1. 한눈에 보는 머신러닝

▼ 1.1 머신러닝이란?

정의

- 데이터에서부터 학습하도록 컴퓨터를 프로그래밍하는 과학(또는 예술)
- 명시적인 프로그래밍 없이 컴퓨터가 학습하는 능력을 갖추게 하는 연구 분야
- 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 것이다.

용어

- 훈련 세트: 시스템이 학습하는 데 사용하는 샘플
- 훈련 사례(샘플): 각 훈련 데이터
- 훈련 데이터
- 정확도: 성능 측정

▼ 1.2 왜 머신러닝을 사용하는가?

- 일반 메일에 비해 스팸에 자주 나타나는 패턴을 감지하여 어떤 단어와 구절이 스팸 메일을 판단하는 데 좋은 기준인지 자동으로 학습한다.

  → 프로그램이 훨씬 짧아지고 유지 보수하기 쉬우며, 대부분 정확도가 높다.

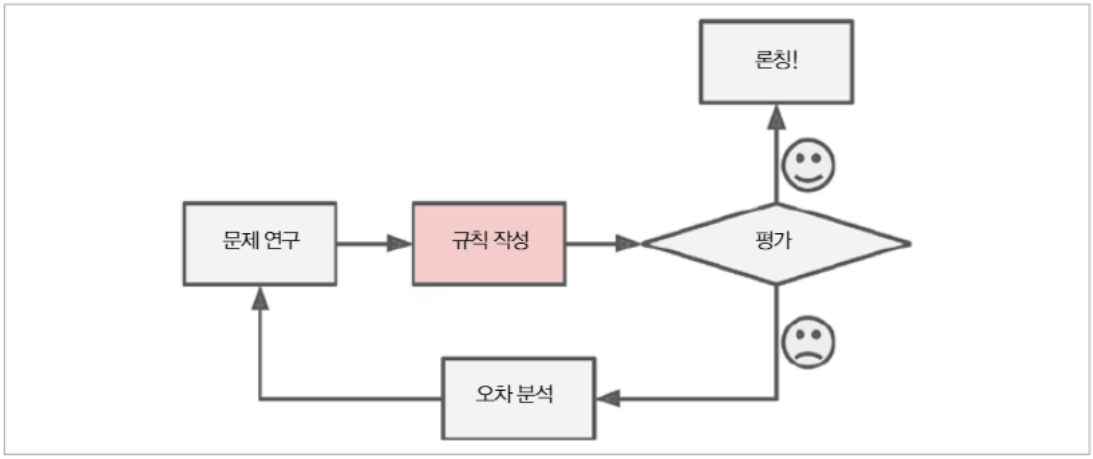

그림 1-1 전통적인 접근 방법

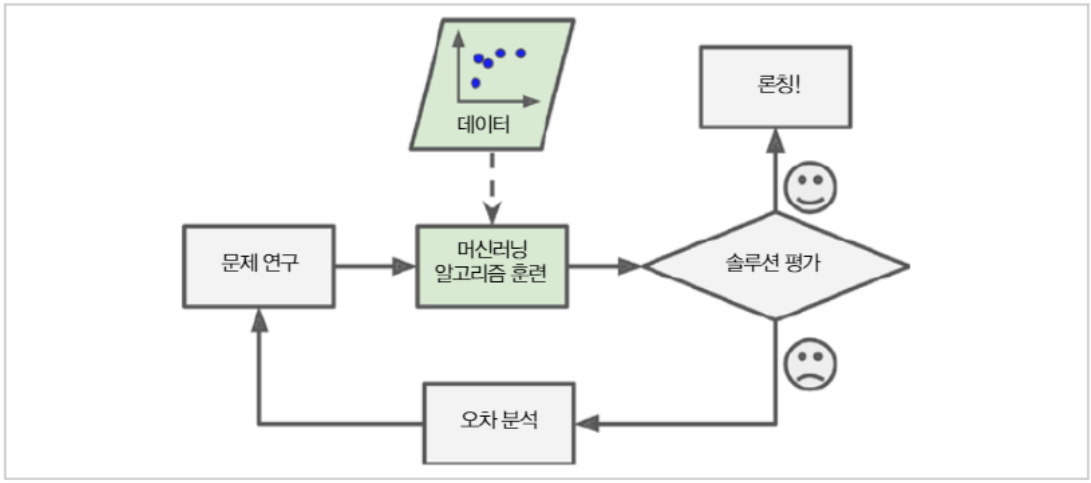

그림 1-2 머신러닝 접근 방법

- 전통적인 방식으로는 너무 복잡하거나 알려진 알고리즘이 없는 문제를 해결할 때 사용한다.

