1.5.1 과소적합과 과잉적합

- [그림 1.13]의 1차 모델은 "과소적합"
  - 모델의 '용량이 작아' 오차가 클 수밖에 없는 현상
- 비선형 모델을 사용하는 대안
  - [그림 1-13]의 2차, 3차, 4차, 12차는 다항식 곡선을 선택한 예
  - 1차(선형)에 비해 오차가 크게 감소함

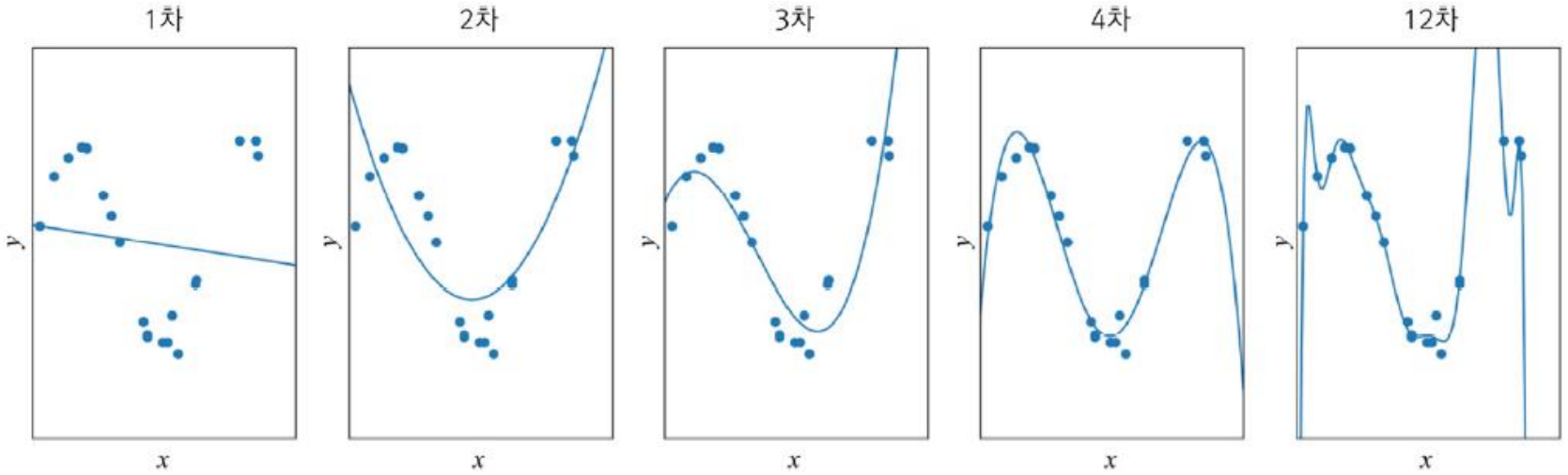


그림 1-13 과소적합과 과잉적합 현상

1.5.1 과소적합과 과잉적합

과잉적합

- 12차 다항식 곡선을 채택한다면 훈련집합에 대해 거의 완벽하게 근사화함
- 하지만 '새로운' 데이터를 예측한다면 큰 문제 발생
  - x_0에서 빨간 막대 근방을 예측해야 하지만 빨간 점을 예측
- 이유는 '용량이 크기' 때문. 학습 과정에서 **잡음까지 수용** → 과잉적합 현상
- 적절한 용량의 모델을 선택하는 모델 선택 작업이 필요함

