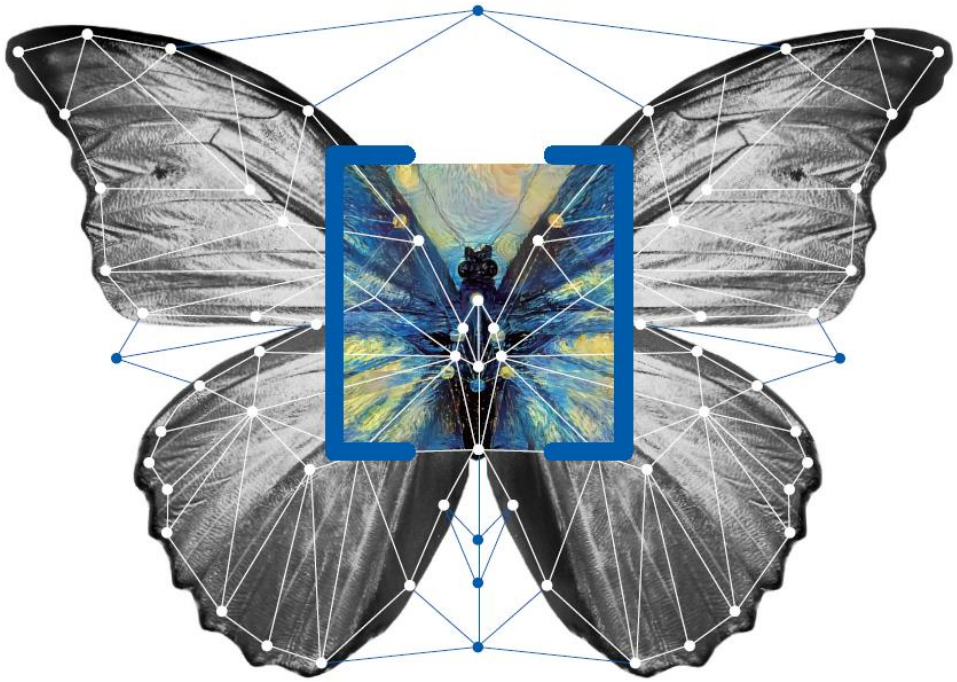
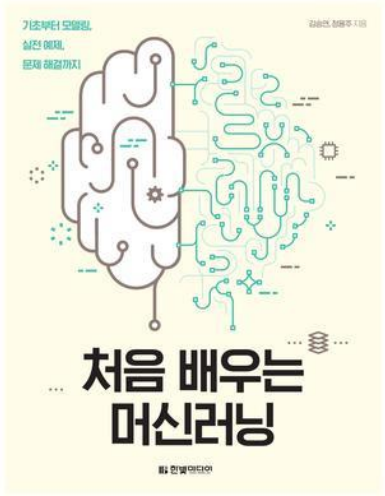




MACHINE LEARNING 기계 학습

기계 학습의 주요 개념

PREVIEW

- 머신 러닝의 4가지 핵심 개념
  - 모델 : 데이터를 바라보는 시점과 가정
  - 손실 함수 : 모델의 수식화된 학습목표
  - 최적화 : 손실 함수로 표현된 모델을 실제로 학습
  - 모델 평가 : 모델의 성능이 실제 상황에서 어떨지 추정

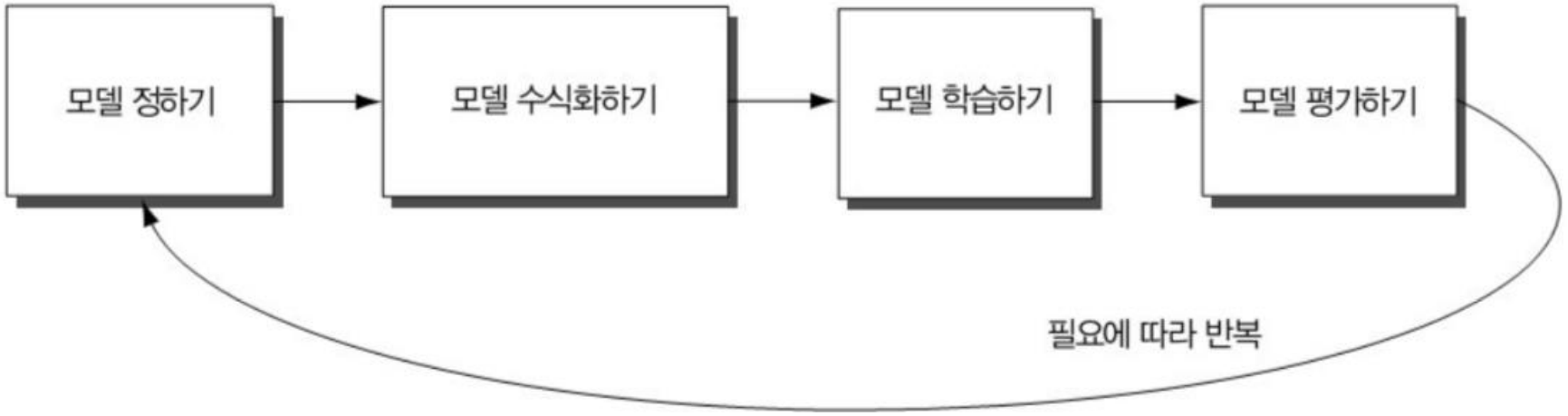
모델이란

- 데이터 분석을 하거나 머신 러닝으로 문제를 해결하려면 무엇부터 해야 할까요?
  - "데이터가 어떤 패턴을 가지지 않을까? 그 패턴들을 이용하면 어떻게든 되지 않을까?
  - 패턴이 있을 것이라는 생각도 어찌 보면 데이터 자체에 대한 믿음
  - 그러한 믿음을 수학에서는 **가정**이라고 함
  - 이러한 가정을 한데 모은 것을 머신 러닝에서는 **모델**이라고 함
  - 현재 상태를 어떠한 시각으로 바라보고 어떠한 기대를 하고 있는가 하는 것이 모델임

머신러닝의 과정

- 일반적인 머신 러닝의 과정
  - 모델 정하기(데이터가 어떻게 생겼을지 가정하기)
  - 모델의 학습 목표를 수식화하기
  - 실제 데이터로 모델 학습하기(최적화)
  - 평가하기

그림 2-1 머신러닝의 과정



머신 러닝의 과정

- 과정 1에서 데이터가 어떻게 생겼을지 이미 가정을 해놓고 왜 또 과정3에서 실제데이터로 학습이 필요할까?

- 과정 1 "이 데이터에서 x와 y는 선형적인 관계를 가진다. 즉, 임의의 w에 대해 y=wx와 같은 관계를 가질 것이다"

- 과정 3 "데이터를 토대로 추측해본 결과 y=2x와 같은 형태를 가진다. 즉, 데이터에 가장 잘 맞는 w는 2이다"

- 정해진 함수(위에서 y=wx) 안에서 함수의 파라미터(여기서는 w)를 데이터를 통해 추측하는 것을 학습이라고 함
  - 학습이라는 것은 그 모델이 표현하는 함수 집합 중에서 가장 데이터에 적합한 함수를 고를 과정

간단한 모델

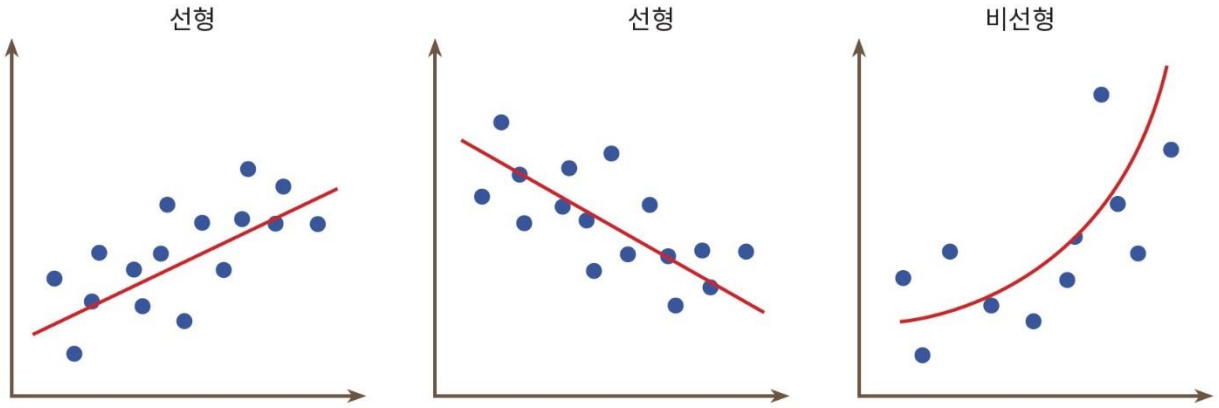
- 모델이 간단하다 = “데이터 구조가 간단하다“

- 선형 모델(linear model)
  - 가장 간단하지만 아주 효과적인 모델
  - 선형 회귀(linear regression)
  - 예측할 결과값을 y라 하고 예측에서 사용하는 값을 $x_1, x_2, \ldots$,라 가정
  - 선형 회귀 정의

