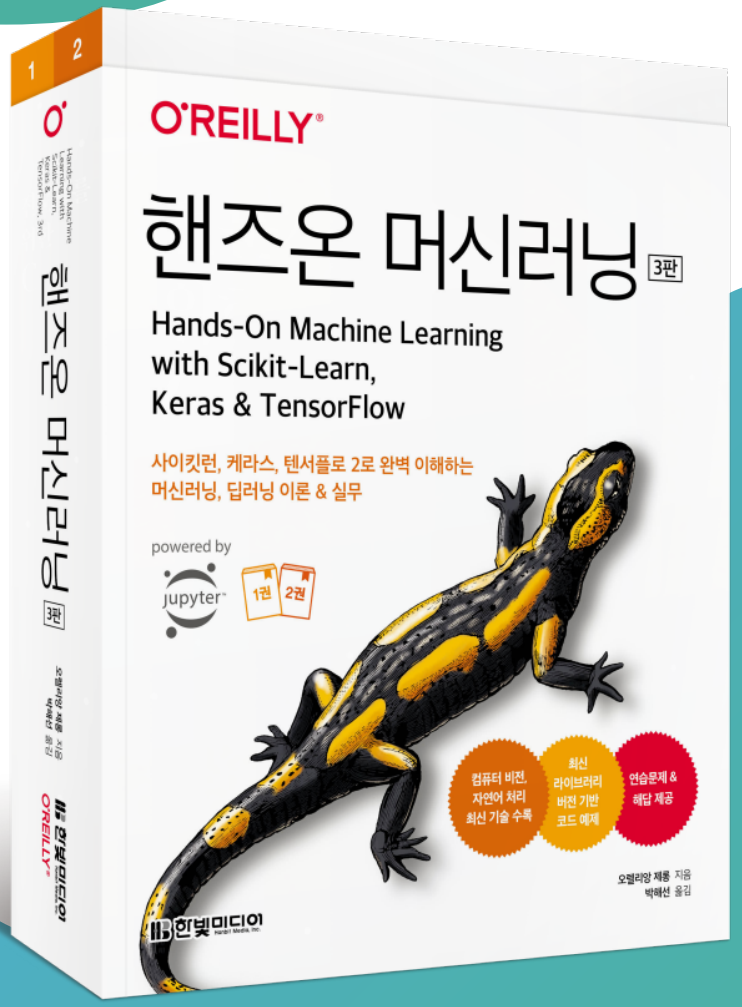
핸즈온
머신러닝(3판)


1
2
O'REILLY
핸즈온 머신러닝 3판
Hands-On Machine Learning
with Scikit-Learn,
Keras & TensorFlow
사이킷런, 케라스, 텐서플로 2로 완벽 이해하는
머신러닝, 딥러닝 이론 & 실무
powered by
jupyter
1권
2권
컴퓨터 비전,
자연어 처리
최신 기술 수록
최신
라이브러리
버전 기반
코드 예제
연습문제 &
해답 제공
오렐리앙 제롱 지음
박해선 옮김
한빛미디어

지은이: 오렐리앙 제롱 Aurélien Géron

머신러닝 컨설턴트. 2013년에서 2016년까지 구글에서 유튜브 동영상 분류 팀을 이끌었다. 2002년에서 2012년까지 프랑스의 모바일 ISP 선두 주자인 위퍼스트Wifirst를 설립하고 CTO로 일했다. 2001년에는 폴리콘셀Polyconseil을 설립하고 CTO로 일했다. 이 회사는 지금 전기차 공유 서비스인 오토립Autolib'을 운영하고 있다.
그전에는 재무(J. P. 모건과 소시에테 제네랄Société Générale), 방위(캐나다 국방부), 의료(수혈) 등 다양한 분야에서 엔지니어로 일했고, C++, WiFi, 인터넷 구조에 관한 몇 권의 기술 서적을 썼으며 한 프랑스 공과대학에서 컴퓨터과학을 가르쳤다.

옮긴이: 박해선 haesun.park@tensorflow.blog

기계공학을 전공했지만 졸업 후엔 줄곧 코드를 읽고 쓰는 일을 했다. 블로그(*tensorflow.blog*)에 글을 쓰고 머신러닝과 딥러닝에 관한 책을 집필, 번역하면서 소프트웨어와 과학의 경계를 흥미롭게 탐험하고 있다.
『챗GPT로 대화하는 기술』(한빛미디어, 2023), 『혼자 공부하는 머신러닝+딥러닝』(한빛미디어, 2020), 『혼자 공부하는 데이터 분석 with 파이썬』(한빛미디어, 2023), 『Do it! 딥러닝 입문』(이지스퍼블리싱, 2019) 등 집필했다.

[코드 예제] *https://github.com/rickiepark/handson-ml3* 에서 주피터 노트북으로 제공

3판의 주요 변경 내용

1. 최신 라이브러리 버전으로 전체 코드 업데이트
2. 특성 이름 추적, 히스토그램 기반 그레이디언트 부스팅, 레이블 전파 등 사이킷런에 새롭게 추가된 다양한 기능
3. 하이퍼파라미터 튜닝을 위한 케라스 튜너(Keras Tuner) 라이브러리, 자연어 처리를 위한 허깅 페이스(Hugging Face)의 트랜스포머스(Transformers) 라이브러리 및 케라스의 새로운 전처리 및 데이터 증식 층
4. 여러 비전 모델(ResNeXt, DenseNet, MobileNet, CSPNet, EfficientNet)과 올바른 모델을 선택하기 위한 가이드라인
5. 〈15장 RNN과 CNN을 사용한 시퀀스 처리〉: 합성된 시계열 대신 시카고 버스 및 철도 탑승객 데이터를 분석하며 ARMA 모델과 그 변형
6. 〈16장 RNN과 어텐션을 사용한 자연어 처리〉: 인코더-디코더 RNN, 트랜스포머 모델을 사용한 영어-스페인어 번역 모델 구축, 스위치 트랜스포머(Switch Transformer), DistilBERT, T5, PaLM(사고 사슬 프롬프트 포함)과 같은 언어 모델 비전 트랜스포머(ViT) 소개, DeiT, 퍼시비어, DINO와 같은 트랜스포머 기반 비전 모델을 비롯한 CLIP, DALL·E, 플라밍고, GATO 등 몇 가지 대형 멀티모달 모델 개요
7. 〈17장 오토인코더, GAN 그리고 확산 모델〉: 확산 모델 소개, DDPM 구현
8. 〈19장 대규모 텐서플로 모델 훈련과 배포〉: 구글 클라우드 AI 플랫폼에서 구글 버텍스 AI로 마이그레이션, 대규모 하이퍼파라미터 검색을 위한 분산 케라스 튜너, TensorFlow.js 코드
9. PipeDream과 Pathways를 비롯한 추가적인 분산 훈련 기법

[참조] https://homl.info/changes3

이 책의 학습 목표

1부 머신러닝

1장: 한눈에 보는 머신러닝	데이터 과학자가 꼭 알아야 할 기초 개념과 용어
2장: 머신러닝 프로젝트 처음부터 끝까지	주택 가격을 예측하는 회귀 작업을 살펴보면서 선형 회귀, 결정 트리, 랜덤 포레스트 등 여러 알고리즘 학습
3장: 분류	분류 시스템 학습
4장: 모델 훈련	신경망 구축에 필요한 모델 훈련 알고리즘 학습
5장: 서포트 벡터 머신	SVM의 핵심 개념, 사용 방법, 작동 원리 학습
6장: 결정 트리	결정 트리의 훈련, 시각화, 예측 방법과 사이킷런의 CART 훈련 알고리즘 등
7장: 앙상블 학습과 랜덤 포레스트	투표 기반 분류기, 배깅과 페이스팅 앙상블, 랜덤 포레스트, 부스팅, 스태킹 앙상블 등
8장: 차원 축소	고차원 공간과 차원 축소 기법
9장: 비지도 학습	비지도 학습과 알고리즘

이 책의 학습 목표

2부 신경망과 딥러닝

10장: 케라스를 사용한 인공 신경망 소개	인공 신경망과 케라스를 이용한 구현 방법
11장: 심층 신경망 훈련	심층 신경망의 문제와 해결 방법
12장: 텐서플로를 사용한 사용자 정의 모델과 훈련	텐서플로와 저수준 파이썬 API를 이용한 사용자 정의 모델과 훈련 알고리즘
13장: 텐서플로를 사용한 데이터 적재와 전처리	tf.data API와 TFRecord 포맷, 케라스 전처리 층 및 tf.data API 사용 방법
14장: 합성곱 신경망을 사용한 컴퓨터 비전	CNN 구성 요소, 케라스를 사용한 구현 방법
15장: RNN과 CNN을 사용한 시퀀스 처리	RNN 개념과 WaveNet 구현
16장: RNN과 어텐션을 사용한 자연어 처리	문장 수준의 RNN과 어텐션 메커니즘
17장: 오토인코더, GAN 그리고 확산 모델	오토인코더 차원 축소, 특성 추출, 비지도 사전 훈련 방법과 생성 모델
18장: 강화 학습	강화 학습 개념, 정책 그레이디언트, 심층 Q-네트워크
19장: 대규모 텐서플로 모델 훈련과 배포	TF 서빙과 구글 버텍스 AI 플랫폼에 모델을 배포하는 방법

이 책의 학습 목표

3부 부록

부록 A: 연습문제 정답

부록 B: 머신러닝 프로젝트 체크리스트

부록 C: 자동 미분

부록 D: 특수한 데이터 구조

부록 E: 텐서플로 그래프

1.1 머신러닝이란?

- 머신러닝은 데이터에서 학습하도록 컴퓨터를 프로그래밍하는 과학(또는 예술)
  - “머신러닝은 명시적인 프로그래밍 없이 컴퓨터가 학습하는 능력을 갖추게 하는 연구 분야” - 아서 새뮤얼Arthur Samuel, 1959
  - “어떤 작업 T에 대한 컴퓨터 프로그램의 성능을 P로 측정했을 때 경험 E로 인해 성능이 향상됐다면, 이 컴퓨터 프로그램은 작업 T와 성능 측정 P에 대해 경험 E로 학습한 것” - 톰 미첼Tom Mitchell, 1997

- 스팸 필터는 (사용자가 스팸이라고 지정한) 스팸 메일과 일반 메일의 샘플을 이용해 스팸 메일 구분법을 배울 수 있는 머신러닝 프로그램의 하나

- 기본 용어
  - 훈련 세트training set: 시스템이 학습하는 데 사용하는 샘플
  - 훈련 사례training instance(혹은 샘플sample): 각 훈련 데이터
  - 모델model: 머신러닝 시스템에서 학습하고 예측을 만드는 부분
    - 신경망neural network, 랜덤 포레스트random forest
  - 작업 T: 새로운 메일이 스팸인지 구분하는 것
  - 경험 E: 훈련 데이터training data
  - 성능 측정 P: 직접 정의해야 하며, 이 성능 측정을 정확도accuracy라고 부르며 분류 작업에 자주 사용

1.2 왜 머신러닝을 사용하나요?(1)

- 전통적 프로그래밍 기법으로는 규칙이 점점 길고 복잡해지므로 유지 보수하기 매우 힘듦

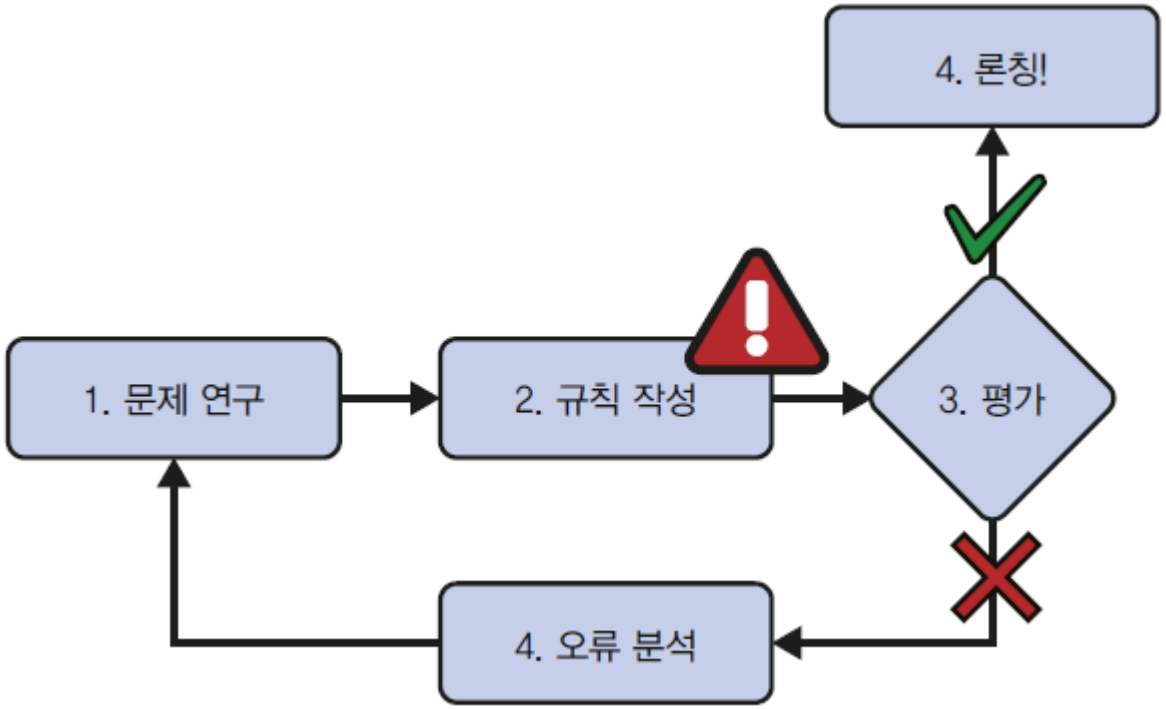


그림 1-1 전통적인 접근 방법

1.2 왜 머신러닝을 사용하나요?(2)

- 프로그램이 훨씬 짧아지고 유지 보수하기 쉬우며 대부분 정확도가 더 높음
  - 머신러닝 기법에 기반을 둔 스팸 필터는 일반 메일에 비해 스팸에 자주 나타나는 패턴을 감지하여 어떤 단어와 구절이 스팸 메일을 판단하는 데 좋은 기준인지 자동으로 학습

