13 확률과 통계 3

학습목표

- 조건부 확률의 개념을 이해한다.
- 베이즈의 정리가 인공지능에서 어떻게 활용되는지 알아본다.
- 전확률과 베이즈의 정리에 대해 알아본다.
- 주어진 표본 데이터로부터 가장 적합한 분포를 찾는 가능도함수에 대해 알아본다.
- 정보량에 대한 다양한 정의와 개념을 이해한다.

학습목차

13.1 조건부확률
13.2 전확률과 베이즈의 정리
13.3 가능도
13.4 정보량

13.1 조건부 확률

어떤 환자가 병원에서 진단을 받았는데 오진일 경우가 있다. 그러면 오진율은 얼마나 되는지에 관심이 생긴다. 간단한 채혈만으로 중대한 질환을 예측할 뿐만 아니라 정확도가 90% 이상이라는 연구 결과를 신뢰할 때, 실제 질병에 걸렸다고 진단받았을 경우 오진 가능성은 얼마나 되는지가 매우 중요하다. 따라서 검사 결과 병에 걸렸다고 판정받았는데 실제는 그 병에 걸려 있지 않을 가능성이 있는 반면에 검사 결과 병에 걸리지 않았다고 판정받았는데 실제는 그 병에 걸려있을 가능성도 있다. 이러한 문제는 뒤에서 배우게 될 베이지안 통계학의 핵심적인 내용이다.

베이지안 통계학을 이해하려면 조건부 확률의 개념을 정확히 알고 있어야 한다. 그래야 베이즈의 정리를 이해할 수 있으며 베이즈의 정리를 활용하여 머신러닝을 구현할 수 있다. 인공지능의 딥러닝에서는 근본적으로 확률값을 계산해야 하는 부분이 많으므로 확률론의 다양한 개념을 알고 있어야 한다. 더구나 조건부 확률에 대한 개념은 인공지능의 머신러닝에서 빼놓을 수 없는 중요 개념이다. 먼저 조건부 확률에 대해 학습하기로 하자.

조건부 확률

베이즈의 정리는 인공지능의 딥러닝 학습에 반드시 알고 있어야 하는 내용이다. 인공지능의 딥러닝 기법을 적용하기 위해 베이지안 통계의 개념을 명확하게 이해하고 있어야 한다는 뜻이다. 베이즈의 정리를 이해하려면 우선 조건부 확률이라는 개념을 먼저 알아야 한다. 이 절에서는 조건부 확률을 간단히 설명한다.

어떤 실험을 했을 때 우리는 표본 공간 S를 얻게 된다고 했다. 그리고 그 표본 공간의 부분 집합(또는 $\wp(S)$의 원소)을 사건이라고 했다. 사건 A와 B에 대하여 사건 B가 일어날 때 사건 A가 일어날 조건부 확률은 다음과 같이 표기한다.

$$P(A\,|\,B) \text{ 또는 } P_B(A)$$

이때 조건부 확률의 계산은 다음 공식을 이용하여 구한다.

$$P(A\,|\,B) = \frac{P(A \cap B)}{P(B)} \tag{13.1}$$

수식 (13.1)에서 표기한 확률 $P(A|B)$의 의미는 무엇일까? 수식 (13.1)에서 $P(B)$는 사건 B가 일어날 확률이고, $P(A \cap B)$는 두 사건 A와 B가 동시에 일어날 확률이다.

일단 $P(B) > 0$인 사건에 대해 조건부 확률의 의미를 좀 더 자세히 이해해보자. 이 의미를 확률론적 관점에서 표본 공간의 모든 부분 집합들을 B와 교집합 하여 생성된 집합을 생각해보자. 즉, 공간

$$\wp_B(S) = \{A \cap B | A \in \wp(S)\} \tag{13.2}$$

를 생각해보자. 그러면 모든 $A \in \wp(S)$에 대해 $A \cap B$는 다시 $\wp(S)$의 원소이며 $P(A \cap B)$도 잘 정의된다. 다음 그림 13.1에서 사건 B는 $P(B) > 0$인 사건이고, 사건 A와 A_3는 B와 교집합이 있으며, A_2는 B와 교차하지 않는 사건이다. 또한 A_1은 B에 완전히 포함되는 사건이다. 따라서 A_1은 위에서 정의한 $\wp_B(S)$에 속하는 원소이고, $A \cap B$와 $A_3 \cap B$도 $\wp_B(S)$의 원소이다. 하지만 A_2는 B와 교집합이 $\varnothing$이므로 $\wp_B(S)$에 들어가지 않는다.

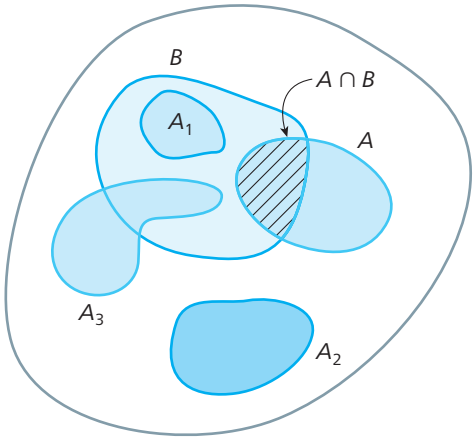


그림 13.1 $\wp_B(S)$의 기하학적 의미

$\wp_B(S)$는 사건 B를 중심으로 볼 때 $\wp(B)$라고 볼 수 있다(그림 13.2 참조).

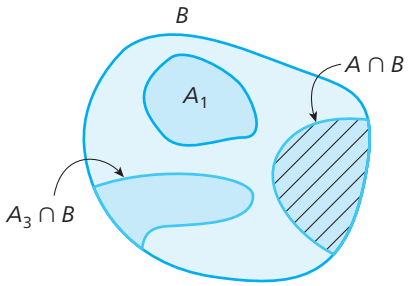


그림 13.2 사건 B 발생 전제하에 사건

이 공간에서 확률함수 Q는 다음과 같이 정의한다.

$$Q(A) = P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{13.3}$$

수식 (13.3)과 같이 정의된 확률함수

$$Q: \wp_B(S) \to [0, 1]$$
$$A \mapsto \frac{P(A \cap B)}{P(B)}$$

는 확률의 공리를 만족한다. 즉, 다음 세 가지가 성립한다.

(1) $0 \le Q(A) \le 1$

(2) $Q(S) = 1$

(3) $A_i \cap A_j = \varnothing (i \ne j) \Rightarrow Q\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} Q(A_i)$

확률함수 Q를 정의한 사람은 러시아 수학자 콜모고로프(Kolmogorov)이다. 이 확률함수가 바로 사건 B가 주어졌을 때 조건부 확률함수이다.

조건부 확률을 정의해보자. 사건 B가 발생했다는 조건에 사상 A가 일어나는 **조건부 확률**(conditional probability) $P(A|B)$는 다음과 같이 정의한다.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \qquad (P(B) > 0) \tag{13.4}$$

수식 (13.4)를 아래와 같이 나타낼 수도 있는데 이를 **곱셈 법칙**(multiplication theorem)이라고 한다.

$$\begin{aligned} P(A \cap B) &= P(B)P(A|B) \\ &= P(A)P(B|A) \end{aligned} \tag{13.5}$$

조건부 확률의 예를 하나 들어보자. 어느 개인이 특정일에 기침이 나는 확률이 5%라고 한다. 하지만 어떤 개인이 아프면 그 사람이 기침할 확률은 훨씬 높아질 것이다. 몸 상태가 좋지 않은 어떤 개인이 기침할 확률이 75%라고 하면

$$P(\text{기침하다}) = 5\%$$
$$P(\text{기침하다}|\text{몸 상태가 좋지 않다}) = 75\%$$

가 된다.

조건부 확률은 확률론에서 가장 중요하고 기본적인 개념이지만 조건부 확률을 정확히 이해하고 활용할 때는 많은 주의가 필요하다. 예를 들어 사건 A와 사건 B가 동시에 일어날 필요는 없다.

조건부 확률의 오해

통계의 오류라는 범주에 속하는 내용 중 조건부 확률을 오해하여 이상한 결과를 믿는 상황이 있다. 조건부 확률은 그 특성상 의미를 오해하기 쉬워서 통계로 계산된 수치는 사실인데 그 수치를 보고 잘못 받아들여서 의도치 않게 통계의 함정에 걸릴 수 있다. 유명한 예시 중 하나가 몬티 홀 문제(Monty Hall problem)이다. 원래 몬티 홀 문제는 미국 TV 게임쇼 '거래를 합시다(Let's Make a Deal)'에서 유래한 퍼즐 문제이다. 여기서는 이 함정을 이해하기 위해 다음과 같은 예를 들어보자.