  - 수식 : $y = w_0 + w_1 x_1 + w_2 x_2 + \ldots$
  - 출력 값(y)이 입력 값(피처)($x_1, x_2, \ldots$)에 대해 선형적인 관계

선형 회귀

- "회귀"란 일반적으로 데이터들을 2차원 공간에 찍은 후에 이들 데이터들을 가장 잘 설명하는 직선이나 곡선을 찾는 문제라고 할 수 있다.
- y = f(x)에서 출력 y가 실수이고 입력 x도 실수일 때 함수 f(x)를 예측하는 것이 회귀이다.



선형 회귀의 예

- 부모의 키와 자녀의 키의 관계 조사
- 면적에 따른 주택의 가격
- 연령에 따른 실업율 예측
- 공부 시간과 학점 과의 관계
- CPU 속도와 프로그램 실행 시간 예측

선형 회귀

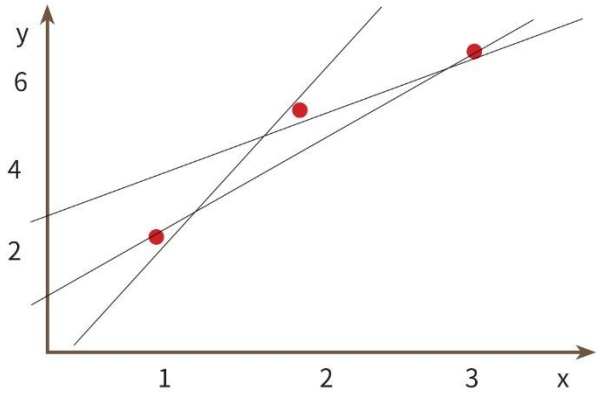
- 직선의 방정식: f(x) = mx+b
- 선형 회귀는 입력 데이터를 가장 잘 설명하는 기울기와 절편값을 찾는 문제이다

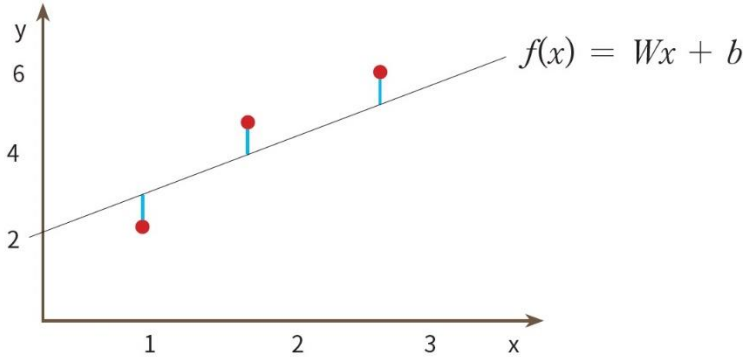
- 선형 회귀의 기본식: f(x) = Wx+b
  - 기울기->가중치
  - 절편->바이어스

선형 회귀의 원리

- 선형 관계

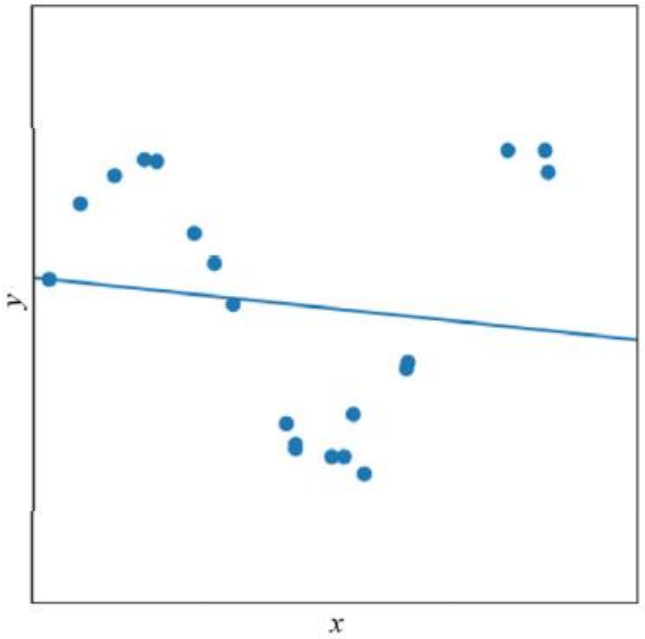
x	y
1	2
2	5
3	6





간단한 모델

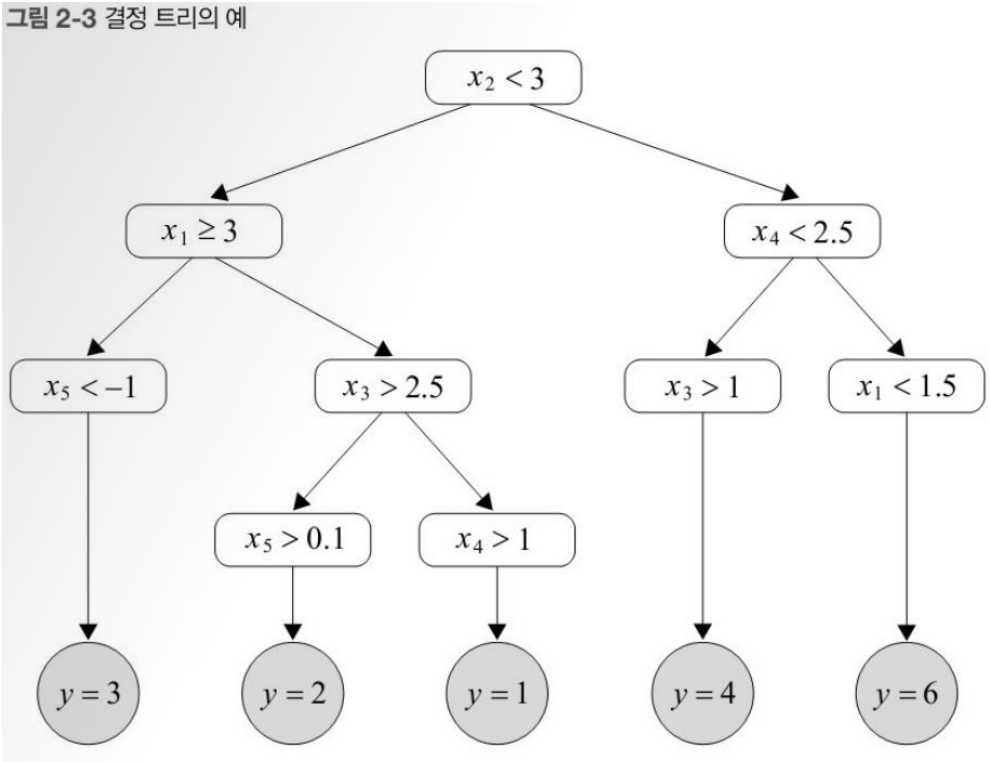
- 간단한 모델은 복잡한 관계를 학습할 수 없다
  - 입출력 데이터의 관계가 단순한 선형 관계가 아니면 어떻게 될까?
  - 예) $w_0x_1/(w_1+x_1)$ 같은 관계의 경우



