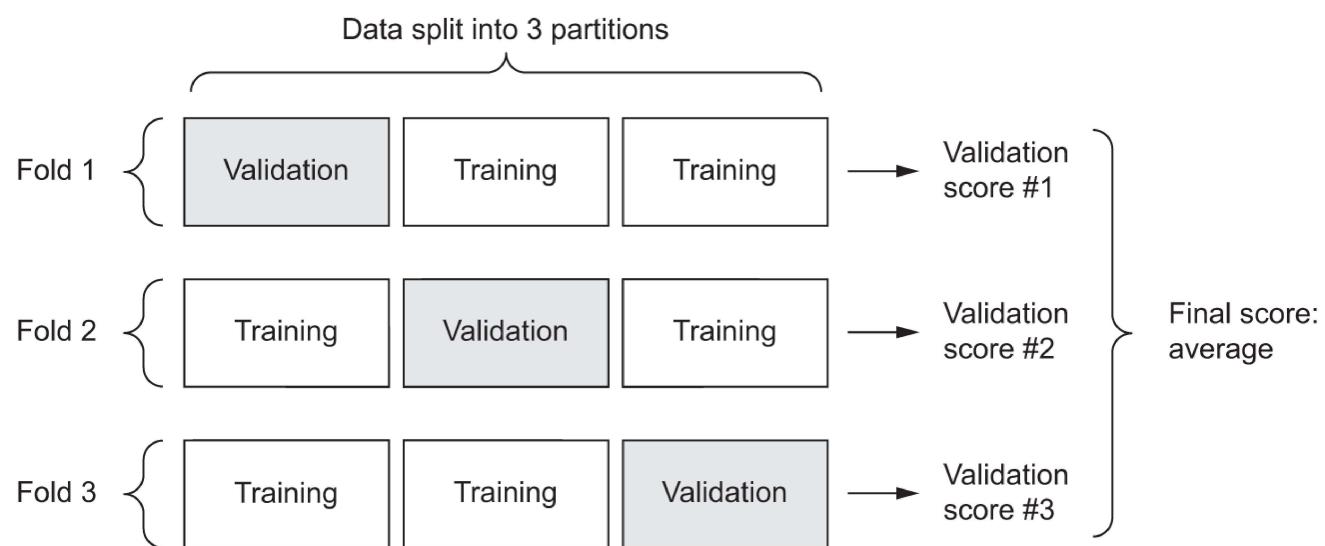
- **대표성:** 데이터셋 무작위 섞은 후 훈련셋, 검증셋, 데이터셋 준비
- **순서 준수:** 미래를 예측하는 모델을 훈련시킬 때, 테스트셋의 데이터는 훈련셋의 데이터보다 시간상 뒤쪽에 위치하도록 해야 함.
- **중복 데이터 제거:** 훈련셋과 테스트셋에 동일한 데이터가 들어가지 않도록 중복 데이터를 제거해야 함.

# 검증셋 활용법

## 홀드아웃 hold-out 검증



## K-겹 교차검증



## 모델 성능 평가의 기준선

- MNIST 데이터셋: 10%의 정확도
- IMDB 데이터셋: 50%의 정확도
- 로이터 통신 기사: 18-19%의 정확도. 기사들이 균등하게 분포되어 있지 않음.