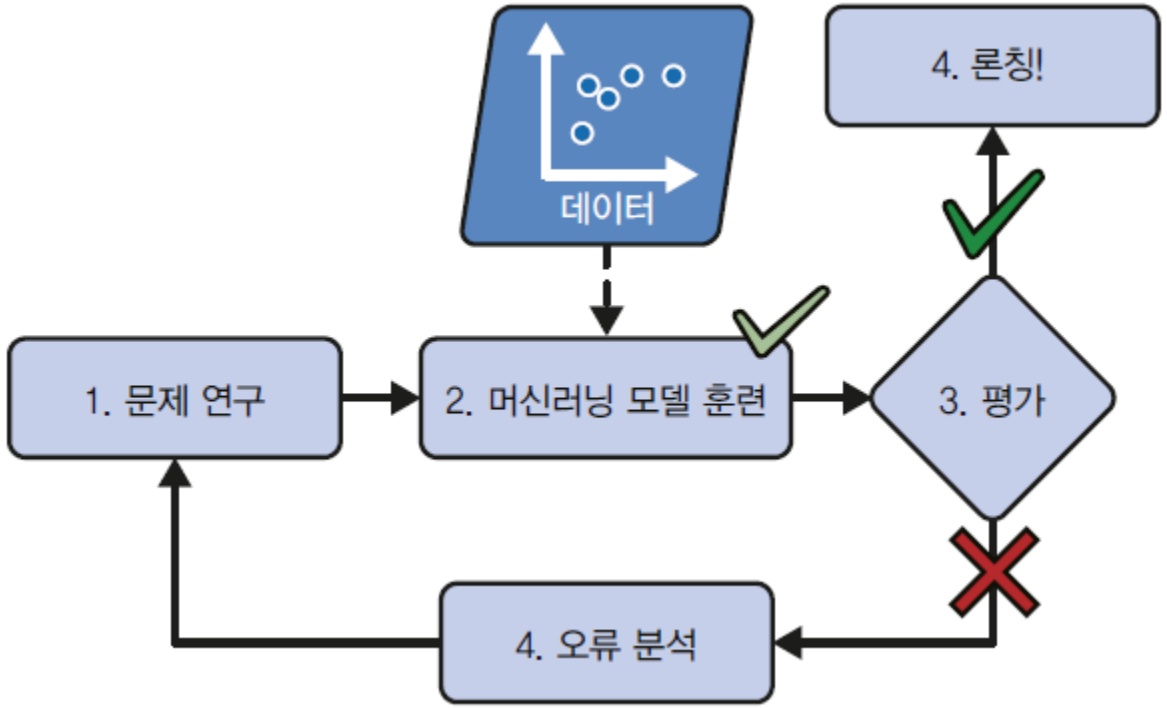


그림 1-2 머신러닝 접근 방법

1.2 왜 머신러닝을 사용하나요?(3)

- 자동으로 변화에 적응
  - 머신러닝 기반의 스팸 필터는 사용자가 스팸으로 지정한 메일에 유독 'For U'가 자주 나타나는 것을 자동으로 인식하고 별도의 작업을 하지 않아도 이 단어를 스팸으로 분류

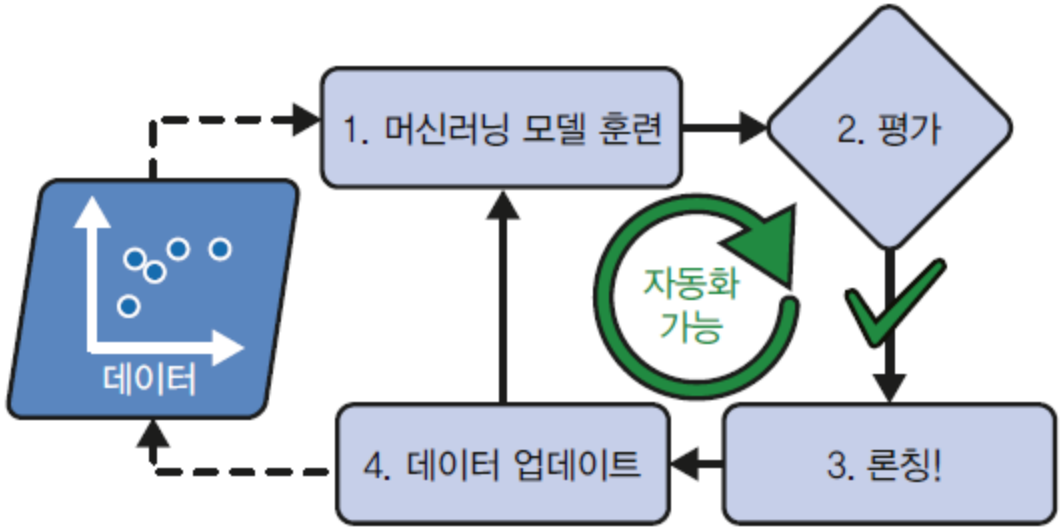


그림 1-3 자동으로 변화에 적응

1.2 왜 머신러닝을 사용하나요?(4)

- **데이터 마이닝**data mining
  - 대용량의 데이터를 분석하여 숨겨진 패턴을 발견

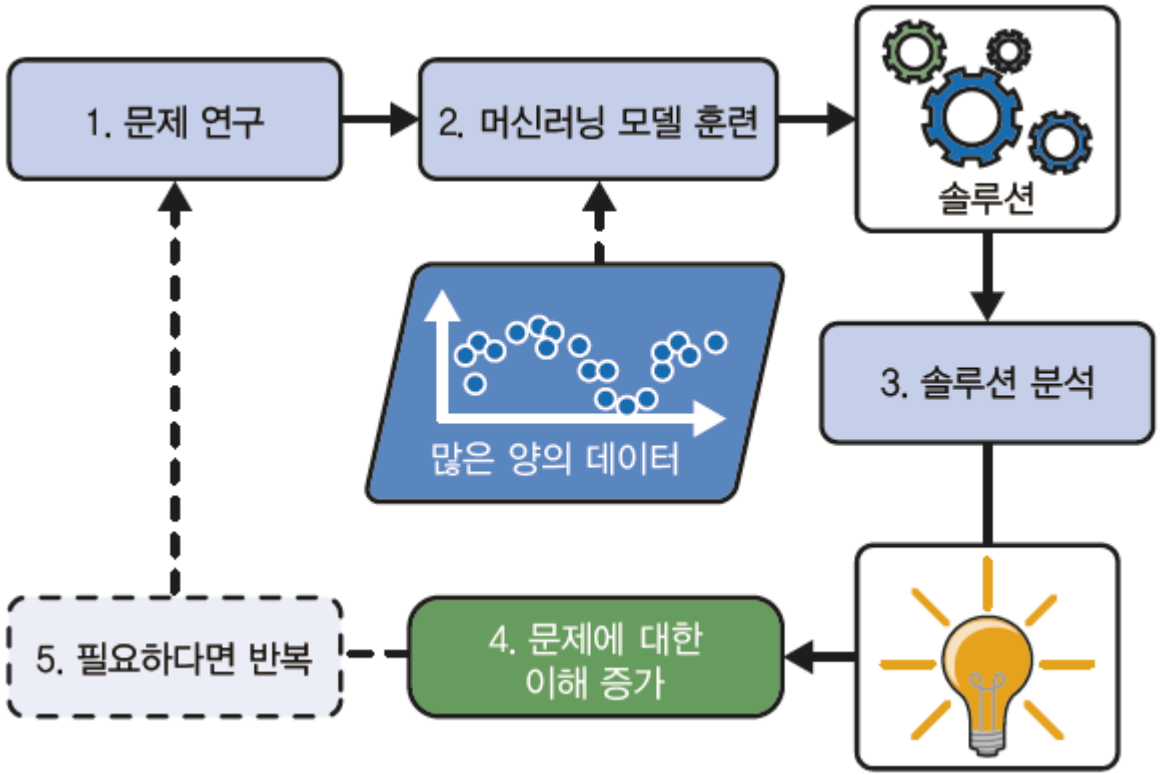


그림 1-4 머신러닝을 통해 학습

1.2 왜 머신러닝을 사용하나요?(5)

- **머신러닝의 강점 분야**
  - 기존 솔루션으로는 많은 수동 조정과 규칙이 필요한 문제
    - 머신러닝 모델이 코드를 간단하게 만들고 전통적인 방법보다 더 잘 수행
  - 전통적인 방식으로는 해결 방법이 없는 복잡한 문제
    - 가장 뛰어난 머신러닝 기법으로 해결 방법을 찾음
  - 유동적인 환경
    - 머신러닝 시스템은 새로운 데이터로 쉽게 재훈련할 수 있어 항상 최신 상태를 유지
  - 복잡한 문제와 대량의 데이터에서 인사이트 얻기

1.3 애플리케이션 사례

- 생산 라인에서 제품 이미지를 분석해 자동으로 분류하기
- 뇌를 스캔하여 종양 진단하기
- 자동으로 뉴스 기사 분류하기
- 토론 포럼에서 부정적인 코멘트를 자동으로 구분하기
- 긴 문서를 자동으로 요약하기
- 챗봇 또는 개인 비서 만들기
- 여러 가지 성과 자료를 바탕으로 회사의 내년도 수익 예측하기
- 음성 명령에 반응하는 앱 만들기
- 신용카드 부정 거래 감지하기
- 구매 이력을 기반으로 고객을 나누고 각 집합마다 다른 마케팅 전략을 계획하기
- 고차원의 복잡한 데이터셋을 명확하고 의미 있는 그래프로 표현하기
- 과거 구매 이력을 기반으로 고객이 관심을 가질 수 있는 상품 추천하기
- 지능형 게임 봇 만들기

1.4 머신러닝 시스템의 종류(1)

- 넓은 범주의 분류
  - 훈련 지도 방식
    - 지도, 비지도, 준지도, 자기 지도, 강화 학습
  - 실시간으로 점진적인 학습을 하는지 아닌지
    - 온라인 학습과 배치 학습
  - 단순하게 알고 있는 데이터 포인트와 새 데이터 포인트를 비교 또는 과학자들이 하는 것처럼 훈련 데이터셋에서 패턴을 발견하여 예측 모델을 생성
    - 사례 기반 학습과 모델 기반 학습

1.4 머신러닝 시스템의 종류(2)

1.4.1 훈련 지도 방식

- 머신러닝 시스템을 학습하는 동안의 지도 형태나 정보량에 따라 분류

지도 학습supervised learning

- 알고리즘에 주입하는 훈련 데이터에 레이블(label)이라는 원하는 답이 포함
  - 분류classification - 스팸 필터
  - 특성feature - 주행 거리, 연식, 브랜드 등을 사용해 중고차 가격 같은 타깃target 수치를 예측
    - 회귀regression

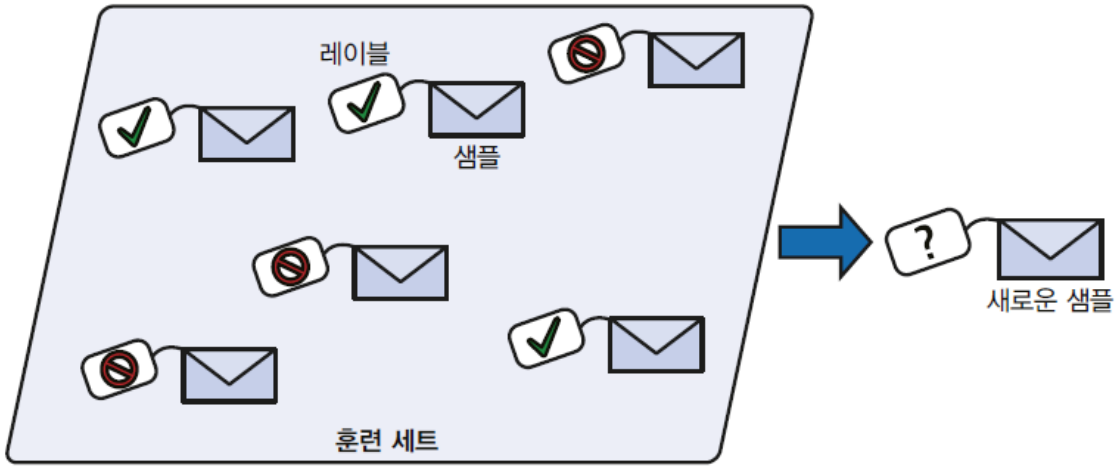


그림 1-5 스팸 분류를 위한 레이블된 훈련 세트(지도 학습의 예)

1.4 머신러닝 시스템의 종류(3)

- 회귀 알고리즘을 분류에 사용할 수도 있음
  - 반대로 일부 분류 알고리즘을 회귀에 사용할 수도 있음
    예) 분류에 널리 쓰이는 로지스틱 회귀logistic regression는 클래스에 속할 확률을 출력

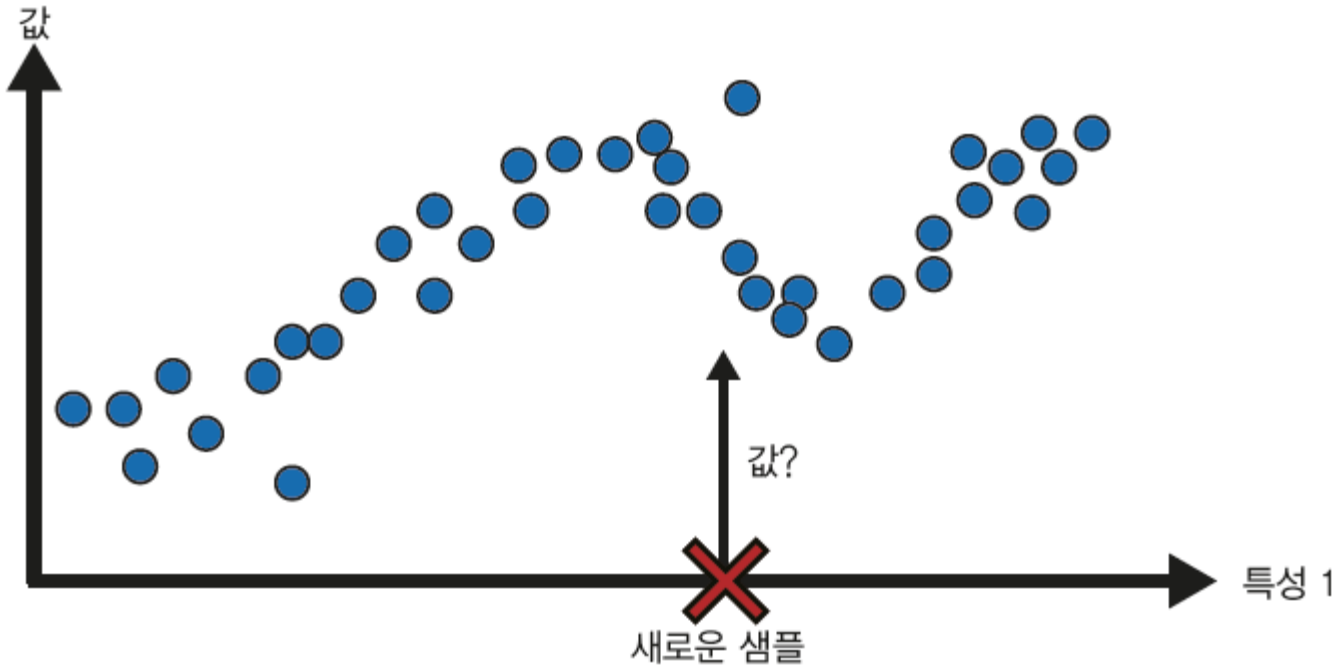


그림 1-6 회귀 문제: 주어진 입력 특성으로 값을 예측

1.4 머신러닝 시스템의 종류(4)