- 간단한 모델의 정리
  - 데이터가 복잡하지 않고 간단하게 생겼다고 가정
  - 결과를 이해하기 쉽다
  - 학습이 쉽다
  - 가정 자체가 강력해서 모델의 표현 능력에 제약이 많다

복잡한 모델

- 복잡한 모델
  - 간단한 모델에 비해 상대적으로 더 복잡하게 생겼다고 가정할 때 사용
  - 상대적인 구분
  - 복잡한 모델은 모델의 유연성을 더 중요시 함
  - 예) 결정 트리(decision tree)



결정 트리

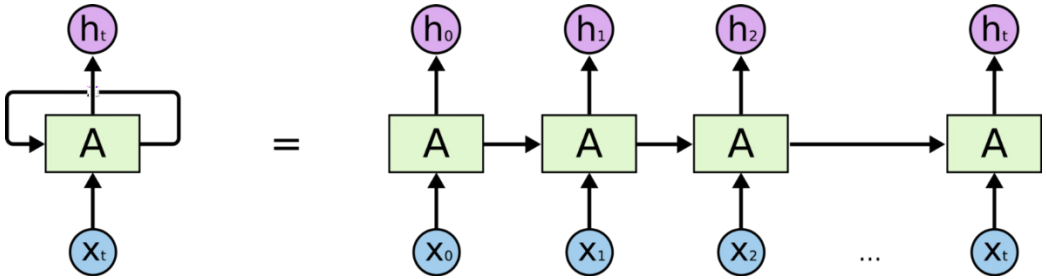
- 결정 트리의 정의
  - 트리의 한 분기점마다 한가지 조건(보통 입력의 한 부분)을 검사하여 분기를 함
  - 모든 분기가 끝나는 리프노드(맨 끝의 노드)에는 결과값이 들어 있다

- 결정 트리의 특징
  - 선형 모델과 달리 복수의 비교식으로 정의
  - 비교식은 무수히 늘어날 수 있기 때문에 더 일반적이고 유연한 가정을 할 수 있음
  - 예) 앞서 설명한 선형 모델도 결정 트리로 구현 가능(각 분기점마다 모든 값에 대한 결과를 작성해서 결과를 저장)

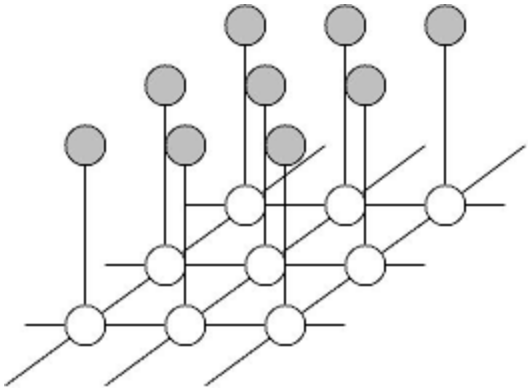
- 복잡한 모델의 장점
  - 복잡한 데이터의 모델링에 적합
  - 유연성이 뛰어난 모델은 많은 종류의 데이터를 모델링
  - 데이터의 모든 부분에 대해 일일이 가정을 만듦으로써 불필요한 노이즈까지 학습하여 성능이 나빠짐

구조가 있는 모델

- 몇가지 특정 상황에서 요긴하게 쓰이는 모델
  - 단순히 입력과 출력의 상관 관계를 학습할 뿐만 아니라 데이터 구조 자체를 모델링 하는 특이한 모델
  - 구조가 있다는 말은 입력과 출력 요소가 서로 연관 관계가 있는 것으로 이해해야 함
    - 순차 모델 (sequence model)
      - RNN(Recurrent(반복적, 되풀이) Neural Net) : 언어의 경우 단어 다음에 단어가 오는 것이 확실, 음성인식, 단어의 의미 파악, 대화
    - 그래프 모델 (graphical model)
      - 마르코프 랜덤 필드 : 관측된 값은 각 위치에서의 숨겨진 상태에 의해 결정



RNN



마르코프 랜덤 필드

좋은 모델이란?

- 편향-분산 트레이드 오프
  - 어떤 모델을 데이터 x로 학습한 결과를 $\hat{f}(x)$, 가능한 모든 모델 중에서 가장 좋은 모델을 f, 원하는 출력값(정답)을 y라 할 때 가장 기본적인 에러인 평균 제곱근 에러(mean squared error)는 다음과 같은 성질을 만족 ($E[]$는 기대값, σ줄일 수 없는 에러)
    - *1)* $E[(y - \hat{f}(x))^2] = Bias[\hat{f}(x)]^2 + Var[\hat{f}(x)] + \sigma^2$
    - *2)* $Bias[\hat{f}(x)] = E[(\hat{f}(x))^2] - f(x)$
    - *3)* $Var[\hat{f}(x)] = E[(\hat{f}(x) - \hat{f}(x))^2]$

    - 1번식 : 모델이 데이터를 예측할 때 생기는 오류가 편향(bias)의 제곱과 분산으로 쪼개진다. 모델이 더 나은 성능을 내려면 편향을 줄이거나 분산을 줄여야 한다
    - 2번식 : 데이터의 학습한 결과와 이상적인 모델 간의 차이. 데이터로 학습한 결과가 모델의 표현력이 부족해서 이상적인 모델과 많은 차이를 보인다면 값이 커지게 된다. 간단한 모델일수록 가정이 강합니다. 따라서 표현력이 부족하므로 편향이 크게 나타나게 된다. 약한 가정을 하는 복잡한 모델이라 할지라도 데이터와 터무니없이 맞지 않으면 편향이 강하게 나타남
    - 3번식 : 데이터를 이용해서 얻은 모델이 학습할 때마다 얼마나 달라질 수 있는지 나타냄. 일반적으로 모델이 복잡할 수록 학습할 때마다 나타나는 모델 편차가 크다. 즉, 모델이 복잡할 수록 분산이 더 크게 나타남.

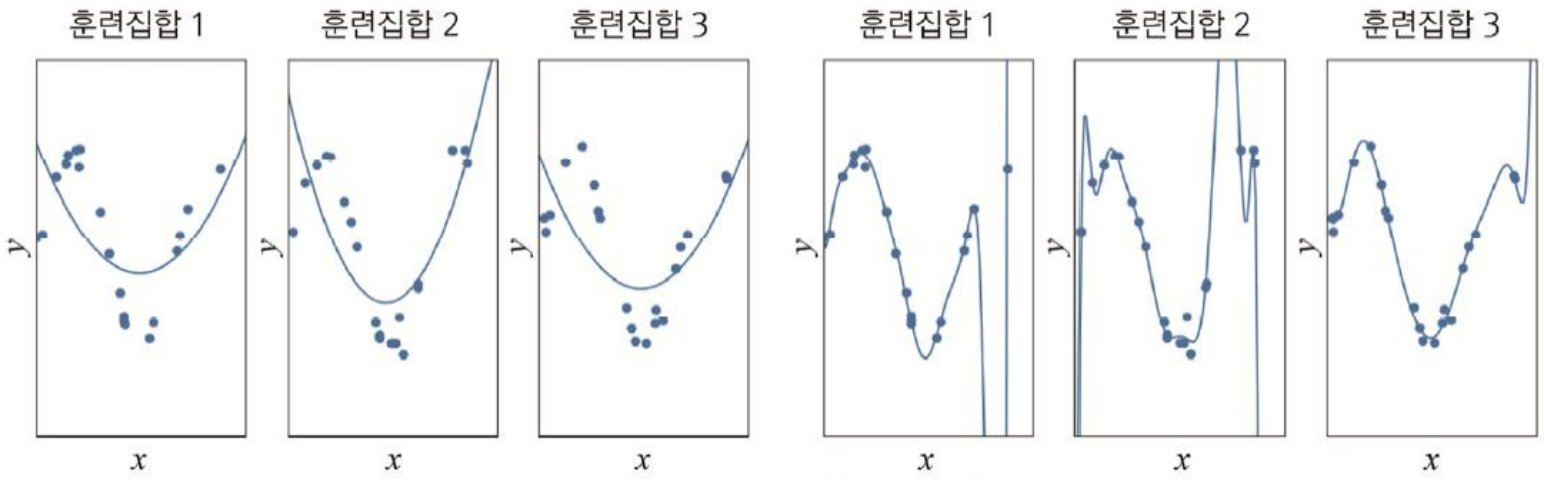
모델의 성능을 극대화 하려면 어떤 모델을?