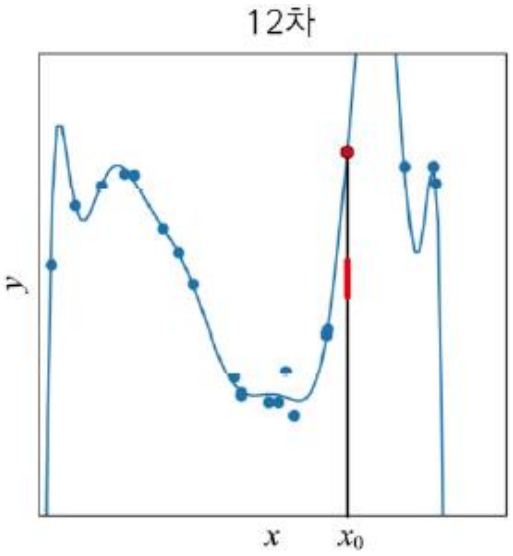


그림 1-14 과잉적합되었을 때 부정확한 예측 현상

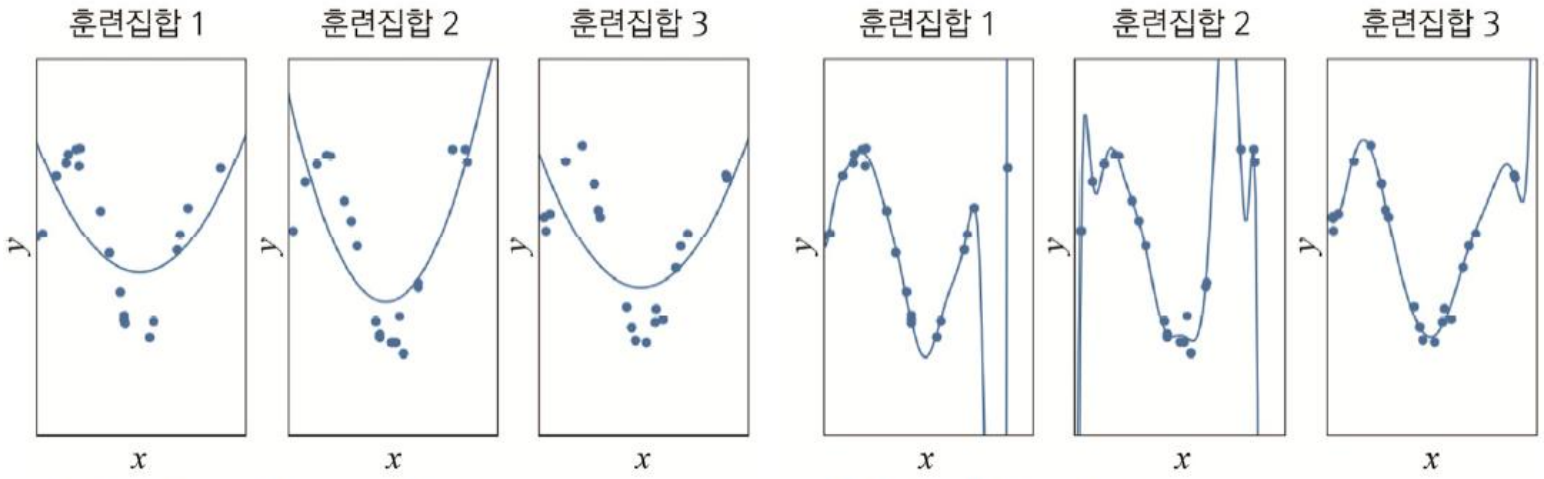
1.5.2 바이어스와 분산

1차~12차 다항식 모델의 비교 관찰

- 1~2차는 훈련집합과 테스트집합 모두 낮은 성능
- 12차는 훈련집합에 높은 성능을 보이나 테스트집합에서는 낮은 성능 → 낮은 일반화 능력
- 3~4차는 훈련집합에 대해 12차보다 낮겠지만 테스트집합에는 높은 성능 → 높은 일반화 능력

1.5.2 바이어스와 분산

- 훈련집합을 여러 번 수집하여 1차~12차에 적용하는 실험
  - 2차는 매번 큰 오차 → 바이어스가 큼. 하지만 비슷한 모델을 얻음 → 낮은 분산
  - 12차는 매번 작은 오차 → 바이어스가 작음. 하지만 크게 다른 모델을 얻음 → 높은 분산
  - 일반적으로 용량이 작은 모델은 바이어스는 크고 분산은 작음. 복잡한 모델은 바이어스는 작고 분산은 큼
  - 바이어스와 분산은 트레이드오프 관계



(a) 2차 모델(바이어스는 크고, 분산은 작음)

(b) 12차 모델(바이어스는 작고, 분산은 큼)