자동차 사고로 사망한 사람의 40%는 안전띠를 매지 않았다고 한다. 반대로 말하면 자동차 사고로 사망한 사람의 60%는 안전띠를 매고도 죽었다는 뜻인데, 그렇다면 안전띠가 더 위험한 것 아닌가?

이 예시의 본질을 살펴보면 조건부 확률의 함정에 빠지기 쉽다는 말이다. 주어진 통계 자료가 말하는 것은 자동차 사고로 사망한 사람 중에서 안전띠를 맨 사람의 비율이 60%라는 사실이다. 그런데 이것을 안전띠를 맸을 때 자동차 사고로 사망할 확률이 더 높다는 식으로 오해하는 오류이다. 두 조건부 확률 $P(A|B)$와 $P(B|A)$가 서로 다르기 때문에 이런 오류가 발생한다.

이 오류를 반박해보자. 먼저 '전체 운전자 중에서 안전띠를 맨 사람의 비율'이 필요하다. 전체 운전자 중에서 95%가 안전띠를 매고 나머지 5%는 안전띠를 매지 않는다고 가정해보자. 또한 전체 운전자 1만 명 중 1명이 자동차 사고로 사망한다고 가정하자. 운전자가 안전띠를 매는 사건을 A라 하고 운전자가 자동차 사고로 사망하는 사건을 B라고 하면, 위 인용문은 사망했다는 전제하에 사망한 사람이 안전띠를 착용했다는 확률인 $P(A|B) = 0.6$이다. 또한 가정에 의하면 $P(A) = 0.95$, $P(B) = 0.0001$이다.

그렇다면 안전띠를 맸을 때 자동차 사고로 사망할 확률은 얼마일까. 이를 계산해 보면 다음과 같다.

$$P(B|A) = \frac{P(A \cap B)}{P(B)} = \frac{P(A|B)P(B)}{P(A)} = \frac{0.6 \times 0.0001}{0.95} = 0.000063$$

이 값은 약 16,000명당 1명에 해당하는 값이다. 이번에는 안전띠를 매지 않았을 때 자동차 사고로 사망할 확률을 계산하면 다음과 같다.

$$P(B|A^c) = \frac{P(A^c \cap B)}{P(A^c)} = \frac{P(A^c|B)P(B)}{P(A^c)} = \frac{0.4 \times 0.0001}{0.05} = 0.0008$$

이 값은 1,250명당 1명에 해당하는 값이다. 즉, 예상했듯이 안전띠를 매지 않을 때 자동차 사고로 사망할 확률이 안전띠를 맸을 때보다 12배 이상 큰 것을 알 수 있다.

독립사상

어느 표본 공간의 사건 A와 사건 B가 주어졌을 때, B가 발생했을 경우 A가 발생할 조건부 확률 $P(A|B)$는 $P(A)$와 같은 값을 가질 수도 있고 그렇지 않을 수도 있다. 만약

$$P(A|B) = P(A) \tag{13.6}$$

가 성립하면 두 개의 사건 A와 B는 **독립**(independent)인 사건이라고 한다. 이 경우에 어느 한 사건에 대해 정보를 알고 있다고 하더라도 그 사실은 다른 사건이 일어날 확률에 어떠한 영향도 미치지 못한다.

수식 (13.6)과 다른 방식으로 다음과 같이 독립사상을 정의하기도 한다.

$$P(A \cap B) = P(A)P(B) \tag{13.7}$$

수식 (13.7)이 성립하면 두 사건 A와 B는 독립이라고 한다. 수식 (13.7)은 앞의 (13.6)에서 정의한 독립과 동치이다. 그 이유를 알아보자. $P(A \cap B) = P(A)P(B)$라고 하면

$$P(A) = \frac{P(A)P(B)}{P(B)} = \frac{P(A \cap B)}{P(B)} = P(A|B)$$

이다. 반대로 $P(A|B) = P(A)$라고 하면

$$P(A \cap B) = P(A|B)P(B) = P(A)P(B)$$

이다. 따라서

$$P(A \cap B) = P(A)P(B) \Leftrightarrow P(A) = \frac{P(A \cap B)}{P(B)} = P(A|B)$$

이다. 또는

$$P(A \cap B) = P(A)P(B) \Leftrightarrow P(B) = P(B|A)$$

이다. 결국 사건 B의 발생이 사건 A의 확률에 영향을 전혀 미치지 못하고 그 반대도 성립한다.

독립사상의 예

독립사상의 예를 들어보자. 제비뽑기에서 10개 중 2장의 제비가 당첨 제비라고 가정해보자. 앨리스가 먼저 한 장을 뽑아서 확인한 후 다시 집어넣는다고 하자. 그다음에 밥도 제비를 한 장 뽑아 확인한 후 다시 집어넣는다고 하자. 한 번만 시행할 경우에 표본 공간은 다음과 같다.

$$S = \{(\text{당첨, 당첨}), (\text{당첨, 실패}), (\text{실패, 당첨}), (\text{실패, 실패})\}$$

사건 A를 앨리스가 당첨되는 사건이라고 하고, 사건 B를 밥이 당첨되지 않는 사건이라고 하자. 그러면

$$A = \{(\text{당첨, 당첨}), (\text{당첨, 실패})\}$$

이고

$$B = \{(\text{당첨, 실패}), (\text{실패, 실패})\}$$

이다. 각각 확률을 살펴보면

$$P(A) = P((\text{당첨, 당첨})) + P((\text{당첨, 실패})) = \frac{2}{10}\frac{2}{10} + \frac{2}{10}\frac{8}{10} = \frac{2}{10}$$

이고

$$P(B) = P((\text{당첨, 실패})) + P((\text{실패, 실패})) = \frac{2}{10}\frac{8}{10} + \frac{8}{10}\frac{8}{10} = \frac{8}{10}$$

이다. 앨리스는 당첨되고 밥은 실패할 확률은

$$P(A \cap B) = \frac{2}{10}\frac{8}{10} = \frac{16}{100}$$

이다. 앨리스가 당첨되었다고 했을 때 밥이 실패할 확률은

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{16/100}{2/10} = \frac{8}{10} = P(B)$$

이다. 따라서 $P(B|A) = P(B)$이므로 두 사건 A와 B는 독립사건이다.

위와 같은 제비뽑기 실험을 10회 반복했을 때 앨리스가 당첨 제비를 뽑은 횟수를 확률변수 X라고 하고, 밥이 당첨 제비를 뽑은 횟수를 Y라고 하자. 그러면 $X \sim B(10, 1/5)$이고, $Y \sim B(10,$

1/5)이다. 각각 확률변수를 구하면 다음과 같다.

$$P(X=x)=\binom{10}{x}\left(\frac{2}{10}\right)^{x}\left(\frac{8}{10}\right)^{10-x},\ x=0,\ 1,\ \ldots,\ 10$$

$$P(Y=y)=\binom{10}{y}\left(\frac{2}{10}\right)^{y}\left(\frac{8}{10}\right)^{10-y},\ y=0,\ 1,\ \ldots,\ 10$$

두 개의 확률변수의 독립에 대해서는 정의하지 않지만, 위에 예제로 들은 두 확률변수는 독립이라고 한다. 그 이유는 X가 갖는 값에 상관없이 Y의 값의 확률이 결정되기 때문이다. 이를

$$P(Y|X)=P(Y),\ \text{또는}\ P(X|Y)=P(X)$$

라고 나타내기도 한다. 여기서는 더 이상 두 개의 확률변수에 대한 독립성을 전개하지 않기로 한다. 위의 예제에서 앨리스가 어떤 제비를 뽑던 밥은 항상 같은 조건에서 제비를 뽑을 수 있다는 뜻이 바로 독립이라는 의미이다.

조건부 확률 $P(A|B)$와 $P(B|A)$는 완전히 다른 정의이다. 예를 들어 어떤 사람이 전염성이 강한 질병을 앓고 있다고 가정해보자. 그러면 검사를 했을 때 90%의 확률로 양성 반응이 나타난다. 이 경우 질병에 걸렸다는 사건을 B라고 했을 때, 검사 결과가 양성으로 나오는 사건을 A라고 하자. 그러면 조건부 확률

$$P(A|B)=90\%$$

이다. 반대로 어떤 사람이 검사 결과로 양성이 나왔다고 가정해보자. 그러나 실제로 이 질병에 걸렸을 확률은 15%밖에 되지 않는다고 한다. 그 이유는 검사 시약의 성능이 많이 떨어지기 때문이다. 이 경우에 검사에서 양성이 나오는 사건을 A라고 하고, 질병에 걸린 사건을 B라고 하자. 그러면 조건부 확률

$$P(B|A)=15\%$$

이다. 두 가지 조건부 확률은 혼동하지 말기를 바란다. 다만 베이즈의 정리에서는 조건부 확률을 바꾸어 쓸 수 있다. 이에 대해서는 뒤의 베이즈의 정리에서 더 자세히 알아보기로 한다.