- 모델의 성능을 극대화하기 위해서는
  - 편향이 너무 크기 않아 적당히 유연하면서
  - 너무 복잡하지 않아 분산이 적게 나오는 모델이 적합
  - 표현력이 크다고 항상 좋은 것은 아니며
  - 항상 단순한 모델만 이용해서는 제대로 된 결과를 얻지 못함.

- 편향이나 분산을 직접적으로 줄이는 대표적인 예
  - 부스팅(boosting) : 간단한 모델을 여러 개 조합하여 편향을 줄이는 방법
  - 랜덤 포레스트(random forest) : 복잡한 모델인 결정 트리를 여러 개 조합하여 분산을 줄이는 방법

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

바이어스와 분산

- 기계 학습의 목표
  - 낮은 바이어스와 낮은 분산을 가진 예측기 제작이 목표. 즉 왼쪽 아래 상황

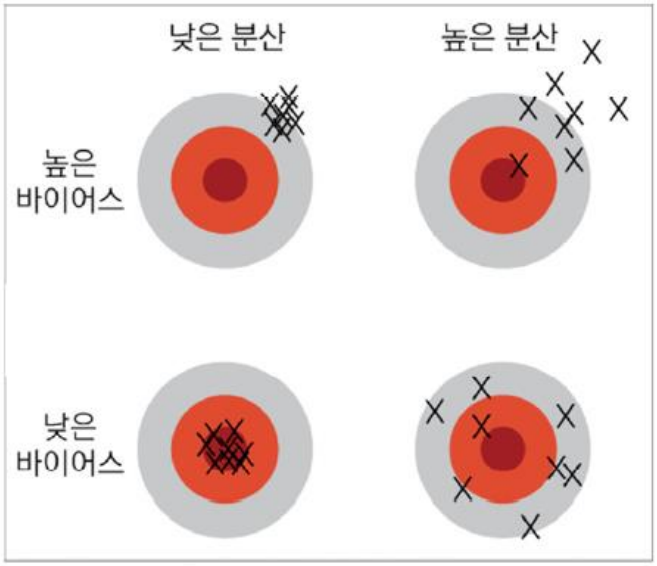


그림 1-16 바이어스와 분산

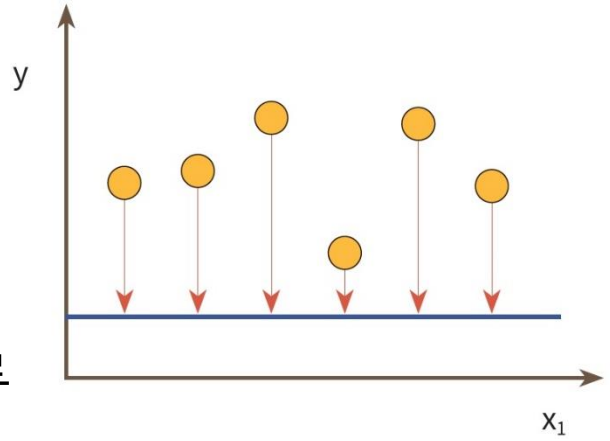
  - 하지만 바이어스와 분산은 트레이드오프 관계
  - 따라서 바이어스 희생을 최소로 유지하며 분산을 최대로 낮추는 전략 필요

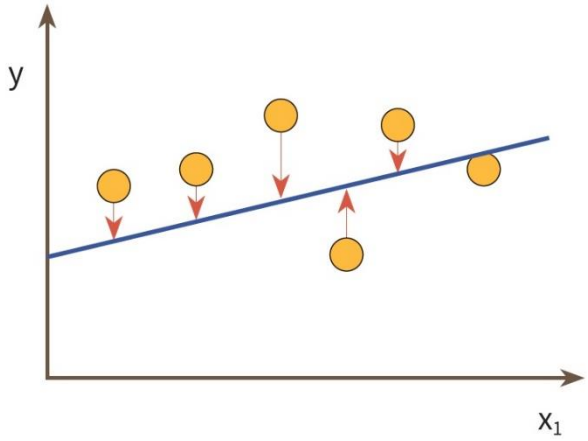
정규화

- 정해진 모델이 필요 이상으로 복잡해지지 않도록 조절하는 트릭
  - 모델이 데이터에 비해 필요 이상으로 복잡하면 불필요한 노이즈까지 학습해서 학습할 때는 성능이 좋지만, 실제로 사용할 때는 좋지 않은 성능을 보임
- 모델의 복잡도를 줄이는 방법
  - 모델 변경 : 데이터를 표현하는 방법을 완전히 새롭게 전환해서 적합한 모델을 찾는 방법
  - 정교화 : 모델에 들어있는 인자에 제한을 두어 모델이 필요 이상으로 복잡해지지 않게 하는 방법
  - 예) 선형회귀 : 입력은 3개(x_1, x_2, x_3), 출력은 1개(y)
    - 선형 회귀 모델 : $y = w_0 + w_1x_1 + w_2x_2 + w_3x_3$
    - 이 모델을 간단하게 만든다면
    - $y = w_0 + w_1x_1$
    - 더 복잡한 연관관계를 이용하는 모델은
    - $y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1x_2 + w_5x_2x_3 + w_6x_1x_3$
    - 다음과 같은 정규화 가능
    - $y = w_0 + w_1x_1 + w_2x_2 + w_3x_3$ (단, w_1, w_2, w_3, 중에 두개만 0이 아닌 값을 가진다)

손실 함수 : 모델의 수식화된 학습 목표

- 손실 함수
  - 모델이 실제로 데이터를 바르게 표현했는지 혹은 얼마나 예측이 정확한지 수학적으로 표현하는 것
  - 예) 선형회귀에서 만들어진 직선에서 데이터가 얼마나 떨어져 있는지 계산하는 함수
  - 손실 함수의 값이 작을수록 모델이 더 정확하게 학습된 것을 의미





- 손실 함수의 종류
  - 산술 손실 함수
  - 확률 손실 함수
  - 랭킹 손실 함수

산술 손실 함수

- 모델로 산술값을 예측할 때는 각 데이터에 대한 예측 값과 실제 관측 값의 차이를 산술적으로 계산하는 손실 함수를 많이 사용
  - 차이의 제곱을 사용하는 제곱 손실함수
  - 차이의 절대값을 사용하는 손실함수
  - 예) 제곱 손실 함수
    - 주어진 데이터 출력값(y)과 모델의 예측값($\hat{y}$)의 차이의 제곱을 계산
    - $loss(f) = (y - \hat{y})^2$
    - f: 모델, y: 데이터로부터 주어진 출력, $\hat{y}$:모델에 데이터로부터 주어진 입력을 넣어서 계산한 값
    - 선형 모델 f: x -> $\hat{y}$ = w_0 +w_1x 의 경우 제곱 손실 함수는
    - $loss(f) = (y - w_0 + w_1 x)^2$
    - 1 시도) x=1, y =3, w_0 = w_1 = 0, 손실 함수 $(3 - 0 - 0*1)^2 = 9$
    - 2 시도) w_1=2가 더 좋을 것이라 추측, 손실 함수 $(3 - 0 - 2*1)^2 = 1$
    - 2의 손실 함수 값이 훨씬 작은 것을 가진다 (더 작은 에러를 가진다는 것은 인자가 모두 0인 모델보다 더 정확하게 학습된 모델이라는 뜻)