  ex) 음성 인식

  피치(pitch)를 분석하고 사운드 강도를 측정하는 알고리즘을 하드코딩하는 방법은 소음이 있는 환경에서 여러 사람이 말하는 경우에는 사용할 수 없다.

  → 각 단어를 녹음한 샘플을 사용해 스스로 학습하는 알고리즘을 작성하는 것이 가장 좋은 방법

- 머신러닝을 통해 배울 수도 있다.
  - 머신러닝 알고리즘이 학습한 것을 조사할 수 있다.
  - **데이터 마이닝**: 머신러닝 기술을 적용해서 대용량의 데이터를 분석하면 겉으로는 보이지 않던 패턴을 발견할 수 있다.

[요약]

1. 기존 솔루션으로는 많은 수동 조정과 규칙이 필요한 문제 : 하나의 머신러닝 모델이 코드를 간단하게 만들고 전통적인 방법보다 더 잘 수행되도록 할 수 있습니다.
2. 전통적인 방식으로는 해결 방법이 없는 복잡한 문제 : 가장 뛰어난 머신러닝 기법으로 해결 방법을 찾을 수 있습니다.
3. 유동적인 환경 : 머신러닝 시스템은 새로운 데이터에 적응할 수 있습니다.
4. 복잡한 문제와 대량의 데이터에서 통찰 얻기

▼ 1.3 애플리케이션 사례

1. **생산 라인에서 제품 이미지를 분석해 자동으로 분류하기**
    - 이미지 분류 작업
    - 일반적으로 합성곱 신경망convolutional neural network(CNN, 14장)을 사용하여 수행
2. **뇌를 스캔하여 종양 진단**
    - 시맨틱 분할 작업
    - 일반적으로 CNN을 사용해 이미지의 각 픽셀을 분류 (종양의 정확한 위치와 모양을 결정해야 한다).
3. **자동으로 뉴스 기사를 분류하기**
    - 자연어 처리natural language processing(NLP) 작업
    - 더 구체적으로 말하면 텍스트 분류
    - 순환 신경망recurrent neural network(RNN), CNN, 트랜스포머 Transformer(16장 참조)를 사용해 해결할 수 있다.
4. **토론 포럼에서 부정적인 코멘트를 자동으로 구분하기**
    - 텍스트 분류 작업입니다.
    - NLP 도구를 사용
5. **긴 문서를 자동으로 요약하기**
    - 텍스트 요약이라 불리는 NLP의 한 분야
    - NLP 도구를 사용
6. **챗봇chatbot 또는 개인 비서 만들기**

- 자연어 이해natural language understanding(NLU)와 질문-대답question-answering 모듈을 포함해 여러 가지 NLP 컴포넌트가 필요하다.

7. **다양한 성능 지표를 기반으로 회사의 내년도 수익을 예측하기**
    - 회귀regression 작업(즉 숫자로 값을 예측).
    - 선형 회귀linear regression나 다항 회귀polynomial regression 모델 (4장), 회귀 SVM(5장), 회귀 랜덤 포레스트random forest(7장), 인공 신경망artificial neural network(10장)과 같은 회귀 모델을 사용해서 해결할 수 있다.
    - 지난 성능 지표의 시퀀스를 고려한다면 RNN, CNN 또는 트랜스포머 (15장, 16장)를 사용할 수 있다.
8. **음성 명령에 반응하는 앱을 만들기**
    - 음성 인식 작업
    - 오디오 샘플을 처리해야 한다.
    - 이는 길고 복잡한 시퀀스이므로 일반적으로 RNN, CNN 또는 트랜스포머(15장, 16장)를 사용한다.
9. **신용 카드 부정 거래 감지하기**
    - 이상치 탐지 작업(9장).
10. **구매 이력을 기반으로 고객을 나누고 각 집합마다 다른 마케팅 전략을 계획하기**
    - 군집clustering 작업(9장).
11. **고차원의 복잡한 데이터셋을 명확하고 의미 있는 그래프로 표현하기**
    - 데이터 시각화 작업
    - 차원 축소dimensionality reduction 기법을 많이 사용한다 (8장).
12. **과거 구매 이력을 기반으로 고객이 관심을 가질 수 있는 상품 추천하기**
    - 추천 시스템
    - 과거 구매 이력을 (그리고 고객에 관한 다른 정보를) 인공 신경망(10장)에 주입하고 다음에 구매할 가능성이 가장 높은 상품을 출력하는 것이 한 가지 방법
    - 일반적으로 모든 고객의 구매 이력을 기반으로 훈련
13. **지능형 게임 봇bot 만들기**
    - 보통 강화 학습reinforcement learning(RL, 18장)으로 해결