조건부 확률의 예

조건부 확률의 예를 들어보자. 특정 전염병으로 사망한 사람 중에서 해당 질병에 대한 백신을 접종받았던 사람이 30%라고 한다. 다시 말해서 사망한 사람 중 30%는 백신 접종을 받았는데도 사망한 것이다. 그렇다면 백신 접종을 하는 게 안전한 걸까? 이 문제를 조건부 확률의 개념으로 풀어보자. 어떤 사람이 백신 접종을 받는 사건을 A라고 하자. 해당 전염병으로 사망하

는 사건을 B라고 하자. 해당 전염병으로 사망한 사람 중에서 30%는 백신 접종자이므로 $P(A|B) = 0.3$이다. 사건 A의 여사건인 A^c은 어떤 사람이 백신 접종을 받지 않는 사건이다. 그러면 $P(A^c|B) = 0.7$이다. 해당 질병으로 사망할 확률이 천명 중 한 명인 $P(B) = 0.001$이라고 가정해 보자. 또한 사람들이 백신 접종을 받을 확률 $P(A) = 0.85$라고 해보자. 그러면

$$P(B|A) = \text{백신 접종을 받았는데 해당 전염병으로 사망할 확률}$$

은 어떻게 구하면 될까? 조건부 확률 공식을 이용해서 구해보자.

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A|B)P(B)}{P(A)} = \frac{(0.3)(0.001)}{0.85} = \frac{0.0003}{0.85} \simeq 0.000353$$

백신 접종을 받았는데 해당 전염병으로 사망할 확률은 10만 명에 약 35명 정도가 된다는 의미이다. 백신 접종을 받지 않았는데 해당 전염병으로 사망할 확률은 어떻게 될까?

$$P(B|A^c) = \frac{P(B \cap A^c)}{P(A^c)} = \frac{P(A^c|B)P(B)}{P(A^c)} = \frac{(0.7)(0.001)}{0.15} = \frac{0.0007}{0.15} \simeq 0.00467$$

백신 접종을 받지 않았는데 전염병으로 사망할 확률은 10만 명 중에서 467명이 된다는 의미이다.

다른 조건부 확률의 예를 들어보자. 2개의 주사위를 동시에 던지는 실험을 한다고 해보자. 사건 A = {눈의 합이 8}이고 사건 B = {두 눈이 같다}라고 하자. 조건부 확률 $P(B\ A)$를 구해보자. 우선 이 실험의 표본 공간 S와 각 사건을 집합으로 표시해보자.

$$S = \{(i, j) \mid i, j = 1, 2, 3, 4, 5, 6\}$$
$$A = \{(i, j) \in S \mid i + j = 8\} = \{(2, 6), (3, 5), (4, 4), (5, 3), (6, 2)\}$$
$$B = \{(i, j) \in S \mid i = j\} = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6)\}$$

이고

$$A \cap B = \{(4, 4)\}$$

이므로 구하는 조건부 확률 $P(B|A)$는 다음과 같다.

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{\frac{1}{36}}{\frac{5}{36}} = \frac{1}{5}$$

곱셈 공식의 확장

곱셈 법칙을 수식 (13.5)에 나타냈다. 이 개념을 좀 더 확장해보자. $n+1$개의 사건 $A_0, A_1, \ldots,$ A_n에 대하여 $P(A_0 \cap A_1 \cap \ldots \cap A_{n-1}) > 0$이면

$$\begin{aligned} &P(A_0 \cap A_1 \cap \ldots \cap A_n) \\ &= P(A_0)P(A_1|A_0)P(A_2|A_0 \cap A_1) \ldots P(A_n|A_0 \cap A_1 \cap \ldots \cap A_{n-1}) \end{aligned} \tag{13.8}$$

이 성립한다. 수식 (13.8)을 수학적 귀납법으로 증명할 수 있다.

$$n = 1\text{일 경우 } P(A_0) > 0$$

이고 곱셈 법칙인 수식 (13.5)에 의해

$$P(A_0 \cap A_1) = P(A_0)P(A_1|A_0)$$

이 성립한다. $n = \mathrm{k}$일 때 성립한다고 가정하고 $n = k+1$일 때 수식 (13.8)이 성립함을 보이면 된다. $A_0 \cap \ldots \cap A_k$를 하나의 사건으로 보면

$$\begin{aligned} &P(A_0 \cap A_1 \cap \ldots \cap A_k \cap A_{k+1}) \\ &= P(A_0 \cap A_1 \cap \ldots \cap A_k)P(A_{k+1}|A_0 \cap A_1 \cap \ldots \cap A_k) \\ &= P(A_0)P(A_1|A_0) \ldots P(A_n|A_0 \cap \ldots \cap A_{k-1}) \cdot P(A_{k+1}|A_0 \cap A_1 \cap \ldots \cap A_k) \end{aligned}$$

가 성립한다. 따라서 $n = k+1$일 때도 수식 (13.8)이 성립한다. 따라서 수식 (13.8)은 모든 자연수 n에 대하여 성립한다.

곱셈 공식이 적용되는 예를 들어보자. 주머니에 5개의 붉은색 공, 4개의 파란색 공이 들어있다. 이 주머니에서 임의로 1개의 공을 꺼내서 다음 색을 확인하고 다시 주머니에 넣는다. 그리고 꺼낸 공과 같은 색의 공 2개를 추가로 주머니에 넣는다. 이런 실험을 세 번 반복한다. 이때 세 번 모두 붉은색 공을 꺼낼 확률을 구해보자.

이 실험의 표본 공간 $S = \{(x, y, z) | x, y, z = Red \text{ 또는 } Blue\}$이다. i번째 시행에서 붉은색 공을 꺼내는 사건을 A_i라고 하면

$$\begin{aligned} A_1 &= \{(Red, y, z) | y, z = Red \text{ or } Blue\} \\ A_2 &= \{(x, Red, z) | x, z = Red \text{ or } Blue\} \\ A_3 &= \{(x, y, Red) | x, y = Red \text{ or } Blue\} \end{aligned}$$

이다. 세 번 모두 붉은색 공을 꺼내는 사건은 $A_1 \cap A_2 \cap A_3$이고, 여기에 확장된 곱셈 법칙을 적용해보면 다음과 같다.

$$P(A_1 \cap A_2 \cap A_3) = P(A_1)P(A_2|A_1)P(A_3|A_1 \cap A_2)$$

이다.

$$P(A_1) = \frac{5}{5+4} = \frac{5}{9}$$

$$P(A_2|A_1) = \frac{5+2}{5+4+2} = \frac{7}{11}$$

$$P(A_3|A_1 \cap A_2) = \frac{5+2+2}{5+4+2+2} = \frac{9}{13}$$

이므로

$$P(A_1 \cap A_2 \cap A_3) = \frac{5}{9}\frac{7}{11}\frac{9}{13} = \frac{35}{143}$$

이다. 이제 베이즈의 정리에 대해 알아보자.

13.2 전확률과 베이즈의 정리

베이즈의 정리는 인공지능뿐만 아니라 다양한 분야에서 이용되는 매우 유용한 개념이다. 앞에서 조건부 확률에 관해 학습하였고 이 절에서는 베이즈의 법칙을 설명하기로 한다. 대부분의 머신러닝과 패턴인식을 다루는 학술적 문서의 전반부는 대부분 베이즈의 정리와 베이즈의 분류기에 관한 내용으로 전개된다. 그 이유는 크게 두 가지로 말할 수 있다. 첫 번째는 베이즈의 분류기가 다른 머신러닝 방법들보다 상대적으로 알고리즘이 간단하지만, 현실 세계의 많은 문제를 효과적으로 풀 수 있다는 장점이 있기 때문이다. 두 번째는 베이즈의 정리를 통해 확률론에 대한 보다 확고한 개념을 잡을 수 있기 때문이다.

베이지안 통계

베이지안 통계에서 데이터를 관측하기 전의 지식이나 경험과 연관된 정보를 토대로 가설이 올바를 확률을 예상해둔다. 이러한 확률을 **사전 확률**(prior probability)이라 한다. 그다음 관측한 데이터를 사용하여 사전에 예측했던 확률을 갱신한다. 이 확률을 **사후 확률**(posterior probability)이라고 한다. 새로운 데이터를 관측할 때마다 그것을 반영하여 다시 추정하고 좀 더 정확한 사후 확률을 구한다. 이를 **베이즈 갱신**(Bayesian update)이라고 한다.