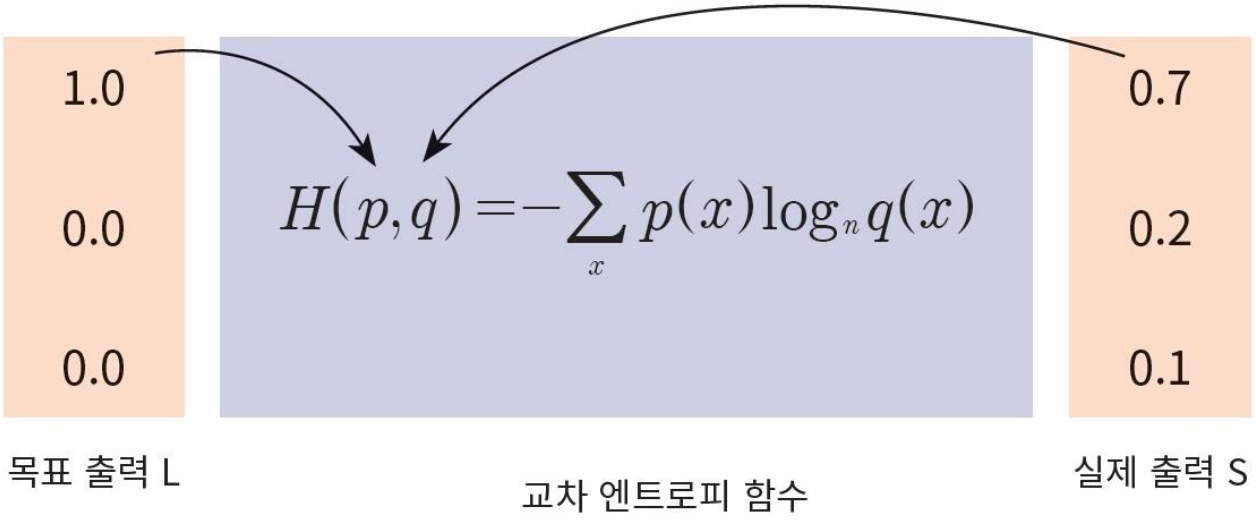
확률 손실함수

- 산술값을 예측하는 데에는 산술 손실함수가 적합하나, 특정 항목을 고를 분류 모델에는 확률 손실 함수가 더 적합
  - 모델이 관측된 데이터를 예측할 확률을 최대화하는 방식으로 계산
  - MLE(maximum likelihood estimation), 쿨백-라이블러 발산(kullback leibler divergence)
  - 예) 교차 엔트로피(cross entropy) : 딥러닝에서 많이 사용됨
    - 교차 엔트로피는 2개의 확률분포 간의 거리를 측정한 것이다.
    - 교차 엔트로피는 2개의 확률 분포 p, q에 대해서 다음과 같이 정의된다.

$$H(p,q) = -\sum_{x} p(x) \log_n q(x)$$

    - 교차 엔트로피가 크면, 2개의 확률 분포가 많이 다른 것이다. 교차 엔트로피가 작으면 2개의 확률 분포가 거의 일치한다고 볼 수 있다.

교차 엔트로피 손실 함수



$$H(p,q) = -\sum_x p(x)\log_n q(x)$$
$$= -(1.0*\log 0.7 + 0.0*\log 0.2 + 0.0*\log 0.1)$$
$$= 0.154901$$

교차 엔트로피의 계산

입력 샘플	실제 출력			목표 출력		
샘플 #1	0.1	0.3	0.6	0	0	1
샘플 #2	0.2	0.6	0.2	0	1	0
샘플 #3	0.3	0.4	0.3	1	0	0

▷ 첫 번째 샘플에 대하여 교차 엔트로피를 계산해보자. - (log(0.1) * 0 + log(0.3) * 0 + log(0.6) * 1) = - (0 + 0 -0.51) = 0.51이 된다.

▷ 두 번째 샘플의 교차 엔트로피는 - (log(0.2) * 0 + log(0.6) * 1 + log(0.2) * 0) = - (0 -0.51 + 0) = 0.51이다.

▷ 세 번째 샘플의 교차 엔트로피는 - (log(0.3) * 1 + log(0.4) * 0 + log(0.3) * 0) = - (-1.2 + 0 + 0) = 1.20이다.

▷ 따라서 3개 샘플의 평균 교차 엔트로피 오류는 (0.51 + 0.51 + 1.20) / 3 = 0.74가 된다.

원-핫 인코딩(One-Hot-Encoding)

- 분류 문제의 출력을 간단하게 표현하는 방법
  - 항목 수가 C개면, C 크기의 벡터(배열)을 만들고 모두 0으로 채운 후 정답 항목만 1로 채움
  - 5가지 항목 중에서 4번째 항목이 정답이라면 y는 다음과 같이 표현

0
0
0
1
0

5가지 항목 중에서 4번째 항목이 정답인 원-핫 인코딩

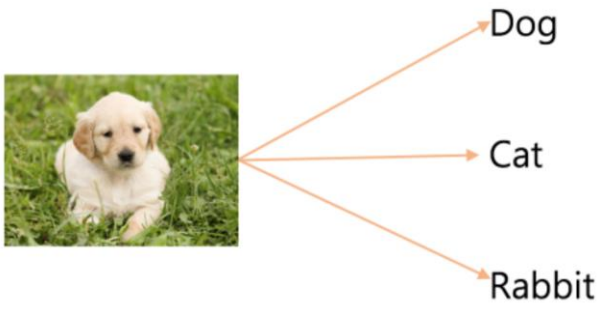
0·p(y=1|x)
0·p(y=2|x)
0·p(y=3|x)
1·p(y=4|x)
0·p(y=5|x)

원-핫 인코딩 각 항목에 각각의 예측 확률을 곱함

  - Y가 주어졌을 때, 입력 x에 대한 손실 함수는 원-핫인코딩된 y의 각 항목에 예측확률($\hat{y}$)를 각각 곱함
  - 전체 항목을 모두 더한 후 –log를 씌우면 결과적으로 –logp(y=4|x)가 나옴

One-Hot 인코딩 손실함수의 예

- 강아지 이미지를 입력 데이터로 넣고
  - 강아지일 확률 0.1, 고양이일 확률 0.8, 토끼일 확률 0.3



  - $loss(f) = -\sum_i [y_i \log \hat{y}_i]$

y =

1(dog)
0(cat)
0(rabbit)

$\hat{y}$ =

0.1
0.8
0.3

  - 손실 함수 계산은 –(1 * ln(0.1) + 0 * ln(0.8) + 0 * ln(0.3))
  - 정답을 잘 맞힐수록 더 작은 값을 가지게 되고, 그 결과 더 작은 손실 함수를 가지게 됨

랭킹 손실함수

- 산술 손실 함수나 확률 손실 함수와 달리 특정한 결과값에 대한 손실이 아님
- 모델이 예측해낸 결과값의 순서가 맞는지만 판별
- 목록에서 몇 가지를 추천하는 추천시스템이나 랭킹 학습 분야에서 사용
  - 예) 데이터의 순서가 $x_1 > x_2 > x_3 > x_4 > x_5$라고 들어오면 각 데이터의 순서를 x의 의미에 따라 학습
  - X는 숫자를 의미한다기보다는 어떤 개념을 나타냄(x_1은 사과, x_2는 바나나 등)

랭킹 손실 함수의 예 [1/2]

- 모든 쌍의 순서가 맞았는지 틀렸는지 확인하는 방식이 가장 간단
  - 모델이 만들어 낸 결과 : $x_3 > x_1 > x_2 > x_4 > x_5$ (정답 : $x_1 > x_2 > x_3 > x_4 > x_5$)
  - 결과값 :

$x_3 > x_1$
$x_3 > x_2$
$x_3 > x_4$
$x_3 > x_5$
$x_1 > x_2$
$x_1 > x_4$
$x_1 > x_5$
$x_2 > x_4$
$x_2 > x_5$
$x_4 > x_5$

  - 10가지 관계 중에서 $x_3 > x_1$, $x_3 > x_2$ 두개의 관계가 잘못되었으므로 손실 함수 결과는 2
  - 페어와이즈 제로-원 손실 함수(pairwise zero-one loss)

랭킹 손실 함수의 예[2/2]

- 편집거리(edit distance) 랭킹 손실 함수
  - 모델이 예측한 순서 목록에서 몇 번의 맞바꿈을 해야 원래 순서로 돌아갈 수 있는지 측정
  - 앞의 예제에서 $x_3>x_1>x_2>x_4>x_5$ 는 두 번의 맞바꿈으로 맞는 순서가 됨

