한빛아카데미
Hanbit Academy, Inc.

MACHINE 기계 학습 LEARNING

오일석 지음

2장. 기계 학습과 수학

PREVIEW

기계 학습에서 수학의 역할

- 수학은 목적함수를 정의하고, 목적함수가 최저가 되는 점을 찾아주는 최적화 이론 제공
- 최적화 이론에 규제, 모멘텀, 학습률, 멈춤조건과 같은 제어를 추가하여 알고리즘 구축
- 사람은 알고리즘을 설계하고 데이터를 수집함

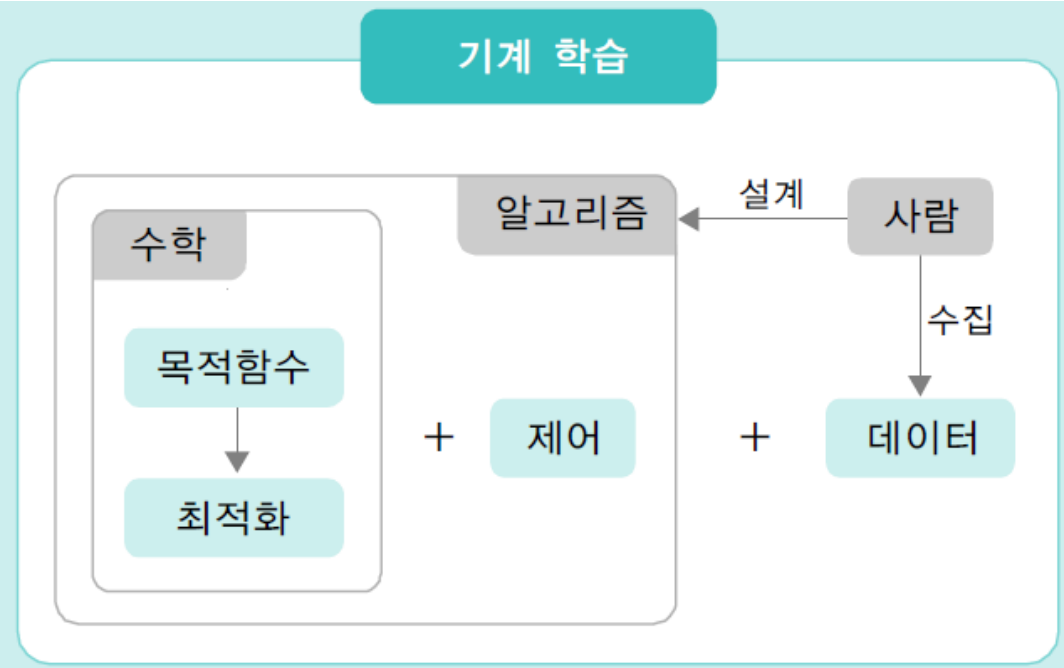


그림 2-1 기계 학습의 개념도

각 절에서 다루는 내용

- 2.1절: 선형대수를 다룬다.
- 2.2절: 확률과 통계를 다룬다.
- 2.3절: 최적화 이론을 다룬다.

- 선형대수: 이 분야의 개념을 이용하면 학습 모델의 매개변수집합, 데이터, 선형연산의 결합 등을 행렬 또는 텐서로 간결하게 표현할 수 있다. 데이터를 분석하여 유용한 정보를 알아내거나 특징 공간을 변환하는 등의 과업을 수행하는 데 핵심 역할을 한다.
- 확률과 통계: 데이터에 포함된 불확실성을 표현하고 처리하는 데 활용한다. 베이즈 이론과 최대 우도 기법을 이용하여 확률 추론을 수행한다.
- 최적화: 목적함수를 최소화하는 최적해를 찾는 데 활용하며, 주로 미분을 활용한 방법을 사용한다. 수학자들이 개발한 최적화 방법을 기계 학습이라는 도메인에 어떻게 효율적으로 적용할지가 주요 관심사이다.

2.2.6 정보이론

- 메시지가 지닌 정보를 수량화할 수 있나?
  - "고비 사막에 눈이 왔다"와 "대관령에 눈이 왔다"라는 두 메시지 중 어느 것이 더 많은 정보를 가지나?
  - 정보이론의 기본 원리 → 확률이 작을수록 많은 정보
- 자기 정보self information 확률변수 X의 정의역을 $\{e_1, e_2, \ldots, e_k\}$ 라고 하자.
  - 사건(메시지) e_i의 정보량 (단위: 비트(밑이 2인 로그함수 사용시) 또는 나츠(밑이 e인 로그함수))
  - 확률이 1/2일 때 1비트의 정보량을 가진다.

$$h(e_i) = -\log_2 P(e_i) \quad \text{또는} \quad h(e_i) = -\log_e P(e_i) \qquad (2.44)$$

확률변수 X의 자기정보량: $I(X) = -\log(P(X))$

- 엔트로피(entropy)
  - 자기정보가 특정사건 e_i의 자기정보를 나타내는 대신 엔트로피는 확률변수 X의 무질서 정도 또는 불확실성을 측정함. $E[h(e)]$ 또는 $E[I(X)]$

이산 확률분포 $$H(x) = -\sum_{i=1,k} P(e_i)\log_2 P(e_i) \quad \text{또는} \quad H(x) = -\sum_{i=1,k} P(e_i)\log_e P(e_i) \qquad (2.45)$$

연속 확률분포 $$H(x) = -\int_{\mathbb{R}} P(x)\log_2 P(x) \quad \text{또는} \quad H(x) = -\int_{\mathbb{R}} P(x)\log_e P(x) \qquad (2.46)$$

2.2.6 정보이론

“Deep learning - Information theory & Maximum likelihood.”

Jan 5, 2017

Information theory

Information theory quantifies the amount of information present. In information theory, the amount of information is characterized as:

- Predictability:
  - Guaranteed events have zero information.
  - Likely events have little information. (Biased dice have little information.)
  - Random events process more information. (Random dice have more information.)
- Independent events add information. Rolling a dice twice with heads have twice the information of rolling the dice once with a head.

> *In information theory, chaos processes more information.*

Information of an event is defined as:

$$I(x) = -\log(P(x))$$

Source: https://jhui.github.io/2017/01/05/Deep-learning-Information-theory/

2.2.6 정보이론

Entropy

In information theory, entropy measures the amount of information.

We define entropy as:

$$H(x) = E_{x \sim P}[I(x)]$$

So

$$H(x) = -\mathbb{E}_{x \sim P}[logP(x)]$$
$$H(x) = -\sum_{x} P(x) \log P(x)$$

If $\log$ has a base of 2, it measure the number of bits to encode the information. In information theory, information and random-ness are positively correlated. High entropy equals high randomness and requires more bits to encode it.

Source: https://jhui.github.io/2017/01/05/Deep-learning-Information-theory/

2.2.6 정보이론

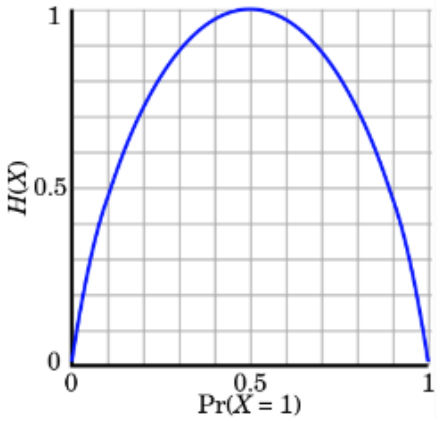
Example

Let's comput the entropy of a coin. For a fair coin:

$$H(X) = -p(head) \cdot \log_2(p(head)) - p(tail) \cdot log_2(p(tail)) = -\log_2 \frac{1}{2} = 1$$

Therefore we can use 1 bit to represent head (0 = head) and 1 bit to represent tail (1 = tail).

The entropy of a coin peaks when $p(head) = p(tail) = 0.5$.



(Source Wikipedia)

For a fair die, $H(X) = \log_2 6 \approx 2.59$. A fair die has more entropy than a fair coin because it is less predictable.

Source: https://jhui.github.io/2017/01/05/Deep-learning-Information-theory/

2.2.6 정보이론

자기 정보와 엔트로피 예제

예제 2-8

윷을 나타내는 확률변수를 x라 할 때 x의 엔트로피는 다음과 같다.

$$H(x) = -\left(\frac{4}{16}\log_2\frac{4}{16} + \frac{6}{16}\log_2\frac{6}{16} + \frac{4}{16}\log_2\frac{4}{16} + \frac{1}{16}\log_2\frac{1}{16} + \frac{1}{16}\log_2\frac{1}{16}\right) = 2.0306\text{비트}$$

주사위는 눈이 6개인데 모두 1/6이라는 균일한 확률을 가진다. 이 경우 엔트로피를 계산하면 다음과 같다.

$$H(x) = -\left(\frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6}\right) = 2.585 \text{ 비트}$$

- 주사위가 윷보다 엔트로피가 높은 이유는?

주사위는 모든 사건이 동일한 확률을 가진다. 즉, 어떤 사건이 일어날지 윷보다 예측하기 어렵다. 다시 말해서 주사위가 윷보다 더 무질서하고 불확실성이 더 크다. 따라서 엔트로피가 더 높다(분산과 비슷). 사실 주사위처럼 모든 사건이 동일한 확률을 가질 때 엔트로피가 최고이다.

2.2.6 정보이론

- 교차 엔트로피cross entropy (두 확률분포간의 불확실성 정도)

  (가정: 확률분포 P는 진짜값들의 분포, Q는 모형으로부터 예측값들의 확률분포)

  - 두 확률분포 P와 Q 사이의 교차 엔트로피 (앞의 식(2.45)는 한 확률분포의 무질서 정도를 측정하였음)

$$H(P,Q) = -\sum_{x} P(x)\log_2 Q(x) = -\sum_{i=1,k} P(e_i)\log_2 Q(e_i) \qquad (2.47)$$

  - 식을 전개하면,

$$\begin{aligned} H(P,Q) &= -\sum_{x} P(x)\log_2 Q(x) \\ &= -\sum_{x} P(x)\log_2 P(x) + \sum_{x} P(x)\log_2 P(x) - \sum_{x} P(x)\log_2 Q(x) \\ &= H(P) + \sum_{x} P(x)\log_2 \frac{P(x)}{Q(x)} \end{aligned}$$

KL 다이버전스(KL divergence)

2.2.6 정보이론

- KL 다이버전스
  - 식 (2.48)은 P와 Q 사이의 KL 다이버전스
  - 두 확률분포 사이의 거리를 계산할 때 주로 사용

$$KL(P \parallel Q) = \sum_x P(x) \log_2 \frac{P(x)}{Q(x)} \qquad (2.48)$$

- 교차 엔트로피와 KL 다이버전스의 관계

$$P와\ Q의\ 교차\ 엔트로피\ H(P,Q) = H(P) + \sum_x P(x) \log_2 \frac{P(x)}{Q(x)} \qquad (2.49)$$
$$= P의\ 엔트로피 + P와\ Q\ 간의\ KL\ 다이버전스$$

- KL 다이버전스는 두 확률분포가 얼마나 다른지 측정한다. P와 Q가 같을 때 0이 된다. 따라서 거리 개념을 내포하고 있기는 하지만, $KL(P||Q) \neq KL(Q||P)$ 이므로 엄밀한 의미에서 거리는 아니다.
- 딥러닝에서는 목적함수를 평균제곱오차(MSE)대신에 교차엔트로피를 주로 사용한다(5.1절).

2.2.6 정보이론

Cross entropy

If entropy measures the minimum number of bits to encode information, cross entropy measures the minimum of bits to encode x with distribution P using the wrong optimized encoding scheme from Q.

Cross entropy is defined as:

$$H(P,Q) = -\sum_{x} P(x) \log Q(x)$$

> *In deep learning, P is the distribution of the true labels, and Q is the probability distribution of the predictions from the deep network.*

Source: https://jhui.github.io/2017/01/05/Deep-learning-Information-theory/

2.2.6 정보이론

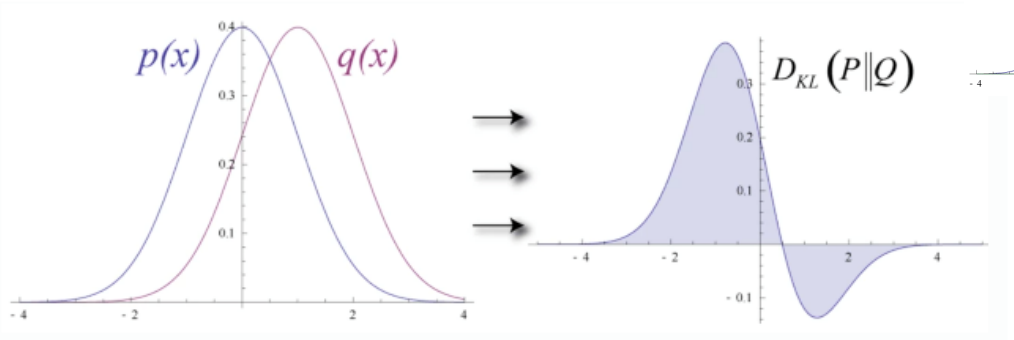
KL Divergence

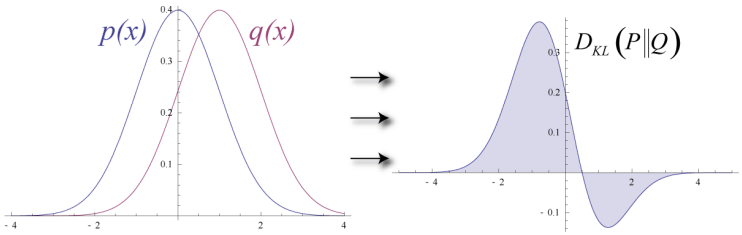
In deep learning, we want a model predicting data distribution Q resemble the distribution P from the data. Such difference between 2 probability distributions can be measured by KL Divergence which is defined as:

$$D_{KL}(P||Q) = \mathbb{E}_x \log \frac{P(x)}{Q(x)}$$

So,

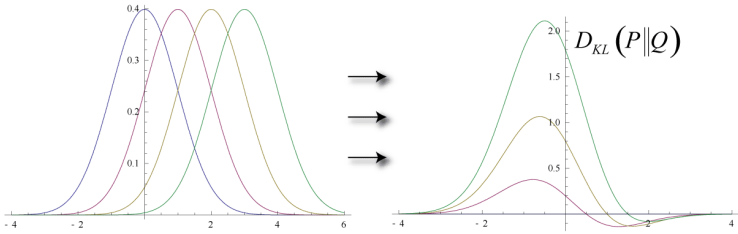
$$D_{KL}(P||Q) = \sum_{x=1}^{N} P(x) \log \frac{P(x)}{Q(x)}$$
$$= \sum_{x=1}^{N} P(x)[\log P(x) - \log Q(x)]$$





Original Gaussian PDF's

KL Area to be Integrated



(Source Wikipedia.)