그림 1-15 모델의 바이어스와 분산 특성

1.5.2 바이어스와 분산

기계 학습의 목표

- **낮은 바이어스와 낮은 분산**을 가진 예측기 제작이 목표. 즉 왼쪽 아래 상황

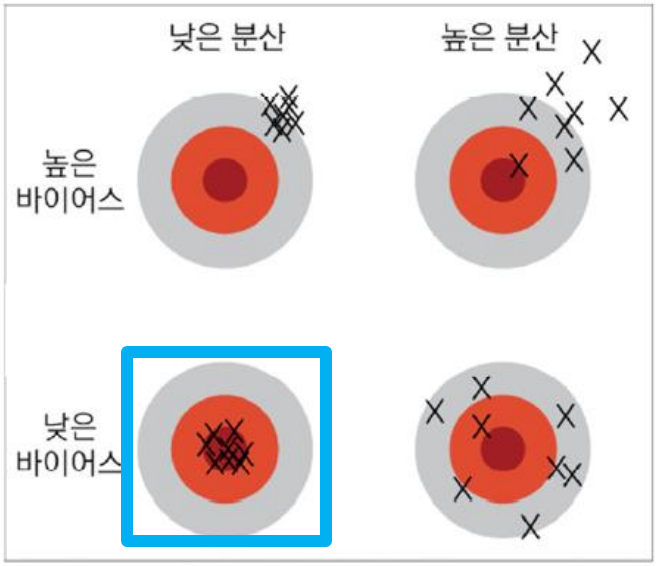


그림 1-16 바이어스와 분산

- 하지만 바이어스와 분산은 트레이드오프 관계
- 따라서 **바이어스 희생을 최소로 유지하며 분산을 최대로 낮추는 전략** 필요

1.5.3 검증집합과 교차검증을 이용한 모델 선택 알고리즘

검증집합을 이용한 모델 선택

- 훈련집합과 테스트집합과 다른 별도의 검증집합을 가진 상황

알고리즘 1-2 검증집합을 이용한 모델 선택

입력 : 모델집합 Ω, 훈련집합, 검증집합, 테스트집합
출력 : 최적 모델과 성능

```
1  for (Ω에 있는 각각의 모델)
2      모델을 훈련집합으로 학습시킨다.
3      검증집합으로 학습된 모델의 성능을 측정한다.  // 검증 성능 측정
4  가장 높은 성능을 보인 모델을 선택한다.
5  테스트집합으로 선택된 모델의 성능을 측정한다.
```

훈련, 검증, 테스트에 사용하는 데이터는 절대 겹치면 안됨.

1.5.3 검증집합과 교차검증을 이용한 모델 선택 알고리즘

교차검증cross validation

- 비용 문제로 별도의 검증집합이 없는 상황에 유용한 모델 선택 기법
- 훈련집합을 등분하여, 학습과 평가 과정을 여러 번 반복한 후 평균 사용

알고리즘 1-3 교차검증에 의한 모델 선택

입력 : 모델집합 Ω, 훈련집합, 테스트집합, 그룹 개수 k

출력 : 최적 모델과 성능

```
1  훈련집합을 k개의 그룹으로 등분한다.
2  for (Ω에 있는 각각의 모델)
3        for (i=1 to k)
4              i번째 그룹을 제외한 k-1개 그룹으로 모델을 학습시킨다.
5              학습된 모델의 성능을 i번째 그룹으로 측정한다.
6        k개 성능을 평균하여 해당 모델의 성능으로 취한다.
7  가장 높은 성능을 보인 모델을 선택한다.
8  테스트집합으로 선택된 모델의 성능을 측정한다.
```

k는 주로 4 또는 5를 사용

1.5.3 검증집합과 교차검증을 이용한 모델 선택 알고리즘

부트스트랩boot strap

난수를 이용한 샘플링 반복

알고리즘 1-4 부트스트랩을 이용한 모델 선택

입력 : 모델집합 Ω, 훈련집합, 테스트집합, 샘플링 비율 $p(0<p\leq 1)$, 반복횟수 T
출력 : 최적 모델과 성능

```
1  for (Ω에 있는 각각의 모델)
2      for (i=1 to T)
3          훈련집합 𝕏에서 pn개 샘플을 뽑아 새로운 훈련집합 𝕏′를 구성한다. 이때 대치를 허용한다.
4          𝕏′로 모델을 학습시킨다.
5          𝕏 − 𝕏′를 이용하여 학습된 모델의 성능을 측정한다.
6      T개 성능을 평균하여 해당 모델의 성능으로 취한다.
7  가장 높은 성능을 보인 모델을 선택한다.
8  테스트집합으로 선택된 모델의 성능을 측정한다.
```