- 시간이 지나면 (게임 같은) 주어진 환경에서 보상이 최대가 되는 행동을 선택하는 (봇과 같은) 에이전트를 훈련하는 머신러닝의 한 분야 (예를 들어 상대 플레이어가 점수를 잃은 때마다 봇이 보상을 받을 수 있다).
- 바둑 세계 챔피언을 이긴 유명한 알파고AlphaGo가 강화 학습을 사용해 구축되었다.

▼ 1.4 머신러닝 시스템의 종류

▼ 1.4.1 지도 학습과 비지도 학습

지도 학습

- 지도 학습(supervised learning)에는 알고리즘에 주입하는 훈련 데이터에 **레이블**(label)이라는 원하는 답이 포함된다

그림 1-5 스팸 분류를 위한 레이블된 훈련 세트(지도 학습의 예)

- **분류**가 전형적인 지도 학습 작업

  ex) 스팸 필터

- **예측 변수** 라 부르는 **특성**(주행거리, 연식, 브랜드 등)을 사용해 중고차 가격 같은 **타깃** 수치를 예측하는 것도 전형적인 작업이다

  → **회귀** 라고 부름

📒 머신러닝에서
- **속성**(attribute): 데이터 타입(예를 들면 주행거리)
- **특성:** 문맥에 따라 여러 의미를 갖지만 일반적으로 속성과 값이 합쳐진 것(예를 들면 주행거리 =15,000)
하지만 많은 사람이 속성과 특성을 구분하지 않고 사용한다

일부 회귀 알고리즘은 분류에 사용할 수도 있다. 반대로 일부 분류 알고리즘을 회귀에 사용할 수도 있다.

ex) **로지스틱 회귀**는 클래스에 속할 확률을 출력한다.

[지도 학습 알고리즘]

1. k-최근접 이웃(k-nearest neighbors)
2. 선형 회귀(linear regression)
3. 로지스틱 회귀(logistic regression)
4. 서포트 벡터 머신(support vector machine): SVM
5. 결정 트리(decision tree)와 랜덤 포레스트(random forest)
6. 신경망(neural networks)

비지도 학습

- **비지도 학습**: 훈련 데이터에 레이블이 없는 학습

[비지도 학습 알고리즘]

- 군집(clustering)
  - k-평균(k-means)
  - DBSCAN
  - 계층 군집 분석(hierarchical cluster analysis): HCA
  - 이상치 탐지(outlier detection)와 특이치 탐지(novelty detection)
  - 원-클래스(one-class SVM)
  - 아이솔레이션 포레스트(isolation forest)

- 시각화(visualization)와 차원 축소(dimensionality reduction)

  차원 축소: 너무 많은 정보를 잃지 않으면서 데이터를 간소화 하는 것

  - 주성분 분석(principal component analysis): PCA
  - 커널(kernel) PCA
  - 지역적 선형 임베딩(locally linear embedding): LLE
  - t-SNE(t-distributed stochastic neighbor embedding)
- 연관 규칙 학습(association rule learning)

  : 대량의 데이터에서 특성 간의 흥미로운 관계를 찾는 것

  - 어프라이어리(Apriori)
  - 이클렛(Eclat)