    1. 첫번째 맞바꿈 : $x_1>x_3>x_2>x_4>x_5$
    2. 두번째 맞바꿈 : $x_1>x_2>x_3>x_4>x_5$

  - 편집 거리 손실 함수의 결과 역시 2로 나옴

    (2가지 손실 함수의 결과가 항상 동일하게 나타나는 것은 아님)

최적화

- 순수 수학 최적화와 기계 학습 최적화의 차이
  - 순수 수학의 최적화 예) $f(x_1, x_2) = -\left(\cos\left(x_1{}^2\right) + \sin\left(x_2{}^2\right)\right)^2$의 최저점을 찾아라.
  - 기계 학습의 최적화는 단지 훈련집합이 주어지고, 훈련집합에 따라 정해지는 목적(손실)함수의 최저점을 찾아야 함
    - 데이터로 미분하는 과정 필요 → 오류 역전파 알고리즘
    - 주로 SGD (스토캐스틱 경사 하강법) 사용

- 기계 학습이 해야 할 일을 식으로 정의하면, (Θ는 매개변수, $J(\Theta)$는 목적함수)

$J(\boldsymbol{\Theta})$를 최소로 하는 최적해 $\hat{\boldsymbol{\Theta}}$을 찾아라. 즉, $\hat{\boldsymbol{\Theta}} = \underset{\Theta}{\operatorname{argmin}} J(\boldsymbol{\Theta})$ (2.50)

간단한 최적화 방법

- 최적화 문제 해결
  - 낱낱탐색exhaustive search 알고리즘
    - 차원이 조금만 높아져도 적용 불가능
    - 예) 4차원 Iris에서 각 차원을 1000구간으로 나눈다면 총 1000^4개의 점을 평가해야 함
  - 무작위 탐색 알고리즘
    - 아무 전략이 없는 순진한 알고리즘

알고리즘 2-1 낱낱탐색 알고리즘

입력 : 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$
출력 : 최적해 $\hat{\boldsymbol{\Theta}}$

1 가능한 해를 모두 생성하여 집합 S에 저장한다.
2 min을 충분히 큰 값으로 초기화한다.
3 for (S에 속하는 각 점 $\boldsymbol{\Theta}_{current}$에 대해)
4 　　if($J(\boldsymbol{\Theta}_{current}) < min$) $min = J(\boldsymbol{\Theta}_{current})$, $\boldsymbol{\Theta}_{best} = \boldsymbol{\Theta}_{current}$
5 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}_{best}$

알고리즘 2-2 무작위 탐색 알고리즘

입력 : 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$
출력 : 최적해 $\hat{\boldsymbol{\Theta}}$

1 min을 충분히 큰 값으로 초기화한다.
2 repeat
3 　　무작위로 해를 하나 생성하고 $\boldsymbol{\Theta}_{current}$라 한다.
4 　　if($J(\boldsymbol{\Theta}_{current}) < min$) $min = J(\boldsymbol{\Theta}_{current})$, $\boldsymbol{\Theta}_{best} = \boldsymbol{\Theta}_{current}$
5 until(멈춤 조건)
6 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}_{best}$

경사 하강법

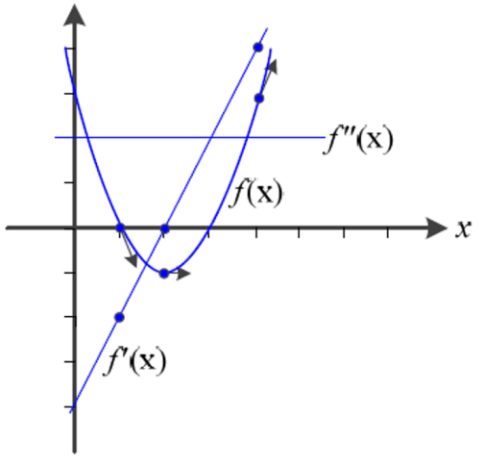
- 기계 학습이 사용하는 전형적인 알고리즘
  - 라인 3에서는 목적 함수가 작아지는 방향을 주로 미분으로 찾아냄

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}, \qquad f''(x) = \lim_{\Delta x \to 0} \frac{f'(x+\Delta x) - f'(x)}{\Delta x} \tag{2.51}$$

  - 1차 도함수 $f'(x)$는 함수의 기울기, 즉 값이 커지는 방향을 지시함
  - 따라서 $-f'(x)$ 방향에 목적함수의 최저점이 존재
  - [알고리즘]에서 $d\boldsymbol{\Theta}$로 $-f'(x)$를 사용함← 경사 하강 알고리즘의 핵심 원리

입력 : 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$
출력 : 최적해 $\hat{\boldsymbol{\Theta}}$

1 난수를 생성하여 초기해 $\boldsymbol{\Theta}$을 설정한다.
2 repeat
3 $J(\boldsymbol{\Theta})$가 작아지는 방향 $d\boldsymbol{\Theta}$를 구한다.
4 $\boldsymbol{\Theta} = \boldsymbol{\Theta} + d\boldsymbol{\Theta}$
5 until(멈춤 조건)
6 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