대표적인 응용사례로 알파고(AlphaGo)와 인간의 바둑대국에서 알파고는 대량의 기보를 분석하여 다양하고 획기적인 패턴들을 식별하고, 바둑돌을 최적의 위치에 놓아 승리할 수 있도록 설계된 알고리즘으로 무장하였다. 여기에 적용한 기법이 바로 베이지안 통계에 기초한 머신러닝이었다. 바로 인공지능의 딥러닝 기법이다.

또 다른 예를 들어보자. 자율주행이 주요한 이슈이다. 자율주행은 0부터 5까지 6개의 레벨로 구분한다. 레벨5란 완전자율주행을 뜻한다. 실제 레벨3부터 인공지능이 운전을 주도하고 인간은 보조자 역할을 한다. 레벨5는 운전자가 아예 불필요한 단계이다. 완전자율주행을 위해서는 인공지능 알고리즘이 스스로 도로 위 상황 자료를 수집하고 분석하여 어떻게 주행할지를 판단할 수 있어야 한다. 상황을 인지하는 알고리즘 자체는 그렇게 어렵지 않게 만들 수 있다. 촬영된 영상과 레이더 및 각종 센서를 이용하여 도로나 주변 상황을 감지할 수 있다. 중요한 것은 인간의 주 영역이라고 여겨졌던 판단 능력이다. 감지된 상황을 두고 어떠한 판단을 내릴 수 있는지가 중요하다.

일단 판단했다면 이는 사전 확률을 계산했다는 의미이고 그에 따른 행동에 돌입한다. 하지만

상황이 원래 계획한 대로 진행되지 않으면 새로 입력받은 데이터를 기반으로 사후 확률값을 계산한다. 그리고 이를 반영하여 원래의 행동을 수정할 수 있어야 한다. 이는 사람이 상황을 인지하고 순간순간 상황판단을 통해 실수를 교정하여 다음 행동에 반영함으로써 큰 위험을 피해 가는 것과 같은 프로세스이다. 과학자들은 이런 능력을 인공지능 알고리즘이 가질 수 있도록 다양한 연구를 하고 있다. 그중 하나가 바로 베이즈의 정리를 기반으로 하는 베이즈의 갱신 방법이다.

전확률

먼저 하나의 표본 공간 S가 주어졌을 때 이에 대한 **분할**(partition)이 무엇인지 말해보자. 하나의 사건들의 집합인 $A_1, A_2, \ldots, A_n$이 S의 한 분할이라는 뜻은 다음 두 가지를 만족하는 경우를 말한다.

(1) $A_i \cap A_j = \varnothing (i \neq j)$

(2) $\bigcup_{i=1}^{n} A_i = S$

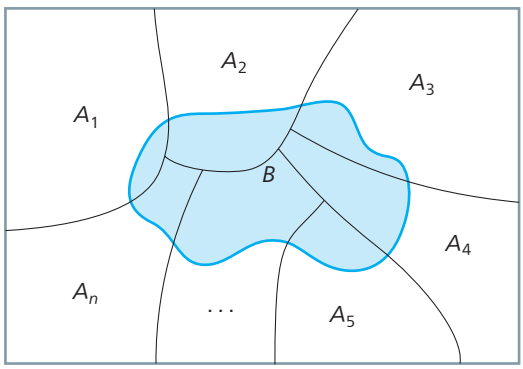


그림 13.3 분할의 개념

그림 13.3에 분할의 개념을 그림으로 나타냈다. 표본 공간 S의 한 분할 $\{A_1, \ldots, A_n\}$이 주어졌다고 하자. 임의의 사건 B에 대하여

$$B = S \cap B = \left(\bigcup_{i=1}^{n} A_i\right) \cap B = \bigcup_{i=1}^{n} (A_i \cap B)$$

이다. 또한 $A_i \cap B (i = 1, \ldots, n)$은 서로 교집합을 갖지 않는다. 따라서 확률의 덧셈 규칙에 따라,

$$P(B) = P(A_1 \cap B) + P(A_2 \cap B) + \ldots + P(A_n \cap B)$$

이다. 각각의 항에 확률의 곱셈 법칙을 적용하면

$$P(B) = P(A_1)P(B|A_1) + P(A_2)P(B|A_2) + \ldots + P(A_n)P(B|A_n) \tag{13.9}$$

이 성립한다. 이를 **전확률 공식**(total probability)이라고 한다.

베이즈의 정리

이제 베이즈의 정리를 기술할 때가 되었다. 표본 공간 S의 한 분할 $\{A_1, \ldots, A_n\}$이 주어졌다고 하자. 각 k에 대하여

$$P(A_k|B) = \frac{P(A_k \cap B)}{P(B)} \tag{13.10}$$

가 성립한다. 수식 (13.10)에 사건 B에 대한 전확률 공식 (13.9)와 곱셈의 법칙

$$P(A_k \cap B) = P(A_k)P(B|A_k) \tag{13.11}$$

를 적용해보자.

$$P(A_k|B) = \frac{P(A_k)P(B|A_k)}{\sum_{i=1}^{n} P(A_i)P(B|A_i)} \tag{13.12}$$

를 얻는다. 수식 (13.12)가 바로 **베이즈의 정리**(Bayes' Theorem)이다.

조건부 확률에서는 새로운 정보를 획득했을 때 확률을 수정하여 새로운 값을 얻을 수 있다. 어떤 실험에서 나온 결과를 이용하여 사전 확률을 개선할 수 있다. 이렇게 개선된 확률을 사후 확률이라고 했다. 확률값을 개선할 수 있도록 해주는 것이 바로 베이즈의 정리이다. 그 절차는 다음 그림 13.4에 나타낸 것과 같이 4단계로 나타낼 수 있다.

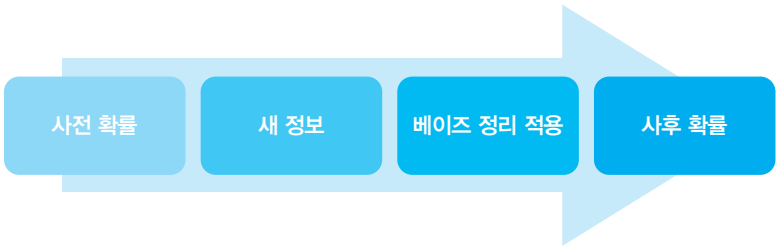


그림 13.4 베이즈의 정리 적용단계

수식 (13.12)에서 하나의 사건만 있다면

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{13.13}$$

로 표현할 수 있다. $P(A)$가 A의 사전 확률이고 $P(A|B)$는 A의 사후 확률이고, $P(B|A)$는 뒤에서 학습하게 될 **가능도**(likelihood)이며, $P(B)$는 B의 사전 확률이다. 수식 (13.13)을 이용하여 하나의 예제에 베이즈의 정리가 어떻게 적용되는지 알아보자.

베이즈의 정리 예

평소에 70%의 확률로 거짓말을 하는 사람이 있다고 하자. 우리는 90%의 정확도를 가진 거짓말 탐지기를 사용하여 이 사람의 말이 거짓인지 판단하는 실험을 해보자. 실제로 이 문제는 거짓말 탐지기의 결과로 어떤 사람의 말이 사실인지 거짓인지 판단하는 문제이다. 이를 수식 (13.13)과 연관 지어 설명해보자.

사건 A는 이 사람의 말이 거짓말인 사건이다. 그래서 사건 A는 우리가 알고 있는 사전 지식으로 이 사람이 평소에 거짓말할 확률이 0.7이라는 뜻이다. 즉, $P(A) = 0.7$이다. 보통 가능도 또는 우도라고 하는 $P(B|A)$도 일반적으로 알려진 사실이다. 여기서는 이 사람이 거짓말을 했을 때 실제로 그 말이 거짓말일 확률이다. 즉, 거짓말 탐지기의 정확도가 바로 가능도이다. 가능도란 뒤에서 자세히 학습하겠지만 거짓말 탐지기가 정확하게 거짓말을 거짓말이라고 판정할 수 있는 정도를 의미한다. 따라서 여기서 말하는 가능도는 $P(B|A) = 0.9$이다.