준지도 학습

- 데이터에 레이블을 다는 것은 일반적으로 시간과 비용이 많이 들기 때문에 레이블이 없는 샘플이 많고 레이블된 샘플은 적은 경우가 많다. 어떤 알고리즘은 일부만 레이블이 있는 데이터를 다룰 수 있다.
- 대부분 지도 학습과 비지도 학습의 조합으로 이루어져 있다.

  ex) **심층 신뢰 신경망**(deep belief network: DBN)

  > 여러 겹으로 쌓은 **제한된 볼츠만 머신**(restricted Boltzman machine: RBM)이라 불리는 비지도 학습에 기초한다. RBM이 비지도 학습 방식으로 순차적으로 훈련된 다음 전체 시스템이 지도 학습 방식으로 세밀하게 조정된다.

강화 학습

- **에이전트**: 학습하는 시스템
- 환경을 관찰해서 행동을 실행하고, 그 결과로 **보상(**또는 벌점)을 받는다.
- 시간이 지나면서 가장 큰 보상을 얻기 위해 **정책**이라고 부르는 최상의 전략을 스스로 학습한다.
- 정책은 주어진 상황에서 에이전트가 어떤 행동을 선택해야 할지 정의한다.

▼ 1.4.2 배치 학습과 온라인 학습

배치 학습

- 시스템이 점진적으로 학습할 수 없다

  → 가용한 데이터를 모두 사용해 훈련시켜야 한다.

- 보통 오프라인에서 수행된다.

  시간과 자원을 많이 소모하기 때문이다.

- **오프라인 학습**
  - 시스템을 훈련시키고 그런 다음 제품 시스템에 적용하면 더 이상의 학습 없이 실행된다
  - 즉, 학습한 것을 단지 적용만 한다
- 새로운 데이터에 대해 학습하려면 전체 데이터를 사용하여 시스템의 새로운 버전을 처음부터 다시 훈련

  → 이전 시스템을 중지, 새 시스템으로 교체

  → 데이터를 업데이트 하고 시스템의 새 버전을 필요한 만큼 자주 훈련시키면 된다.

  → 시간이 많이 소요될 수 있으므로 더 능동적인 방법 필요 + 많은 컴퓨팅 자원이 필요

  → 큰 비용 발생

- 자원이 제한된 시스템이 스스로 학습해야 할 때 많은 양의 훈련 데이터를 나르고, 학습을 위해 매일 몇 시간씩 많은 자원을 사용하면 심각한 문제 일으킴

온라인 학습

- 데이터를 순차적으로 한 개씩 또는 **미니배치**(mini-batch, 작은 묶음 단위)로 주입하여 시스템을 훈련시킴
- 매 학습 단계가 빠르고 비용이 적게 들어 시스템은 데이터가 도착하는 대로 즉시 학습할 수 있다.
- 연속적으로 데이터를 받고 빠른 변화에 스스로 적응해야 하는 시스템에 적합하다
- 컴퓨팅 자원이 제한된 경우에도 좋다.

  → 학습이 끝난 데이터는 더는 필요하지 않으므로 버리면 되기 때문에 많은 공간 절약 가능

[외부 메모리 학습]

- 컴퓨터 한 대의 메인 메모리에 들어갈 수 없는 아주 큰 데이터셋을 학습하는 시스템
- 알고리즘이 데이터 일부를 읽어 들이고 훈련 단계를 수행하고, 전체 데이터가 모두 적용될 때까지 이 과정을 반복함
- **학습률**: 변화하는 데이터에 얼마나 빠르게 적응할 것인지
  - 높을 때: 시스템이 데이터에 빠르게 적응하지만 예전 데이터를 금방 잊어버림
  - 낮을 때: 시스템의 관성이 더 커져서 느리게 학습된다. 하지만 새로운 데이터에 있는 잡음이나 대표성 없는 데이터 포인트에 덜 민감해짐

▼ 1.4.3 사례 기반 학습과 모델 기반 학습

사례 기반 학습

- 시스템이 훈련 샘플을 기억함으로써 학습함
- 유사도 측정을 사용해 새로운 데이터와 학습한 샘플을 비교하는 식으로 일반화한다.