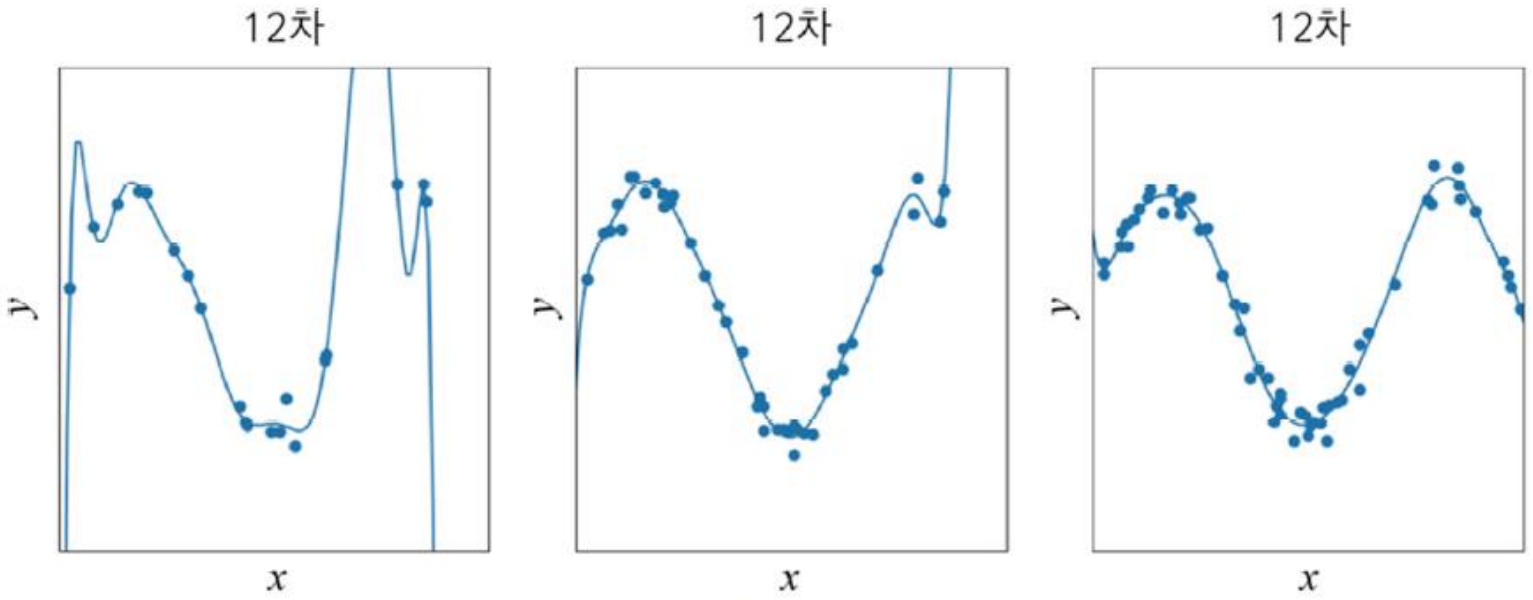
n 은 훈련집합의 사이즈

1.5.4 모델 선택의 한계와 현실적인 해결책

- **[알고리즘 1-2, 1-3, 1-4]에서 모델 집합 Ω**
  - [그림 1-13]에서는 서로 다른 차수의 다항식이 Ω인 셈
  - 현실에서는 아주 다양
    - 신경망(3,4,8장), 강화 학습(9장), 확률 그래피컬 모델(10장), SVM(11장), 트리 분류기(12장) 등이 선택 대상
    - 신경망을 채택하더라도 MLP(3장), 깊은 MLP(4장), CNN(4장) 등 아주 많음
- **현실에서는 경험으로 큰 틀 선택한 후**
  - 모델 선택 알고리즘으로 세부 모델 선택하는 전략 사용
  - 예) CNN을 사용하기로 정한 후, 은닉층 개수, 활성함수, 모멘텀 계수 등을 정하는데 모델 선택 알고리즘을 적용함