사건 B는 거짓말 탐지기가 거짓이라고 판정하는 사건이다. 그래서 사전 확률 $P(B)$는 거짓말 탐지기가 거짓이라고 판정할 확률이다. 이 경우는 두 가지가 있을 수 있다. 이 사람이 진실을 말했는데 거짓말 탐지기가 거짓이라고 판정하는 확률과 거짓말을 했는데 거짓말 탐지기가 거짓이라고 판정하는 확률이다. 다음과 같이 두 경우의 확률을 더하면 $P(B)$를 구할 수 있다.

$$\begin{aligned} P(B) &= P(B|A)P(A) + P(B|A^c)P(A^c) \\ &= (0.9)(0.7) + (0.1)(0.3) \\ &= 0.63 + 0.03 \\ &= 0.66 \end{aligned}$$

이제 이 사람의 말이 거짓일 사후 확률 $P(A|B)$를 베이즈의 정리를 이용하여 구해보면

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{(0.9)(0.7)}{0.66} = \frac{0.63}{0.66} \simeq 0.9545$$

가 된다. 그러므로 이 사람의 말이 거짓말일 확률은 95.45% 정도라고 할 수 있다. 이 결과의 의미를 그림 13.4에서 언급했던 '사전 확률 → 새 정보 → 베이즈의 정리 적용 → 사후 확률 단계'

를 가지고 좀 더 자세히 살펴보자. 이 사람이 어떤 말을 했을 때 그 말이 거짓말일 확률이 70%라고 알고 있다. 그런데 이 사람이 자기가 한 말이 진실이라고 주장한다고 해보자. 그래서 이를 검증하기 위해 거짓말 탐지기를 이용해서 판단해보기로 한 것이다. 이 거짓말 탐지기는 오랜 데이터를 축적해서 약 90%의 정확도를 가지고 있다고 알려진 기계이다. 거짓말 탐지기가 실제로 판정하는 사례는 다음 표와 같이 총 네 가지가 있다.

정치인 / 탐지기	A^c: 진실을 말했다	A: 거짓을 말했다
B^c: 참이라고 판정한다	$P(B^c \mid A^c)$ = 진실을 말했는데, 참이라고 판정할 확률	$P(B^c \mid A)$ = 거짓을 말했는데, 참이라고 판정할 확률
B: 거짓이라고 판정한다	$P(B \mid A^c)$ = 진실을 말했는데, 거짓이라고 판정할 확률	$P(B \mid A)$ = 거짓을 말했는데, 거짓이라고 판정할 확률

즉, 거짓말을 했을 때 거짓말이라고 판정하는 확률이 90%라는 뜻이다. 거짓말 탐지기로 이 사람의 말을 검사했더니 거짓이라고 판정이 났다. 그래서 이 사람이 거짓말을 했을 확률인 사후 확률이 95.45%로 업데이트된다.

즉, 베이즈의 정리를 통해 알 수 있는 사실은 관찰을 통해 새로운 정보를 획득하면 사전 확률보다 개선된 사후 확률로 업데이트할 수 있다는 점이다.

베이즈의 정리 활용 예

대체로 우리가 배우는 통계학은 빈도수를 기반으로 정립된 이론이다. 이와는 매우 다른 특징을 가지고 있는 이론이 베이즈의 정리이다. 새로운 정보, 관찰 및 결과 등이 나타나면 이를 반영하여 새로운 확률을 찾는다는 점에서 일반 통계학의 확률계산과 다르다. 이러한 과정을 거쳐 진실 혹은 원하는 답에 접근해간다. 어떤 사실이 바뀌면 확률값이 바뀌고 그에 따라 판단과 전략도 바뀐다. 이는 매우 과학적이며 타당성이 있고 실용적인 면에서도 적용 범위가 매우 넓다. 특히 자료가 많지 않으면 타당한 결론을 도출하는 데 매우 유용한 통계적 기법이다.

우리나라 대검찰청 자료에 의하면 폭행상해 사범 기소율이 22%라고 한다. 실제 범죄를 저지른 사람이 기소될 때도 있지만 범죄를 저지르지 않았는데 기소될 경우도 있다. 또한 실제 범죄를 저질렀고 기소될 때도 있고 범죄를 저질렀지만 기소되지 않을 때도 있다.

여기서는 검찰이 기소한 사람들에 대해서만 고려해보자. 검사의 기소 데이터와 축적된 데이터를 통해 기소된 사람 중에서 약 85%만이 실제 범인인 것을 알게 되었다.

즉, 기소된 사람 중에서 15%는 범죄를 저지르지 않은 사람이다. 또한 재판하는 판사의 역량을 축적된 데이터를 통해 검토한 결과 기소된 실제 범죄자에게 유죄 판결을 내리는 확률이 80%라고 한다.

빈도주의와 베이즈주의[1]

확률론을 연구하는 학자를 '빈도주의자(Frequentism)'와 '베이즈주의자(Bayesianism)'로 구분한다. 빈도주의자에게 확률은 특정 사건을 시행할 때 A가 나타날 경우의 수를 이 사건이 일어나는 모든 경우의 수로 나눠 구하는 것을 의미한다. 예를 들어 주사위를 던지면 1부터 6까지 여섯 가지 경우의 수가 나오는데 이때 1이 나올 경우의 수는 한 가지뿐이다. 결국 주사위를 1번 던질 때 1이 나올 확률은 6분의 1이라고 할 수 있다. 빈도적 확률 개념을 적용한 대화형 AI라면 어떤 상황을 입력해도 주사위를 던져 1이 나올 확률은 6분의 1이라고 대답할 것이다.

반면에 베이즈주의자는 이와 다르게 본다. 무한히 주사위를 던졌을 때 1이 나올 확률이 6분의 1인 것은 분명하다. 그런데 어떤 사람이 주사위를 10번 던졌는데 경험적으로 1이 7번 나왔다고 주장할 수 있다. 이 사람에게 1이 나올 확률은 10분의 7인 것이다. 만약 주사위를 10번 던진 사람과 20번 던진 사람이 있다면, 이 두 사람은 실제로 경험한 결과에 따라 1이 나올 확률을 다르게 느낀다. 교과서에서는 이를 사건 A가 일어났을 때 사건 B가 일어날 확률이라는 의미를 가진 조건부 확률의 개념으로 설명하고 있다.

조건부 확률은 베이즈의 정리나 베이즈 확률이라고도 한다. 이를 적용하면 사건에 영향을 미칠 수 있는 상황을 최대한 고려할 수 있다. 여러 상황을 가정한 AI의 알고리즘일수록 연산 과정이 복잡해질 수밖에 없다. 수학자나 컴퓨터공학자는 빈도적 확률 개념과 베이즈 확률 개념을 이용해 AI의 머신러닝 알고리즘을 설계한다.

판사가 기소된 사람에게 유죄라고 판정했을 때 이 사람이 실제로는 범죄를 저질렀을 사후 확률은 얼마나 되는지 알아보자. 다음과 같이 사건을 정의해보자.

A = 기소된 자가 범죄를 저질렀다.
B = 판사가 기소된 자를 무죄라고 판정한다.

다음과 같은 표로 네 가지 결과를 분류할 수 있다. 기소된 사람 입장에서 억울한 경우는 범죄를 저지르지 않았는데 유죄 판결을 받는 경우이다. 이런 경우가 발생하지 않도록 $P(B^c|A^c)$을 최소화할 필요가 있다. 또한 사회 정의라는 측면에서 볼 때 폭행 피해자 입장에서 억울한 경우는 기소된 자가 실제로 범죄를 저질렀는데 무죄 판결을 받는 경우이다. 이런 경우가 발생하지 않도록 $P(B|A)$를 최소화할 필요가 있다.

기소된 자 / 판사	A^c: 범죄를 저지르지 않았다	A: 범죄를 저질렀다
B^c: 유죄 판결을 받았다	$P(B^c\|A^c)$ = 범죄를 저지르지 않았는데 유죄 판결을 받을 확률	$P(B^c\|A)$ = 범죄를 저질렀고 유죄 판결을 받을 확률
B: 무죄 판결을 받았다	$P(B\|A^c)$ = 범죄를 저지르지 않았고 무죄 판결을 받을 확률	$P(B\|A)$ = 범죄를 저질렀지만 무죄 판결을 받을 확률

[1] 출처: 동아사이언스;https://www.dongascience.com/news.php?idx = 46883

사전 확률은 무엇인지 생각해보자. 사전 확률은 $P(A) = 0.85$이다. 우도인 $P(B^c|A) = 0.8$이 판사가 기소된 자가 실제 범죄자인데 유죄 판결을 내리는 확률이다. 그래서 판사가 유죄라고 판결하는 확률은 기소된 자가 실제 범죄를 저질렀는데 유죄 판결을 내리는 경우와 기소된 자가 범죄를 저지르지 않았는데 유죄 판결을 내리는 경우의 합으로 계산해야 한다.

$$\begin{aligned} P(B^c) &= P(B^c|A)P(A) + P(B^c|A^c)P(A^c) \\ &= (0.8)(0.85) + (0.2)(0.15) \\ &= 0.68 + 0.03 \\ &= 0.71 \end{aligned}$$

그러면 기소된 자에게 판사가 유죄라고 판결을 내렸는데 그가 실제로 범죄자일 확률은 얼마나 될까?

$$P(A|B^c) = \frac{P(B^c|A)P(A)}{P(B^c)} = \frac{(0.8)(0.85)}{0.71} = \frac{0.68}{0.71} \simeq 0.9577$$

이 된다. 즉, 판사가 기소된 자를 유죄라고 판결했다면 실제 범인일 확률이 96% 정도 되며, 기소된 자가 유죄 판결을 받았다 하더라도 약 4% 정도의 사람은 실제 범인이 아니라는 뜻이다. 사전 확률 85%에 비해 사후 확률이 96%로 높아지는 것을 알 수 있다.