모델 기반 학습

- 샘플들의 모델을 만들어 예측에 사용하는 것
- **효용 함수**: 모델이 얼마나 좋은지 측정하는 함수
- **비용 함수**: 모델이 얼마나 나쁜지 측정하는 함수

머신러닝 프로젝트 과정

1. 데이터 분석
2. 모델 선택
3. 훈련 데이터로 모델 훈련

   (학습 알고리즘이 비용 함수를 최소화하는 모델 파라미터 찾기)
4. 새로운 데이터에 모델을 적용해 예측하고 이 모델이 잘 일반화되길 기대

▼ 1.5 머신러닝의 주요 도전 과제

▼ 1.5.1 충분하지 않은 양의 훈련 데이터

- 대부분의 머신러닝 알고리즘이 잘 작동하려면 데이터가 많아야 한다.
- 아주 간단한 문제에서조차도 수천 개의 데이터가 필요하고 이미지나 음성 인식 같은 복잡한 문제라면 수백만 개가 필요할지도 모른다.

▼ 1.5.2 대표성 없는 훈련 데이터

- 일반화가 잘되려면 우리가 일반화하기 원하는 새로운 사례를 훈련 데이터가 잘 대표하는 것이 중요하다.
- 샘플이 작으면 **샘플링 잡음**(우연에 의한 대표성 없는 데이터)이 생긴다.
- **샘플링 편향**: 샘플이 매우 크더라도 표본 추출 방법이 잘못되면 대표성을 띠지 못할 수 있다.

▼ 1.5.3 낮은 품질의 데이터

- 훈련 데이터가 에러, 이상치(outlier), 잡음(예를 들면 성능이 낮은 측정 장치 때문에)으로 가득하다면 머신러닝 시스템이 내재된 패턴을 찾기 어려워 잘 작동하지 않을 것이다.
- 그렇기 때문에 훈련 데이터 정제에 시간을 투자할 만한 가치는 충분하다.

[훈련 데이터 정제가 필요한 경우]

1. 일부 샘플이 이상치라는 게 명확하면 간단히 그것들을 무시하거나 수동으로 잘못된 것을 고치는 것이 좋다.
2. 일부 샘플에 특성 몇 개가 빠져있다면 (예를 들면 고객 중 5%가 나이를 기록하지 않음), 이 특성을 모두 무시할지, 이 샘플을 무시할지, 빠진 값을 채울지(예를 들면 평균 나이로), 또는 이 특성을 넣은 모델과 제외한 모델을 따로 훈련시킬 것인지 결정해야 한다.

▼ 1.5.4 관련 없는 특성

- 훈련 데이터에 관련 없는 특성이 적고 관련 있는 특성이 충분해야 시스템이 학습할 수 있을 것이다.
- 성공적인 머신러닝 프로젝트의 핵심 요소는 훈련에 사용할 좋은 특성들을 찾는 것

[특성 공학]

- **특성 선택**: 가지고 있는 특성 중에서 훈련에 가장 유용한 특성을 선택

- **특성 추출:** 특성을 결합하여 더 유용한 특성을 만듭니다. 앞서 본 것처럼 차원 축소 알고리즘이 도움이 될 수 있다.
- 새로운 데이터를 수집해 새 특성을 만든다.

▼ 1.5.5 훈련 데이터 과대적합

- 모델이 훈련 데이터에 너무 잘 맞지만 일반성이 떨어진다는 뜻
- 심층 신경망 같은 복잡한 모델은 데이터에서 미묘한 패턴을 감지할 수 있지만, 훈련 세트에 잡음이 많거나 데이터셋이 너무 작으면 (샘플링 잡음이 발생하므로) 잡음이 섞인 패턴을 감지하게 된다.

  → 당연히 이런 패턴은 새로운 샘플에 일반화되지 못한다.

[과대적합 해결 방법]

- 파라미터 수가 적은 모델을 선택하거나(예를 들면 고차원 다항 모델보다 선형 모델), 훈련 데이터에 있는 특성 수를 줄이거나, 모델에 제약을 가하여 단순화시킨다.
- 훈련 데이터를 더 많이 모은다.
- 훈련 데이터의 잡음을 줄인다(예를 들면 오류 데이터 수정과 이상치 제거).

▼ 1.5.6 훈련 데이터 과소적합

- 과대적합의 반대
- 모델이 너무 단순해서 데이터의 내재된 구조를 학습하지 못할 때 일어난다.