비지도 학습unsupervised learning

- 훈련 데이터에 레이블이 없음
  - 시스템이 아무런 도움 없이 학습

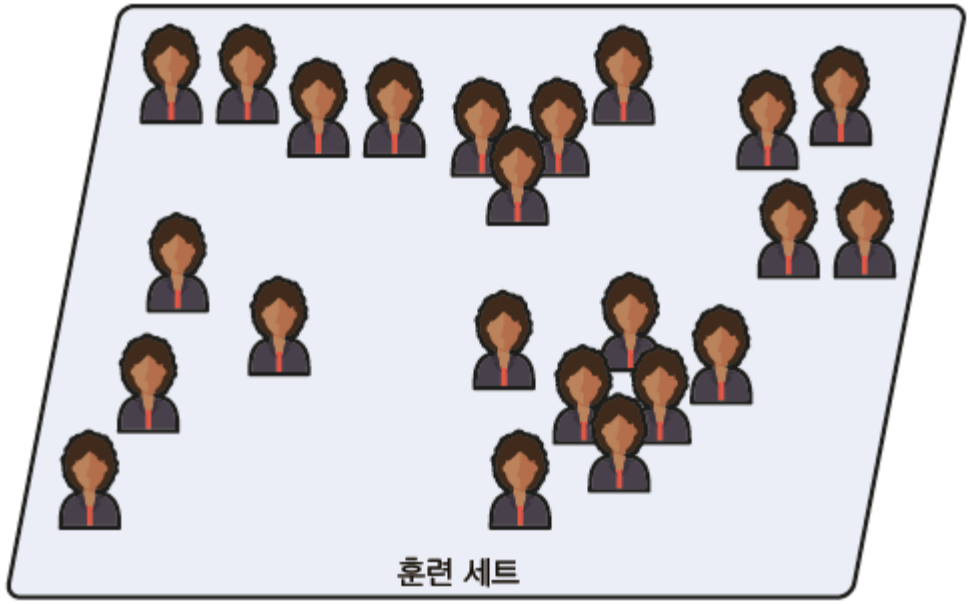


그림 1-7 비지도 학습에서 레이블이 없는 훈련 세트

1.4 머신러닝 시스템의 종류(5)

- 계층 군집hierarchical clustering 알고리즘을 사용하여 각 그룹을 더 작은 그룹으로 세분화

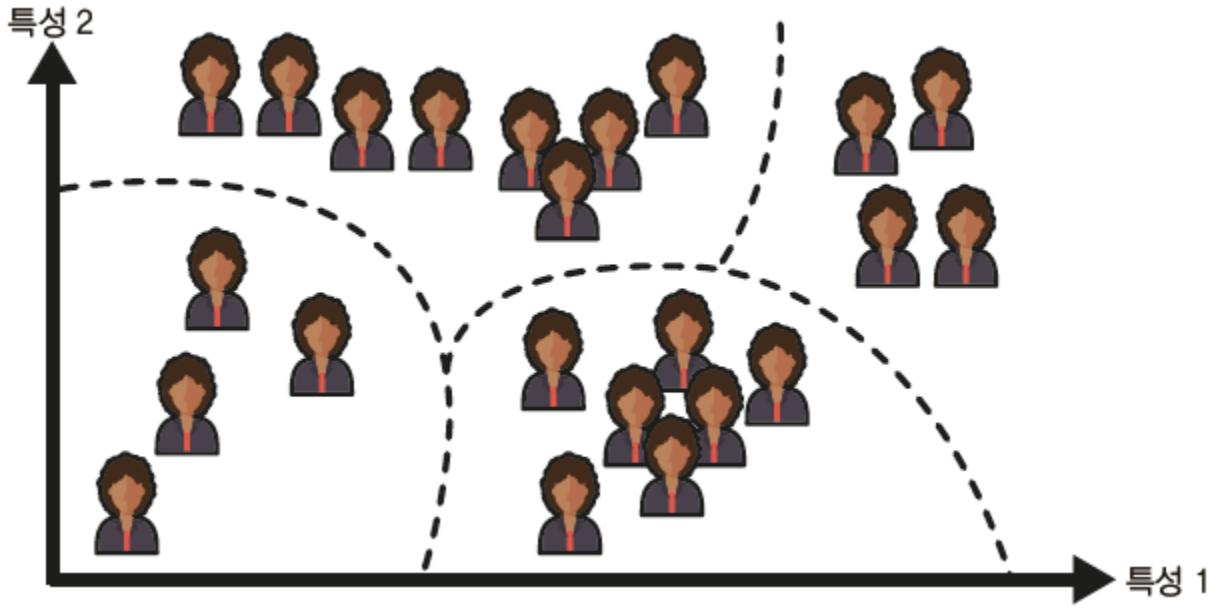


그림 1-8 군집

1.4 머신러닝 시스템의 종류(6)

- 시각화visualization
- 차원 축소dimensionality reduction
- 특성 추출feature extraction

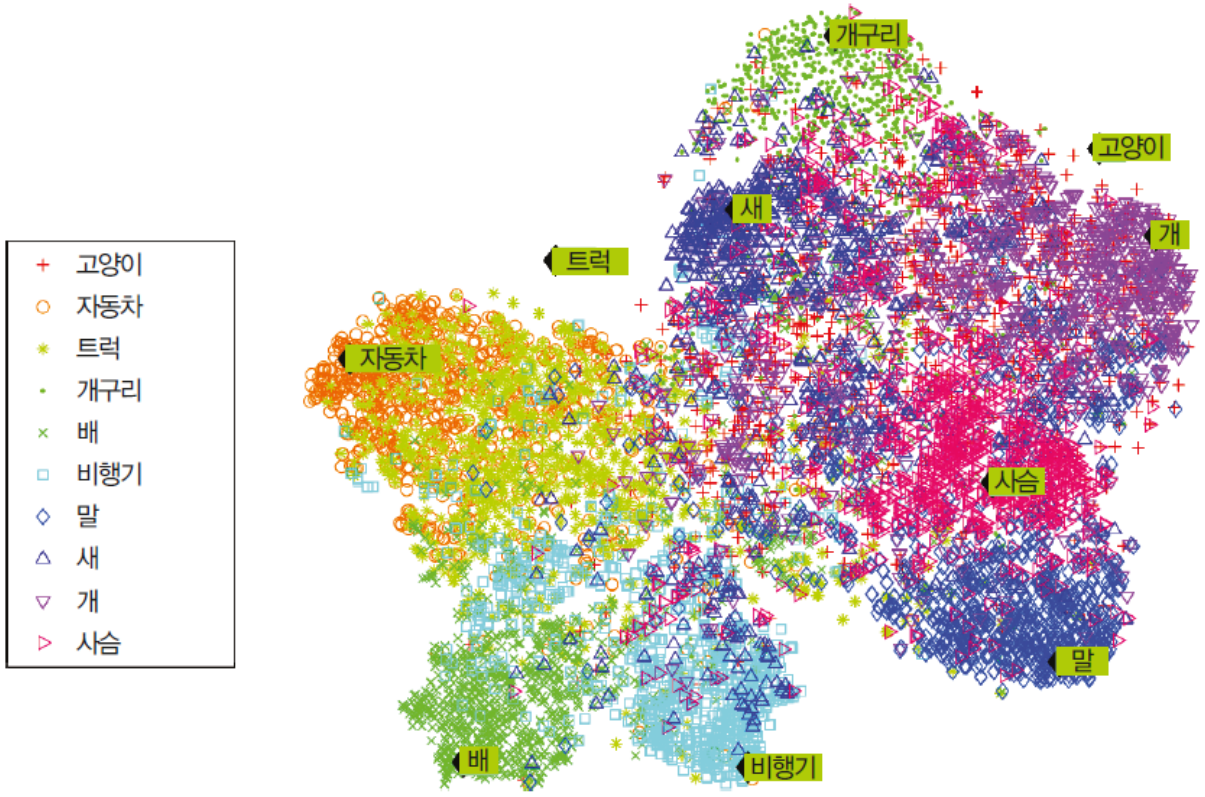


그림 1-9 의미 있는 군집을 강조한 t-SNE 시각화의 예

1.4 머신러닝 시스템의 종류(7)

- 이상치 탐지outlier detection
- 특이치 탐지novelty detection
- 연관 규칙 학습association rule learning

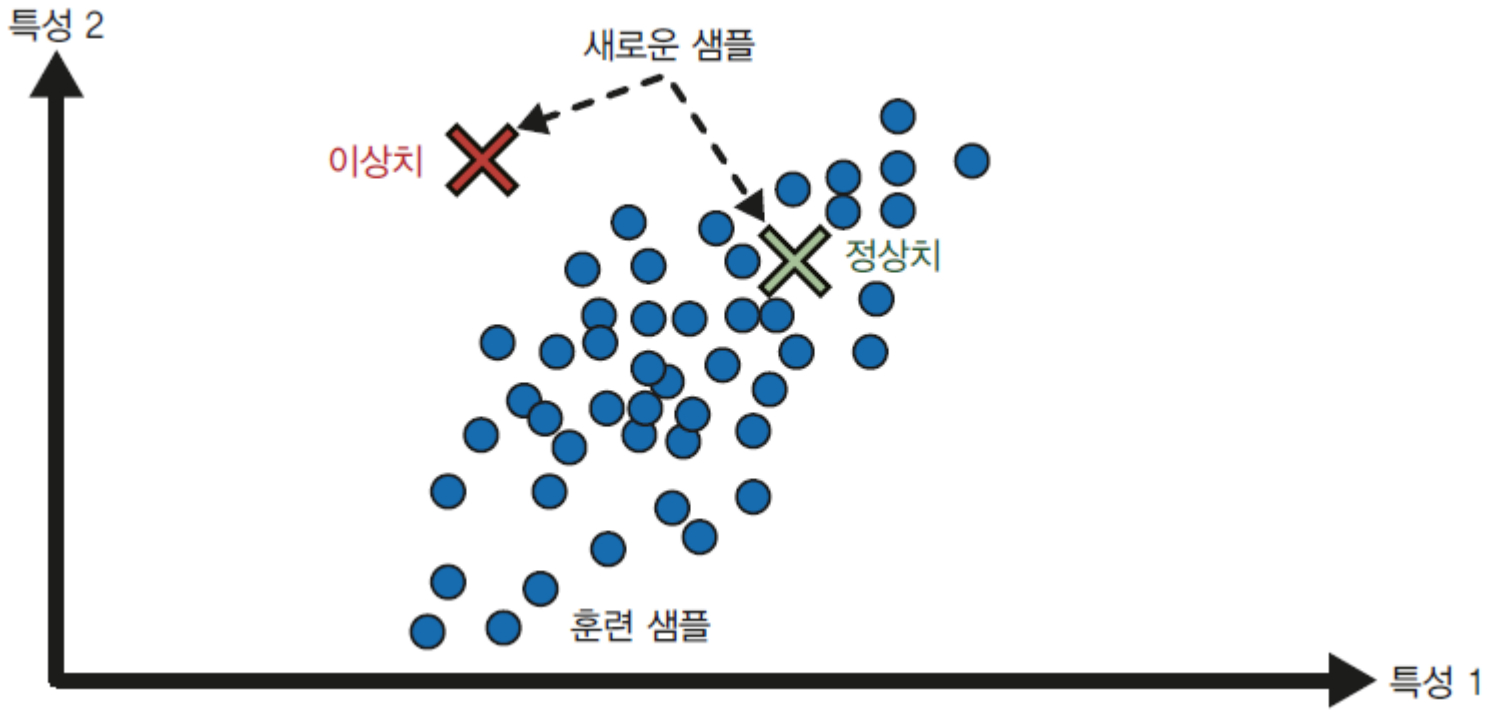


그림 1-10 이상치 탐지

1.4 머신러닝 시스템의 종류(8)

준지도 학습semi-supervised learning

- 레이블이 일부만 있는 데이터를 다루는 알고리즘
  - 구글 포토 서비스
- 대부분의 준지도 학습 알고리즘은 지도 학습과 비지도 학습의 조합으로 구성

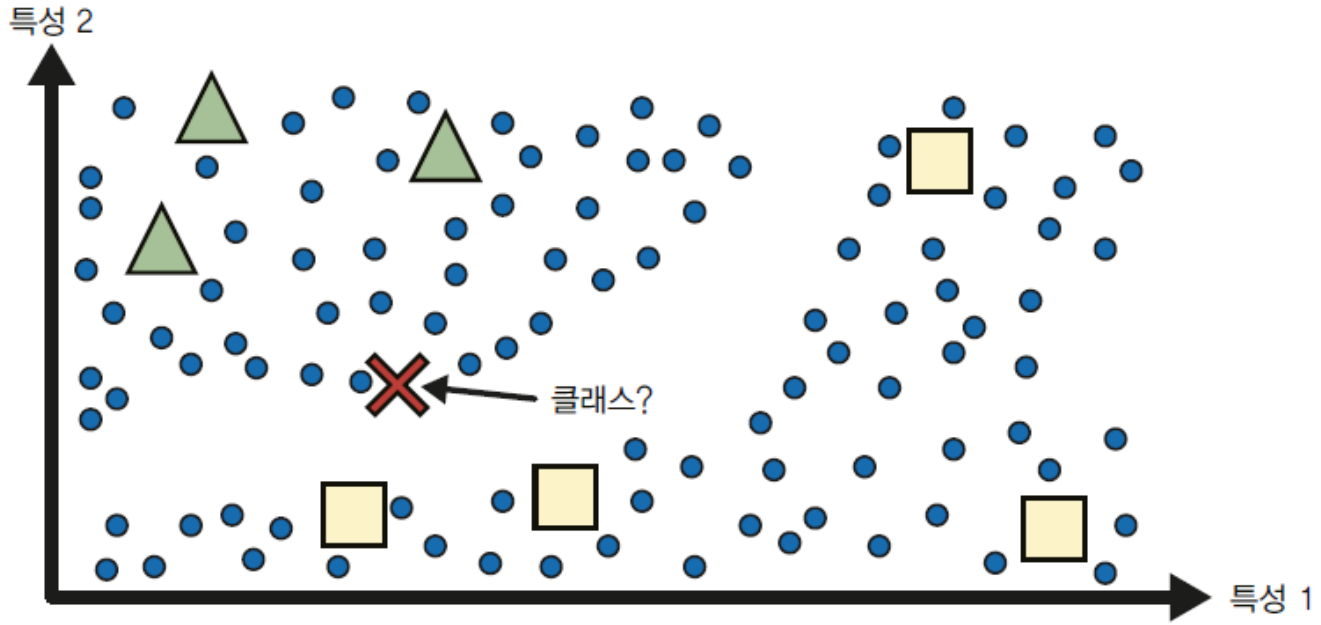


그림 1-11 두 개의 클래스(삼각형과 사각형)를 사용한 준지도 학습: 새로운 샘플(곱셈 기호)은 레이블이 있는 사각형 샘플에 더 가깝지만 레이블이 없는 샘플(원)이 이 샘플을 삼각형 클래스로 분류하는 데 도움