참고로 위의 사실을 언뜻 보면 100명 중에서 4명이 죄를 짓지 않았는데도 유죄 판결을 받는 것처럼 보이지만 사실은 기소된 사람 중에서 판사가 판결한 것이므로 실제로 무죄인 사람이 기소되는 경우가 1.5% 정도 된다고 할 때 100명 중에서 1.5명이 무죄이면서 기소된다는 말이고 그 중에서 0.04%가 유죄 판결을 받는다는 뜻이므로 죄를 짓지 않았는데 기소도 되고 유죄 판결까지 받는 경우는 100명 중 0.06명이라는 의미이다. 즉, 유죄 판결을 받은 만 명 중 6명 정도가 죄를 짓지 않았는데 유죄 판결을 받았다고 할 수 있으며 이 확률이 바로 $P(B^c|A^c) = 0.0006$이다.

베이즈의 정리 구현

베이즈의 정리를 정리하기 위해서 추론 과정을 파이썬 코드로 확인해보자.

예제 13.1

가수일까, 직장인일까?

철수라는 사람이 있다. 철수는 춤을 잘 추고 노래 부르는 걸 좋아하며 사람들의 관심을 즐길 줄 아는 사교성을 가지고 있다. 여러분은 철수가 가수라고 생각하는가? 아니면 직장인이라고 생각하는가?

이 설명을 보면 철수는 가수일 가능성이 더 높다고 생각하고 많은 사람이 이에 동의할 것이다. 하지만 여기에 직업적인 배경분포를 가정해보자. 가령, 남자 가수와 남자 직장인의 비율이 1:24라고 한다면 철수는 통계적으로 직장인일 가능성이 더 크다.

이러한 오류를 해결하기 위해서 베이지안 추론을 적용해보자. 계산의 편의성을 위해 이 세상에 가수와 직장인이라는 두 가지 직업만 존재한다고 가정하자. 그리고 직장인이 가수보다 24배 많다. 이에 대해 앞서 배운 사전 확률, 사후 확률을 파악하고 계산해보자.

철수가 가수일 사건을 A라고 하자. 철수에 대한 정보가 없다면 $P(A) = 1/25 = 0.04$가 사전 확률이 된다. 이제 철수의 친구로부터 그에 대한 정보를 얻었다고 가정하자. 이를 X라고 하자. 우리가 알고 싶은 확률은 철수의 정보를 안다는 가정에서 철수가 가수일 확률인 $P(A|X)$이다. 이를 계산하기 위해, 베이즈의 정리를 적용해보자.

$$P(A|X) = P(X|A)P(A)/P(X)$$

$P(X|A)$는 철수가 가수일 경우, 그의 친구가 앞서 설명한 바와 같이 그를 서술할 가능성이다. 아마 거의 1에 가까울 것이다. 우리는 90%라고 하자.

$P(X)$는 누군가가 그의 친구가 그를 설명한 대로 진술할 가능성이다. 바로 계산하기 어려우므로 다음과 같은 논리를 구성해서 구해보자.

$$P(X) = P(X \cap A) + P(X \cap A^c) = P(X|A)P(A) + P(X|A^c)P(A^c)$$

$P(A^c)$은 철수가 가수가 아닐 확률, 즉 직장인일 확률을 가리키므로 $P(A^c) = 1 - P(A) = 1 - 0.04 = 0.96$이다. 또한 $P(X|A)$는 철수가 농부라는 전제로 친구가 철수를 설명한 내용 X의 확률이 90%라고 하고, $P(X|A^c)$는 철수가 직장인이라는 전제로 친구가 철수를 설명한 내용 X의 확률이 20%라고 하자.

$$P(X) = 0.9 \times 0.04 + 0.2 \times 0.96 = 0.228$$

이 된다. 이를 모두 합하면 다음과 같다.

$$P(A|X) = \frac{(0.9 \times 0.04)}{0.228} = 0.158$$

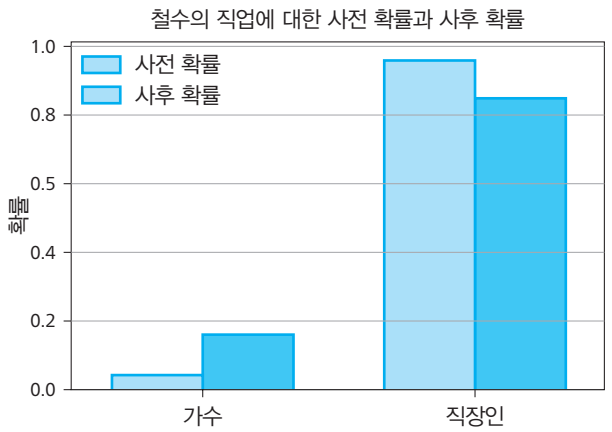
절대적인 수치로는 가수보다 직장인일 확률이 높게 나온다. 하지만 여기서 우리가 주목해야 할 점은 철수의 정보를 고려하여 사후 확률을 계산하였을 때, 사전 확률과 확연한 차이를 보인다는 것이다. 이게 바로 베이지안 추론에서의 믿음의 업데이트 과정이라고 할 수 있다.

사전 확률과 사후 확률을 파이썬으로 구현해보자.

In

```
import numpy as np
from matplotlib import pyplot as plt
import matplotlib
matplotlib.rc('font', family='Malgun Gothic')
                                          # plot에 한글을 출력하기 위해서 필요하다

colors = ['#348ABD', "#A60628"]
prior = [1/25., 24/25.]          # 사전 확률: [P(A), P(~A)]
posterior = [0.158, 1-0.158]     # 사후 확률: [P(A|X), P(~A|X)]

plt.bar([0, .7], prior, alpha=0.7, width=0.25,
   color=colors[0], label="사전 확률",
   lw="3", edgecolor='#348ABD')
plt.bar([0+0.25, .7+0.25], posterior, alpha=0.7,
   width=0.25, color=colors[1], label='사후 확률',
   lw="3", edgecolor='#A60628')

plt.xticks([0.12, 0.82], ['가수', '직장인'])
plt.title('철수의 직업에 대한 사전 확률과 사후 확률')
plt.ylabel('확률')
plt.legend(loc='upper left')

plt.show()
```

Out



13.3 가능도

통계에서 우도(likelihood)란 표현이 자주 나오는데 이 용어 자체가 직관적이지 않아 이해하기가 어렵다. 우리 책에서는 앞으로 우도라고 표현하지 않고 직관적인 의미가 담긴 **가능도**(likelihood)라고 나타내기로 한다. 가능도는 정규분포부터 회귀분석과 최신 인공지능 알고리즘에 이르기까지 통계학의 모든 부분에서 빠질 수 없는 개념이다.

예를 들어 동전 1개를 던져서 표면을 확인하는 실험을 생각해보자. 동전을 한 번 던져서 앞면이 나올 확률은 얼마일까? 대부분의 보통 사람들도 바로 1/2이라고 답한다. 이미 선입관으로 동전은 매우 공정해서 앞면과 뒷면이 반반씩 나올 거라고 알고 있기 때문이다. 왜냐하면 동전의 특성은 잘 알려져서 우리가 잠재적으로 앞면이 나올 확률이 1/2이라는 사실을 인정하고 있기 때문에 의심 없이 이를 받아들인다.

하지만 우리가 잘 모르는 상황에서 얻게 되는 데이터로부터 어떤 확률을 추정해야 할 때가 있다. 예를 들어 주머니 속에 공이 3개가 들어있다고 가정해보자. 단, 주머니 속 사정을 우리가 모르고 있다고 한다. 이 주머니에서 1개의 공을 꺼내서 색을 확인하고 다시 집어넣은 다음 다시 공을 하나 꺼내서 확인하는 실험을 10회 실시한다고 해보자. 실험 결과 10번 중 붉은색 공이 7번 나왔고 파란색 공이 3번 나왔다고 한다. 과연 이 주머니에 들어있는 공들은 무슨 색의 공이 몇 개씩 들어있는 것일까? 또 다른 실험 결과로 10번 중 붉은색 공이 2번 나왔고 파란색 공이 8번 나왔다고 한다면 주머니의 공 성분은 어떻게 되어 있다고 판단하게 될까? 이런 문제를 해결하기 위한 도구가 바로 가능도이다.