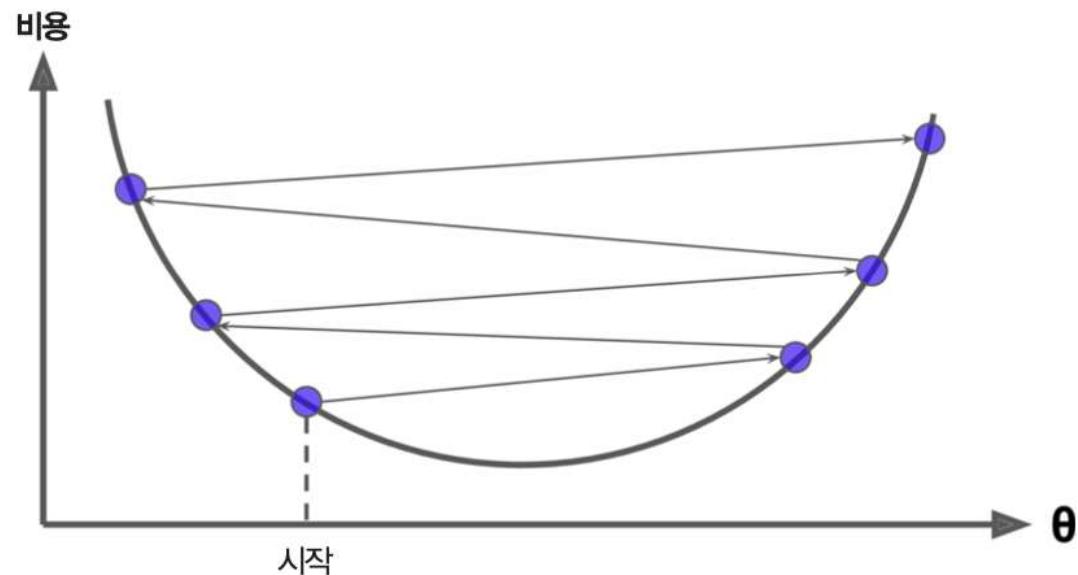
# 모델 훈련 최적화

## 첫째 경우: 경사하강법 관련 파라미터 조정

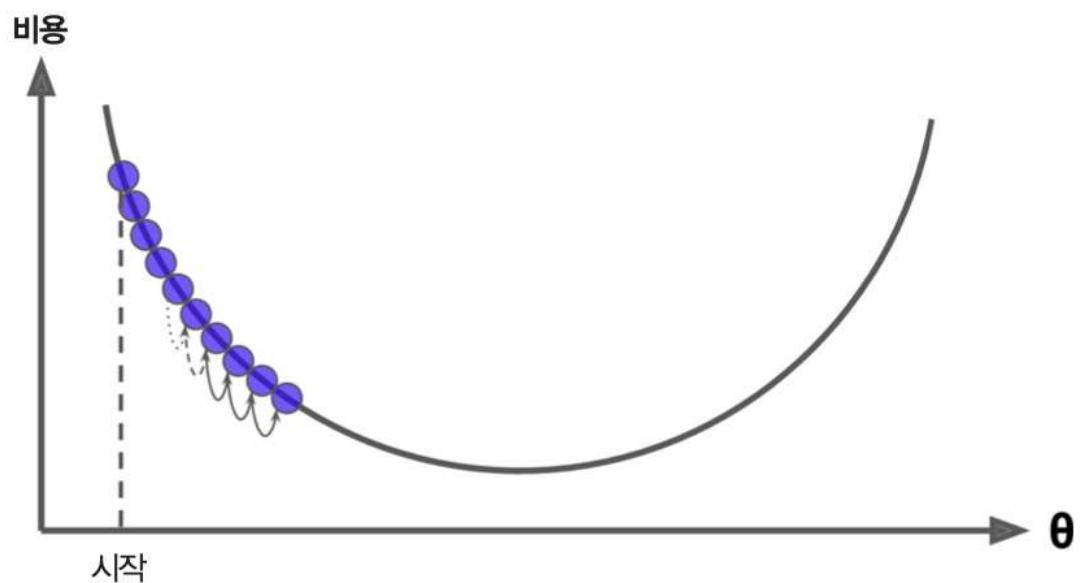
- 훈련셋의 손실값이 줄어들지 않거나 진동하는 등 훈련이 제대로 이루어지지 않는 경우
- 학습률과 배치 크기 조절