1.4 머신러닝 시스템의 종류(9)

자기 지도 학습self-supervised learning

- 레이블이 전혀 없는 데이터셋에서 레이블이 완전히 부여된 데이터셋을 생성
  - 레이블이 없는 이미지로 구성된 대량의 데이터셋이 있다면 각 이미지의 일부분을 랜덤하게 마스킹masking하고 모델이 원본 이미지를 복원하도록 훈련
  - 훈련하는 동안 마스킹된 이미지는 모델의 입력으로 사용되고 원본 이미지는 레이블로 사용

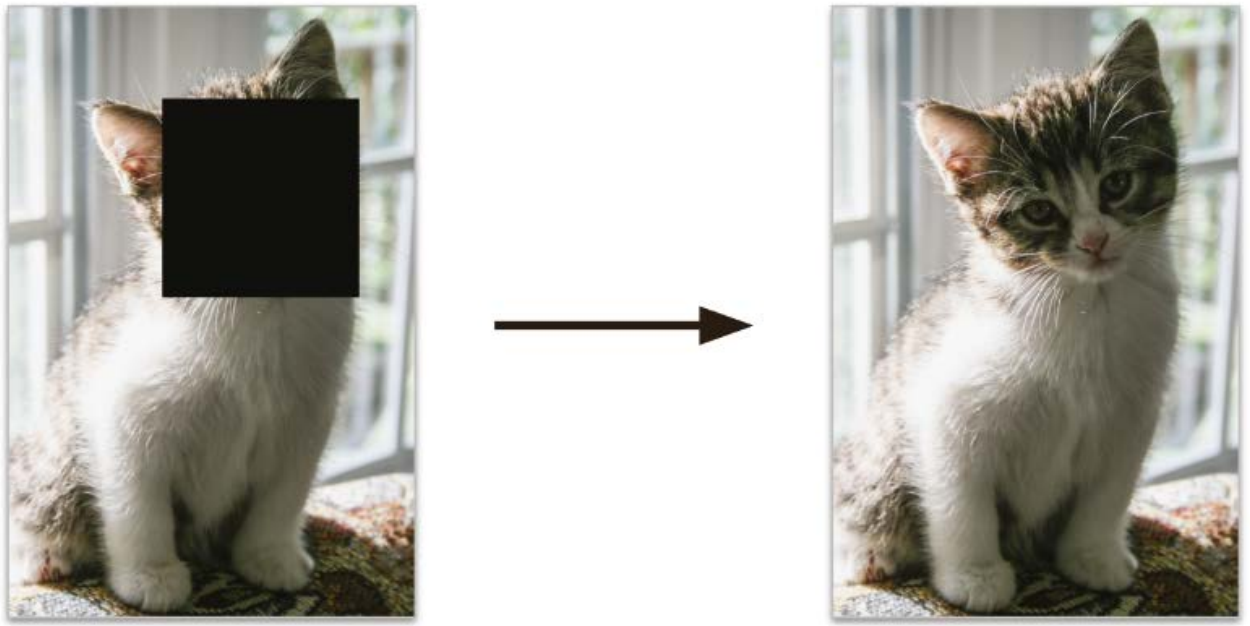
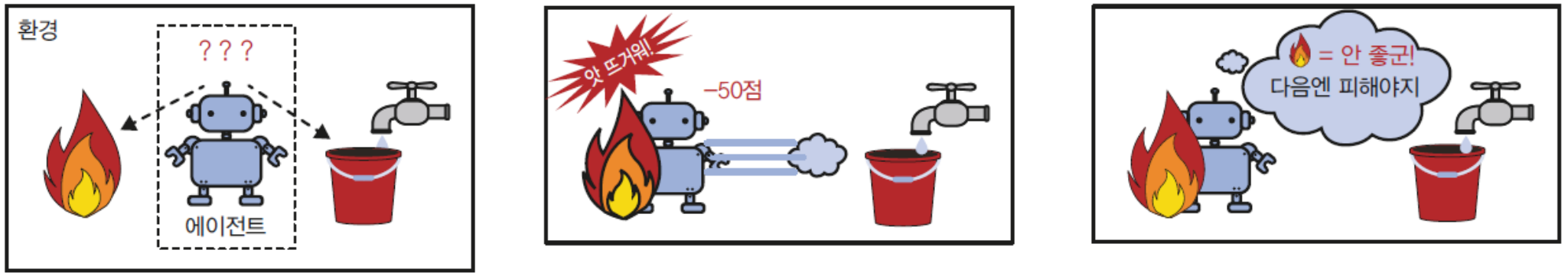


그림 1-12 자기 지도 학습의 예: 입력(왼쪽)과 타깃(오른쪽)

1.4 머신러닝 시스템의 종류(10)

강화 학습reinforcement learning

- 에이전트agent - 학습하는 시스템
  - 환경environment을 관찰해서 행동action을 실행하고 그 결과로 보상reward 또는 벌점penalty 부과
- 정책policy
  - 시간이 지나면서 가장 큰 보상을 얻기 위한 최상의 전략을 스스로 학습
  - 정책은 주어진 상황에서 에이전트가 어떤 행동을 선택해야 할지 정의



① 관찰
② 정책에 따라 행동을 선택

③ 행동 실행!
④ 보상이나 벌점을 받음

⑤ 정책 수정(학습 단계)
⑥ 최적의 정책을 찾을 때까지 반복

그림 1-13 강화 학습

1.4 머신러닝 시스템의 종류(11)

1.4.2 배치 학습과 온라인 학습

배치 학습batch learning

- 오프라인 학습offline learning
  - 가용한 데이터를 모두 사용해 훈련
  - 일반적으로 이 방식은 시간과 자원을 많이 소모하므로 오프라인에서 수행
- 모델 부패model rot 또는 데이터 드리프트data drift
  - 모델의 성능이 시간 경과에 따라 천천히 감소하는 경향

- 전체 데이터셋을 사용해 훈련하는 데 많은 시간이 소요
- 전체 데이터셋을 사용해 훈련한다면 많은 컴퓨팅 자원이 필요
- 자원이 제한된 시스템(예: 스마트폰 또는 화성 탐사 로버)이 스스로 학습해야 할 때 많은 양의 훈련 데이터를 나르고 매일 몇 시간씩 학습을 위해 많은 자원을 사용하면 심각한 문제 발생
  - 점진적으로 학습할 수 있는 알고리즘을 사용하는 것이 효과적

1.4 머신러닝 시스템의 종류(12)

온라인 학습online learning

- 데이터를 순차적으로 한 개씩 또는 미니배치mini-batch라 부르는 작은 묶음 단위로 주입하여 시스템을 훈련
- 매 학습 단계가 빠르고 비용이 적게 들어 시스템은 데이터가 도착하는 대로 즉시 학습

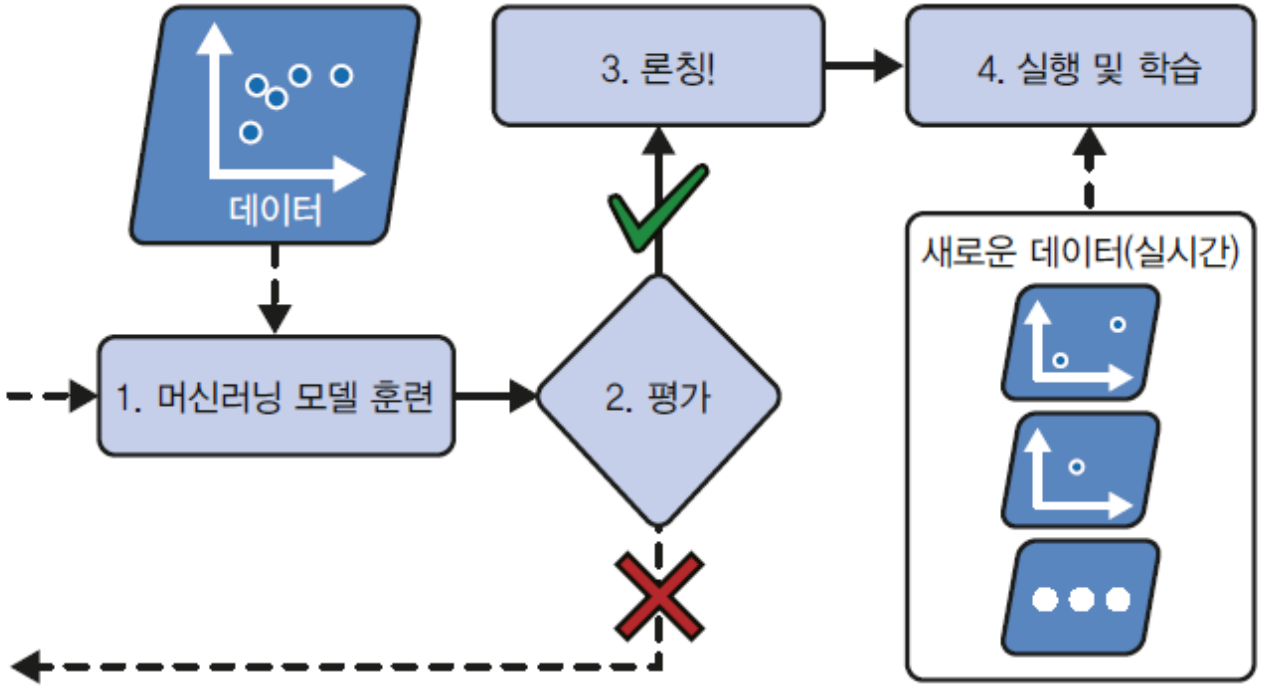


그림 1-14 온라인 학습에서는 모델을 훈련하고 제품에 론칭한 뒤에도 새로운 데이터가 들어오면 계속 학습

1.4 머신러닝 시스템의 종류(13)

- 외부 메모리 학습out-of-core learning
  - 온라인 학습 알고리즘을 사용하여 컴퓨터 한 대의 메인 메모리에 들어갈 수 없는 아주 큰 데이터셋에서 모델을 훈련
  - 알고리즘이 데이터 일부를 읽어들이고 훈련 단계를 수행
  - 전체 데이터가 모두 적용될 때까지 이 과정을 반복

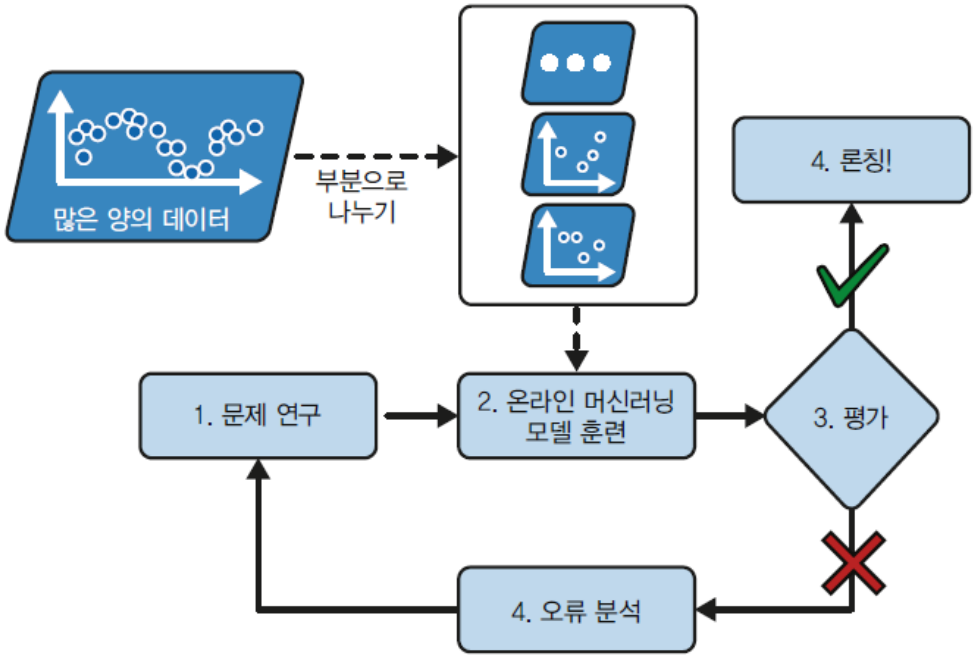


그림 1-15 온라인 학습을 사용한 대량의 데이터 처리

1.4 머신러닝 시스템의 종류(14)

- 학습률learning rate
  - 온라인 학습 시스템에서 변화하는 데이터에 얼마나 빠르게 적응할 것인지의 파라미터
  - 학습률이 높으면 - 시스템이 데이터에 빠르게 적응하지만 예전 데이터를 금방 잊음
  - 학습률이 낮으면 - 시스템의 관성이 더 커져서 더 느리게 학습. 하지만 새로운 데이터에 있는 잡음이나 대표성 없는 데이터 포인트에 덜 민감

- 온라인 학습에서 가장 큰 문제점
  - 시스템에 나쁜 데이터가 주입되었을 때 시스템 성능이 감소
  - 시스템을 면밀히 모니터링하고 성능 감소가 감지되면 즉각 학습을 중지

1.4 머신러닝 시스템의 종류(15)

1.4.3 사례 기반 학습과 모델 기반 학습

사례 기반 학습instance-based learning

- 시스템이 훈련 샘플을 기억함으로써 학습
- 유사도 측정을 사용해 새로운 데이터와 학습한 샘플(또는 학습한 샘플 중 일부)을 비교하는 식으로 일반화