KL-divergence is not commutative: $D_{KL}(P||Q) \neq D_{KL}(Q||P)$.

Source: https://jhui.github.io/2017/01/05/Deep-learning-Information-theory/

2.2.6 정보이론

Recall:

$$
\begin{aligned}
H(P) &= -\sum P \log P, \\
H(P,Q) &= -\sum P \log Q, \quad \text{and} \\
D_{KL}(P||Q) &= \sum P \log \frac{P}{Q}.
\end{aligned}
$$

We can rewrite the cross entropy equation with KL divergence:

$$
\begin{aligned}
H(P,Q) &= -\sum P \log Q \\
&= -\sum P \log P + \sum P \log P - \sum P \log Q \\
&= H(P) + \sum P \log \frac{P}{Q} \\
H(P,Q) &= H(P) + D_{KL}(P||Q)
\end{aligned}
$$

So cross entropy is the sum of entropy and KL-divergence. Cross entropy $H(P,Q)$ is larger than $H(P)$ since we require extra amount of information (bits) to encode data with less optimized scheme from Q if $P \neq Q$. Hence, KL-divergence is always positive for $P \neq Q$ or zero otherwise.

K-L divergence is always positive 인 이유:
"K-L divergence Wikipedia" 와 Gibb's inequality 또는 Jensen's Inequality 이용하여 증명 가능. (Wikipedia 참조)

$H(P)$ only depends on P: the probability distribution of the data. Since it is un-related with the model θ we build, we can treat $H(P)$ as a constant. Therefore, minimizing the cross entropy is equivalent to minimize the KL-divergence.

$$
\begin{aligned}
H(P,Q_\theta) &= H(P) + D_{KL}(P||Q_\theta) \\
\nabla_\theta H(P,Q_\theta) &= \nabla_\theta (H(P) + D_{KL}(P||Q_\theta)) \\
&= \nabla_\theta D_{KL}(P||Q_\theta)
\end{aligned}
$$

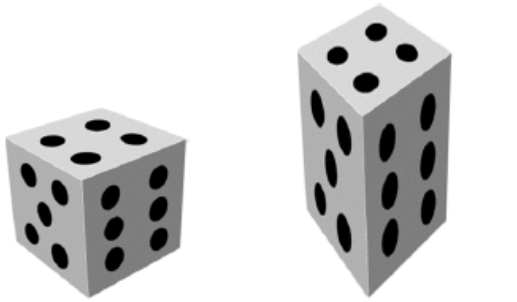
Source: https://jhui.github.io/2017/01/05/Deep-learning-Information-theory/

2.2.6 정보이론

예제 2-9

[그림 2-21]과 같이 정상적인 주사위와 찌그러진 주사위가 있는데, 정상적인 주사위의 확률분포는 P, 찌그러진 주사위의 확률분포는 Q를 따르며, P와 Q가 다음과 같이 분포한다고 가정하자.

$$P(1)=\frac{1}{6},\ P(2)=\frac{1}{6},\ P(3)=\frac{1}{6},\ P(4)=\frac{1}{6},\ P(5)=\frac{1}{6},\ P(6)=\frac{1}{6}$$
$$Q(1)=\frac{3}{12},\ Q(2)=\frac{1}{12},\ Q(3)=\frac{1}{12},\ Q(4)=\frac{1}{12},\ Q(5)=\frac{3}{12},\ Q(6)=\frac{3}{12}$$



(a) 정상 주사위 (b) 찌그러진 주사위

그림 2-21 확률분포가 다른 두 주사위

확률분포 P와 Q 사이의 교차 엔트로피와 KL 다이버전스는 다음과 같다.

$$H(P,Q)=-\left(\frac{1}{6}\log_2\frac{3}{12}+\frac{1}{6}\log_2\frac{1}{12}+\frac{1}{6}\log_2\frac{1}{12}+\frac{1}{6}\log_2\frac{1}{12}+\frac{1}{6}\log_2\frac{3}{12}+\frac{1}{6}\log_2\frac{3}{12}\right)=2.7925$$
$$KL(P\parallel Q)=\frac{1}{6}\log_2\frac{2}{3}+\frac{1}{6}\log_2 2+\frac{1}{6}\log_2 2+\frac{1}{6}\log_2\frac{2}{3}+\frac{1}{6}\log_2\frac{2}{3}=0.2075$$

[예제 2-8]에서 P의 엔트로피 $H(P)$는 2.585이었다. 따라서 식 (2.49)가 성립함을 알 수 있다.

2.3 최적화

- 2.3.1 매개변수 공간의 탐색
- 2.3.2 미분
- 2.3.3 경사 하강 알고리즘

- 순수 수학 최적화와 기계 학습 최적화의 차이
  - 순수 수학의 최적화 예) $f(x_1, x_2) = -(\cos(x_1^2) + \sin(x_2^2))^2$의 최저점을 찾아라. 주로 미분을 사용하며, 미분가능한 함수의 최저점을 찾는 문제.
  - 기계 학습의 최적화는 단지 훈련집합이 주어지고, 훈련집합에 따라 정해지는 목적함수의 최저점을 찾아야 함
    - 잡음이 섞여있고, 변화가 심한 열악한 데이터를 잘 이용하여 미분하는 과정 필요 → 오류 역전파 알고리즘 (3.4절)
    - 데이터로 미분을 계산하면서 최저점을 찾아가는 주로 SGD(스토캐스틱 경사 하강법) 사용

2.3.1 매개변수 공간의 탐색

- 학습 모델의 매개변수 공간
  - 높은 차원에 비해 훈련집합의 크기가 작아 참인 확률분포를 구하는 일은 불가능함
  - 따라서 기계 학습은 적절한 모델을 선택하고, 목적함수를 정의하고, 모델의 매개변수 공간을 탐색하여 목적함수가 최저가 되는 최적점을 찾는 전략 사용 → 특징 공간에서 해야 하는 일을 모델의 매개변수 공간에서 하는 일로 대치한 셈
  - [그림 2-22]는 여러 예제 (Θ는 매개변수, $J(\Theta)$는 목적함수)

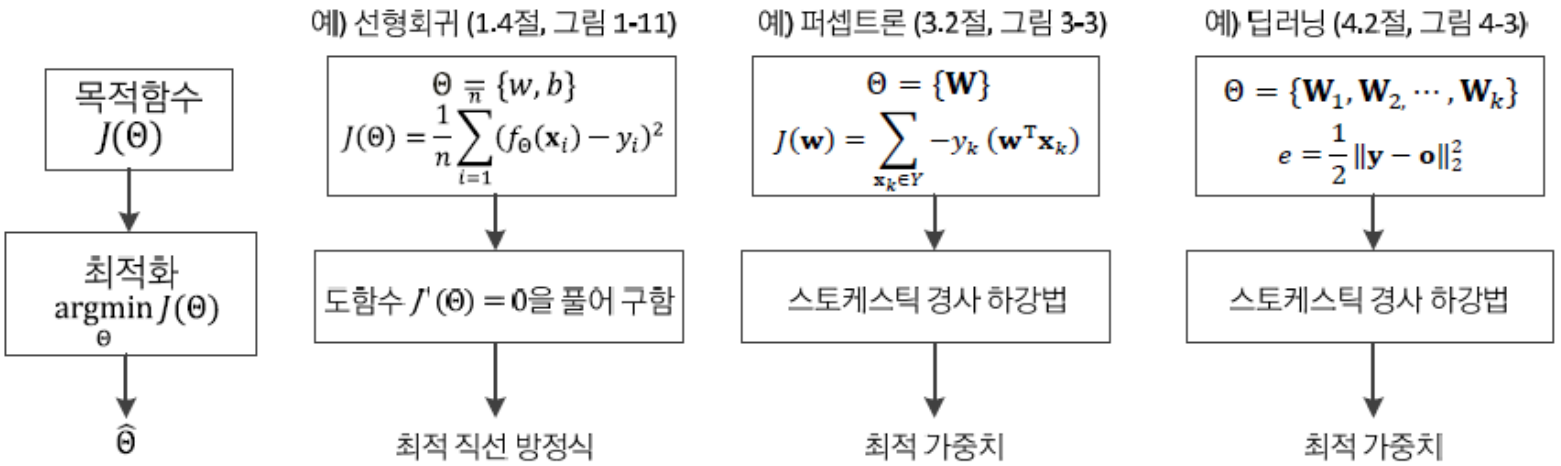


그림 2-22 최적화를 이용한 기계 학습의 문제풀이 과정

2.3.1 매개변수 공간의 탐색

- 학습 모델의 매개변수 공간
  - 특징 공간보다 수 배~수만 배 넓음
    - [그림 2-22]의 선형회귀에서는 특징 공간은 1차원 , 매개변수 공간은 2차원
    - MNIST 인식하는 딥러닝 모델은 784차원 특징 공간, 수십만~수백만 차원의 매개변수 공간
  - [그림 2-23] 개념도의 매개변수 공간: $\hat{x}$은 전역 최적해, x_2와 x_4는 지역 최적해
  - x_2와 같이 전역 최적해(global optimal solution)에 가까운 지역 최적해(local optimal solution)를 찾고 만족하는 경우 많음

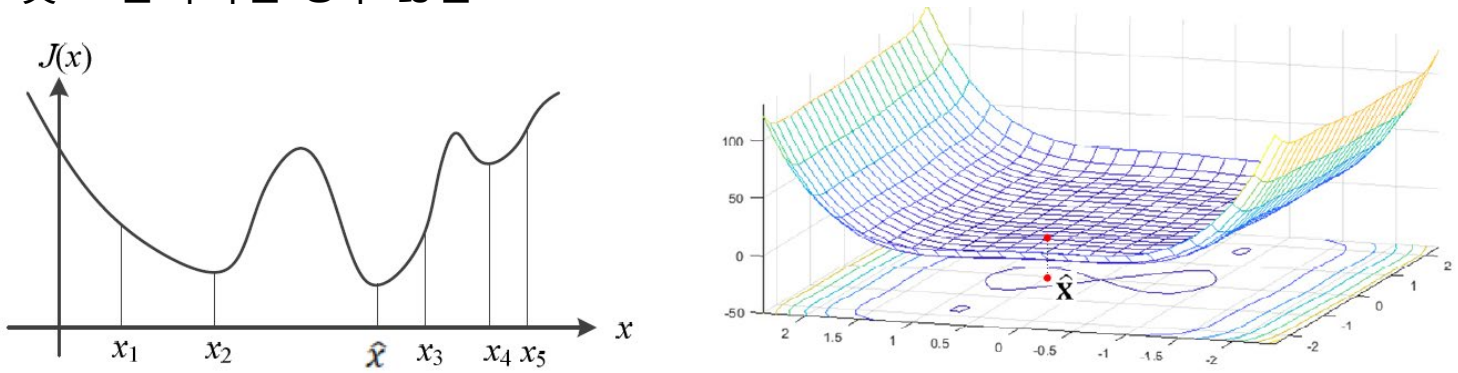


그림 2-23 최적해 탐색

- 기계 학습이 해야 할 일을 식으로 정의하면,

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

2.3.1 매개변수 공간의 탐색

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 random search 알고리즘
  - 멈춤조건(예를들면 미리 정한 시간이 다하며 끝냄)까지 실행.
  - 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 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}_{best}$

2.3.1 매개변수 공간의 탐색

- [알고리즘 2-3]은 기계 학습이 사용하는 전형적인 알고리즘
  - 라인 3에서는 목적함수가 작아지는 방향을 주로 미분으로 찾아냄
  - 미분은 함수의 기울기를 알려주는데, 기울기는 함수값이 가장 급격하게 작아지는 방향으로 $d\boldsymbol{\Theta}$를 사용할 수 있다.

알고리즘 2-3 기계 학습이 사용하는 전형적인 탐색 알고리즘(1장의 [알고리즘 1-1]과 같음)

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

```
1  난수를 생성하여 초기해 Θ을 설정한다.
2  repeat
3      J(Θ)가 작아지는 방향 dΘ를 구한다.
4     Θ = Θ + dΘ
5  until(멈춤 조건)
6  Θ̂ = Θ
```

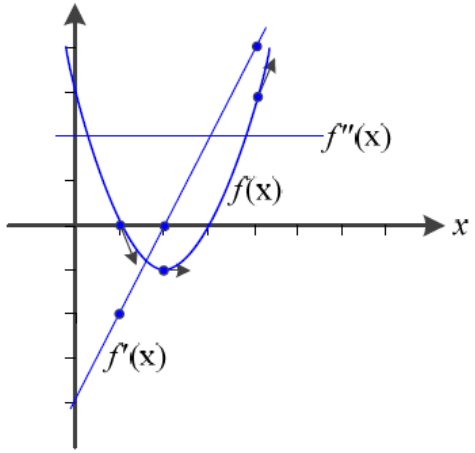
2.3.2 미분

미분에 의한 최적화

- 미분의 정의

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

- 1차 도함수 $f'(x)$는 함수의 기울기, 즉 x가 미세하게 증가하였을 때 함수값의 변화율을 알려줌. 또는 값이 커지는 방향을 지시함
- 따라서 $-f'(x)$ 방향에 목적함수의 최저점이 존재
- [알고리즘 2-3]에서 $d\boldsymbol{\Theta}$로 $-f'(x)$를 사용함← 경사 하강 알고리즘의 핵심 원리



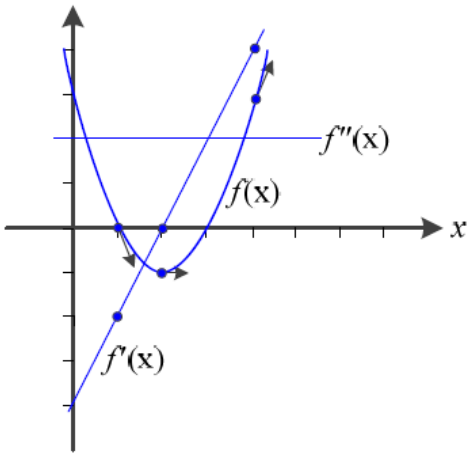
$y = f(x) = x^2 - 4x + 3$

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

그림 2-24 간단한 미분 예제

2.3.2 미분

- 미분에 의한 최적화



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

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

그림 2-24 간단한 미분 예제

- 몇 개의 점을 살펴보면서 최저점을 찾아가는 방법을 구상하자.
- $x = 4$에서의 1차 도함수 값은 $f'(4) = 2 * 4 - 4 = 4$ 인데, 최저점은 왼쪽에 있다.
- $x = 1$에서의 1차 도함수 값은 $f'(1) = 2 * 1 - 4 = -2$ 인데, 최저점은 오른쪽에 있다.
- 도함수 값이 +이면 –방향으로 가야 최저점을 만나고, -이면 +방향으로 가야 최저점을 찾을 수 있다.
- 따라서 $-f'(x)$ 방향에 목적함수의 최저점이 존재한다.
- [알고리즘 2-3]에서 라인3에서 $-f'(x)$ 가 바로 $d\boldsymbol{\theta}$ 이다.
- $f(x)$의 입장에서가 아니라 x축의 입장에서 $f(x)$가 최소가 되는 x축의 점 x를 찾기 위해서 x가 어느방향으로 움직이는가에 촛점을 두어야 함.