1.6.1 데이터 확대

- 데이터를 더 많이 수집하면 일반화 능력이 향상됨



(a) 훈련집합의 크기 = 20 (b) 훈련집합의 크기 = 40 (c) 훈련집합의 크기 = 60

그림 1-17 데이터를 확대하여 일반화 능력을 향상함

1.6.1 데이터 확대

- 데이터 수집은 많은 비용이 듦
  - 그라운드 트루스를 사람이 일일이 레이블링해야 함
- 인위적으로 데이터 확대
  - 훈련집합에 있는 샘플을 변형함
  - 약간 회전 또는 와핑 (부류 소속이 변하지 않게 주의)

문제점은 ?

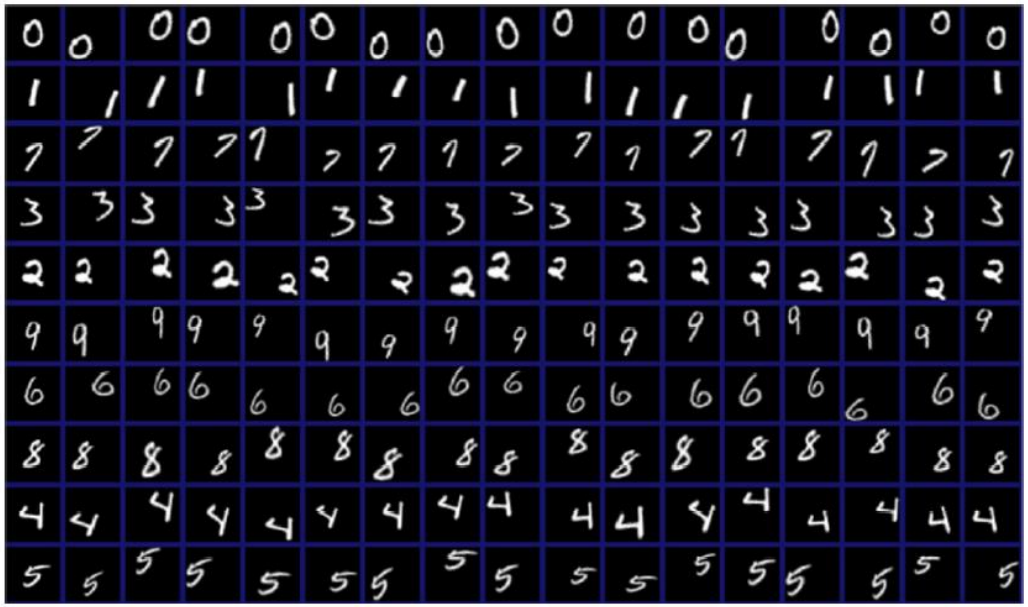


그림 5-24 필기 숫자 데이터의 다양한 변형[8]