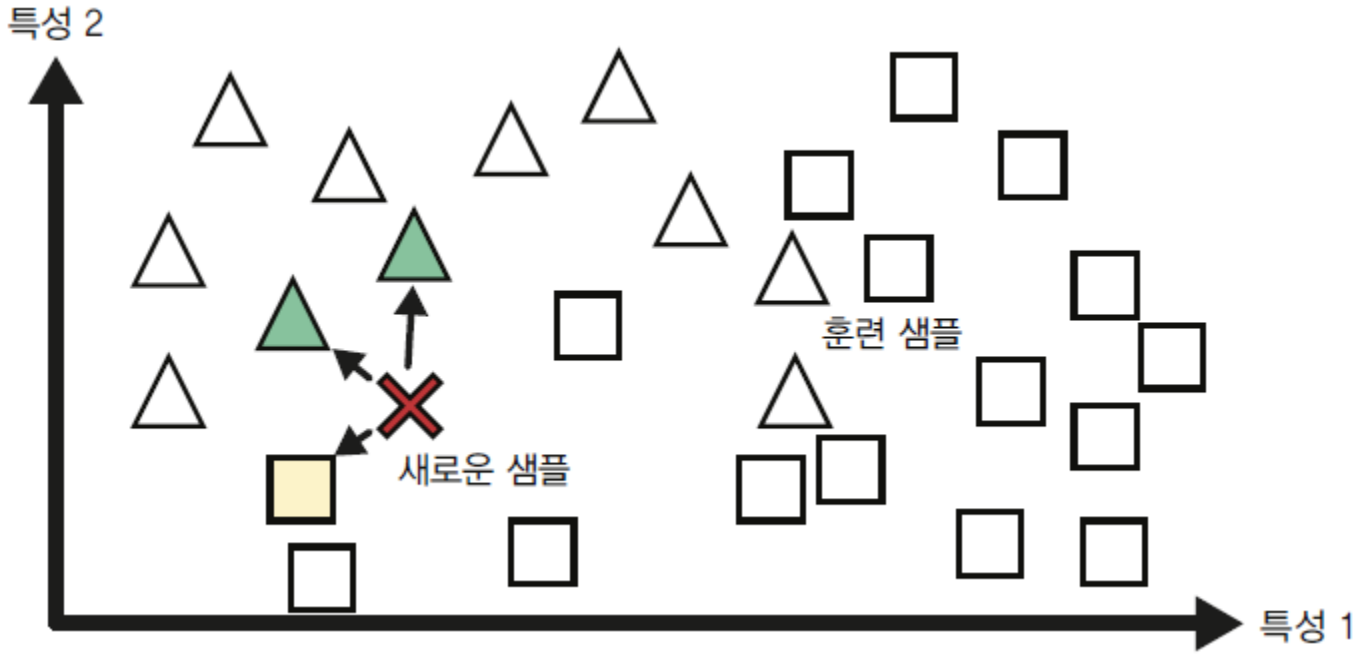


그림 1-16 사례 기반 학습

1.4 머신러닝 시스템의 종류(16)

모델 기반 학습model-based learning

- 샘플들의 모델을 만들어 예측prediction에 사용

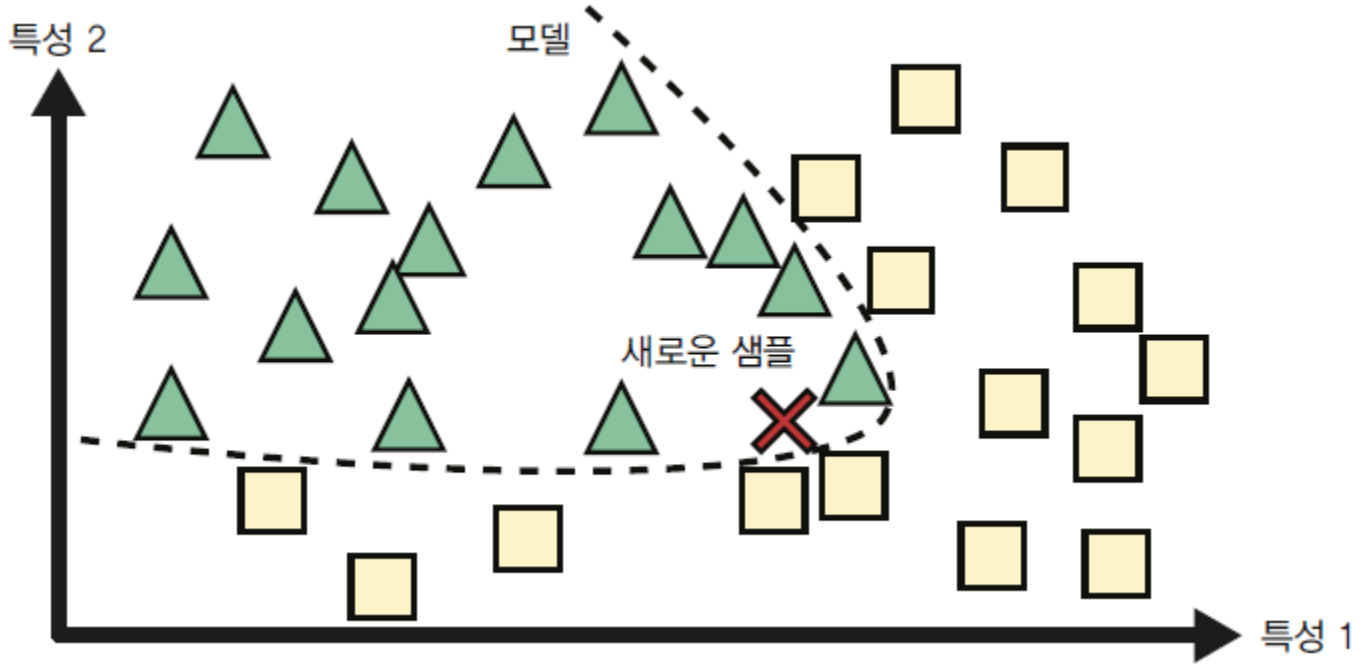


그림 1-17 모델 기반 학습

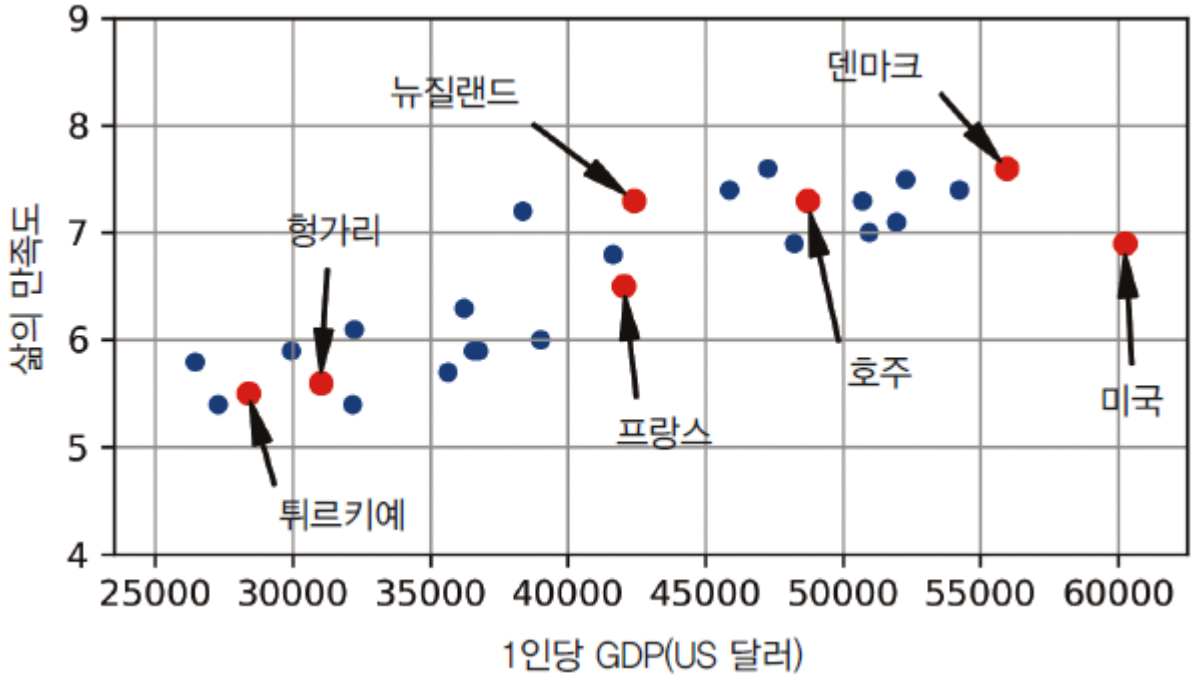
1.4 머신러닝 시스템의 종류(17)

- 모델 기반 학습의 예시
  - 돈이 사람을 행복하게 만드는지 알아본다고 가정(OECD, 세계은행 통계 활용)

표 1-1 돈이 사람을 행복하게 만드는가?

국가	1인당 GDP(US 달러)	삶의 만족도
튀르키예	28,384	5.5
헝가리	31,008	5.6
프랑스	42,026	6.5
미국	60,236	6.9
뉴질랜드	42,404	7.3
호주	48,698	7.3
덴마크	55,938	7.6

그림 1-18 어떤 경향이 보이나?



1.4 머신러닝 시스템의 종류(18)

- 모델 선택model selection
  - 1인당 GDP라는 특성 하나를 가진 삶의 만족도에 대한 선형 모델linear model을 선택
  - 이 모델은 두 개의 모델 파라미터model parameter r θ_0과 θ_1을 가짐
  - 측정 지표 정의
    - 효용 함수utility function 또는 적합도 함수fitness function
    - 비용 함수cost function
  - 훈련training
    - 알고리즘에 훈련 데이터를 공급하여 데이터에 가장 잘 맞는 선형 모델의 파라미터를 찾음

식 1-1 간단한 선형 모델

$$삶의\_만족도 = \theta_0 + \theta_1 \times 1인당\_GDP$$

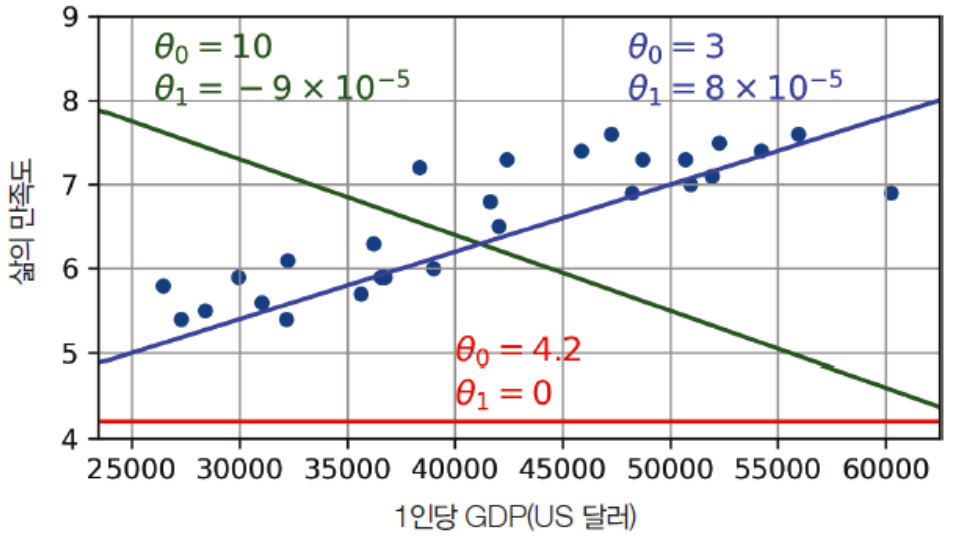


그림 1-19 가능한 몇 가지 선형 모델

1.4 머신러닝 시스템의 종류(19)

- 최적화한 선형 모델
  - OECD 데이터에 없는 키프로스Cyprus 사람들이 얼마나 행복한지 알아보기 위해 이 모델을 사용
  - 키프로스의 1인당 GDP를 보면 37,655달러이므로 이를 모델에 적용해 $3.75 + 37{,}655 \times 6.78 \times 10^{-5} = 6.30$이라는 삶의 만족도를 계산

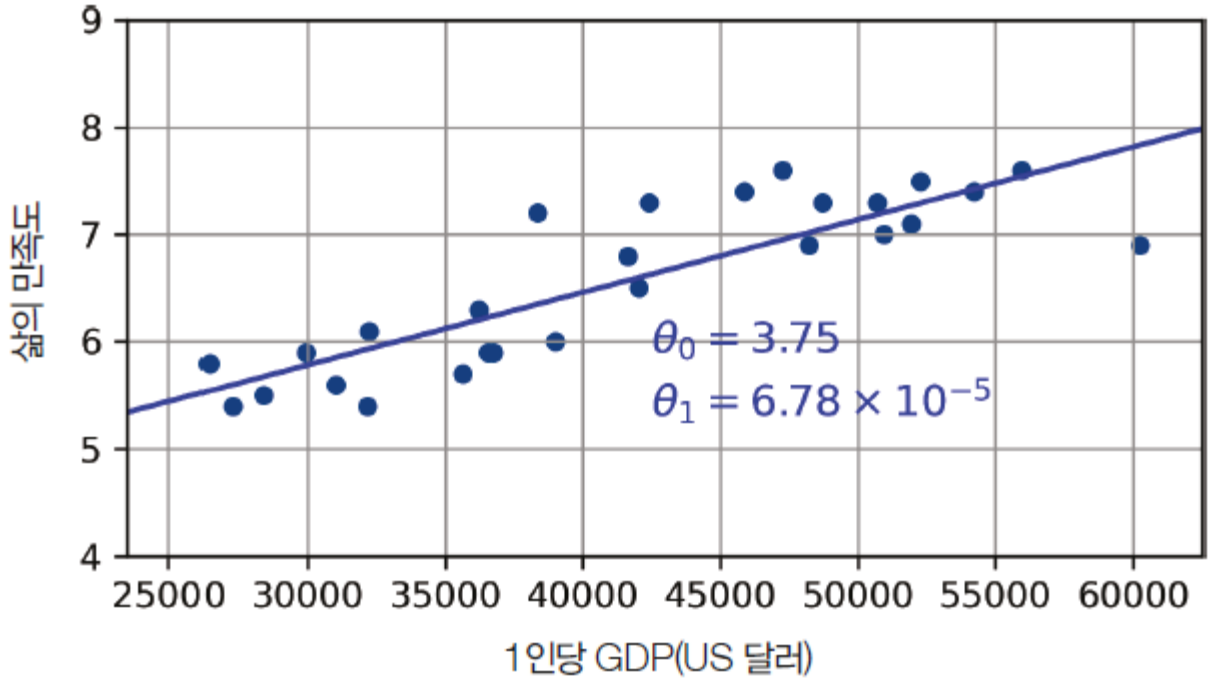


그림 1-20 훈련 데이터에 최적인 선형 모델

1.4 머신러닝 시스템의 종류(20)