## 학습률 조정

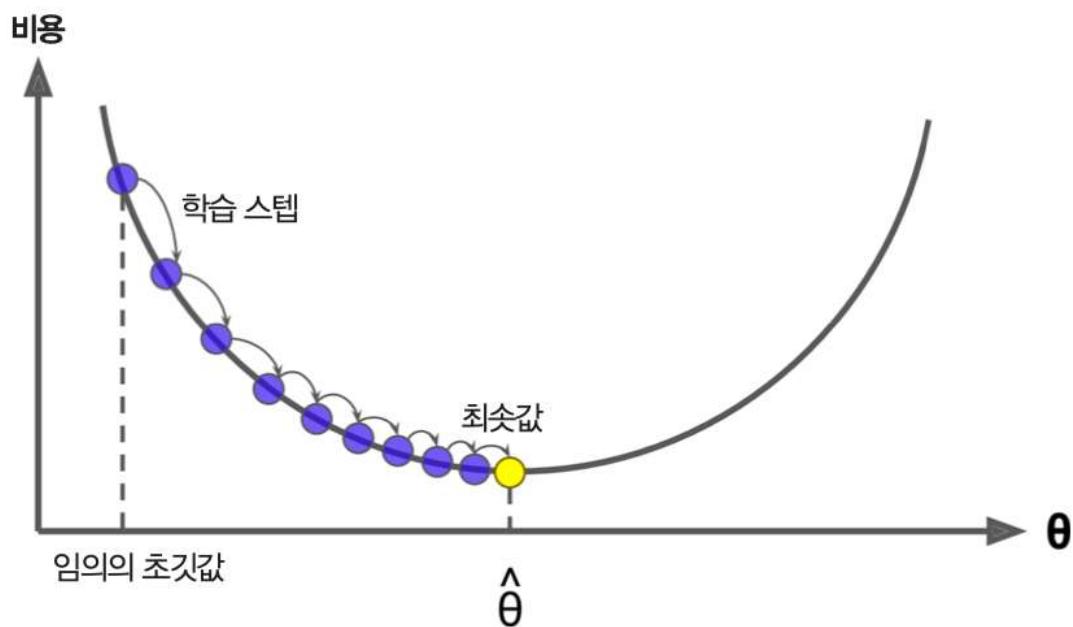
- 매우 큰 학습률을 사용하는 경우: 모델이 제대로 학습되지 않는다.



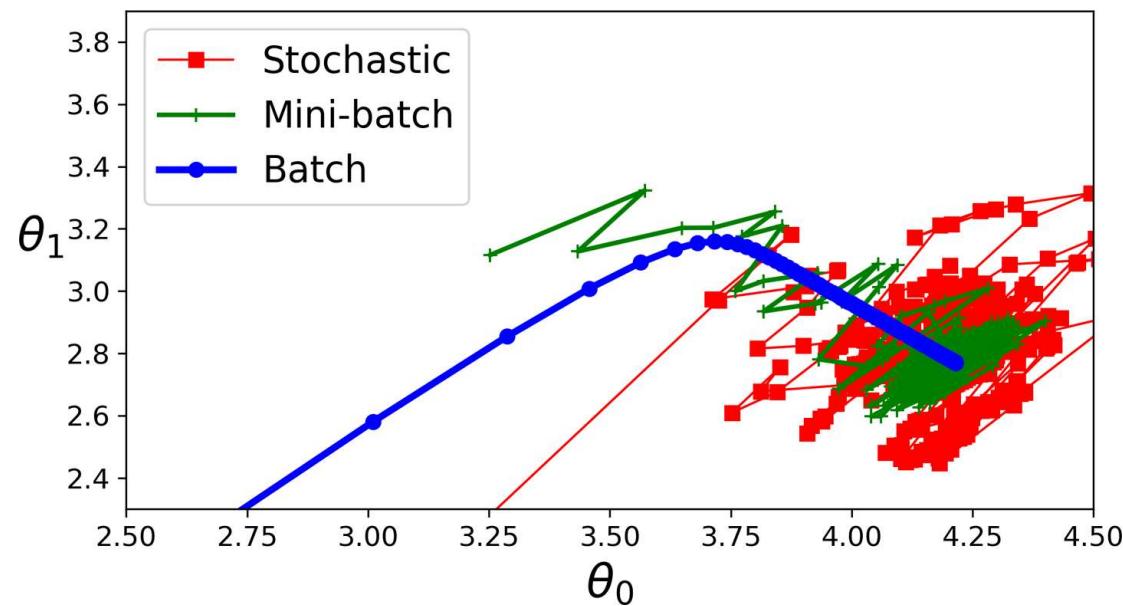
- 매우 작은 학습률을 사용하는 경우: 모델이 너무 느리게 학습된다.