1.6.2 가중치 감쇠

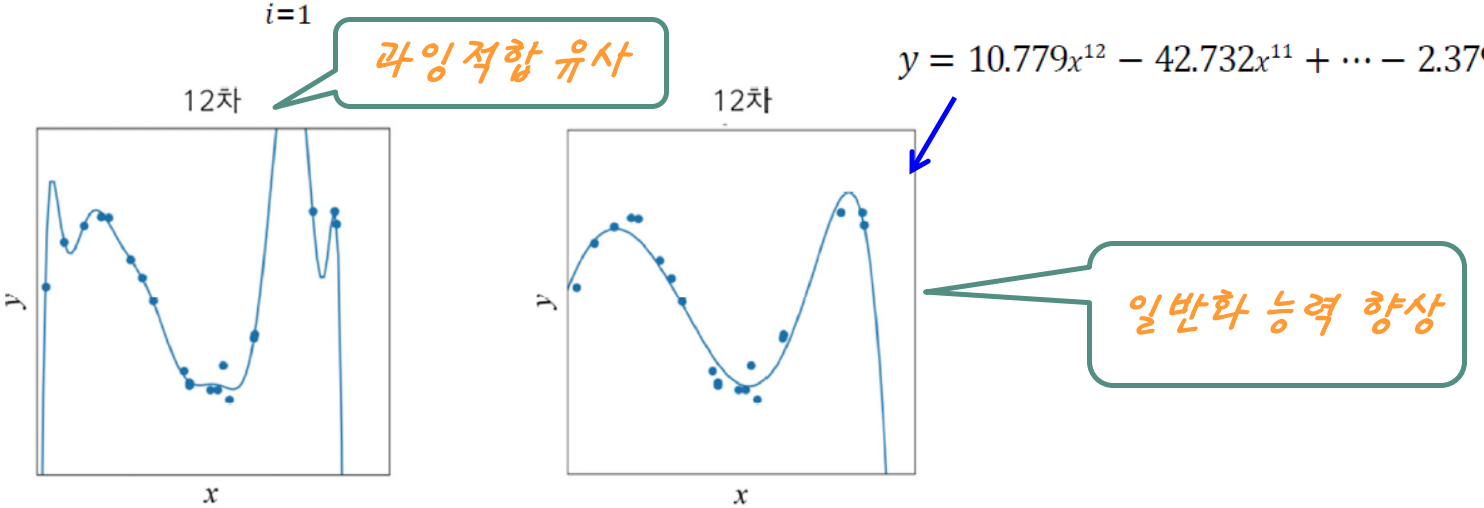
가중치를 작게 조절하는 기법

- [그림 1-18(a)]의 12차 곡선은 가중치가 매우 큼

$$y = 1005.7x^{12} - 27774.4x^{11} + \cdots - 22852612.5x^{1} - 12.8$$

- 가중치 감쇠는 개선된 목적함수를 이용하여 가중치를 작게 조절하는 규제 기법
  - 식 (1.11)의 두 번째 항은 규제 항으로서 가중치 크기를 작게 유지해줌

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n}(f_{\Theta}(\mathbf{x}_i) - y_i)^2 + \lambda\|\Theta\|_2^2 \qquad (1.11)$$



(a) 가중치 감쇠 적용 안 함[식 (1.8)의 목적함수] (b) 가중치 감쇠 적용함[식 (1.11)의 목적함수]

그림 1-18 가중치 감쇠에 의한 규제 효과

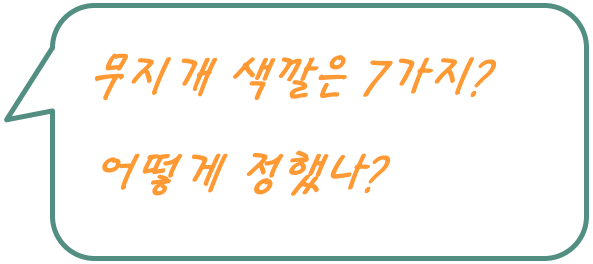
1.7.1 지도 방식에 따른 유형

지도 학습

- 특징 벡터 $\mathbb{X}$와 목푯값 $\mathbb{Y}$가 모두 주어진 상황
- 회귀와 분류 문제로 구분

비지도 학습

- 특징 벡터 $\mathbb{X}$는 주어지는데 목푯값 $\mathbb{Y}$가 주어지지 않는 상황
- 군집화 과업 (고객 성향에 따른 맞춤 홍보 응용 등)
- 밀도 추정, 특징 공간 변환 과업
- 6장의 주제



1.7.1 지도 방식에 따른 유형

강화 학습

- 목푯값이 주어지는데, 지도 학습과 다른 형태임
- 예) 바둑
  - 수를 두는 행위가 샘플인데, 게임이 끝나면 목푯값 하나가 부여됨
    - 이기면 1, 패하면 -1을 부여
  - 게임을 구성한 샘플들 각각에 목푯값을 나누어 주어야 함

샘플 그룹 당 목표값 하나씩

- 9장의 주제

준지도 학습

데이터 수집의 비용과 시간을 줄이기 위함

- 일부는 $\mathbb{X}$와 $\mathbb{Y}$를 모두 가지지만, 나머지는 $\mathbb{X}$만 가진 상황
- 인터넷 덕분으로 $\mathbb{X}$의 수집은 쉽지만, $\mathbb{Y}$는 수작업이 필요하여 최근 중요성 부각
- 7장의 주제