- 파이썬 코드 - 산점도 시각화 및 사이킷런을 이용한 선형 모델 훈련과 예측

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression

# 데이터를 다운로드하고 준비합니다.
data_root = "https://github.com/ageron/data/raw/main/"
lifesat = pd.read_csv(data_root + "lifesat/lifesat.csv")
X = lifesat[["GDP per capita (USD)"]].values
y = lifesat[["Life satisfaction"]].values

# 데이터를 그래프로 나타냅니다.
lifesat.plot(kind='scatter', grid=True,
             x="GDP per capita (USD)", y="Life satisfaction")
plt.axis([23_500, 62_500, 4, 9])
plt.show()

# 선형 모델을 선택합니다.
model = LinearRegression()

# 모델을 훈련합니다.
model.fit(X, y)

# 키프로스에 대해 예측을 만듭니다.
X_new = [[37_655.2]]       # 2020년 키프로스 1인당 GDP
print(model.predict(X_new)) # 출력: [[6.30165767]]
```

1.4 머신러닝 시스템의 종류(21)

NOTE

- *k*-최근접 이웃k-nearest neighbors 회귀
  - 이전 코드에서 선형 회귀 모델을 *k*-최근접 이웃 회귀로 바꾸려면 간단히 아래 두 줄을 변경

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

⇩

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=3)
```

1.4 머신러닝 시스템의 종류(22)

- **머신러닝 프로젝트 과정 요약**
  1. 데이터 분석
  2. 모델 선택
  3. 훈련 데이터로 모델을 훈련
     - 학습 알고리즘이 비용 함수를 최소화하는 모델 파라미터를 찾음
  4. 잘 일반화되기를 기대하면서 새로운 데이터에 모델을 적용해 예측 - 추론inference

1.5 머신러닝의 주요 도전 과제(1)

믿기 힘든 데이터의 효과

- 충분한 데이터가 주어지면 아주 간단한 모델을 포함하여 여러 다른 머신러닝 알고리즘이 복잡한 자연어 중의성 해소disambiguation 문제를 거의 비슷하게 잘 처리한다는 사실

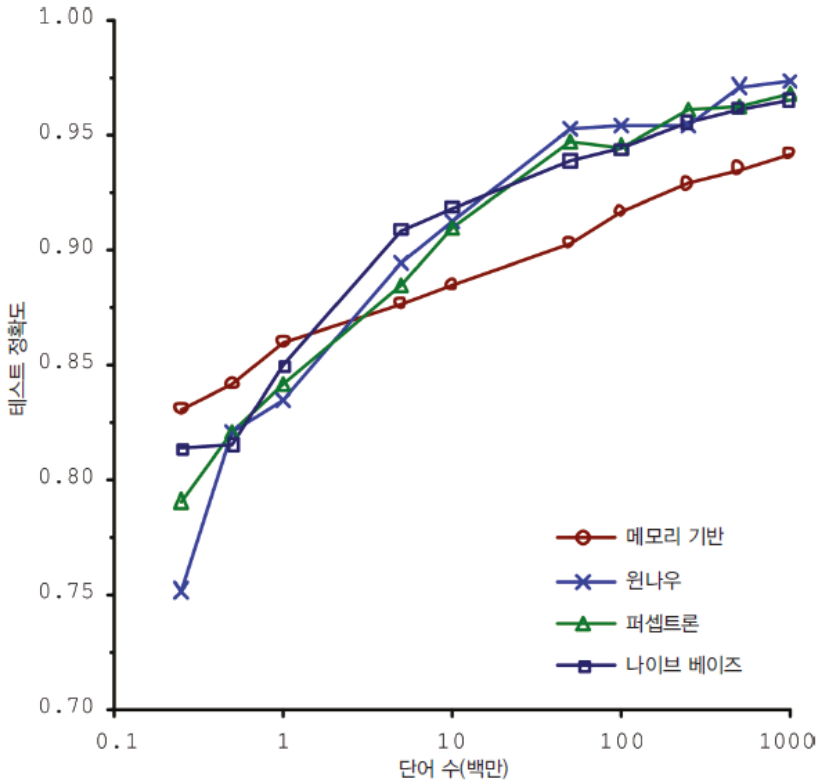


그림 1-21 알고리즘 대비 데이터의 중요성

1.5 머신러닝의 주요 도전 과제(2)

1.5.1 충분하지 않은 양의 훈련 데이터

- 머신러닝 알고리즘이 잘 작동하려면 데이터 가 많아야 함
- 간단한 문제에서도 수천 개의 데이터가 필요하고 이미지나 음성 인식 같은 복잡한 문제라면 수백만 개가 필요

1.5.2 대표성 없는 훈련 데이터

- 일반화가 잘되려면 훈련 데이터가 일반화하고 싶은 새로운 사례를 잘 대표하는 것이 중요
- 샘플이 작으면 샘플링 잡음sampling noise(우연에 의한 대표성 없는 데이터) 발생
- 매우 큰 샘플도 표본 추출 방법이 잘못되면 대표성을 띠지 못할 수 있음 - 샘플링 편향sampling bias

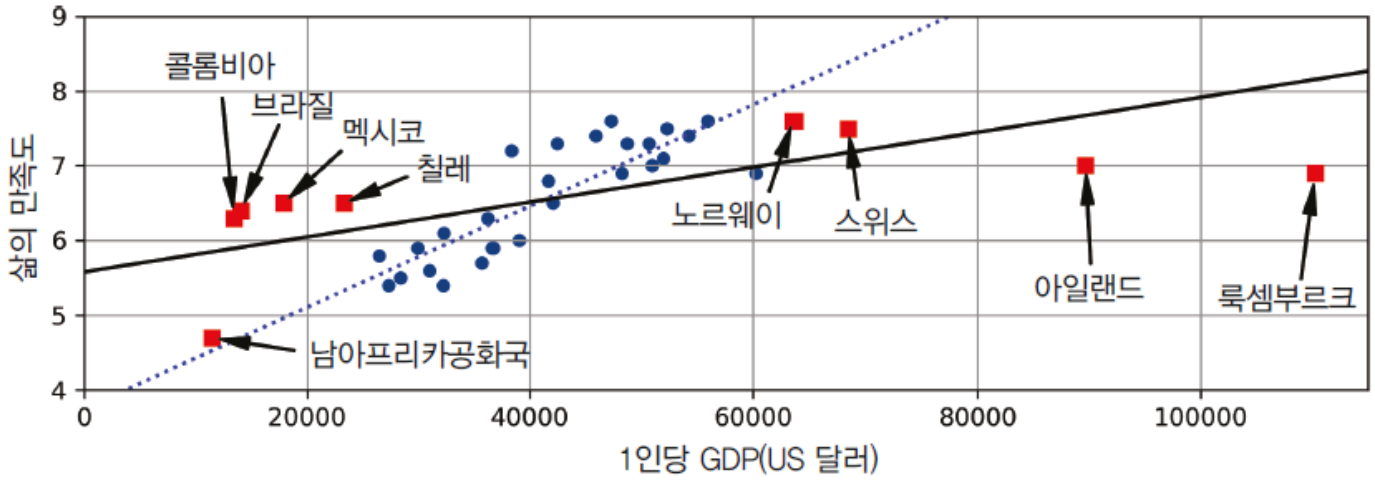


그림 1-22 대표성이 더 큰 훈련 샘플

1.5 머신러닝의 주요 도전 과제(3)

1.5.3 낮은 품질의 데이터

- 훈련 데이터 정제가 필요한 경우
  - 일부 샘플이 이상치라는 게 명확하면 간단히 해당 샘플들을 무시하거나 수동으로 잘못된 것을 고침
  - 일부 샘플에 특성 몇 개가 빠져있다면(예: 고객 중 5%가 나이를 기록하지 않음) 이 특성을 모두 무시할지, 이 샘플을 무시할지, 빠진 값을 채울지(예: 평균 나이로 채움), 또는 이 특성을 넣은 모델과 제외한 모델을 따로 훈련시킬 것인지 결정

1.5.4 관련없는 특성

- 특성 공학feature engineering
  - 특성 선택feature selection - 가지고 있는 특성 중에서 훈련에 가장 유용한 특성을 선택
  - 특성 추출feature extraction - 특성을 결합하여 더 유용한 특성을 만듦
  - 데이터 수집 - 새로운 데이터를 수집해 새 특성을 만듦

1.5 머신러닝의 주요 도전 과제(4)

1.5.5 훈련 데이터 과대적합

- 모델이 훈련 데이터에는 너무 잘 맞지만 일반성이 떨어짐

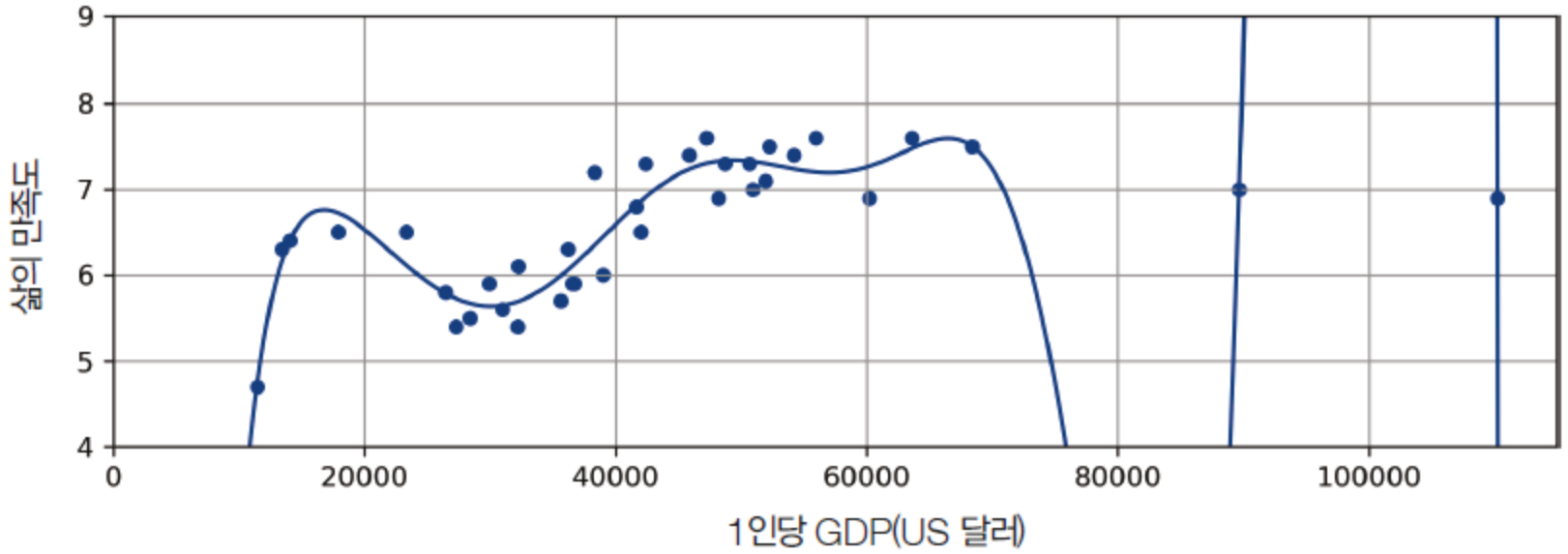


그림 1-23 훈련 데이터에 과대적합된 사례

1.5 머신러닝의 주요 도전 과제(5)

- **규제**regularization
  - 모델을 단순하게 하고 과대적합의 위험을 줄이기 위해 모델에 제약을 가하는 것
  - 학습하는 동안 적용할 규제의 양은 하이퍼파라미터hyperparameter가 결정

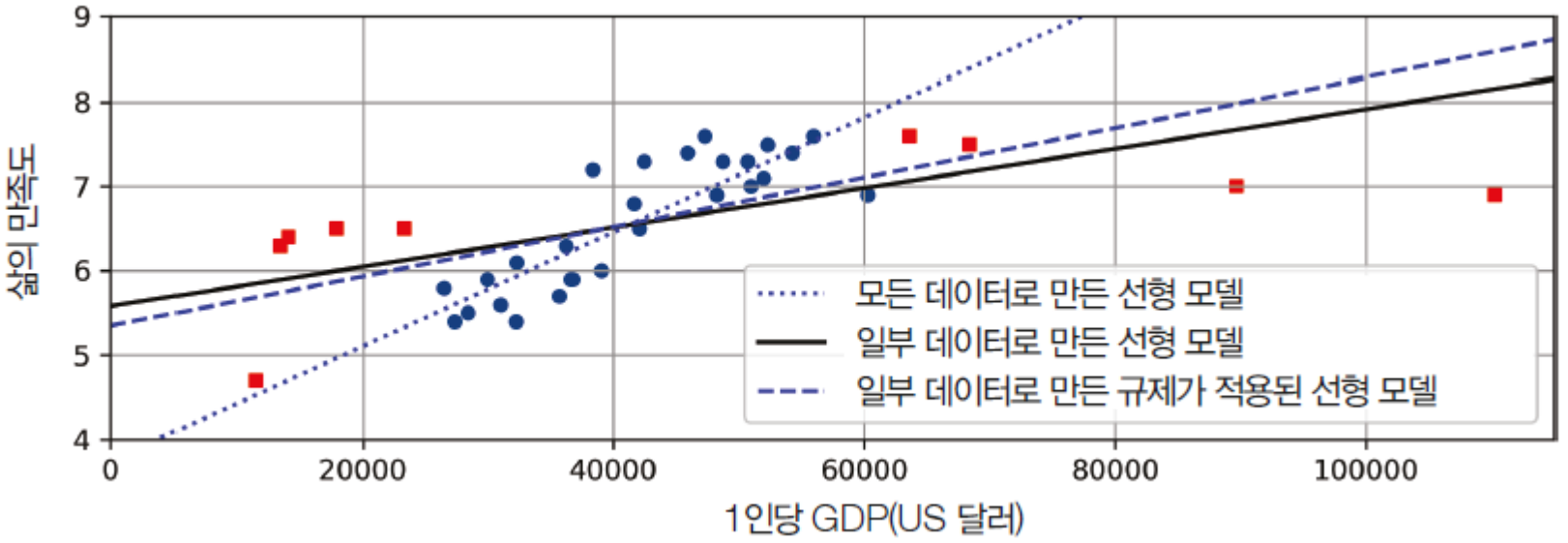


그림 1-24 규제는 과대적합의 위험을 감소

1.5 머신러닝의 주요 도전 과제(6)