2.3.2 미분

- 수학에서는 $f'(x) = 0$ 을 풀면 최저점을 구할 수 있는데, $f'(x) = 2x - 4 = 0 \Rightarrow x = 2$ 기울기가 0인 지점. => 분석적 방법
- [알고리즘 2-3]과 같이 $d\boldsymbol{\theta}$ 만큼 조금씩 이동을 반복하면서 최저점 찾는 방법=> 수치적 방법
- 분석적 방법은 차원이 낮고 목적함수가 단순한 경우 적용가능하나, 기계학습은 이런 조건을 벗어나므로 기계학습 알고리즘은 수치적 방법을 사용한다.

2.3.2 미분

편미분(partial differentiation)

- 변수가 여러 개인 함수의 미분. 변수 각각에 대해 독립적으로 미분: 편미분
- 편미분의 결과는 벡터 형태가 되는데, 이 미분값이 이루는 벡터를 그레이디언트(gradient)라 부름
- 여러 가지 표기: $\nabla f, \frac{\partial f}{\partial \mathbf{x}}, \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)^{\mathrm{T}}$
- 예) 변수가 2개 이므로 1차 도함수도 2개의 변수로 된 2차원 벡터이다. 이러한 미분을 편미분이라 함.

$$\left.\begin{aligned} f(\mathbf{x}) = f(x_1, x_2) &= \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \\ \nabla f = f'(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}} &= \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)^{\mathrm{T}} = (2x_1^5 - 8.4x_1^3 + 8x_1 + x_2, 16x_2^3 - 8x_2 + x_1)^{\mathrm{T}} \end{aligned}\right\} \quad (2.52)$$

- 식(2.52)에서 $\frac{\partial f}{\partial x_1}$=0 과 $\frac{\partial f}{\partial x_2} = 0$ 을 풀면, (-0.08980, 0.7216)$^{\mathrm{T}}$ 와 (0.0898, -0.7126)$^{\mathrm{T}}$ 등의 복수 해를 얻음=> 분석적 방법
- [알고리즘 2-3]은 그레디언트 $\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)^{\mathrm{T}}$ 을 구하여, 벡터 덧셈으로 다음 점을 구한다. 다음 페이지 [예제 2-10] .

기계 학습에서 편미분

- 매개변수 집합 Θ에 많은 변수가 있으므로 편미분을 많이 사용

2.3.2 미분

- 편미분으로 얻은 그레이디언트에 따라 최저점을 찾아가는 예제

예제 2-10

초기점 $\mathbf{x}_0$=(-0.5,0.5)^{T}라고 하자. $\mathbf{x}_0$에서의 그레이디언트는 $f'(\mathbf{x}_0) = (-2.5125, -2.5)^T$ 즉, $\nabla f|_{\mathbf{x}_0} = (-2.5125, -2.5)^T$이다. [그림 2-25]는 $\mathbf{x}_0$에서 그레이디언트를 화살표로 표시하고 있어, $-f'(\mathbf{x}_0)$은 최저점의 방향을 제대로 가리키는 것을 확인할 수 있다. 하지만 얼마만큼 이동하여 다음 점 $\mathbf{x}_1$로 옮겨갈지에 대한 방안은 아직 없다. 2.3.3절에서 공부하는 경사 하강법은 이에 대한 답을 제공한다.

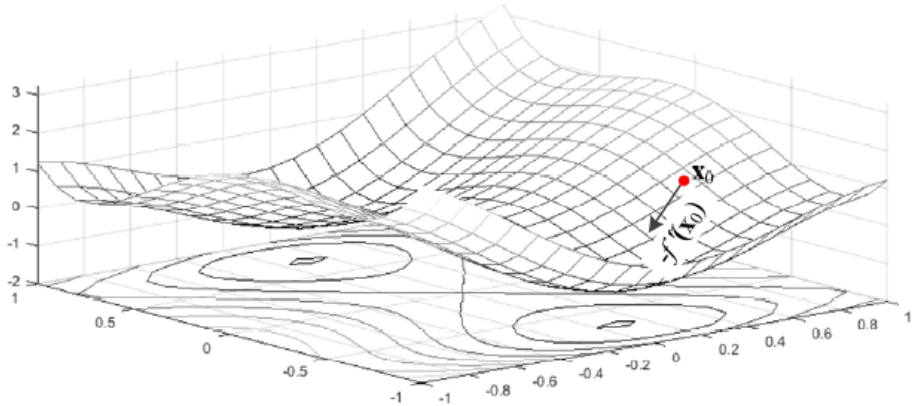


그림 2-25 그레이디언트는 최저점으로 가는 방향을 알려 줌

2.3.2 미분

독립변수와 종속변수의 구분

- 식 (1.2)에서 x는 독립변수, y는 종속변수

$$y = wx + b \qquad (1.2)$$

- 기계 학습에서 이런 해석은 무의미 (왜냐하면 이러한 수식적용은 예측 단계에 해당하며, 예측단계라는 말은 최적화 문제를 이미 다 풀어 기계학습이 끝났다는 말이다. 즉,예측 단계를 위한 해석에 불과)
- 최적화는 예측 단계가 아니라 학습 단계에 필요
  - 식 (1.8)에서 Θ가 독립변수이고 $e = J(\Theta)$라 하면 e가 종속변수임

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n}(f_{\Theta}(\mathbf{x}_i) - y_i)^2 \qquad (1.8)$$

  - [그림 2-22]는 여러 가지 사례를 보여줌

2.3.2 미분

- 연쇄법칙(chain rule)
  - 합성함수 $f(x) = g(h(x))$ 또는 $f(x) = g(h(i(x)))$ 의 미분

$$\left.\begin{aligned} f'(x) &= g'\big(h(x)\big)h'(x) \\ f'(x) &= g'\big(h(i(x))\big)h'(i(x))i'(x) \end{aligned}\right\} \quad (2.53)$$

  - 예) $f(x) = 3(2x^2-1)^2 - 2(2x^2-1) + 5$ 일 때 $h(x) = 2x^2 - 1$ 로 두면,

$$f'(x) = \underbrace{(3 * 2(2x^2-1) - 2)}_{g'(h(x))}\underbrace{(2 * 2x)}_{h'(x)} = 48x^3 - 32x$$

  - 연쇄법칙 사용 안한다면 f(x)를 완전히 전개한 후에 미분해야 하는 번거로움이 있음. 고차원 방정식이라면 훨씬 더 번거로움=> 연쇄법칙 사용

- 다층 퍼셉트론은 합성함수
  - $\frac{\partial o_i}{\partial u_{23}^1}$를 계산할 때 연쇄법칙 적용
  - 3.4절(오류 역전파)에서 설명

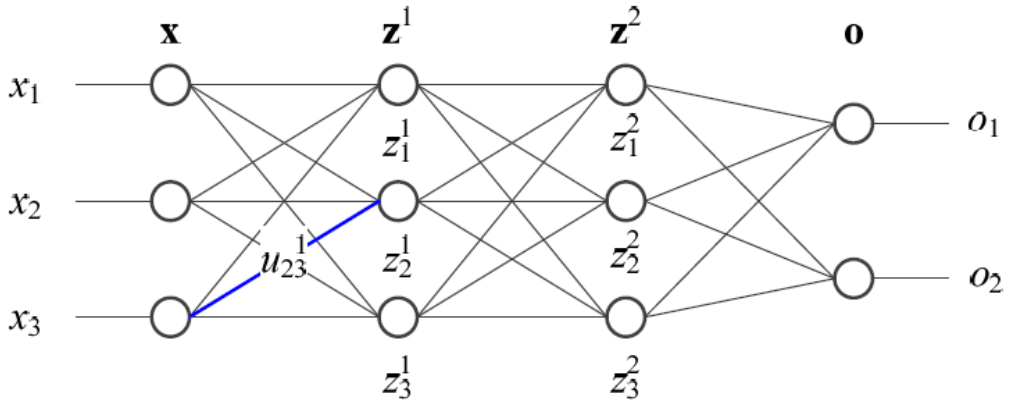


그림 2-26 다층 퍼셉트론은 합성함수

2.3.2 미분

- 야코비언 행렬(Jacobian matrix)
  - d차원 벡터 $\mathbf{x}$를 입력받아 m 차원벡터를 출력하는 함수를 $\mathbf{f}: \mathbb{R}^d \mapsto \mathbb{R}^m$ 라 표현하고, $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T$ 라 표기한다.
  - 함수 $\mathbf{f}: \mathbb{R}^d \mapsto \mathbb{R}^m$을 미분하여 얻은 $m * d$ 행렬 $\mathbf{J}$: 야코비안 행렬: 각 변수로 편미분한 도함수로 표시

$$\text{야코비안 행렬 } \mathbf{J} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_d} \end{pmatrix}$$

예) $\mathbf{f}: \mathbb{R}^2 \mapsto \mathbb{R}^3$인 $\mathbf{f}(\mathbf{x}) = (2x_1 + x_2^2, -x_1^2 + 3x_2, 4x_1x_2)^{\mathrm{T}}$

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))^T = (f_1(x_1, x_2), f_2(x_1, x_2), f_3(x_1, x_2))^T$$
$$= (2x_1 + x_2^2 \quad , \quad -x_1^2 + 3x_2 \quad , \quad 4x_1x_2)^T$$

$$\mathbf{J} = \begin{pmatrix} 2 & 2x_2 \\ -2x_1 & 3 \\ 4x_2 & 4x_1 \end{pmatrix}$$

$\boldsymbol{x} = (2,1)^T$ 에서의 야코비안 $\quad \mathbf{J}|_{(2,1)^{\mathrm{T}}} = \begin{pmatrix} 2 & 2 \\ -4 & 3 \\ 4 & 8 \end{pmatrix}$

야코비안은 여러 개의 함수 f_i 에 대하여 여러 개의 x_j 에 대하여 편미분한 행렬이고,

헤시안은 하나의 함수 f 에 대하여 여러 개의 x_j 에 대하여 두 번 미분한 행렬이다.

2.3.2 미분

- 헤시안 행렬(Hessian matrix): (그레디언트(f')를 다시 한번 더 미분함. f를 두 번 미분(f''))

  함수 f를 x_j 로 편미분한 결과를 x_i로 다시 편미분한 2차 편도함수들의 $n * n$ 의 대칭인 행렬

2차 편도함수: $\frac{\partial^2 f}{\partial x_i x_j} = \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}\right)$

$$\text{헤시안 행렬 } \mathbf{H} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 x_1} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n x_n} \end{pmatrix}$$

예) 식(2.52)

$$f(\mathbf{x}) = f(x_1, x_2)$$
$$= \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1 x_2 + (-4 + 4x_2^2)x_2^2$$
$$\mathbf{H} = \begin{pmatrix} 10x_1^4 - 25.2x_1^2 + 8 & 1 \\ 1 & 48x_2^2 - 8 \end{pmatrix}$$

$\mathbf{x}=(0,1)^T$ 에서의 헤시안

$$\mathbf{H}|_{(0,1)^T} = \begin{pmatrix} 8 & 1 \\ 1 & 40 \end{pmatrix}$$

야코비안은 여러 개의 함수 f_i 에 대하여 여러 개의 x_j 에 대하여 편미분한 행렬이고,

헤시안은 하나의 함수 f 에 대하여 여러 개의 x_j 에 대하여 두 번 미분한 행렬이다.

2.3.3 경사 하강 알고리즘

- [알고리즘 2-3]을 다시 설명하면 미분으로 알아낸 그레이디언트 $\frac{\partial J}{\partial \boldsymbol{\Theta}}$는 오르막 방향을 가리키므로, 그레이디언트의 음수를 $d\boldsymbol{\Theta}$ 로 사용하여 내리막 방향을 찾아 조금씩 이동하는 일을 반복하여 최적해를 찾아나가는 것이다.
- $d\boldsymbol{\Theta} = -\frac{\partial J}{\partial \boldsymbol{\Theta}} = -\mathbf{g}$
- 식 (2.58)은 경사 하강법이 낮은 곳을 찾아가는 원리

$$\boldsymbol{\Theta} = \boldsymbol{\Theta} - \rho \mathbf{g} \qquad (2.58)$$

  - $\mathbf{g} = d\boldsymbol{\Theta} = \frac{\partial J}{\partial \boldsymbol{\Theta}}$이고, ρ는 학습률(learning rate)
  - 그레이디언트로 방향을 알 수 있지만 얼마만큼 이동해야 최적해에 도달할 수 있는지는 알 수 없으므로 이동량을 조절하기 위해 학습률을 곱한다.

- 식(2.58)을 이용하는 최적화 알고리즘을 경사하강법(gradient descent method)라고 함.