붉은색 공과 파란색 공으로만 들어있을 수도 있고, 붉은색 공과 파란색 공 그리고 알 수 없는 다른 색의 공이 들어있을 수도 있다. 즉, 다음 수식 (13.14)에 나타낸 3가지

$$\{\text{Red, Red, Blue}\},\ \{\text{Red, Blue, Blue}\} \text{ 또는 } \{\text{Red, Blue, ?}\} \tag{13.14}$$

중 한 가지이다. 실험 결과에서 나온 사실을 가지고 이 3가지 경우 중에 어느 게 가장 가능성이 있을지 알아보자. 일단 문제를 좀 더 간단히 만들어보자. 주머니 속에는 붉은색 공과 파란색 공만 들어있다고 가정해보자. 그러면 문제가 좀 더 간단해진다. 즉, 다음 수식 (13.15)로 표현되는 2가지 중 한 가지이다.

$$A = \{\text{Red, Red, Blue}\},\ B = \{\text{Red, Blue, Blue}\} \tag{13.15}$$

표본 수가 한 개일 경우의 가능도

실제로 한 개의 공을 꺼내서 색을 확인하는 실험을 10회 실시한다고 할 때 10회 중 붉은색 공이 나오는 횟수를 확률변수 X라고 하면 X가 가질 수 있는 값은 0, 1, . . . , 10이고 X는 이항분포를 갖는 확률변수 $B(10, p)$이다. 즉, 확률분포함수는 수식 (13.16)과 같다.

$$P(X = x) = \binom{10}{x} p^x (1 - p)^{10-x}, \; x = 0, 1, \ldots, 10 \tag{13.16}$$

이항분포의 가능도함수는

$$L(p) = p^x (1 - p)^{10-x} \tag{13.17}$$

이 된다. 계수인

$$\binom{10}{x}$$

를 생략한 것으로서 확률질량함수와 비슷하다. 우리가 알고 싶은 것은 어떤 p값이 수식 (13.17)을 최대가 되게 만드는가이다.

몇 가지 예를 들어 가능도함수가 어떻게 이용되는지 이해해보자.

첫 번째 예제는 우리가 이미 관측해서 얻는 값이 $n = 10$이고 $x = 7$일 경우이다. 그래서 가능도함수는

$$L(\mathrm{p}) = p^7 (1 - \mathrm{p})^3 \tag{13.18}$$

이다. 여기서 p는 주머니에서 공 한 개를 꺼내 확인한 색이 붉은색일 확률이므로 0에서 1 사이의 값이다. 그러면 어떤 p가 수식 (13.18)을 최대치로 만드는가? 다시 말해서 어떤 이항분포 $B(10, \mathrm{p})$가 수식 (13.18)을 최대치로 만드는가? p를 찾기 위해 p에 0.1부터 0.9까지 넣어서 수식 (13.18)의 값을 계산하면 다음 표와 같다.

p	$p^7(1 - p)^3$
0.1	0.000000073
0.2	0.000006554
0.3	0.000075014
0.4	0.000353894
0.5	0.000976563

p	$p^7(1 - p)^3$
0.6	0.001791590
0.7	**0.002223566**
0.8	0.001677722
0.9	0.000478297

표를 보면 $p = 0.7$일 때 가능도함수의 값이 가장 큰 0.002223566임을 알 수 있다. 실제로 미분을 이용하여 수식 (13.18)이 [0, 1]에서 최댓값을 갖는 p를 찾아보자. 즉, 극댓값을 갖는 p를 구해보자.

$$\frac{d}{dp}L(\mathrm{p}) = 7p^6(1-p)^3 - 3p^7(1-p)^2 = 0$$

을 계산하면 $L(p)$는 $p = 0, 1, 7/10$에서 극값을 가지며, $p = 7/10$일 때 최댓값을 갖는다.

주머니에서 10번 꺼내어 색을 확인해볼 때 7번 붉은색 공이 나오는 현상을 가장 잘 설명하는 p 값이 0.7이라는 뜻이다. 즉, 한 번 던졌을 때 붉은색 공이 나오는 확률을 0.7로 선택할 때 위의 실험 결과와 가장 잘 맞아떨어진다는 의미이다. 다시 말해서 이 표본은 $B(10, 7/10)$에서 추출되었고 한 번 공을 꺼낼 때 그 공이 붉은색일 확률이 7/10이라는 것이다. 공의 수가 3개인 주머니 속의 공이 두 가지 색만 가졌다고 가정했기 때문에 이 사실로부터 우리는 주머니 속에 붉은색 공이 2개, 파란색 공이 1개 들어있는

$$A = \{Red, Red, Blue\}$$

일 거라고 판단할 수 있다.

두 번째 예제를 살펴보자. 이번에는 실험 결과 $n = 10$이고 $x = 2$일 경우이다. 붉은색 공이 2번 나왔고 파란색 공이 8번 나왔다는 뜻이다. 그러면 이 표본은 어떤 $B(10, p)$에서 추출된 것인가? $n = 10$이고 $x = 2$이므로 가능도함수는

$$p^2(1-p)^8$$

이 된다. 미분을 이용하여 극대를 구해보면 이 식의 최댓값은 $p = 2/9$일 때이다. 즉, 이 표본은 $B(10, 2/9)$에서 추출되었다고 말할 수 있다. 다시 말해서 한 번 공을 꺼낼 때 그 공이 붉은색일 확률은 2/9이라는 것이다. 이 사실로부터 우리는 주머니가

$$B = \{Red, Blue, Blue\}$$

일 거라고 판단할 수 있다.

세 번째 예제를 살펴보자. 이번에는 실험 결과 $n = 10$이고 $x = 5$일 경우이다. 붉은색 공이 5번 나왔고 파란색 공이 5번 나왔다는 뜻이다. 그러면 이 표본은 어떤 $B(10, p)$에서 추출된 것인가? $n = 10$이고 $x = 5$이므로 가능도함수는

$$p^5(1-p)^5$$

이 된다. 미분을 이용하여 극대를 구해보면 이 식의 최댓값은 $p = 1/2$일 때이다. 즉, 이 표본은 $B(10, 1/2)$에서 추출되었다고 말할 수 있다. 다시 말해서 한 번 공을 꺼낼 때 그 공이 붉은색일 확률은 1/2이라는 것이다. 그러나 이 사실로부터 우리는 주머니가

$$A = \{Red, Red, Blue\}$$

또는

$$B = \{Red, Blue, Blue\}$$

중 어느 것이라고 판단하기가 어렵다. 왜냐하면 $p = 1/2$은 A라고 할 경우의 확률 2/3와 B라고 할 경우의 확률 1/3의 딱 중간이기 때문이다.

이런 경우는 10번 꺼내기를 단 한 번만 해서 표본의 수가 한 개밖에 되지 않아 생긴다. 10번 꺼내기를 여러 차례 해본다고 해보자. 그러면 표본의 수가 늘어나고 가능도에 의해 더 정밀한 추측을 할 수 있다. 즉, 더 쉽게 주머니 사정을 파악할 수 있다. 이것이 바로 베이지안 통계의 장점이다. 실험 횟수가 증가하면 증가할수록 더 많은 데이터를 얻을 수 있고, 이렇게 얻은 데이터를 이용하여 더 정확한 예측을 할 수 있다.

네 번째 예제로 우리가 주머니에서 공을 꺼내어 색을 확인하는 횟수를 10회가 아니고 그 횟수를 점점 늘려 100회, 1000회, 10000회 등으로 상황을 바꿔가며 붉은색 공의 수를 세어본다고 해보자. 예를 들어 $n = 10000$이고 $x = 6700$이라고 하자. 즉, 10000번 확인했는데 그중에서 붉은색 공이 6700번 나왔다는 의미이다. 수식 (13.18)은

$$p^{6700}(1 - p)^{3300}$$

이 되고 미분을 이용하여 이 식이 최댓값을 갖는 p를 찾아보면 $p = 6700/10000$이다. 다시 말해서 이항분포 $B(10000, 6700/10000)$에서 표본 데이터가 추출되었다고 볼 수 있다. 이처럼 $n \to \infty$로 늘려나가면 이항분포의 가능도함수 최대치는 $p = 2/3$일 때이다.

위의 경우에서 주머니 속 사정은 $A = \{Red, Red, Blue\}$또는 $B = \{Red, Blue, Blue\}$두 가지 중의 하나였으므로 실제로

$$A\text{의 경우라면: } p = \frac{2}{3} \simeq 0.66667$$

$$B\text{의 경우라면: } p = \frac{1}{3} \simeq 0.33333$$

이기 때문에 가능도함수의 값이 가장 큰 $p = 6700/10000 = 67/100 = 0.67$이 2/3에 가장 근접

한 값이다. 그래서 주머니 속 사정이 A일 거라고 추측할 수 있다.