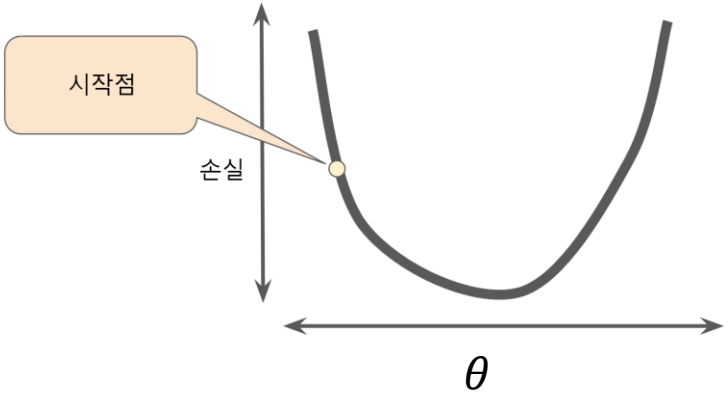


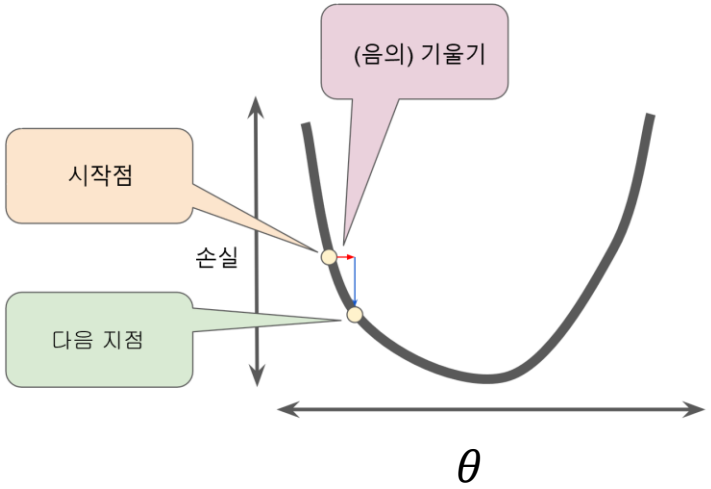
$y = f(x) = x^2 - 4x + 3$

$y' = f'(x) = 2x - 4$

경사 하강법

- 간단한 최적화 방법 중 하나로 임의의 지점에서 시작해서 경사를 따라 내려갈 수 없을 때까지 반복적으로 내려가며 최적화를 수행



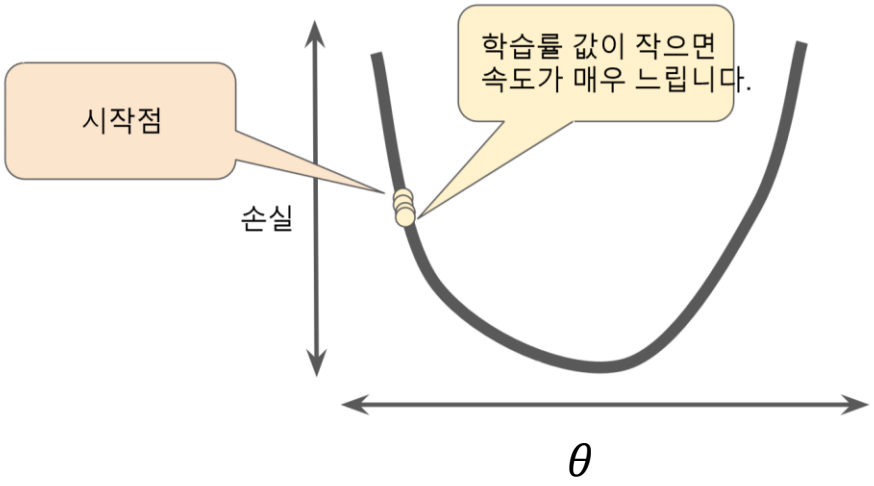


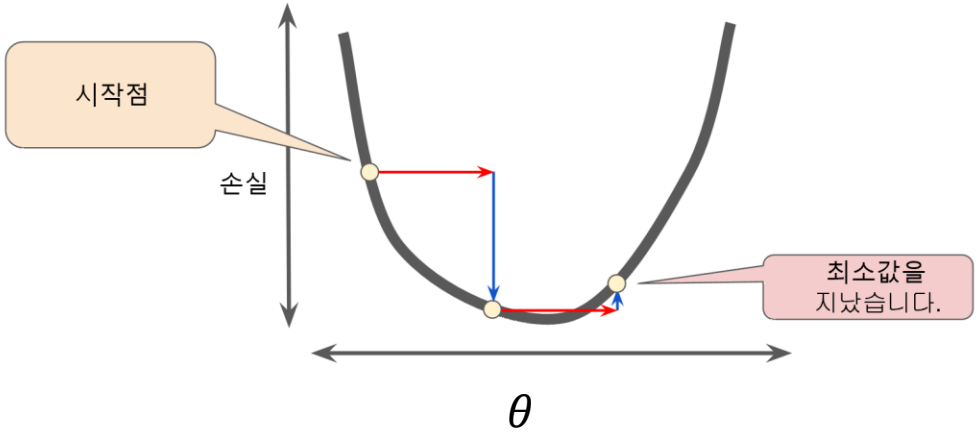
1. 현재 값에서의 경사값(∇)을 구합니다
2. 경사를 따라 η 만큼 내려갑니다 (즉, 경사값에 -1을 곱한 방향으로 η 만큼 움직입니다)

   $\theta_{stept+1} = \theta_{stept} - \eta\nabla$
3. 손실 함수의 출력값이 많이 줄었는지 보고 그렇다면 1로 돌아가 계속 하강합니다. 많이 줄어들지 않았다면 경사를 다 내려온 것이므로 θ 값 구하기를 그만둡니다.

경사하강법

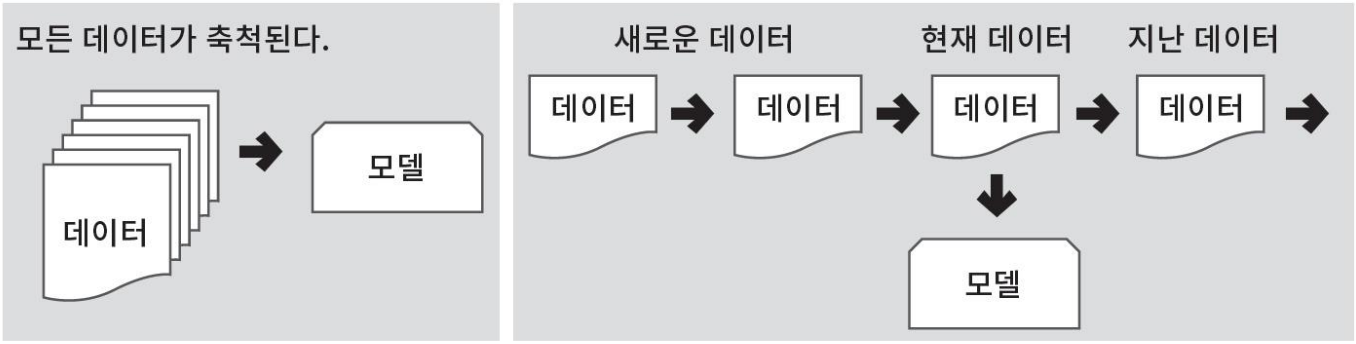
- 경사값은 어떻게 구할 것인가?
  - 함수의 경사값은 그 함수의 미분값
  - 모든 점에서 미분 가능한 제곱 손실 함수를 절댓값 손실함수보다 더 많이 사용
- 얼마나 내려가는지 보여주는 학습률(η)는 어떻게 조절할 것인가?
  - 학습률이 크면 매 스텝마다 θ 를 많이 변화시켜서 빠른 시간에 학습되기는 하지만, 최솟값 근처에서 완전히 수렴하지 못하고 값이 계속 왔다갔다 하는 안정적이지 못한 모습을 보임
  - 학습률이 작다면 수렴에 훨씬 시간이 오래 걸리겠지만, 최솟값 근처에서는 안정적으로 수렴이 가능
  - 실제 구현에서는 여러 가지 학습률을 시도해보고 처음에는 큰 값으로 시작해서 학습이 진행되면서 점점 값을 줄이는 방식으로 사용





확률적 경사하강법과 배치 학습

- 경사하강법은 데이터가 많아지면 계산량이 증가해서 학습시간이 길어짐
- 일부 데이터만 이용해서 손실 함수와 1차 미분값을 근사적으로 계산하는 확률적 경사하강법(SGD)가 만들어짐
  - 하나의 샘플이 주어지면 오차를 계산하여서 바로 가중치를 변경하는 방법(온라인 학습이라고도 함)
- 모든 샘플을 모두 보여준 후에 개별 샘플의 그래디언트를 전부 더해서 이것을 바탕으로 모델의 가중치를 변경하는 배치학습

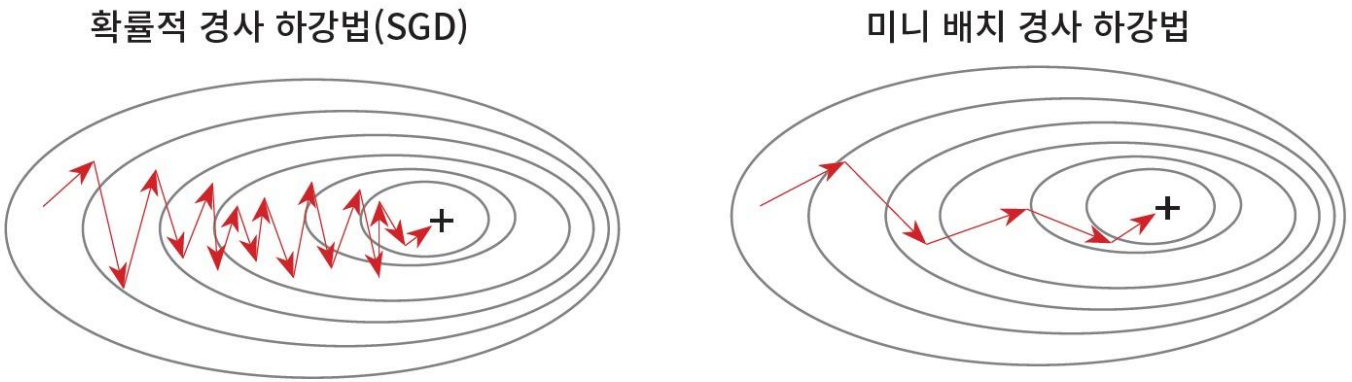


배치학습 온라인 학습

미니 배치

- 온라인 학습과 배치 학습의 중간에 있는 방법이 미니 배치(mini batch)이다.
- 이 방법에서는 훈련 데이터를 작은 배치들로 분리시켜서 하나의 배치가 끝날 때마다 학습을 수행하는 방법이다.