2.3.3 경사 하강 알고리즘

배치 경사 하강 알고리즘

- 라인 3에서 훈련집합 샘플 각각에 대해 그레이디언트를 계산한다. (용어 $\frac{\partial J}{\partial \boldsymbol{\Theta}} = \nabla$)

  훈련집합: $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ $\mathbb{Y} = \{y_1, y_2, \cdots, y_n\}$
- 라인4: 그레이디언트 평균 구함
- 라인5: 평균그레이디언트 음수를 더하여 내리막 경사방향으로 이동
- $\rho = 0.001$과 같이 작은 값을 주고 조금씩 이동. 더 이상 개선되지 않으면 멈춤
- 모든 샘플의 그레이디언트를 평균한 후 한꺼번에 갱신. 따라서 배치(batch)라는 용어 사용

  현대 기계학습에서는 거의 사용하지 않고 대신 SGD 사용=> 다음 페이지

알고리즘 2-4 배치 경사 하강 알고리즘(BGD)

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ

출력: 최적해 $\hat{\boldsymbol{\Theta}}$

1 난수를 생성하여 초기해 $\boldsymbol{\Theta}$를 설정한다.
2 repeat
3 $\mathbb{X}$에 있는 샘플의 그레이디언트 $\nabla_1, \nabla_2, \cdots, \nabla_n$을 계산한다.
4 $\nabla_{total} = \frac{1}{n}\sum_{i=1,n} \nabla_i$ // 그레이디언트 평균을 계산
5 $\boldsymbol{\Theta} = \boldsymbol{\Theta} - \rho \nabla_{total}$
6 until(멈춤 조건)
7 $\hat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

참고: $\frac{\partial J}{\partial \boldsymbol{\Theta}} = \nabla$

2.3.3 경사 하강 알고리즘

- 스토캐스틱 경사 하강SGD(stochastic gradient descent) 알고리즘
  - 한번에 샘플 하나의 그레이디언트를 계산한 후 즉시 매개변수 갱신
  - 라인 3~6을 모든 샘플에 한 번 반복하는 일을 한 세대(epoch)라 부름
  - 알고리즘은 멈춤 조건이 만족할 때까지 세대를 반복한다. **새로운 세대가 시작할 때마다 샘플의 순서를 섞어 알고리즘의 임의성을 투입한다.**
  - 이 임의성 때문에 이름을 스토케이스틱이라는 용어가 붙었다.

알고리즘 2-5 스토케스틱 경사 하강 알고리즘(SGD)

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

```
1  난수를 생성하여 초기해 Θ를 설정한다.
2  repeat
3    X의 샘플의 순서를 섞는다.
4    for (i=1 to n)
5          i번째 샘플에 대한 그레이디언트 ∇_i를 계산한다.
6          Θ = Θ − ρ∇_i
7  until(멈춤 조건)
8  Θ̂ = Θ
```

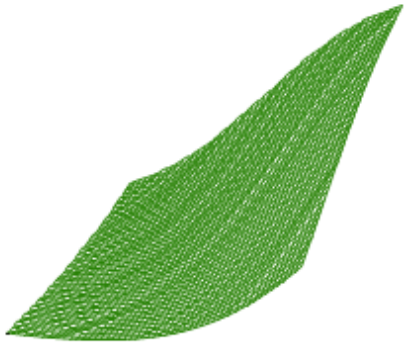
2.3.3 경사 하강 알고리즘

- 스토캐스틱 경사 하강SGD(stochastic gradient descent) 알고리즘

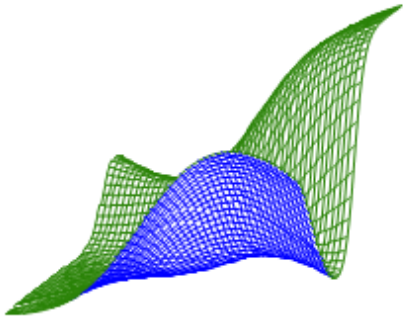
  - [알고리즘 2-5]의 라인 3~6을 다음과 같이 다른 방식의 구현

3 $\mathbb{X}$에서 임의로 샘플 하나를 뽑는다.
4 뽑힌 샘플의 그레이디언트 ∇를 계산한다.
5 $\Theta = \Theta - \rho\nabla$

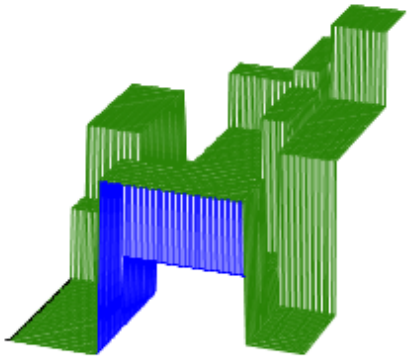
대표적인 빅데이터 분석 모형 (Supervised Learning)



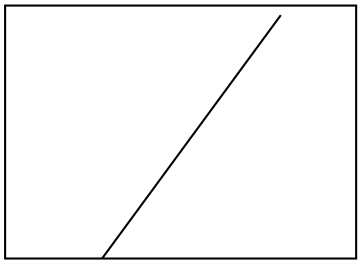
일반화선형모형
(Generalized Linear Models):
회귀모형(Regression) /
로지스틱회귀(Logistic Regression)



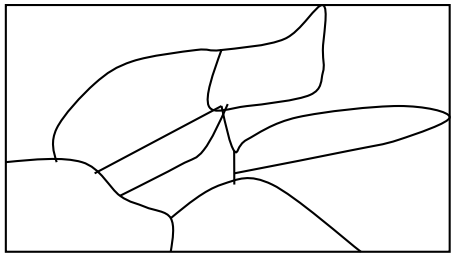
신경망 모형
(Neural
Networks)



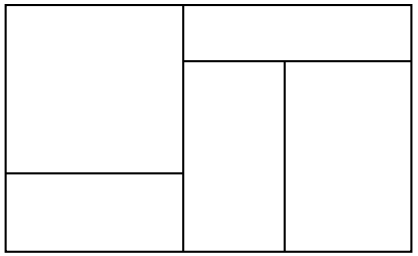
의사결정나무
(Decision
Trees)



Linear regression



Neural networks



Classification trees

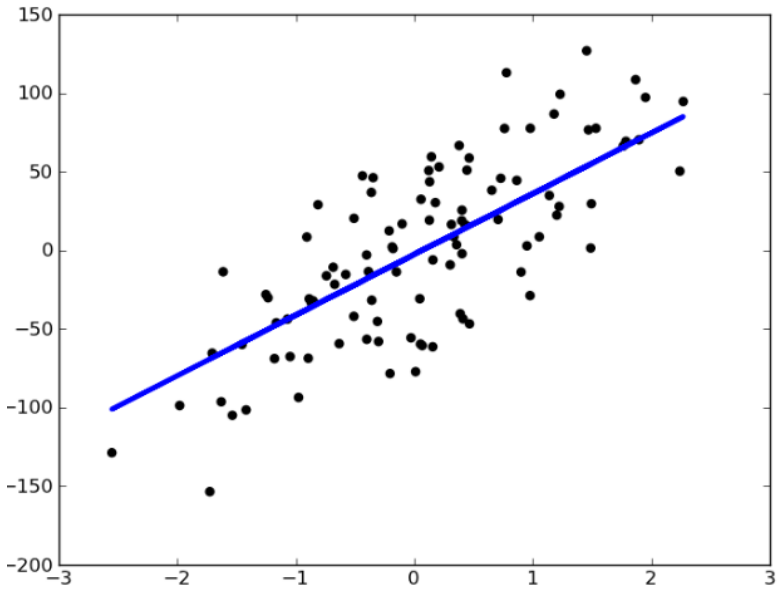
참고: 데이터마이닝기법 ≈ 빅데이터분석기법 ≈ 기계학습(머신러닝)기법

일반화 선형 모델 (Generalized Linear Model, GLM)

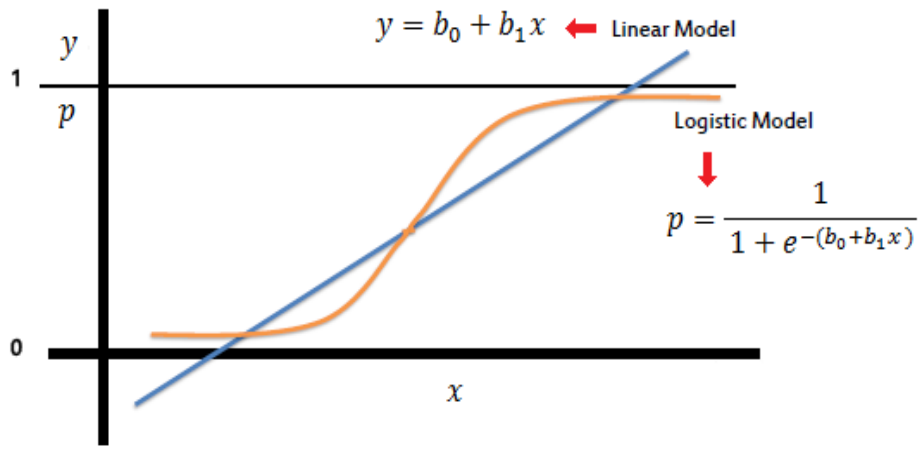
반응변수(Y) / 설명변수(X)	연속형	범주형
연속형	선형회귀 모형	로지스틱 모형
범주형	분산 분석	분할표 분석 (하나의 X) 로지스틱 모형 또는 로그 선형 모형(여러 개의 Xs)
연속형 + 범주형	공분산 분석 선형회귀(가변수) 모형	로지스틱 모형

일반화 선형 모형(Generalized Linear Models)

회귀(Regression) 모형



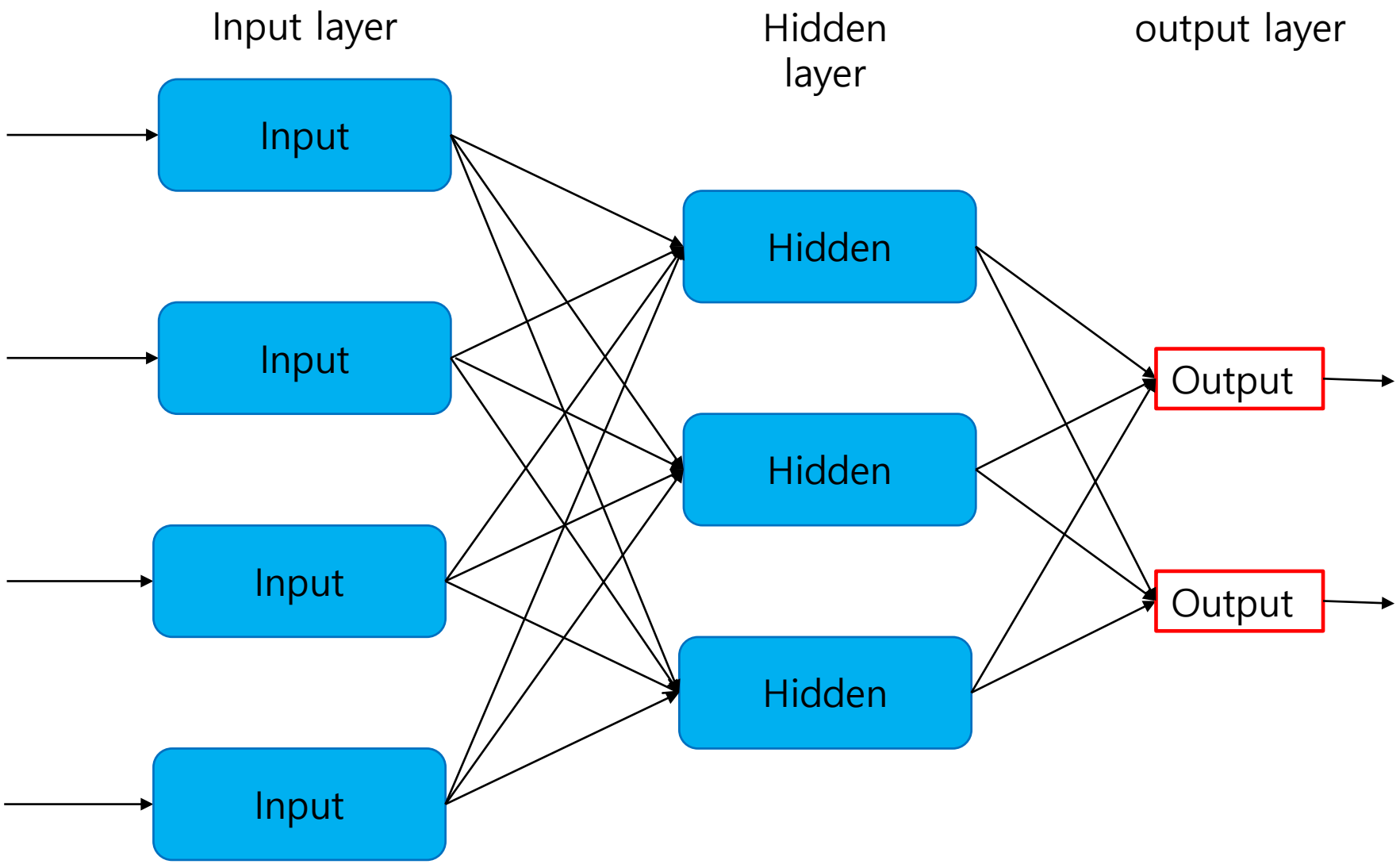
로지스틱회귀(Logistic Regression) 모형



일반화 선형 모형(Generalized Linear Models)