- 적절한 학습률을 사용하는 경우: 모델이 적절한 속도로 학습된다.



## 배치 크기 조정



## 둘째 경우: 보다 적절한 모델 사용

훈련은 잘 진행되는데 검증셋에 대한 성능이 좋아지지 않는다면 다음 두 가지 경우를 의심해 보아야 한다.

- 훈련셋이 적절한지 않은 경우
  - 예제: 라벨이 무작위로 섞인 MNIST 데이터셋
- 사용하는 모델이 적절하지 않은 경우
  - 예제: 선형 분류가 불가능한 데이터셋에 선형분류 모델을 적용하는 경우
  - 예제: 시계열 데이터 분석에 앞서 살펴본 Sequential 모델을 사용하는 경우

## **셋째 경우: 모델의 정보 저장 능력 조정**

- 모델의 훈련셋/검증셋의 평가지표가 계속 향상되지만 과대적합이 발생하지 않는 경우
- 기본적으로 모델의 정보 저장 능력을 키워야 한다.
- 즉, 신경망 모델의 은닉층 또는 층에 사용되는 유닛의 수를 증가시킨다.

# 일반화 향상법

- 양질의 데이터셋
- 특성 공학
- 조기 종료
- 규제

## 양질의 데이터셋

- 양질의 데이터를 보다 많이 수집하는 일이 보다 적절한 모델을 찾으려는 노력보다 값어치가 높음.
- 충분한 양의 샘플: 훈련셋이 크면 클 수록 일반화 성능이 좋아짐.
- 타깃(라벨) 지정 오류 최소화.
- 적절한 데이터 전처리: 데이터 클리닝과 결측치 처리.
- 유용한 특성 선택: 주요 특성에 집중하면 훈련 시간도 줄이면서 동시에 보다 높은 일반화 성능의 모델을 훈련시킬 수 있음.