- 1 < size(매니 배치) < size(훈련 데이터)



미니 배치의 특징

- 전체 훈련 데이터를 전부 처리한 후에 모델의 매개 변수를 업데이트하면 모델이 업데이트되는데 너무 오랜 시간이 걸림. 또한 전체 데이터가 너무 커서 메모리에 들어가지 못함 (배치 학습의 단점)
- 하나의 샘플을 처리한 후에 즉시 모델의 매개 변수를 업데이트 하면 잡음에 너무 취약해짐 (확률적 경사하강기법의의 단점)

- 미치 배치
  - 온라인 학습의 장점 : "빠른 모델의 업데이트와 메모리 효율성"
  - 배치 학습의 장점 : "정확한 모델 업데이트와 계산 효율성 "

모델 평가: 실제 활용에서 성능을 평가하는 방법

- 손실 함수가 모델을 최적화하고자 수식으로 표현하는 방법이라면 모델 평가는 모델이 얼마나 좋은 성능을 보일지 평가하는 방법
- 일반화 : 학습 데이터뿐만 아니라 학습 데이터가 아닌 새로운 데이터가 들어왔을 때도 잘 동작하는지 측정
- 일반화가 중요한 이유
  - 학습에 사용되는 관측된 데이터들은 한정된 패턴들만 보여줌
  - 관측된 데이터에 의존해 학습하면 진짜 분포에서 오히려 멀어짐
  - 학습된 데이터에서만 잘 동작하고 관측될 데이터에 대해서는 성능이 잘 나오지 않음
  - 이런 문제를 과학습(오버피팅 overfitting)이라고 함

모델의 일반화 특성 평가

- 모델 일반화 특성 평가
  - 한정적인 데이터를 이용해서 모델의 일반화 특성을 알아내는것을 목표로 함
  - 일반화는 모델이 관측된 데이터(학습 데이터)가 아닌 데이터에 대해서도 좋은 성능을 내는지를 의미
  - 일반화가 얼마나 잘 되었는지 측정하는 에러를 일반화 에러(generalization error)라고함
  - 일반화 에러를 구하는 방법은 다양하지만 여기서는 가장 유명한 학습-평가 데이터 나누기(train-test data split)와 교차 검증(cross-validation)에 대해 알아봄

학습-평가 데이터 나누기

- 학습-평가 데이터 나누기
  - 데이터를 학습용과 평가용으로 나누어 평가하는 방법
  - 80:20이나 50:50의 비율이 가장 흔하게 사용
  - 보통은 무작위로 해당 비율만큼 데이터를 선택하여 학습용 데이터를 만들고, 나머지를 평가용으로 사용
  - 시간에 따라서 달라지는 데이터를 다룰 때는 오래된 데이터를 학습용으로 사용하고 최근 데이터를 평가용으로 사용하는 것이 일반적
  - 데이터를 나누고 나서는 학습용 데이터로 모델을 학습시키고, 평가용 데이터로 성능을 평가
  - 모델이 학습용 데이터를 단순히 외운 것인지, 아니면 학습용 데이터에서 실제로 유용한 패턴을 학습했는지 검증

교차 검증

교차검증

- 학습-평가 데이터 나누기를 한 번만 하는 것이 아니라 여러 번 반복해서 좀 더 정확하게 일반화 에러를 평가하는 방법
- 교차 검증은 학습-평가 데이터 나누기만큼 유명하고 많이 쓰이는 방법
- K겹 교차검증(K-fold cross-validation)
  1. 데이터 셋을 K개로 나눈다
  2. 그 중 첫번째 세트를 제외하고 나머지에 대해 모델을 학습한다. 그리고 첫번째 세트를 이용해서 평가를 수행한다
  3. 과정 2를 마지막 세트까지 진행
  4. 각 세트에 대해 구했던 평가 결과의 평균을 구한다

1번째 세트	2번째 세트	3번째 세트	4번째 세트	5번째 세트
1번째 세트	2번째 세트	3번째 세트	4번째 세트	5번째 세트
1번째 세트	2번째 세트	3번째 세트	4번째 세트	5번째 세트
1번째 세트	2번째 세트	3번째 세트	4번째 세트	5번째 세트
1번째 세트	2번째 세트	3번째 세트	4번째 세트	5번째 세트

정확도

- 정확도(accuracy)
  - 모델이 데이터를 얼마나 정확하게 분류했는지에 대한 평가 지표
  - 상당수 머신 러닝 시스템은 데이터를 활용하여 몇 가지 선택지 가운데에서 가장 적합한것을 고르는데, 이때 선택이 올바른지 측정하여 수치로 나타낸 것이 정확도임
  - 정확도의 정의
    - $$\text{정확도} = \frac{\text{맞게 분류한 데이터 숫자}}{\text{평가하는데 쓰는 총 데이터 숫자}}$$
  - 예) 정확도가 0.5라면 전체 데이터 중 반은 정답을 맞힌 것이고 반은 틀린 것

    0.9라면 전체 데이터 중에 90%는 정답을 맞힌것

정밀도와 포괄성

- 정확도가 상대적으로 덜 중요한 경우
  - 데이터의 항목이 한쪽으로 치우쳐져 있을 경우
    - 사막에서 다음날 비가 올지 안 올지 예측하는 모델의 경우 사막에는 거의 비가 오지 않으므로 모델이 잘 동작하지 않더라도 비가 오지 않을 거라는 사실을 쉽게 맞출 수 있음
    - 실제도 비가 올 확률이 0.1%라면 항상 '그냥 비가 오지 않는다 ' 는 결과를 돌려주는 모델이더라도 정확도가 0.999가 됨
  - 한 분류가 다른 분류보다 중요한 경우
    - 혈액을 분석하여 암에 결렸는지 판별하는 경우 암에 걸렸는데 그렇지 않다고 잘못 판별하면 자칫 환자의 생명이 위험할 수 있다. 반면 암에 걸리지 않았는데도 걸렸다고 예측할 경우에는 추가 검사를 하는 것이 일반적이므로 부정확 하더라도 암환자를 모두 발견해내는 것이 중요
  - 정확도만으로는 모델이 얼마나 유용한지 판단이 어려움
  - 모델의 유용성을 따질 때는 정밀도(precision)과 포괄성(recall)도 함께 고려

정밀도와 포괄성

- 긍정(positive, 양성)이냐 부정(negative, 음성)이냐를 결정하는 이진화 분류 문제

데이터의 실제값	양성으로 예측	음성으로 예측
실제로 양성	참 양성(진짜 양성)	거짓 음성
실제로 음성	거짓 양성	참 음성(진짜 음성)

- 정밀도 : $\text{정밀도} = \frac{\text{참 양성}}{\text{참 양성} + \text{거짓 양성}}$
  - 1년 중 10일 비를 예측하였는데, 비가 오기로 예측한 날 중에서 실제로 며칠인지 확인
  - 10일 중 8일 비가 왔다면 정밀도는 0.8
- 포괄성 : $\text{포괄성} = \frac{\text{참 양성}}{\text{참 양성} + \text{거짓 음성}}$
  - 암 환자가 10명인데 여기서 암환자를 몇 명이나 찾아냈는지
  - 단순히 포괄성만 높으면 좋지 않음(모든 암 환자를 찾을 수 있기 때문)
- F_1: 정밀도와 포괄성 둘 다 높아야 점수가 높아짐
  - $F_1 = \frac{2 \times \text{정밀도} \times \text{포괄성}}{\text{정밀도} + \text{포괄성}}$

랭킹 평가

- 정밀도@K
  - 항목들의 랭킹을 구한 후 앞에서부터 K번째까지의 결과 중에서 몇 개가 올바른지 검사
  - 예) 검색 엔진 결과
    - 한 페이지에 검색 결과 20개를 보여주는 엔진에서 검색 결과의 품질을 평가하기 위해 모든 결과가 아니라 1페이지의 결과 20개만 확인할 경우엔 정밀도@20dml 평가
    - 사용자에게 노출이 덜 되는 랭킹이 낮은 항목에 대해서는 신경쓰지 않음
- NDCG
  - 정밀도@K 와 비슷하게 랭킹이 높은 결과에 가중치를 줌
  - 정밀도@K 와는 달리 NDCG(Normalized Discounted Cumulative Gain)는 중요도를 순위에 따라 바꾸어가며 평가
  - 예)n개의 검색 결과의 등급을 관련성에 따라 3~0사이의 값을 가진다고 하면 이상적인 합과 현재의 합의 비교가 가능
    - 검색엔진 순위가 3,3,2,0,1,2 라면 이상적인 순위는 3,3,2,2,1