	B	C	D	E	F	G	H	I	J	K	L	M	N
1	site	ymd	Tem_Mean	Tem_max	Tem_min	Rain	WS_Mean	WS_Max	Solar_sum		**Pollen(Y)**	Pollen 0/1	
162	서울	2014-08-08	26.1	30.6	21.9		3.8	6.9	19.64		0	0	
163	서울	2014-08-09	26.5	31.6	22.3		4.6	7.3	22.35		0	0	
164	서울	2014-08-10	23.4	27.7	19.6	42.5	2.6	9.3	6.66		0	0	
165	서울	2014-08-11	23.7	28.6	18.3		2.2	5.2	19.44		1	1	
166	서울	2014-08-12	25.8	30.4	22.3		2.7	4.9	16.57		36	1	
167	서울	2014-08-13	24.2	26.7	22.5	0	2.1	4.6	6.14		1	1	
168	서울	2014-08-14	22.3	24.2	21.4	0.5	2	3.8	4.32		3	1	
169	서울	2014-08-15	24.9	29.6	20.9		1.7	4.5	12.53		9	1	
170	서울	2014-08-16	26.1	31.8	22.2		1.8	5.1	15.5		0	0	
171	서울	2014-08-17	24	25	22.9	3	1.9	4.2	3.2		0	0	
172	서울	2014-08-18	22.3	24.4	20.9	11.5	3.1	5	6.42		0	0	
173	서울	2014-08-19	22.3	25.3	20.4	4.5	1.5	3.5	4.37		0	0	
174	서울	2014-08-20	24.8	28	22.4	0.5	1.1	2.5	6.74		15	1	
175	서울	2014-08-21	23.1	24.6	21.8	46.5	2.4	5.2	4.94		4	1	
176	서울	2014-08-22	24.7	28.2	21.8	15	3.1	5.9	13.56		64	1	
177	서울	2014-08-23	25.2	29.7	22.8	0.5	2.7	4.4	12.29		1	1	
178	서울	2014-08-24	25.6	30.7	20.7	0.1	1.3	4.5	15.04		4	1	
179	서울	2014-08-25	25.9	28.6	24.1	0	3	4.8	8.43		0	0	
180	서울	2014-08-26	25.4	29.6	21.6	11.5	2.5	4.8	11.38		14	1	
181	서울	2014-08-27	24.6	28.6	21.1		2.8	4	14.77		3	1	
182	서울	2014-08-28	25.1	30.7	19.8		2.2	4.6	18.41		63	1	
183	서울	2014-08-29	25.2	30.1	21	6.5	1.8	5	14.21		20	1	
184	서울	2014-08-30	24.6	30.7	19.9		1.9	4.3	17.6		50	1	
185	서울	2014-08-31	24.1	29	20.7		1.4	3	12.18		35	1	
186	서울	2014-09-01	24.1	29.7	20.4	1.5	1.5	5.1	13.6		16	1	
187	서울	2014-09-02	23.1	27.7	19.3	6	2.5	5.6	10.67		39	1	

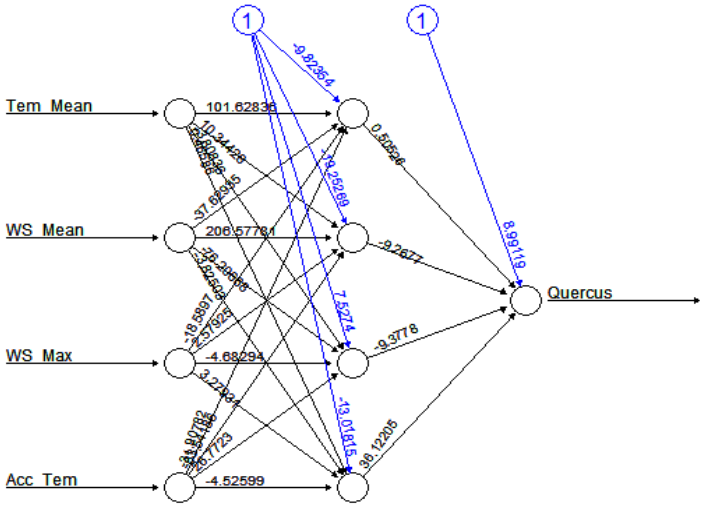
신경망 모형(Neural network model)



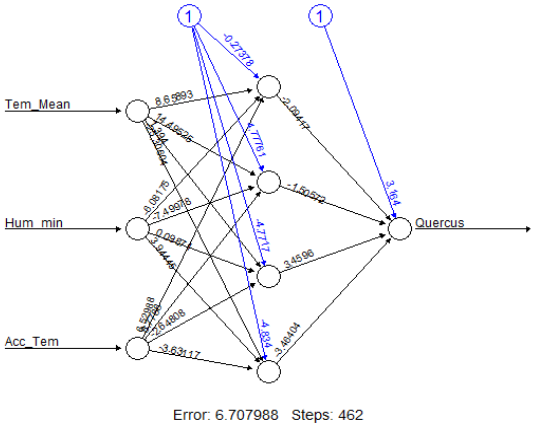
신경망 모형(Neural network model)

- **기상요소와 참나무 꽃가루농도의 신경망모형(neural networks)**

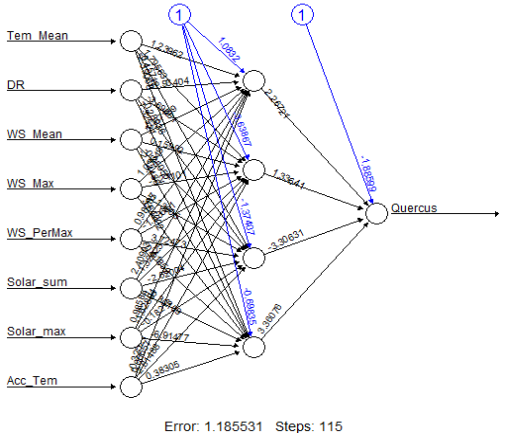
신경망모형 결과(hidden node 갯수 =4)



전체지역



서울지역

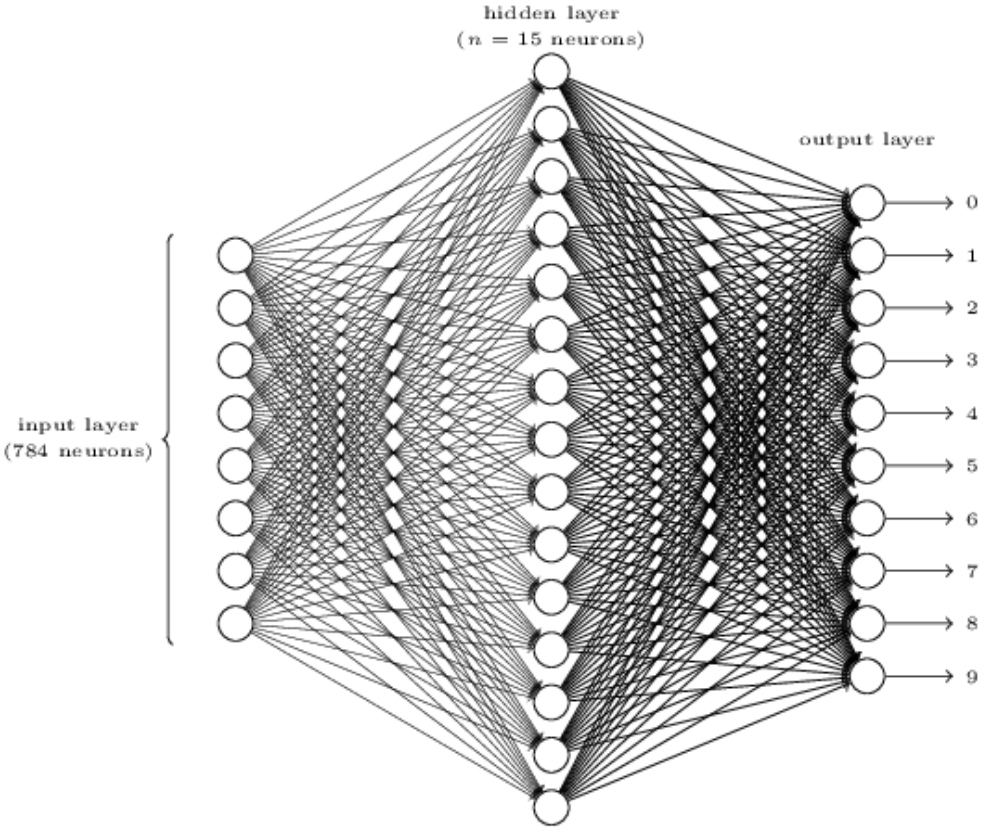


대전지역

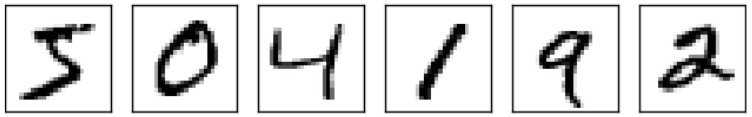
신경망 모형(Neural network model)

Using neural nets to recognize handwritten digits

Training :



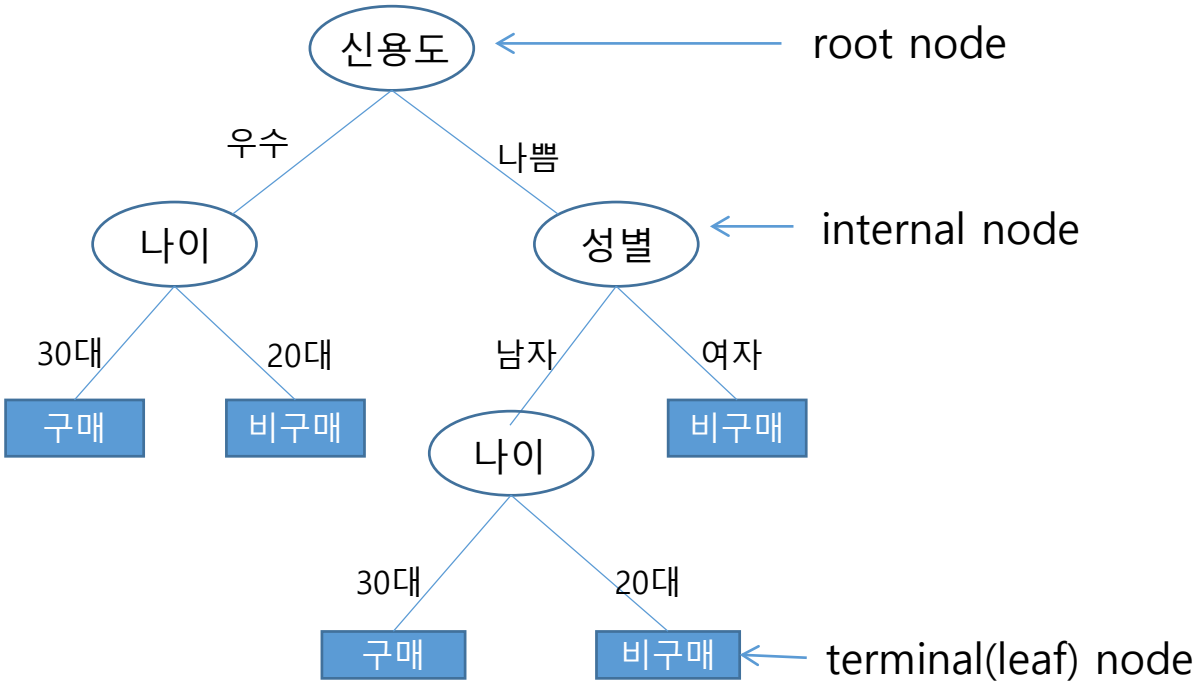
Test:



출처: http://neuralnetworksanddeeplearning.com/chap1.html

의사결정나무 (Decision trees)

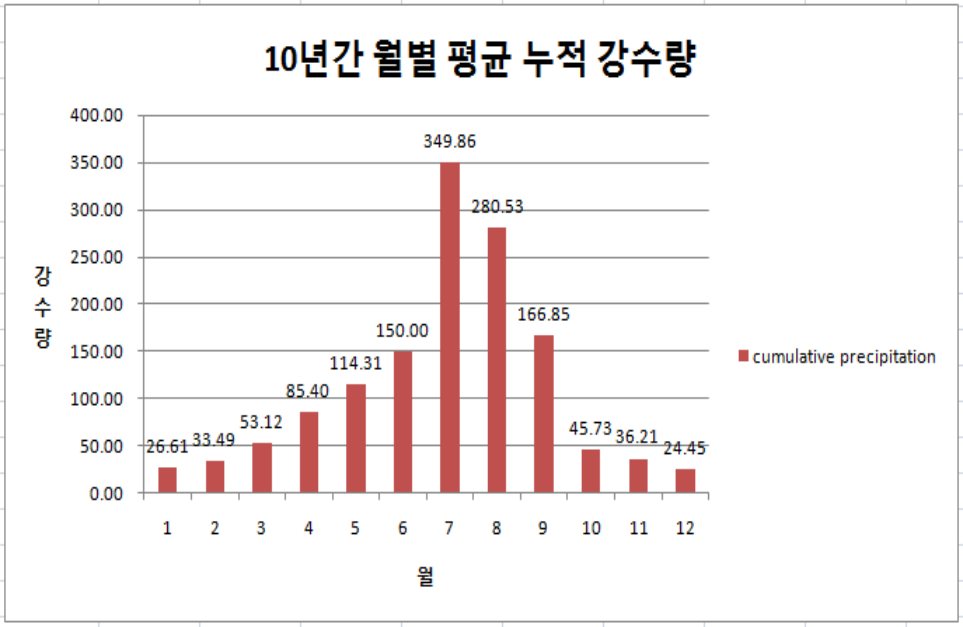
	구매비구매 데이터			
1				
2				
3	성별	나이	신용도	구매유무
4	여자	30대	나쁨	비구매
5	여자	20대	우수	비구매
6	남자	30대	나쁨	구매
7	남자	30대	우수	구매
8	여자	20대	나쁨	비구매
9	남자	20대	나쁨	비구매
10	남자	30대	나쁨	구매
11	여자	20대	우수	비구매
12	남자	30대	나쁨	구매
13	여자	30대	우수	구매
14	남자	20대	나쁨	비구매
15	남자	20대	우수	비구매
16	남자	20대	나쁨	비구매
17	여자	30대	나쁨	비구매
18	여자	30대	우수	구매
19	남자	30대	우수	구매
20	여자	20대	나쁨	비구매
21	여자	30대	우수	구매
22	남자	30대	나쁨	구매



의사결정나무 (Decision trees)

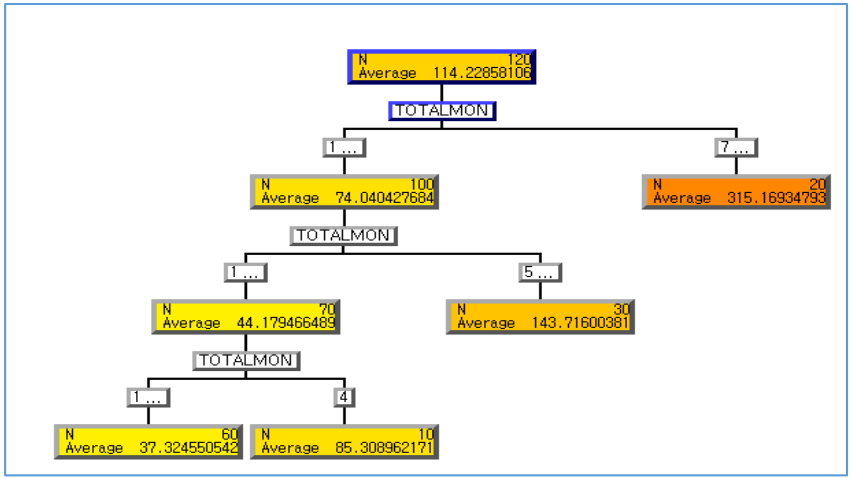
강수량 월별 그룹화(계절 구분)