[과소적합 해결 방법]

- 모델 파라미터가 더 많은 강력한 모델 선택
- 학습 알고리즘에 더 좋은 특성을 제공
- 모델의 제약을 줄이기

▼ 1.6 테스트와 검증

1. 모델이 새로운 샘플에 얼마나 잘 일반화될지 아는 유일한 방법은 새로운 샘플에 실제로 적용해보는 것이다.
2. 훈련 데이터를 훈련 세트와 테스트 세트 두 개로 나누는 것
   a. 훈련 세트를 사용해 모델을 훈련

b. 테스트 세트를 사용해 모델을 테스트

- **일반화 오차(또는 외부 샘플 오차)**
  - 새로운 샘플에 대한 오류 비율
  - 테스트 세트에서 모델을 평가함으로써 이 오차에 대한 추정값estimation을 얻는다.

    → 이 값은 이전에 본 적이 없는 새로운 샘플에 모델이 얼마나 잘 작동할지 알려준다.

    → 훈련 오차가 낮지만 (즉, 훈련 세트에서 모델의 오차가 적음) 일반화 오차가 높다면 이는 모델이 훈련 데이터에 과대적합되었다는 뜻

▼ 1.6.1 하이퍼파라미터 튜닝과 모델 선택

[모델 평가]

1. 모델 여러 개 중에서 선택을 고민하고 있다면 모델을 훈련 세트로 훈련한 뒤, 테스트 세트를 사용해 얼마나 잘 일반화 되는지 비교해보면 된다.
2. 모델을 선택했다면, 여러 개의 하이퍼파라미터 값으로 여러 개의 모델을 훈련시켜 가장 적합한 하이퍼파라미터를 찾으면 된다.

   → 실제로 모델을 돌려보면 성능이 예상만큼 좋지 않다.

   모델과 하이퍼파라미터가 테스트 세트에 최적화된 모델을 만들었기 때문

[해결 방법]

- **홀드아웃 검증(holdout validation)**

  훈련의 일부를 떼어내어 여러 후보 모델을 평가하고 가장 좋은 하나를 선택

  - **검증 세트(validation set / 개발 세트, 데브 세트)**

    새로운 아웃 세트

    1. 훈련 세트에서 다양한 하이퍼파라미터 값을 가진 여러 모델을 훈련
    2. 검증 세트에서 가장 높은 성능을 내는 모델을 선택
    3. 홀드아웃 검증 과정이 끝나면 이 최선의 모델을 (검증 세트를 포함한) 전체 훈련 세트에서 다시 훈련하여 최종 모델을 제작
    4. 최종 모델을 테스트 세트에서 평가하여 일반화 오차를 추정

- **교차 검증(cross-validation)**

작은 검증 세트를 여러 개 사용해서 반복적인 교차 검증 수행

검증 세트마다 나머지 데이터에서 훈련한 모델을 해당 검증 세트에서 평가

- **장점**: 모든 모델의 평가를 평균하면 훨씬 정확한 성능을 측정할 수 있다.
- **단점**: 훈련 시간이 검증 세트의 개수에 비례해 늘어난다.

[데이터 불일치]

어떤 경우에는 쉽게 많은 양의 훈련 데이터 얻을 수 있지만 이 데이터가 실제 제품에 사용될 데이터를 완벽하게 대표하지 못할 수 있다.

- **훈련-개발 세트(train-dev set)**
  - 훈련 사진의 일부를 떼어내어 또 다른 세트를 만드는 것
  - 모델을 (훈련-개발 세트가 아니라 훈련 세트에서) 훈련한 다음 훈련-개발 세트에서 평가한다.
  - 모델이 잘 작동하면 훈련 세트에 과대적합된 것이 아니다. 이 모델이 검증 세트에서 나쁜 성능을 낸다면 이 문제는 데이터 불일치에서 오는 것이다.
  - 모델이 훈련-개발 세트에서 잘 작동하지 않는다면 이는 훈련 세트에 과대적합된 것이다.

    → 모델을 규제하거나 더 많은 훈련 데이터를 모으거나 훈련 데이터 정제를 시도해봐야 한다.