1.8.1 인공지능과 기계 학습의 간략한 역사

1950	인공지능 여부를 판별하는 튜링 테스트[Turing1950]
1956	최초의 인공지능 학술대회인 다트머스 콘퍼런스 개최. '인공지능' 용어 탄생[McCarthy1955]
1958	로젠블렛이 퍼셉트론 제안[Rosenblatt1958]
	인공지능 언어 Lisp 탄생
1959	사무엘이 기계 학습을 이용한 체커 게임 프로그램 개발[Samuel1959]
1969	민스키가 퍼셉트론의 과대포장 지적. 신경망 내리막길 시작[Minsky1969]
	제1회 IJCAI International Joint Conference on Artificial Intelligence 개최
1972	인공지능 언어 Prolog 탄생
1973	Lighthill 보고서로 인해 인공지능 내리막길, 인공지능 겨울AI winter 시작
1974	웨어보스가 오류 역전파 알고리즘을 기계 학습에 도입[Werbos1974]
1975경	의료진단 전문가 시스템 Mycin – 인공지능에 대한 관심 부활
1979	「IEEE Transactions on Pattern Analysis and Machine Intelligence」 저널 발간
1980	제1회 ICML International Conference on Machine Learning 개최
	후쿠시마가 NeoCognitron 제안[Fukushima1980]
1986	「Machine Learning」 저널 발간
	『Parallel Distributed Processing』 출간
	다층 퍼셉트론으로 신경망 부활

1.8.1 인공지능과 기계 학습의 간략한 역사

1987	Lisp 머신의 시장 붕괴로 제2의 인공지능 겨울
	UCI 리포지토리 서비스 시작
	NIPS Neural Information Processing Systems 콘퍼런스 시작
1989	「Neural Computation」 저널 발간
1993	R 언어 탄생
1997	IBM 딥블루가 세계 체스 챔피언인 카스파로프 이김
	LSTM Long short-term memory 개발됨
1998경	SVM이 MNIST 인식 성능에서 신경망 추월
1998	르쿤이 CNN의 실용적인 학습 알고리즘 제안[LeCun1998]
	『Neural Networks: Tricks of the Trade』 출간
1999	NVIDIA 사에서 GPU 공개
2000	「Journal of Machine Learning Research」 저널 발간
	OpenCV 최초 공개
2004	제1회 그랜드 챌린지(자율 주행)
2006	층별학습 탄생[Hinton2006a]
2007경	딥러닝이 MNIST 인식 성능에서 SVM 추월
2007	GPU 프로그래밍 라이브러리인 CUDA 공개

1.8.1 인공지능과 기계 학습의 간략한 역사

	어번 챌린지(도심 자율 주행)
	Scikit-learn 라이브러리 최초 공개
2009	Theano 서비스 시작
2010	ImageNet 탄생
	제1회 ILSVRC 대회
2011	IBM 왓슨이 제퍼디 우승자 꺾음
2012	MNIST에 대해 0.23% 오류율 달성
	AlexNet 발표 (3회 ILSVRC 우승)
2013	제1회 ICLR International Conference on Learning Representations 개최
2014	Caffe 서비스 시작
2015	TensorFlow 서비스 시작
	OpenAI 창립
2016	알파고와 이세돌의 바둑 대회에서 알파고 승리[Silver2016]
	『Deep Learning』 출간
2017	알파고 제로[Silver2017]

1.8.3 사회적 전망

미래의 직업 변화

- 시의적절하고 심사숙고 해야 할 객관적 담론
- 프레이는 702개 직업의 사라질 위기를 확률로 계산 [Frey2017]
- 텔레마케터 99% …… …… …… ….. ….. ….. 오락 치료사 0.28%

기계가 사람을 지배할지 모른다는 두려움

- 알파고 이후 매스컴을 통해 여과 없이 전파. 쓸데없는 과장에 불과
- 냇가에 다리 놓는 일과 목포-제주에 대교를 놓는 일은 규모만 다를 뿐 본질적으로 같은 일
- 오목 프로그램이나 바둑 프로그램은 규모만 다를 뿐 본질적으로 같은 일. 오목은 간단한 규칙으로 구현 가능하나 바둑은 미분을 사용한 복잡한 기계 학습 알고리즘 사용
- 현재 기계 학습은 온통 수학과 컴퓨터 알고리즘일 뿐