자료 구성	
시간적	10년간(2001~2010) 월 평균 누적강수량
공간적	남한 전체 606지점(synop+aws)



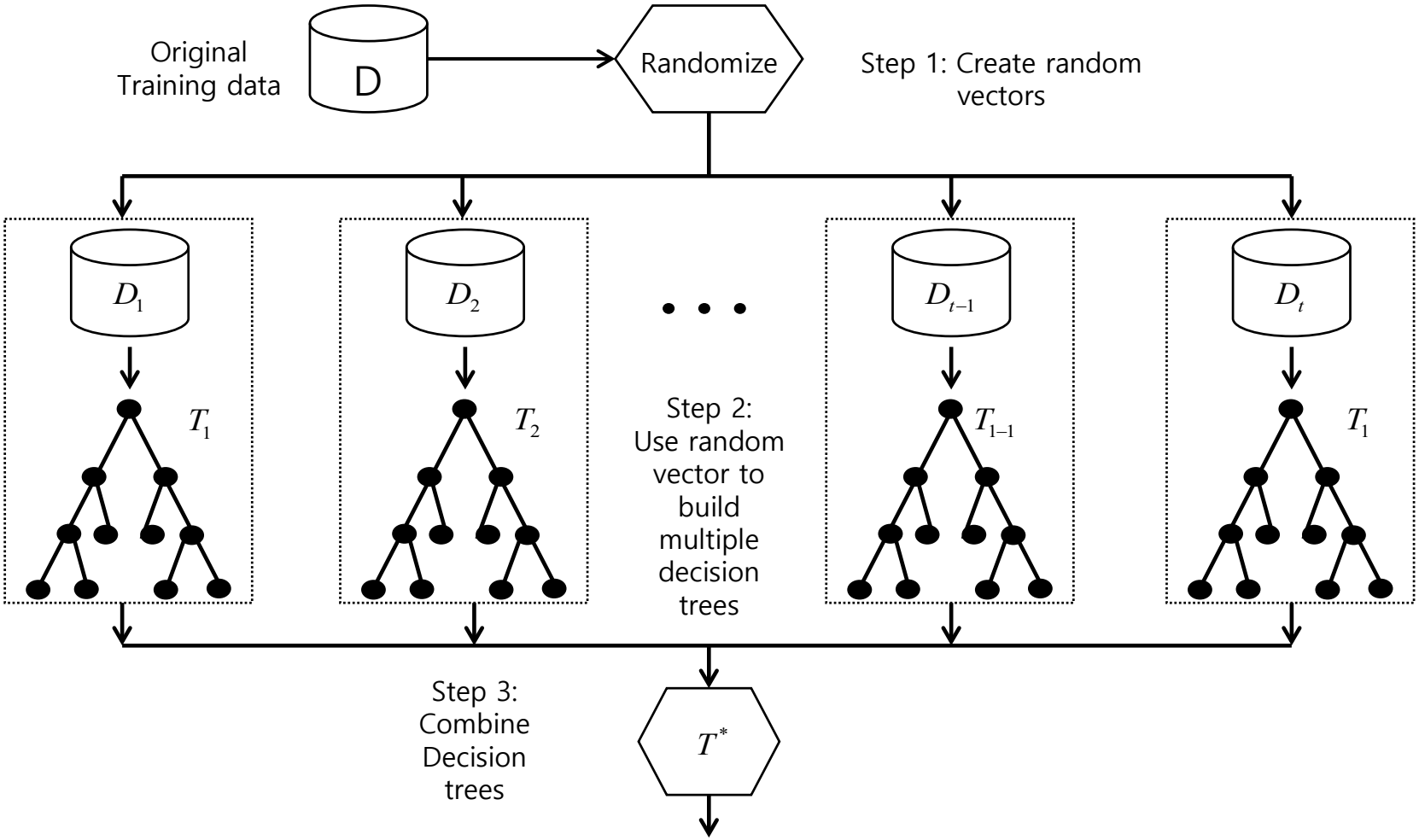
◆ 통계적 의사결정나무(Decision Tree) 기법 적용 하여 계절구분

- 제 1 계절 : 7월, 8월
- 제 2 계절 : 5월, 6월, 9월
- 제 3 계절 : 1월, 2월, 3월, 4월 ,10월, 11월, 12월



앙상블기법(Ensemble methods)

Bagging, Boosting, Random forest, etc.



출처 : INTRODUCTION TO DATA MINING/ PANG-NING TAN, (2005)

(선형)회귀 모형 ((Linear) Regression Model)

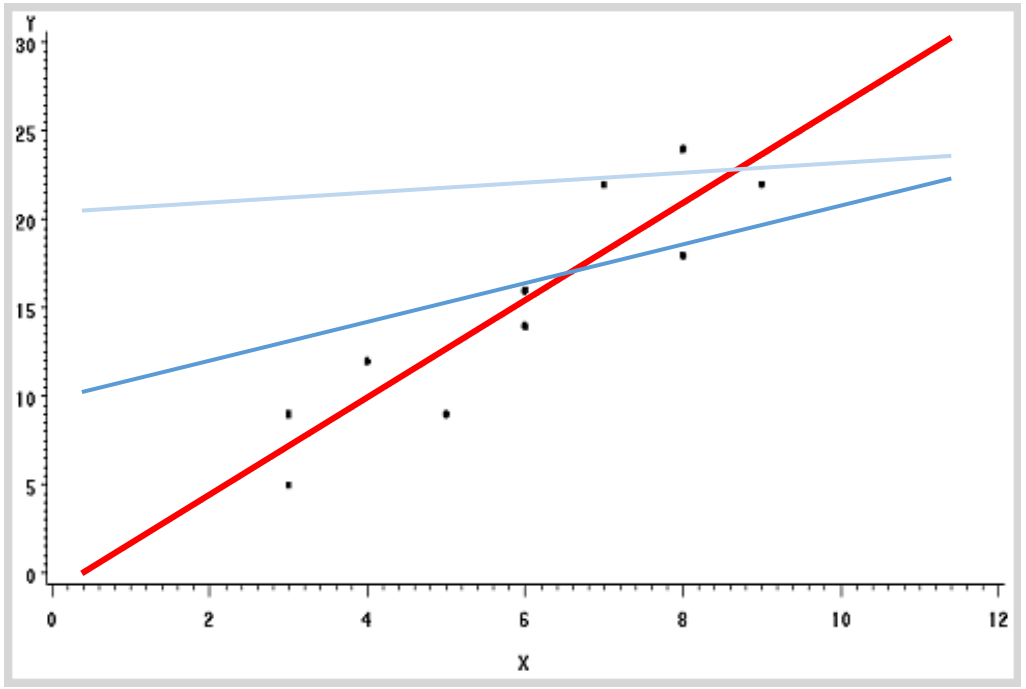
- 단순 선형 회귀 모형 (설명변수(X)가 하나임).

$$y = \beta_0 + \beta_1 x + \varepsilon$$

$$\varepsilon \sim N(0, \sigma^2)$$

$$\hat{y} = b_0 + b_1 x \quad \text{또는}$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

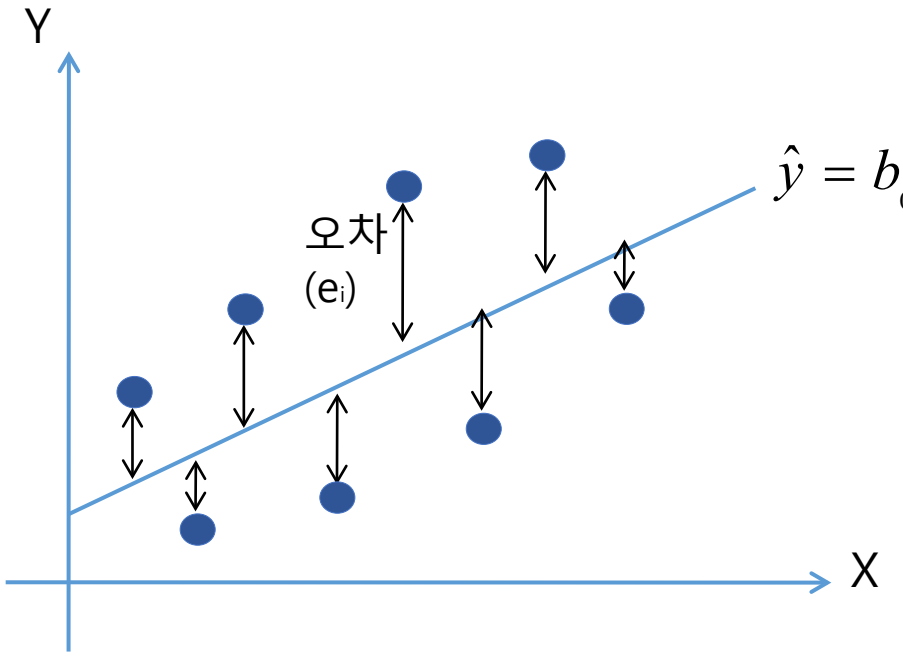


자료들을 가장 잘 대표하고, 가장 잘 지나는 직선(붉은 선)을 찾는다

(선형)회귀 모형 ((Linear) Regression Model)

- 단순 선형 회귀 모형 (설명변수(X)가 하나임).

- 최소 제곱법



$\hat{y} = b_0 + b_1 x$ 또는 $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$

오차(e_i)의 제곱을 더했을 때 최소가 되는 직선

모든 데이터를 가장 잘 표현 할 수 있는 1개의 직선

(선형)회귀 모형 ((Linear) Regression Model)

- 최소 제곱법(계속)

  잔차가 최소가 되도록 다음과 같이 식을 세워 구한다.

$$Q(\hat{\beta}_0, \hat{\beta}_1) = \sum_{i=1}^{n} e_i^{\,2} = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 X)^2$$

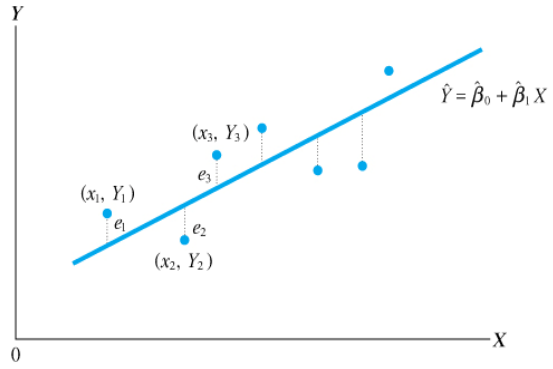
  회귀계수를 구할 때, 위의 식을 최소화하는 값을 구한다. 이와 같은 원리를 최소제곱법(least square method)이라고 한다. 추정량을 구하는 과정은 방정식의 극소값을 구하는 방법으로 $\hat{\beta}_0$, $\hat{\beta}_1$ 를 에 대하여 미분을 취하여 0으로 놓은 후에 와 에 대하여 해를 구한다.

  식을 정리해 얻은 결과는 다음과 같다.

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

그림 12-10 회귀선과 잔차그림



회귀분석 복습

2.2 단순선형회귀모형

단순선형 회귀모형: $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \quad i = 1, 2, \cdots, n$

i : 전체 n개의 관측값 중 i번째 값을 나타내는 첨자

ε_i : 평균이 0, 분산이 σ^2인 오차를 나타내는 확률변수.

ε_i 들은 확률적으로 서로 독립

X_i : 값이 주어진 상수

==> Y_i는 $X = X_i$로 주어질 때 평균이 $\beta_0 + \beta_1 X_i$이고, 분산이 σ^2인 확률변수

Y_i들 역시 독립

기울기 β_1 : 설명변수 X가 한 단위 증가할 때 움직이는 Y의 평균값의 변화량

출처: 회귀분석(제2판), 김충락, 강근석 지음, 교우사 (2010)

회귀분석 복습

2.3 회귀계수의 추정

2.3.1 최소제곱법

표본: $(X_1, Y_1), \cdots, (X_n, Y_n)$

최소제곱법(method of least squares) : 오차의 크기를 "전체적"으로 작게 하는 방법.

- 각 오차의 제곱의 합을 최소로 하는 회귀식을 구하는 방법
- 독일의 수학자 가우스(Carl Friedrich Gauss)에 의해 사용됨.

최소제곱법 :

관측치 Y_i와 모집단회귀식 $\beta_0 + \beta_1 X_i$와의 차이인 오차 ε_i들의 제곱의 합이 최소가 되도록 회귀계수를 추정하는 방법.

즉,

$$S = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2$$

을 최소화하는 β_0와 β_1의 값을 구한다.