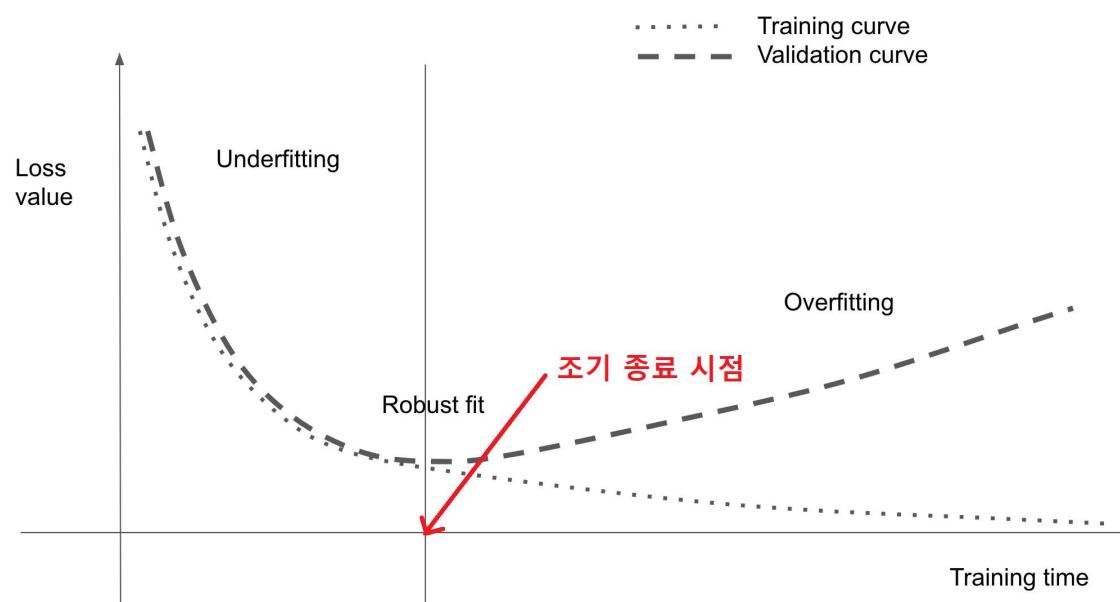
## 특성 공학

- 유용한 특성으로 구성된 훈련셋으로 모델을 훈련하면 보다 적은 양의 데이터로 보다 효율적인 훈련 가능
- 특성 공학: 모델 훈련에 가장 유용한 특성으로 구성된 데이터셋을 준비하는 과정

## 예제: 시간 읽기

Raw data: pixel grid		
Better features: clock hands' coordinates	{x1: 0.7, y1: 0.7} {x2: 0.5, y2: 0.0}	{x1: 0.0, y1: 1.0} {x2: -0.38, y2: 0.32}
Even better features: angles of clock hands	theta1: 45 theta2: 0	theta1: 90 theta2: 140