1.5.6 훈련 데이터 과소적합

- 모델이 너무 단순해서 데이터의 내재된 구조를 학습하지 못할 때 발생
- 문제 해결
  - 모델 파라미터가 더 많은 강력한 모델을 선택
  - 학습 알고리즘에 더 좋은 특성을 제공(특성 공학)
  - 모델의 제약을 줄임(예: 규제 하이퍼파라미터를 감소시킴)

1.5 머신러닝의 주요 도전 과제(7)

1.5.7 핵심 요약

- 머신러닝은 명시적인 규칙을 코딩하지 않고 기계가 데이터로부터 학습하여 어떤 작업을 더 잘하도록 만드는 것
- 여러 종류의 머신러닝 시스템
  - 지도 학습과 비지도 학습, 배치 학습과 온라인 학습, 사례 기반 학습과 모델 기반 학습 등
- 머신러닝 프로젝트에서는 훈련 세트에 데이터를 모아 학습 알고리즘에 주입
  - 학습 알고리즘이 모델 기반이면 훈련 세트에 모델을 맞추기 위해 모델 파라미터를 조정하고(즉, 훈련 세트에서 좋은 예측을 만들기 위해), 새로운 데이터에서도 좋은 예측을 만들 거라 기대
  - 알고리즘이 사례 기반이면 샘플을 기억하는 것이 학습이고 유사도 측정을 사용하여 학습한 샘플과 새로운 샘플을 비교하는 식으로 새로운 샘플에 일반화
- 훈련 세트가 너무 작거나, 대표성이 없거나, 잡음이 많고 관련없는 특성으로 오염되어 있다면 시스템이 잘 작동하지 않음
  - (엉터리가 들어가면 엉터리가 나옴)
  - 모델이 너무 단순하거나(과소적합된 경우) 너무 복잡하지 않아야 함(과대적합된 경우)

1.6 테스트와 검증(1)

- 훈련 세트와 테스트 세트 두 개로 나누어 검증
  - 데이터의 80%를 훈련에 20%는 테스트용으로 분리하며, 데이터셋 크기에 따라 비율이 다름
    - 훈련 세트를 사용해 모델을 훈련하고 테스트 세트를 사용해 모델을 테스트
    - 새로운 샘플에 대한 오류 비율: 일반화 오차 또는 외부 샘플 오차
    - 테스트 세트에서 모델을 평가함으로써 이 오차에 대한 추정값으로, 이전에 본 적이 없는 새로운 샘플에 모델이 얼마나 잘 작동할지 예측
      - 훈련 오차가 낮지만(즉, 훈련 세트에서 모델의 오차가 적음) 일반화 오차가 높다면 이는 모델이 훈련 데이터에 과대적합되었다는 뜻

1.6 테스트와 검증(2)

1.6.1 하이퍼파라미터 튜닝과 모델 선택

- 홀드아웃 검증holdout validation
  - 훈련 세트의 일부를 떼어내어 여러 후보 모델을 평가하고 가장 좋은 하나를 선택
  - 검증 세트가 작을 경우, 반복적으로 교차 검증 수행
  - 최종 모델을 테스트 세트에서 평가하여 일반화 오차를 추정

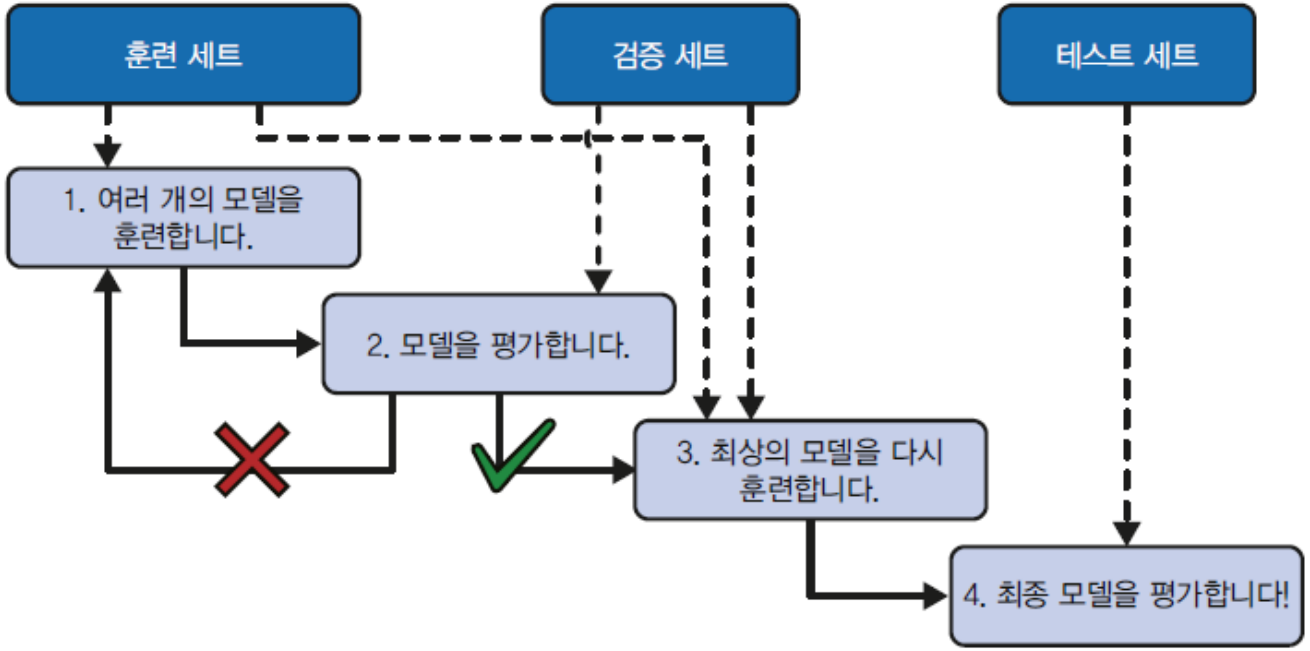


그림 1-25 홀드아웃 검증을 사용한 모델 선택

1.6 테스트와 검증(3)

1.6.2 데이터 불일치

- 훈련-개발 세트train-dev set
  - 모델이 훈련-개발 세트에서 잘 작동한다면 검증 세트에서 평가할 수 있음
  - 만약 성능이 나쁘다면 이 문제는 데이터 불일치에서 오는 것
- 공짜 점심 없음no free lunch(NFL)

그림 1-26 실제 데이터(오른쪽)가 부족할 때 풍부하지만 비슷한 데이터(왼쪽)로 훈련 세트와 과대적합을 평가하기 위한 검증 세트를 만들 수 있음. 실제 데이터는 데이터 불일치(검증 세트)와 모델의 최종 성능(테스트 세트)을 평가하기 위해 사용

연습문제(1)

1. 머신러닝을 어떻게 정의할 수 있나?
2. 머신러닝이 도움을 줄 수 있는 문제 유형 네 가지
3. 레이블된 훈련 세트란 무엇인가?
4. 가장 널리 사용되는 지도 학습 작업 두 가지는?
5. 보편적인 비지도 학습 작업 네 가지는?
6. 사전 정보가 없는 여러 지형에서 로봇을 걸어가게 하려면 어떤 종류의 머신러닝 알고리즘을 사용할 수 있나?
7. 고객을 여러 그룹으로 분할하려면 어떤 알고리즘을 사용해야 하나?
8. 스팸 감지의 문제는 지도 학습과 비지도 학습 중 어떤 문제로 볼 수 있나?
9. 온라인 학습 시스템이 무엇인가?
10. 외부 메모리 학습이 무엇인가?

연습문제(2)

11. 예측을 하기 위해 유사도 측정에 의존하는 학습 알고리즘은 무엇인가?
12. 모델 파라미터와 학습 알고리즘의 하이퍼파라미터 사이에는 어떤 차이가 있나?
13. 모델 기반 알고리즘이 찾는 것은 무엇인가요? 성공을 위해 이 알고리즘이 사용하는 가장 일반적인 전략은 무엇인가? 예측은 어떻게 만드나?
14. 머신러닝의 주요 도전 과제는 무엇인가?
15. 모델이 훈련 데이터에서의 성능은 좋지만 새로운 샘플에서의 일반화 성능이 나쁘다면 어떤 문제가 있나? 가능한 해결책 세 가지는 무엇인가?
16. 테스트 세트가 무엇이고 왜 사용해야 하나?
17. 검증 세트의 목적은 무엇인가?
18. 훈련-개발 세트가 무엇인가? 언제 필요하고 어떻게 사용해야 하나?
19. 테스트 세트를 사용해 하이퍼파라미터를 튜닝하면 어떤 문제가 생기나?

16.6 비전 트랜스포머(1)

- 비주얼 어텐션visual attention을 사용한 이미지 캡션caption 생성

그림 16-12 비주얼 어텐션: 입력 이미지(왼쪽)와 단어 'frisbee'를 생성하기 전 모델의 초점(오른쪽)

16.6 비전 트랜스포머(2)

- 2020년 10월 구글 비전 트랜스포머vision transformer(ViT)
  - 이미지를 16×16의 작은 정사각형으로 자르고 정사각형의 시퀀스를 마치 단어 표현의 시퀀스처럼 취급
- 2020년 12월 페이스북 데이터 효율적인 이미지 트랜스포머data-efficient image transformers(DeiT)
  - 추가 훈련 데이터 없이도 이미지넷에서 경쟁력 있는 결과를 달성
  - 모델의 구조는 원본 ViT와 거의 동일하지만 증류 기법을 사용하여 최신 CNN 모델에서 얻은 지식을 모델로 이동
- 2021년 3월 딥마인드 퍼시비어Perceiver 구조
  - 멀티모달multimodal 트랜스포머
  - 즉, 텍스트, 이미지, 오디오 등 거의 모든 종류의 데이터 입력 가능
- 2021년 4월 마틸드 카론Mathilde Caron DINO
  - 자기 지도 방식으로 레이블 없이 훈련되고 높은 정확도의 시맨틱 분할이 가능한 비전 트랜스포머
  - 훈련 중에 복제되어 한 네트워크는 티처 역할을 하고 다른 네트워크는 스튜던트 역할을 수행
- 2021년 OpenAI의 DALL·E와 DALL·E2
- 2022년 5월 딥마인드의 멀티모달 모델인 GATO

16.7 허깅 페이스의 트랜스포머스 라이브러리(1)

- 허깅 페이스
  - 자연어 처리, 비전 등을 위해 사용하기 쉬운 오픈 소스 도구 생태계를 구축한 AI 회사
  - 사전 훈련된 모델과 이에 상응하는 토크나이저tokenizer
  - 필요한 경우 자체 데이터셋에서 미세 튜닝할 수 있는 트랜스포머스Transformers라이브러리
  - 이 라이브러리는 텐서플로, 파이토치, (Flax 라이브러리를 통해) JAX를 지원
- `transformers.pipeline()` 함수
  - 감성 분석, 개체명 인식, 요약 등 원하는 작업을 지정하기만 하면 사전 훈련된 기본 모델을 다운로드하여 바로 사용

```
from transformers import pipeline

classifier = pipeline("sentiment-analysis") # 가능한 다른 작업이 많습니다.
result = classifier("The actors were very convincing.")
```

16.7 허깅 페이스의 트랜스포머스 라이브러리(2)

- 결과는 입력 텍스트당 하나의 딕셔너리를 담은 파이썬 리스트

```
>>> result
[{'label': 'POSITIVE', 'score': 0.9998071789741516}]
```

- 이 예제에서 모델은 약 99.98%의 신뢰도로 해당 문장이 긍정임을 정확하게 판단
  - 모델에 문장의 배치를 전달할 수도 있음

```
>>> classifier(["I am from India.", "I am from Iraq."])
[{'label': 'POSITIVE', 'score': 0.9896161556243896},
 {'label': 'NEGATIVE', 'score': 0.9811071157455444}]
```

16.7 허깅 페이스의 트랜스포머스 라이브러리(3)

- `pipeline()` 함수는 주어진 작업에 대해 기본 모델을 사용
  - 기본적으로 distilbert-baseuncased-finetuned-sst-2-english 모델을 사용
  - 두 문장을 모순, 중립, 함의의 세 가지 클래스로 분류하는 MultiNLI[Multi-Genre Natural Language Inference] 작업에서 미세 튜닝된 DistilBERT 모델을 사용

```
>>> model_name = "huggingface/distilbert-base-uncased-finetuned-mnli"
>>> classifier_mnli = pipeline("text-classification", model=model_name)
>>> classifier_mnli("She loves me. [SEP] She loves me not.")
[{'label': 'contradiction', 'score': 0.9790192246437073}]
```

16.7 허깅 페이스의 트랜스포머스 라이브러리(4)

- TFAutoModelForSequenceClassification과 AutoTokenizer 클래스를 사용하여 동일한 DistilBERT 모델을 토크나이저와 함께 로드

```
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = TFAutoModelForSequenceClassification.from_pretrained(model_name)
```

- 두 개의 문장 쌍을 토큰화
  - 이 코드에서는 패딩을 활성화하고 파이썬 리스트 대신 텐서플로 텐서를 사용하도록 지정

```
token_ids = tokenizer(["I like soccer. [SEP] We all love soccer!",
                       "Joe lived for a very long time. [SEP] Joe is old."],
                      padding=True, return_tensors="tf")
```

16.7 허깅 페이스의 트랜스포머스 라이브러리(5)

- 출력은 딕셔너리와 비슷한 BatchEncoding 클래스의 객체로 토큰 ID의 시퀀스와 패딩 토큰을 0으로 마스킹한 마스크를 포함

```
>>> token_ids
{'input_ids': <tf.Tensor: shape=(2, 15), dtype=int32, numpy=
array([[ 101, 1045, 2066, 4715, 1012,  102, 2057, 2035, 2293, 4715, 999,
         102,    0,    0,    0],
       [ 101, 3533, 2973, 2005, 1037, 2200, 2146, 2051, 1012, 102, 3533,
        2003, 2214, 1012, 102]], dtype=int32)>,
 'attention_mask': <tf.Tensor: shape=(2, 15), dtype=int32, numpy=
array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], dtype=int32)>}
```

16.7 허깅 페이스의 트랜스포머스 라이브러리(6)

- BatchEncoding 객체를 모델에 직접 전달하면 예측 클래스 로짓을 포함하는 TFSequenceClassifierOutput 객체가 반환

```
>>> outputs = model(token_ids)
>>> outputs
TFSequenceClassifierOutput(loss=None, logits=[<tf.Tensor: [...] numpy=
array([[-2.1123817 , 1.1786783 , 1.4101017 ],
       [-0.01478387, 1.0962474 , -0.9919954 ]], dtype=float32)>], [...])
```