출처: 회귀분석(제2판), 김충락, 강근석 지음, 교우사 (2010)

회귀분석 복습

제곱합의 β_0와 β_1에 대한 편미분:

$$\frac{\partial S}{\partial \beta_0} = (-2)\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i) = 0$$

$$\frac{\partial S}{\partial \beta_1} = (-2)\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i)X_i = 0$$

정규방정식(normal equation)

$$nb_0 + b_1\sum_{i=1}^{n} X_i = \sum_{i=1}^{n} Y_i$$

$$b_0\sum_{i=1}^{n} X_i + b_1\sum_{i=1}^{n} X_i^2 = \sum_{i=1}^{n} X_i Y_i$$

최소제곱추정량(least squares estimator):

$$b_1 = \frac{S_{XY}}{S_{XX}}$$

$$b_0 = \overline{Y} - b_1\overline{X}$$

위에서

$$S_{XY} = \sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})$$

$$S_{XX} = \sum_{i=1}^{n}(X_i - \overline{X})^2$$

b_0와 b_1의 의미: 표본회귀식의 절편과 기울기

b_1 : 설명변수 X가 한 단위 변화할 때 반응변수 Y의 평균변화량

- **b_0와 b_1은 X와 Y변수가 갖는 단위들과 함께 해석되어야 한다**

출처: 회귀분석(제2판), 김충락, 강근석 지음, 교우사 (2010)

회귀분석 복습

3.1 중선형회귀모형

3.1.1 모집단 중회귀모형

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_{p-1} X_{i,p-1} + \varepsilon_i , \quad i = 1, 2, \cdots, n$$

$\varepsilon_i \sim iid \ N(0, \sigma^2)$

X_{ij} = j번째 설명변수 X_j의 i번째 관측치

β_j = Y와 X_j간의 기울기. 다른 설명변수들의 값들이 고정되었을 때

3.2 회귀계수의 추정

3.2.1 최소제곱법

오차제곱합 : $S = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_{i1} - \cdots - \beta_{p-1} X_{i,p-1})^2$

$$\frac{\partial S}{\partial \beta_0} = (-2) \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_{i1} - \cdots - \beta_{p-1} X_{i,p-1}) = 0$$

$$\frac{\partial S}{\partial \beta_j} = (-2) \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_{i1} - \cdots - \beta_{p-1} X_{i,p-1}) X_{ij} = 0 , \quad j = 1, 2, \cdots, p-1$$

출처: 회귀분석(제2판), 김충락, 강근석 지음, 교우사 (2010)

회귀분석 복습

$b_0, b_1, \cdots, b_{p-1}$: β_j들의 최소제곱추정량

정규방정식: p차원 연립방정식

$$nb_0 + b_1 \sum_{i=1}^{n} X_{i1} + \cdots + b_{p-1} \sum_{i=1}^{n} X_{i,p-1} = \sum_{i=1}^{n} Y_i$$

$$b_0 \sum_{i=1}^{n} X_{ij} + b_1 \sum_{i=1}^{n} X_{i1} X_{ij} + \cdots + b_{p-1} \sum_{i=1}^{n} X_{i,p-1} X_{ij} = \sum_{i=1}^{n} X_{ij} Y_i , \quad j = 1, 2, \cdots, p-1$$

추정된 회귀식: $\hat{Y}_i = b_0 + b_1 X_{i1} + \cdots + b_{p-1} X_{i,p-1}$

잔차: $e_i = Y_i - \hat{Y}_i = Y_i - (b_0 + b_1 X_{i1} + \cdots + b_{p-1} X_{i,p-1})$

행렬과 벡터를 이용한 유도

오차제곱합: $S = \sum_{i=1}^{n} \varepsilon_i^2 = \boldsymbol{\varepsilon}' \boldsymbol{\varepsilon} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$

$$\frac{\partial S}{\partial \boldsymbol{\beta}} = -2\mathbf{X}'\mathbf{y} + 2(\mathbf{X}'\mathbf{X})\boldsymbol{\beta}$$ (p.471 부록 A.7.2 참조)

정규방정식: $(\mathbf{X}'\mathbf{X})\mathbf{b} = \mathbf{X}'\mathbf{y}$

최소제곱추정량: $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$: 추정량 b 가 y 들의 선형 함수임.

$\mathbf{X}'\mathbf{X}$ 의 역행렬을 구하기 위해 QR 분해법 등의 계산 방법 이용.

출처: 회귀분석(제2판), 김충락, 강근석 지음, 교우사 (2010)

회귀분석 복습

적합된 반응벡터: $\hat{y} = Xb$

단순회귀의 경우:

$$X'X = \begin{bmatrix} n & \sum_{i=1}^{n} X_i \\ \sum_{i=1}^{n} X_i & \sum_{i=1}^{n} X_i^2 \end{bmatrix}, \qquad X'y = \begin{bmatrix} \sum_{i=1}^{n} Y_i \\ \sum_{i=1}^{n} X_i Y_i \end{bmatrix}$$

$$(X'X)^{-1} = \frac{1}{n\sum_{i=1}^{n} X_i^2 - (\sum_{i=1}^{n} X_i)^2} \begin{bmatrix} \sum_{i=1}^{n} X_i^2 & -\sum_{i=1}^{n} X_i \\ -\sum_{i=1}^{n} X_i & n \end{bmatrix}$$

==> $(X'X)^{-1}X'y$ = 식(2.5) p.51

추정된 회귀식: $\hat{Y}_i = x_i'b$

잔차: $e_i = Y_i - x_i'b$

잔차벡터(residual vector) : $e = y - \hat{y} = y - Xb$

여기에서 $\hat{y} = (\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_n)$, $e = (e_1, e_2, \ldots, e_n)$, Xb 는 기대함수 $X\beta$ 에서 β 대신 추정량 b 를 대입한 것.

출처: 회귀분석(제2판), 김충락, 강근석 지음, 교우사 (2010)

Gradient Descent algorithm 이해하기

Lets take the example of predicting the price of a new price from housing data:

Now, given **historical housing data,** the task is to create a model that predicts the price of a new house given the house size.

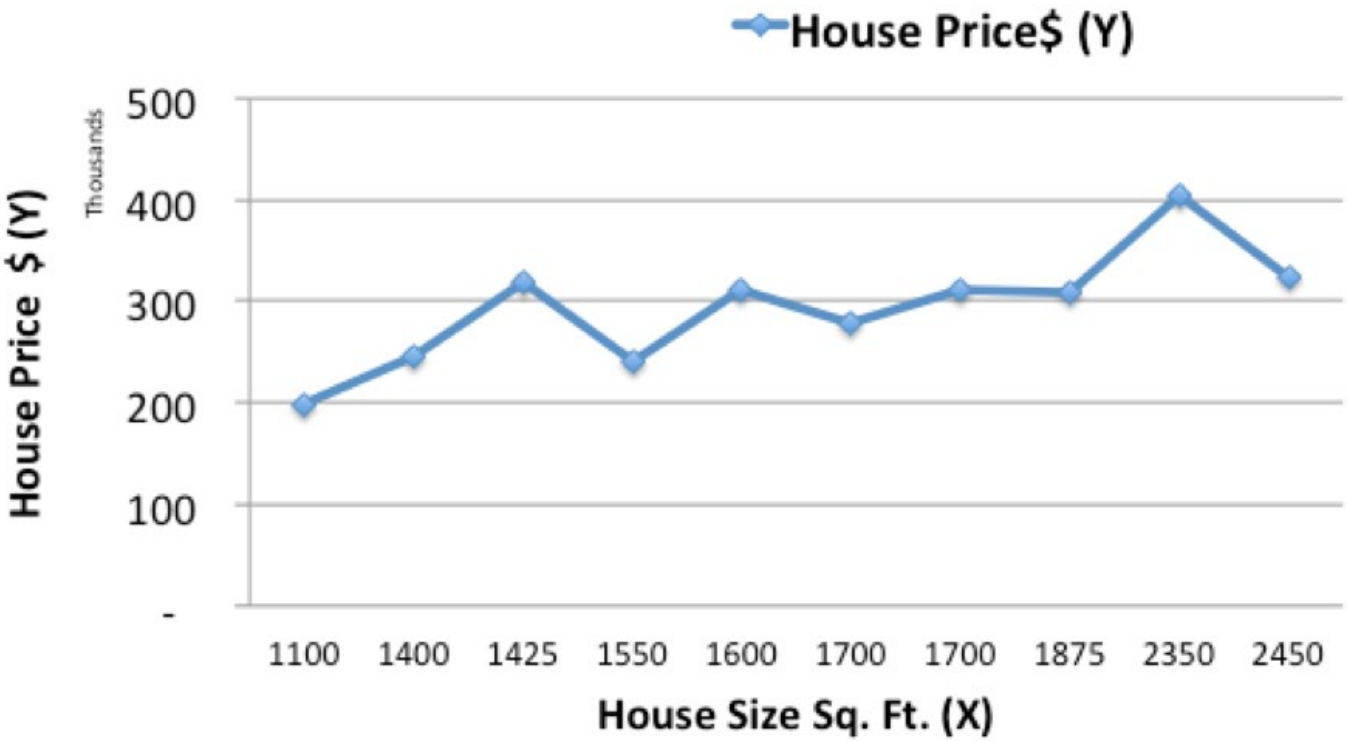
House Size sq.ft (X)	1400	1600	1700	1875	1100	1550	2350	2450	1425	1700
House Price$ (Y)	245,000	312,000	279,000	308,000	199,000	219,000	405,000	324,000	319,000	255,000

The task – for a new house, given its size (X), what will its price (Y) be?

출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

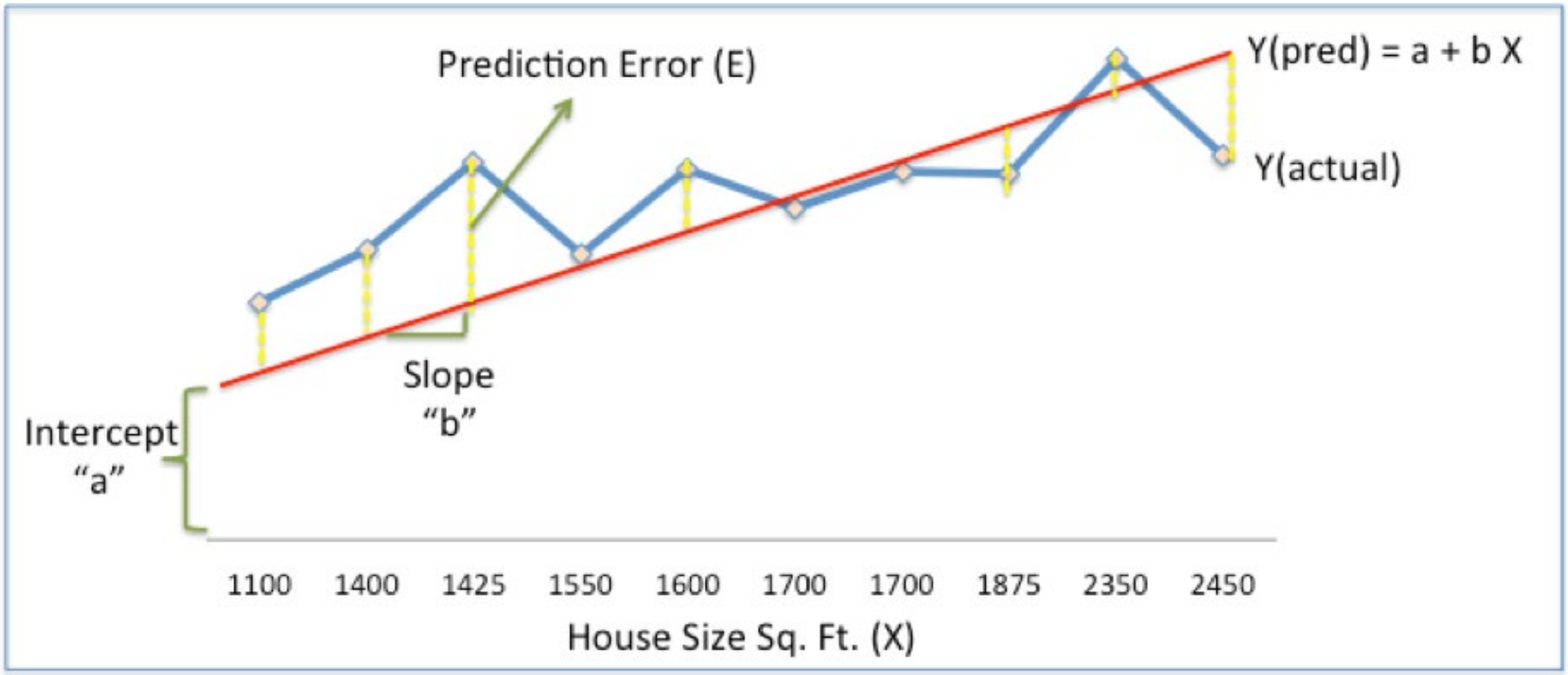
Lets start off by plotting the historical housing data:



출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

Now, we will use a simple linear model, where we fit a line on the historical data, to predict the price of a new house (Ypred) given its size(X)



출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

In the above chart, the red line gives the predicted house price (Ypred) given house size (X).
Ypred = a+bX

The blue line gives the actual house prices from historical data (Yactual)
The difference between Yactual and Ypred (given by the yellow dashed lines) is the prediction error (E)

So, we need to find a line with optimal values of a,b (called weights) that best fits the historical data by reducing the prediction error and improving prediction accuracy.

So, our objective is to find optimal **a, b** that minimizes the error between actual and predicted values of house price (1/2 is for mathematical convenience since it helps in calculating gradients in calculus)

Sum of Squared Errors (SSE) = ½ Sum (Actual House Price – Predicted House Price)^2
= ½ Sum(Y – Ypred)^2

(Please note that there are other measures of Error. SSE is just one of them.)

This is where Gradient Descent comes into the picture. Gradient descent is an optimization algorithm that finds the optimal weights (a,b) that reduces prediction error.

출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

Lets now go step by step to understand the **Gradient Descent algorithm:**

Step 1: Initialize the weights(a & b) with random values and calculate Error (SSE)

Step 2: Calculate the gradient i.e. change in SSE when the weights (a & b) are changed by a very small value from their original randomly initialized value. This helps us move the values of a & b in the direction in which SSE is minimized.

Step 3: Adjust the weights with the gradients to reach the optimal values where SSE is minimized

Step 4: Use the new weights for prediction and to calculate the new SSE

Step 5: Repeat steps 2 and 3 till further adjustments to weights doesn't significantly reduce the Error

출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

We will now go through each of the steps in detail (I worked out the steps in excel, which I have pasted below). But before that, we have to standardize the data as it makes the optimization process faster.

HOUSING DATA	
House Size (X)	House Price (Y)
1,100	1,99,000
1,400	2,45,000
1,425	3,19,000
1,550	2,40,000
1,600	3,12,000
1,700	2,79,000
1,700	3,10,000
1,875	3,08,000
2,350	4,05,000
2,450	3,24,000

Min-Max Standardization	
X (X-Min/Max-min)	Y (Y-Min/Max-Min)
0.00	0.00
0.22	0.22
0.24	0.58
0.33	0.20
0.37	0.55
0.44	0.39
0.44	0.54
0.57	0.53
0.93	1.00
1.00	0.61

출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

Step 1: To fit a line Ypred = a + b X, start off with random values of a and b and calculate prediction error (SSE)

a	b	X	Y	YP=a+bX	SSE=1/2(Y-YP)^2
0.45	0.75	0.00	0.00	0.45	0.101
		0.22	0.22	0.62	0.077
		0.24	0.58	0.63	0.001
		0.33	0.20	0.70	0.125
		0.37	0.55	0.73	0.016
		0.44	0.39	0.78	0.078
		0.44	0.54	0.78	0.030
		0.57	0.53	0.88	0.062
		0.93	1.00	1.14	0.010
		1.00	0.61	1.20	0.176
				Total SSE	**0.677**

출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

Step 2: Calculate the error gradient w.r.t the weights

$\partial SSE/\partial a = -(Y-YP)$
$\partial SSE/\partial b = -(Y-YP)X$

Here, $SSE=\frac{1}{2}(Y-YP)^2 = \frac{1}{2}(Y-(a+bX))^2$
You need to know a bit of calculus, but that's about it!!