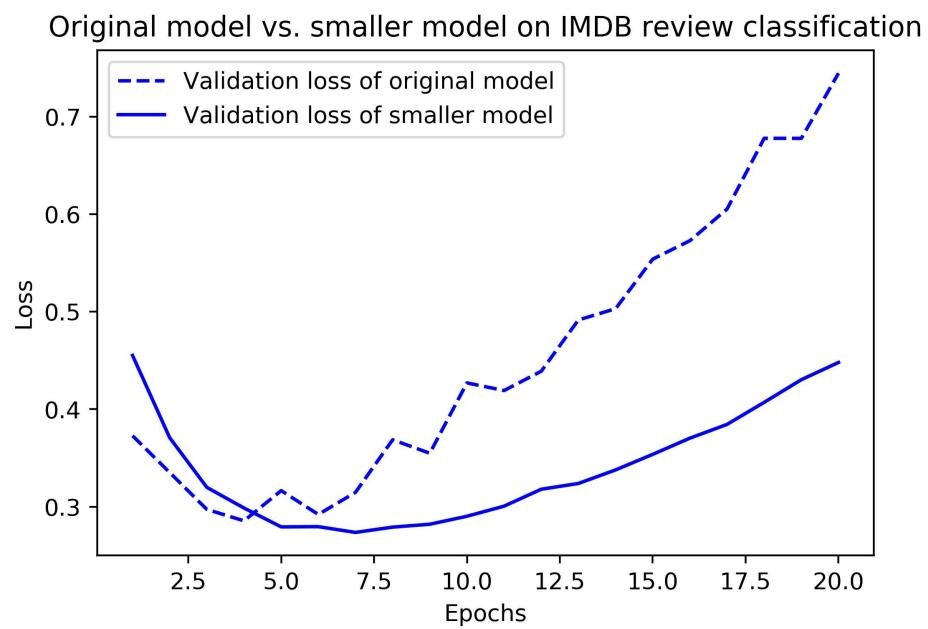
## 조기 종료



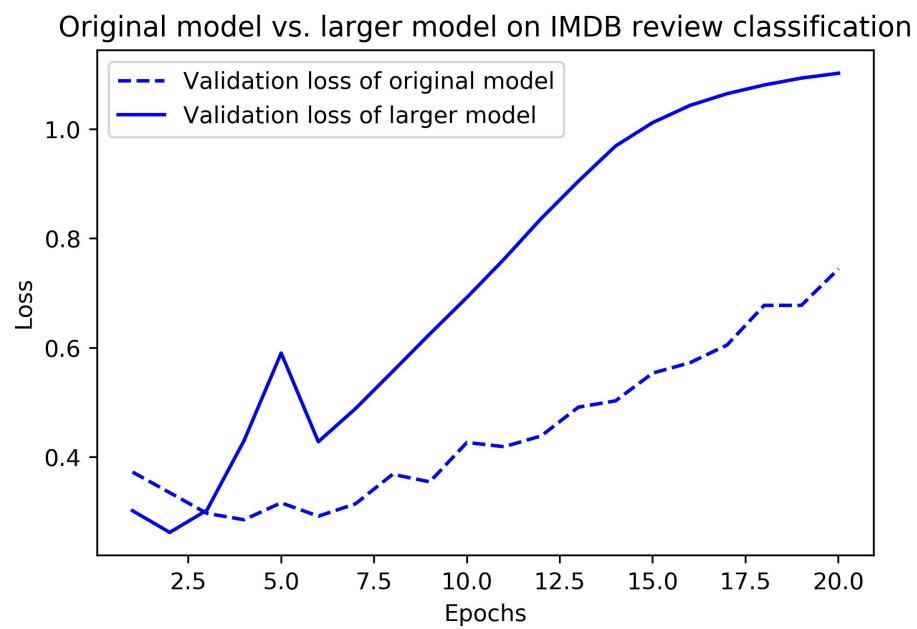
규제

규제 기법 1: 신경망 크기 축소

- 작은 모델



- 큰 모델



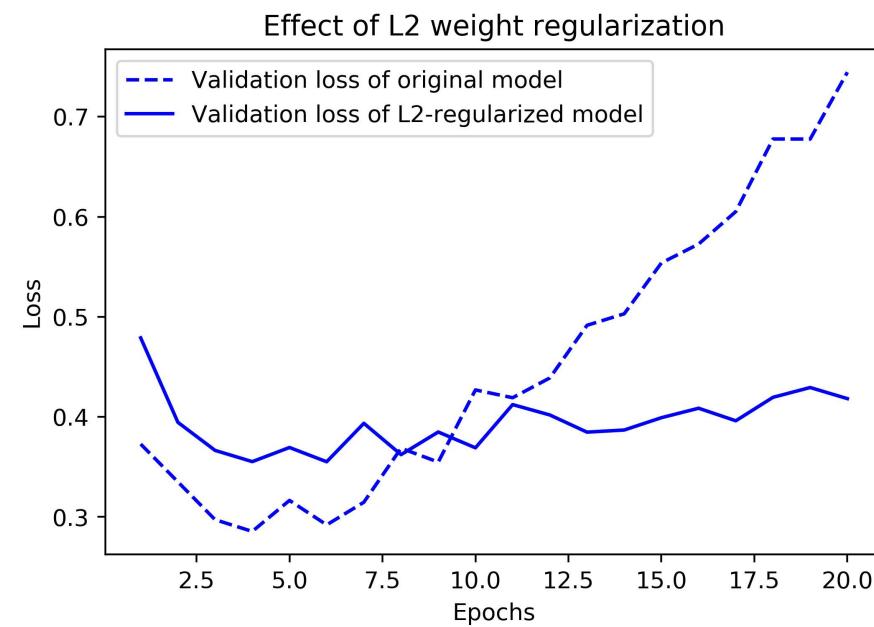
## 규제 기법 2: 가중치 규제

- L1 규제
- L2 규제

```
layers.Dense(16,
             kernel_regularizer=regularizers.l2(0.002),
             activation="relu")

layers.Dense(16,
             kernel_regularizer=regularizers.l1(0.001),
             activation="relu")

layers.Dense(16,
             kernel_regularizer=regularizers.l1_l2(l1=0.001, l2=0.002),
             activation="relu")
```



규제는 훈련 중에만 적용되며 실전에는 사용되지 않는다.

## 규제 기법 3: 드롭아웃 적용