- 소프트맥스 활성화 함수를 적용하여 로짓을 클래스 확률로 변환하고, argmax() 함수를 사용하여 각 입력 문장 쌍에 대해 가장 높은 확률을 가진 클래스를 예측

```
>>> Y_probas = tf.keras.activations.softmax(outputs.logits)
>>> Y_probas
<tf.Tensor: shape=(2, 3), dtype=float32, numpy=
array([[0.01619702, 0.43523544, 0.5485676 ],
       [0.08672056, 0.85204804, 0.06123142]], dtype=float32)>
>>> Y_pred = tf.argmax(Y_probas, axis=1)
>>> Y_pred # 0 = 모순, 1 = 함의, 2 = 중립
<tf.Tensor: shape=(2,), dtype=int64, numpy=array([2, 1])>
```

16.7 허깅 페이스의 트랜스포머스 라이브러리(7)

- 이 모델은 확률 대신 로짓을 출력하기 때문에 sparse_categorical_crossentropy 손실 대신 tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True) 손실을 사용해야 함
  - 또한 이 모델은 훈련 중에 BatchEncoding 입력을 지원하지 않으므로 data 속성을 사용하여 일반 딕셔너리로 준비

```
sentences = [("Sky is blue", "Sky is red"), ("I love her", "She loves me")]
X_train = tokenizer(sentences, padding=True, return_tensors="tf").data
y_train = tf.constant([0, 2]) # 모순, 중립
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(loss=loss, optimizer="nadam", metrics=["accuracy"])
history = model.fit(X_train, y_train, epochs=2)
```

17.9 확산 모델(1)

- 잡음 제거 확산 확률 모델denoising diffusion probabilistic model(DDPM)
  - 2020년에 UC 버클리의 조나단 호Jonathan Ho 등은 매우 사실적인 이미지를 생성할 수 있는 확산 모델을 구축하는 데 성공
- 등방성isotropic
- 정방향 과정forward process

식 17-5 정방향 확산 과정의 확률 분포 q

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

식 17-6 정방향 확산 프로세스의 지름길

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

17.9 확산 모델(2)

- $\mathbf{x}_t$에서 $\mathbf{x}_{t-1}$로 가는 역방향 과정reverse process을 수행할 수 있는 모델을 훈련
  - 이 모델을 사용하여 이미지에서 작은 잡음를 제거하고 모든 잡음이 사라질 때까지 이 작업을 여러 번 반복
  - 고양이 이미지가 많이 포함된 데이터셋에서 모델을 훈련시킨 다음 가우스 잡음으로 가득 찬 사진을 제공하면 모델이 점차 새로운 고양이가 나타나게 될 것임

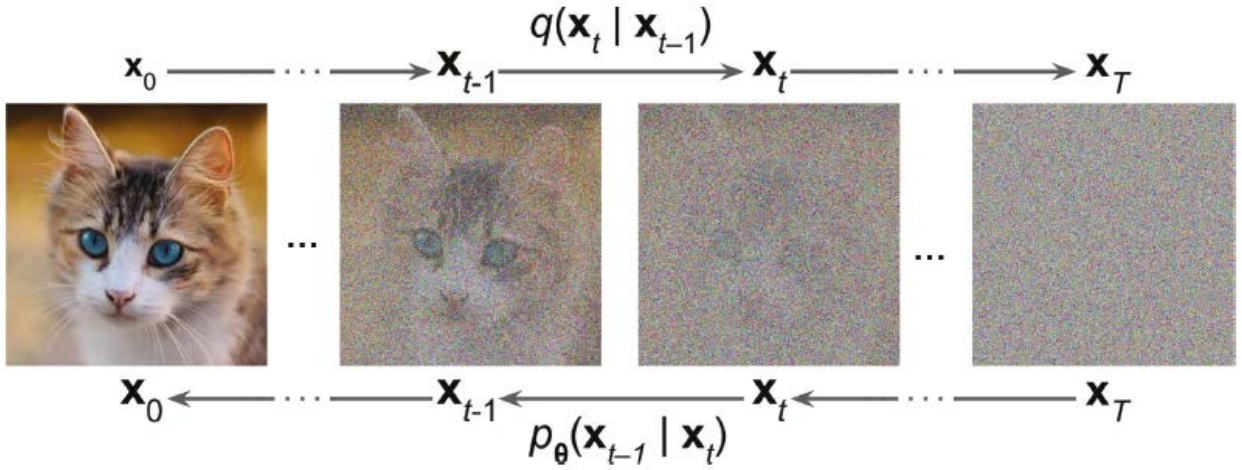


그림 17-20 정방향 프로세스 *q*와 역방향 프로세스 *p*

식 17-7 정방향 확산 프로세스를 위한 분산 스케줄 방정식

$$\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, \text{ 여기에서 } \bar{\alpha}_t = \frac{f(t)}{f(0)} \text{ 이고 } f\left(t\right) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2$$

17.9 확산 모델(3)

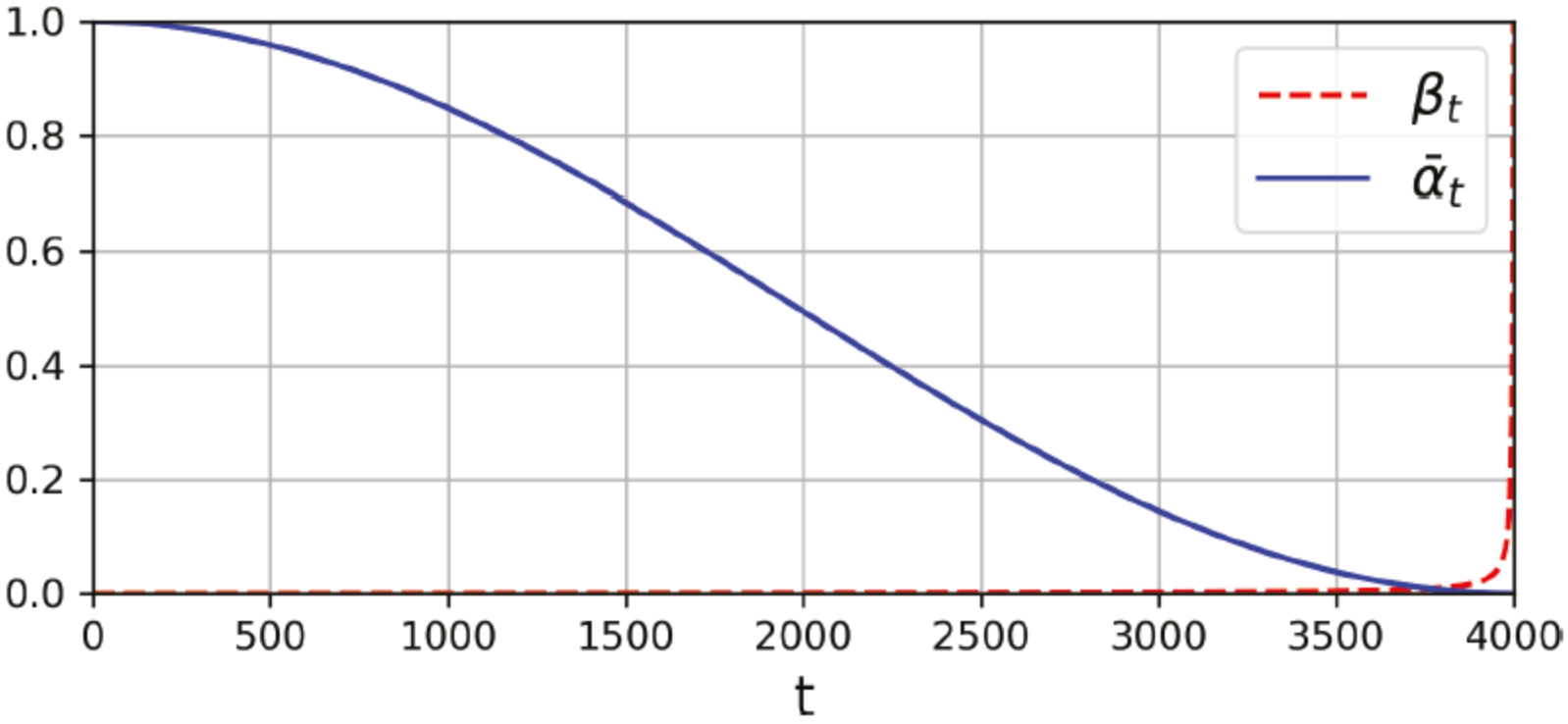


그림 17-21 잡음 분산 스케줄 β_t와 남은 신호의 분산 $\bar{\alpha}_t$

17.9 확산 모델(4)

- α_t, β_t, $\bar{\alpha}_t$를 계산하는 함수를 만들고 T= 4,000으로 호출

```
def variance_schedule(T, s=0.008, max_beta=0.999):
    t = np.arange(T + 1)
    f = np.cos((t / T + s) / (1 + s) * np.pi / 2) ** 2
    alpha = np.clip(f[1:] / f[:-1], 1 - max_beta, 1)
    alpha = np.append(1, alpha).astype(np.float32) # α₀ = 1 추가
    beta = 1 - alpha
    alpha_cumprod = np.cumprod(alpha)
    return alpha, alpha_cumprod, beta # t=0에서 T까지 αₜ , ᾱₜ , βₜ

T = 4000
alpha, alpha_cumprod, beta = variance_schedule(T)
```

17.9 확산 모델(5)

- 확산 과정을 거꾸로 수행하도록 모델을 훈련하려면 정방향 과정의 여러 타임 스텝에서 추출한 잡음 섞인 이미지가 필요
  - 데이터셋에서 깨끗한 이미지의 배치를 가져와 이런 이미지를 만드는 prepare_batch() 함수

```
def prepare_batch(X):
    X = tf.cast(X[..., tf.newaxis], tf.float32) * 2 - 1                          ❶
    X_shape = tf.shape(X)
    t = tf.random.uniform([X_shape[0]], minval=1, maxval=T + 1, dtype=tf.int32)  ❷
     alpha_cm = tf.gather(alpha_cumprod, t)                                      ❸
     alpha_cm = tf.reshape(alpha_cm, [X_shape[0]] + [1] * (len(X_shape) - 1))    ❹
     noise = tf.random.normal(X_shape)                                           ❺
     return {                                                                    ❻
         "X_noisy": alpha_cm ** 0.5 * X + (1 - alpha_cm) ** 0.5 * noise,
         "time": t,
     }, noise
```

17.9 확산 모델(6)

- 훈련 데이터셋과 검증 데이터셋을 만들고 모든 배치에 prepare_batch() 함수를 적용

```
def prepare_dataset(X, batch_size=32, shuffle=False):
    ds = tf.data.Dataset.from_tensor_slices(X)
    if shuffle:
        ds = ds.shuffle(buffer_size=10_000)
    return ds.batch(batch_size).map(prepare_batch).prefetch(1)

train_set = prepare_dataset(X_train, batch_size=32, shuffle=True)
valid_set = prepare_dataset(X_valid, batch_size=32)
```

17.9 확산 모델(7)

- 확산 모델 만들기
  - 잡음 이미지와 타임 스텝을 입력으로 받고 입력에서 뺄 잡음를 예측

```
def build_diffusion_model():
    X_noisy = tf.keras.layers.Input(shape=[28, 28, 1], name="X_noisy")
    time_input = tf.keras.layers.Input(shape=[], dtype=tf.int32, name="time")
    [...]           # 잡음과 타임 스텝을 기반으로 모델을 만듭니다.
    outputs = [...] # (입력 이미지과 같은 크기의) 잡음을 예측합니다.
    return tf.keras.Model(inputs=[X_noisy, time_input], outputs=[outputs])
```

- 모델 훈련

```
model = build_diffusion_model()
model.compile(loss=tf.keras.losses.Huber(), optimizer="nadam")
history = model.fit(train_set, validation_data=valid_set, epochs=100)
```

17.9 확산 모델(8)

- 모델을 훈련하고 나면 이를 사용하여 새로운 이미지를 생성
  - 평균이 0이고 분산이 1인 가우스 분포에서 $\mathbf{x}_T$를 랜덤으로 샘플링한 다음 모델에 전달하여 잡음을 예측
  - 그다음 [식 17-8]을 사용하여 이미지에서 잡음을 빼면 $\mathbf{x}_{T-1}$이 됨
  - x_0이 될 때까지 이 과정을 3,999번 더 반복

식 17-8 확산 프로세스를 한 단계씩 거꾸로 진행하기

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right)\right) + \sqrt{\beta_t}\mathbf{z}$$

17.9 확산 모델(9)

- 역방향 과정을 구현하는 함수를 작성하고 이를 호출하여 몇 개의 이미지를 생성

```
def generate(model, batch_size=32):
    X = tf.random.normal([batch_size, 28, 28, 1])
    for t in range(T, 0, -1):
        noise = (tf.random.normal if t > 1 else tf.zeros)(tf.shape(X))
        X_noise = model({"X_noisy": X, "time": tf.constant([t] * batch_size)})
        X = (
            1 / alpha[t] ** 0.5
            * (X - beta[t] / (1 - alpha_cumprod[t]) ** 0.5 * X_noise)
            + (1 - alpha[t]) ** 0.5 * noise
        )
    return X

X_gen = generate(model) # 생성된 이미지
```

17.9 확산 모델(10)

그림 17-22 DDPM으로 생성된 이미지

모든 것이 최신 버전으로 업데이트된 아마존 인공 지능 분야 부동의 1위 도서

이 책은 이론과 실습을 아우르며 머신러닝과 딥러닝 모두를 관통하는 큰 그림을 머릿속에 그릴 수 있도록 돕습니다. 도식을 활용한 이론 설명과 바로 활용 가능한 최신 버전 코드 예제를 통해 손쉽게 모델을 훈련하고 신경망을 구축하는 방법을 익힐 수 있습니다. 또한 장마다 제공되는 연습문제를 풀면서 배운 내용을 복습하고 자신의 프로젝트에 적용해볼 수 있습니다. 파이썬 프로그래밍 경험만 있다면 시작해보세요. 누구나 머신러닝 전문가가 될 수 있습니다!

*** 추천의 말 ***

3판에는 특히 컴퓨터 비전과 자연어 처리 분야의 최신 이론들이 추가되었고 케라스, 텐서플로, 구글 클라우드 등에서 개선 및 추가된 새로운 기능들을 다루고 있어 더욱 유용합니다.

- 권순선, 구글 Global ML Developer Programs Lead