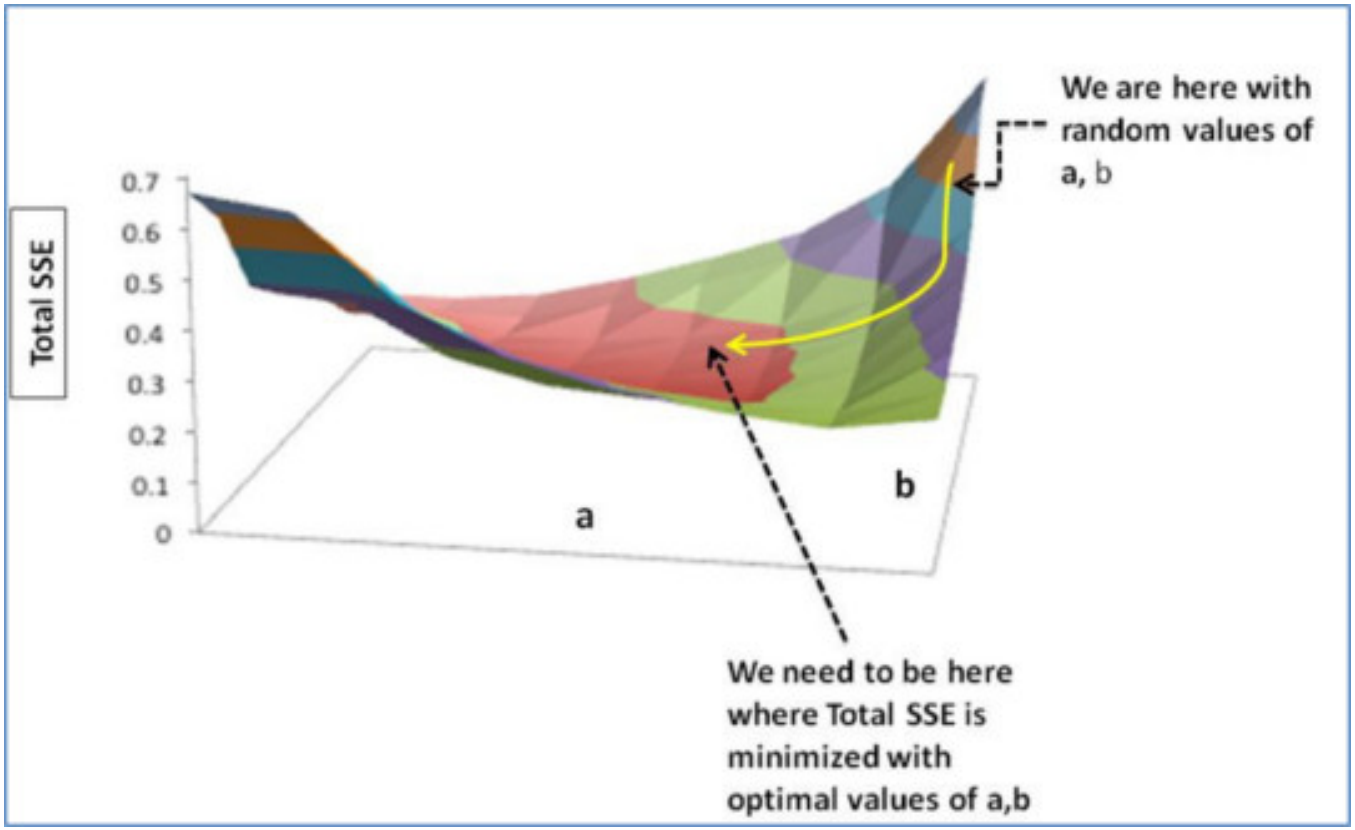
$\partial SSE/\partial a$ and $\partial SSE/\partial b$ are the **gradients** and they give the direction of the movement of a,b w.r.t to SSE.

a	b	X	Y	YP=a+bX		SSE		∂SSE/∂a = -(Y-YP)	∂SSE/∂b = -(Y-YP)X
0.45	0.75	0.00	0.00	0.45		0.101		0.45	0.00
		0.22	0.22	0.62		0.077		0.39	0.09
		0.24	0.58	0.63		0.001		0.05	0.01
		0.33	0.20	0.70		0.125		0.50	0.17
		0.37	0.55	0.73		0.016		0.18	0.07
		0.44	0.39	0.78		0.078		0.39	0.18
		0.44	0.54	0.78		0.030		0.24	0.11
		0.57	0.53	0.88		0.062		0.35	0.20
		0.93	1.00	1.14		0.010		0.14	0.13
		1.00	0.61	1.20		0.176		0.59	0.59
					Total SSE	0.677	Sum	3.300	1.545

출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

Step 3: Adjust the weights with the gradients to reach the optimal values where SSE is minimized



출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

We need to update the random values of a,b so that we move in the direction of optimal a, b.

Update rules:

• a – ∂SSE/∂a

• b – ∂SSE/∂b

So, update rules:

1. New a = a – r * **∂SSE/∂a =** 0.45-0.01*3.300 = 0.42

2. New b = b – r * **∂SSE/∂b=** 0.75-0.01*1.545 = 0.73

here, r is the learning rate = 0.01, which is the pace of adjustment to the weights.

출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

Step 4: Use new a and b for prediction and to calculate new Total SSE

a	b	X	Y	YP=a+bX		SSE		∂SSE/∂a	∂SSE/∂b
0.42	0.73	0.00	0.00	0.42		0.087		0.42	0.00
		0.22	0.22	0.58		0.064		0.36	0.08
		0.24	0.58	0.59		0.000		0.01	0.00
		0.33	0.20	0.66		0.107		0.46	0.15
		0.37	0.55	0.69		0.010		0.14	0.05
		0.44	0.39	0.74		0.063		0.36	0.16
		0.44	0.54	0.74		0.021		0.20	0.09
		0.57	0.53	0.84		0.048		0.31	0.18
		0.93	1.00	1.10		0.005		0.10	0.09
		1.00	0.61	1.15		0.148		0.54	0.54
					Total SSE	**0.553**	**Sum**	**2.900**	**1.350**

You can see with the new prediction, the total SSE has gone down (0.677 to 0.553). That means prediction accuracy has improved.

출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

Gradient Descent algorithm 이해하기

Step 5: Repeat step 3 and 4 till the time further adjustments to a, b doesn't significantly reduces the error. At that time, we have arrived at the optimal a,b with the highest prediction accuracy.

This is the Gradient Descent Algorithm. This optimization algorithm and its variants form the core of many machine learning algorithms like Neural Networks and even Deep Learning.

출처: http://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html
Keep it simple! How to understand Gradient Descent algorithm by Jahnavi Mahanta

목적함수(objective function)

- 평균제곱 오차(MSE) 목적함수 ($\boldsymbol{y}$ 는 원래 데이터의 값 , $\boldsymbol{o}$ 는 모형으로부터의 output)

$$e = \frac{1}{2}\|\mathbf{y} - \mathbf{o}\|_2^2 \qquad (5.1)$$

  - 오차가 클수록 e값이 크므로 벌점으로 훌륭함

출처: 기계학습, 오일석 지음, 한빛아카데미(2017)

교차 엔트로피 (cross entropy) 목적함수

- **교차 엔트로피:** 앞의 평균제곱오차를 목적함수로 사용하면 더딘 학습 현상이 발생한다, 따라서 딥러닝에서는 교차 엔트로피(식 (2.47) : $H(P,Q) = -\sum_y P(y)log_2Q(y)$)를 주로 사용한다.
  - 레이블에 해당하는 y가 확률변수 (부류가 2개라고 가정하면 $y \in \{0,1\}$)
  - 확률 분포: P는 정답 레이블, Q는 신경망 출력

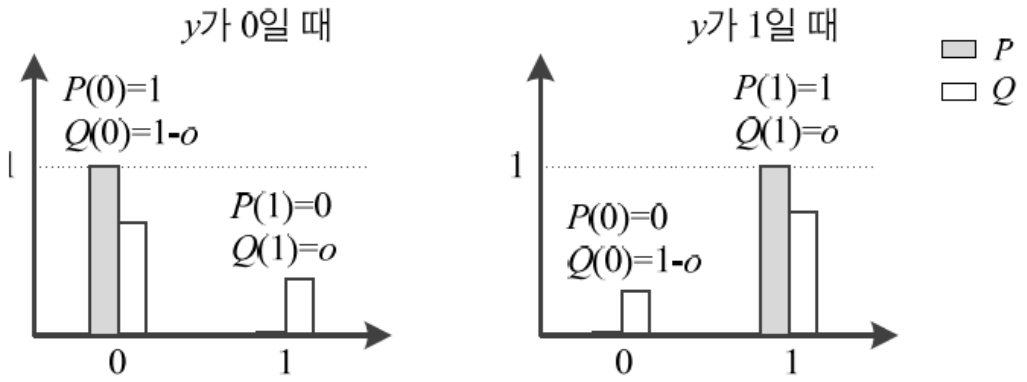


그림 5-3 레이블 y가 0일 때와 1일 때의 P와 Q의 확률분포

  - 확률분포를 통일된 수식으로 쓰면,

$$P(0) = 1 - y \quad Q(0) = 1 - o$$
$$P(1) = y \quad Q(1) = o$$

$y = 0$일 때, $y = 0$일 확률은 1이되어야 함. 즉, $P(y = 0) = 1$, 즉, $P(y = 0) = 1 - y$.

신경망 출력(시그모이드 함수)은 무조건 $P(y = 1|x)$일 확률을 출력함.
즉, $Q(y = 1) = o, Q(y = 0) = 1 - o$.

출처: 기계학습, 오일석 지음, 한빛아카데미(2017)

교차 엔트로피 (cross entropy) 목적함수

- **교차 엔트로피:** 앞의 평균제곱오차를 목적함수로 사용하면 더딘 학습 현상이 발생한다, 따라서 딥러닝에서는 교차 엔트로피(식 (2.47) : $H(P,Q) = -\sum_y P(y) log_2 Q(y)$)를 주로 사용한다.

  - 교차 엔트로피 식은 $H(P,Q) = -\sum_{y\in\{0,1\}} P(y)\log_2 Q(y)$ $(= E(-\log_2 Q(Y))$
  - $H(P,Q) = -\sum_{y\in\{0,1\}} P(Y=y)\log_2 Q(Y=y) = -\sum_{y\in\{0,1\}} P(y)\log_2 Q(y)$

    $= -P(Y=1) log Q(Y=1) - P(Y=0)\log Q(Y=0)$

    $= -y\log\hat{p} - (1-y)\log(1-\hat{p})$

    $= -(y\log\hat{p} + (1-y)\log(1-\hat{p}))$

  여기에서 $\hat{p}$ 은 모형으로 부터 나온 output을 말한다. 앞의 $o = \hat{p}$ 이다.

- $y = 0$일 때, $y = 0$일 확률은 1이되어야 함. 즉, $P(y=0) = 1$, 즉, $P(y=0) = 1 - y$.

- 신경망 출력(시그모이드 함수)은 무조건 $\hat{p} = o = P(y=1|x)$ 일 확률을 출력함.
- 즉, $Q(y=1) = o, Q(y=0) = 1 - o$

출처: 기계학습, 오일석 지음, 한빛아카데미(2017)

교차 엔트로피 (cross entropy) 목적함수

교차 엔트로피 목적함수

$$e = -(y \log_2 o + (1-y)\log_2(1-o)), \quad \text{이때, } o = \sigma(z) \text{이고} z = wx + b \qquad (5.4)$$

- 제구실 하는지 확인
  - y가 1, o가 0.98일 때 (예측이 잘된 경우)

    오류 $e = -(1 \log_2 0.98 + (1-1)\log_2(1-0.98)) = 0.0291$로서 낮은 값
  - y가 1, o가 0.0001일 때 (예측이 엉터리인 경우)

    오류 $e = -(1 \log_2 0.0001 + (1-1)\log_2(1-0.0001)) = 13.2877$로서 높은 값

※참고: 오류함수

엔트로피entropy $= -\sum_i p_i \log p_i$ 여기에서 p_i 는 i번째 사건이 일어날 확률

크로스 엔트로피cross entropy $= -\sum_i p_i \log q_i$ 여기에서 y_i 정답을 이야기함. 크로스 엔트로피의 의미는 p_i 와 q_i 의 값이 얼마나 서로 비슷한가이다. 지도학습에서는 보통 p_i 는 정답을 사용하기 때문에 현재 예측값이 얼마나 정답과 가까운지를 나타내게 된다. 두 값이 가까울수록 전체 식의 값은 줄어들어서 p_i 와 q_i 가 같아지면 식의 값은 최소가 된다.

출처: 기계학습, 오일석 지음, 한빛아카데미(2017)

교차 엔트로피 (cross entropy) 목적함수

- 식 (5.4)를 c개의 출력 노드를 가진 경우로 확장
  - 출력 벡터 $\mathbf{o} = (o_1, o_2, \cdots, o_c)^{\mathrm{T}}$인 상황으로 확장 ([그림 4-3]의 DMLP)

$$e = -\sum_{i=1,c} (y_i \log_2 o_i + (1 - y_i) \log_2 (1 - o_i)) \qquad (5.6)$$

  - y_i 는 실제 라벨 값이고(0 또는 1), o_i는 logistic function으로서 우리의 예측 값이 된다.

출처: 기계학습, 오일석 지음, 한빛아카데미(2017)

목적함수 (objective function)

- Conventional **objective functions** (to be minimized)
  - Cross-Entropy (for binary classification tasks)

$$E(f) = -\frac{1}{N}\sum_i \left[y_i \log\left(f(x_i)\right) + (1 - y_i) \log\left(1 - f(x_i)\right) \right]$$

  - Mean Squared Error (for regression tasks)

$$E(f) = \frac{1}{N}\sum_i \left(y_i - f(x_i)\right)^2$$

- Backpropagation

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$