위의 예제를 살펴보면 우리가 가능도의 개념을 활용하여 두 가지 주머니 환경 중에서 가장 가능성이 높은 것을 선택할 수 있었다. 통계학을 이용하여 좀 더 일반적으로 말해보면 어떤 가설 H에 대한 가능도란 어떤 시행 결과 E가 주어졌을 때, 만약 주어진 가설 H가 참이라면 그러한 결과 E가 나올 정도는 얼마나 되느냐 하는 것이다. 즉, 결과 E가 나온 경우 그러한 결과가 나올 수 있는 여러 가능한 가설들을 평가할 수 있는 지푯값이 바로 가능도이다.

다른 예를 들어 가능도를 이해해보자. 의학 전문가 시스템(expert system)을 이용하여 사람의 신체 정보를 입력하고 병명을 출력한다고 해보자. 하지만 이 전문가 시스템이 어떤 결론을 출력으로 내놓았다고 해서 그것을 우리가 전적으로 신뢰할 수는 없다. 따라서 출력으로 나온 병명은 하나의 가설이라고 생각할 수 있다. 가설이 설정된 뒤에 병명 판정에 대한 새로운 지식이나 입력되는 신체 정보에 변화가 있다면 새로운 병명이 출력될 수도 있다.

전문가 시스템의 불확실성을 평가하기 위해 우리가 앞에서 배운 베이즈의 정리를 활용해보자. 사전 확률에 새로운 정보를 추가하여 사후 확률을 얻는 과정이 베이즈의 정리 핵심이라고 했다. 사전 확률을 부여할 때 독자들은 약간의 의심을 할 수도 있다. 왜냐하면 자의성을 배제하기가 어렵기 때문이다. 이때 가능도를 사용하면 그 자의성을 조금이라도 줄일 수 있어서 사전 확률을 좀 더 정확하게 계산할 수 있다.

복수개 표본 데이터가 있을 경우의 가능도

앞에서 표본의 수가 한 개일 경우에 가능도를 여러 가지 예를 들어 이해했다. 실제로 가능도는 표본의 수가 복수개일 경우에 더 정확한 역할을 한다.

다음과 같은 m개의 데이터를 고려해보자.

$$x_1, x_2, \ldots, x_m$$

이들이 발생할 확률을 각각

$$p_1, p_2, \ldots, p_m$$

이라고 하자. 즉, $P(X = x_i) = p_i (i = 1, \ldots, m)$이다. 이때 가능도는 다음 수식 (13.19)와 같이 정의한다.

$$p_1 p_2 \cdots p_m = \prod_{i=1}^{m} p_i \tag{13.19}$$

위와 같이 가능도는 모든 확률의 곱이다.

정규분포의 확률밀도함수를 다시 생각해보자. 정규분포 $N(\mu, \sigma^2)$을 따르는 확률밀도함수는 평균 μ와 표준편차 σ를 가질 때

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

이다. 이 경우 가능도함수는 표본값이 $x_1, x_2, \ldots, x_m$일 때 두 개의 매개변수 μ와 σ를 변수로 갖는 함수로서

$$\begin{aligned} L(\mu, \sigma_2, x_1, x_2, \ldots, x_m) &= \prod_{i=1}^{m} f(x_i) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^m \prod_{i=1}^{m} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right) \\ &= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^m \exp\left(-\sum_{i=1}^{m} \frac{(x_i-\mu)^2}{2\sigma^2}\right) \end{aligned} \tag{13.20}$$

이다. 매개변수가 μ와 σ 두 개라는 점에 주의하기를 바란다.

앞절에서 예로 들었던 이항분포의 가능도를 다시 생각해보자. $n = 10$이고 실제로 측정된 값이 한 개가 아니고 $x = x_1, x_2, \ldots, x_m$이라고 해보자. 이들이 발생할 확률을 각각

$$p_1, p_2, \ldots, p_m$$

이라고 하면 $P(X = x_i) = p_i (i = 1, \ldots, m)$이다. 이때 가능도함수는 표본값이 $x_1, x_2, \ldots, x_m$일 때 하나의 매개변수 p의 함수로서 다음 수식 (13.21)과 같이 정의한다.

$$L(10, p, x_1, x_2, \ldots, x_m) = p_1 p_2 \ldots p_m = \prod_{i=1}^{m} p_i \tag{13.21}$$

여기서 매개변수는 p이다($n = 10$으로 결정되어 있기 때문에 p만 매개변수로 간주한다).

일반적인 이항분포의 확률질량함수를 다시 생각해보자. 이항분포 $B(n, p)$를 따르는 확률질량함수는 한 번 시행에서 성공할 확률을 p라고 할 때 다음 수식 (13.22)와 같이 표현할 수 있다.

$$f(x) = P(X = x) = p^x (1-p)^{n-x} \tag{13.22}$$

이 경우 가능도함수는 수식 (13.22)에 나타난 모든 확률의 곱이므로 표본값이 $x_1, x_2, \ldots, x_m\}$일 때 두 개의 매개변수 n과 p의 함수로서 다음 수식 (13.23)이 된다.

$$L(n, p, x_1, x_2, \ldots, x_m) = \prod_{i=1}^{m} p^{x_i} (1-p)^{n-x_i} \tag{13.23}$$

여기서 매개변수는 n과 p이다.

가능도는 확률의 곱이기 때문에 이대로는 값이 0에 한없이 가까워져서 문제가 될 수 있다. 또한 식이 곱의 형태이기 때문에 미분으로 다루기 어렵다는 문제도 있다. 그래서 가능도에 로그함수를 취하여 처리하면 곱이 합으로 바뀌어서 미분 처리가 수월해진다. 로그함수는 단조 증가하는 함수이기 때문에, 가능도함수가 극댓값을 가지는 위치와 **로그가능도**(log likelihood)에서 극댓값을 가지는 위치는 같다. 따라서 곱의 형태로 이루어진 가능도함수를 미분하여 극값을 구하는 대신, 합의 형태로 이루어진 로그가능도를 미분하여도 같은 결과를 얻을 수 있다.

이렇게 정의한 로그가능도함수는 정규분포를 가정할 때 다음 수식 (13.24)와 같이 나타낼 수 있다.

$$\log L(\mu, \sigma, x_1, x_2, \ldots, x_m) = \sum_{i=1}^{m} \log p(x_i) = m \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \sum_{i=1}^{m} \frac{(x_i - \mu)^2}{2\sigma^2} \quad (13.24)$$

이와 같이 로그가능도함수도 주어진 표본으로부터 가장 큰 값을 갖게 되는 정규분포의 매개변수인 μ와 σ를 찾는 목적이므로 곱셈보다 덧셈으로 표현된 수식으로 계산하는 것이 더 편리하다. 나중에 이들 문제를 파이썬 코드로 실행하면서 설명하도록 하자.

가능도가 작은 경우

예를 들어보자. 정규분포 $N(\mu, \sigma)$에서 추출한 표본 데이터가

$$x_1 = 2.4,\ x_2 = 1.9,\ x_3 = 2.6,\ x_4 = 0.4,\ x_5 = 1.3,\ x_6 = 1.1,\ x_7 = 1.5,\ x_8 = 1.6,\ x_9 = 1.8$$

일 때 표준화된 정규분포 $N(0, 1)$의 확률밀도함수와 데이터를 겹쳐서 다음 그림 13.5에 나타냈다. 이 그림에서 확률밀도함수의 평균은 0이고 표준편차는 1이다. 데이터를 이 확률밀도함수와 같이 나타내면 데이터 가능도나 로그가능도를 구할 수 있다.

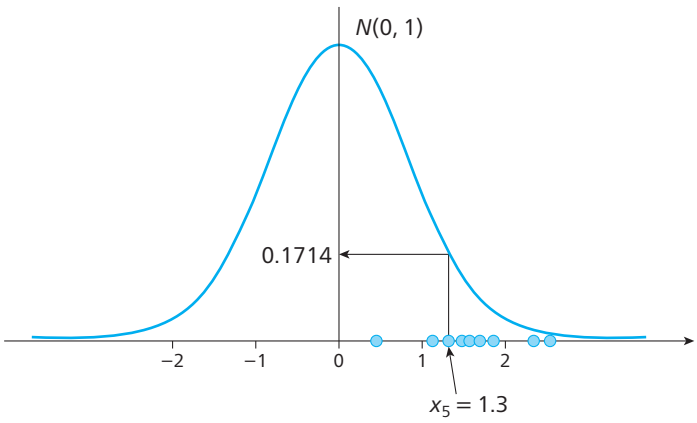


그림 13.5 가능도가 작은 경우

그림 13.5를 보면 데이터를 점으로 나타냈다. 정규분포 확률밀도함수는 주로 오른쪽에 치우쳐 있다. 따라서 표준화된 정규분포가 가정된 경우라면 데이터가 그림 13.5처럼 분포되는 경우는 별로 많지 않다. 따라서 이 데이터들이 표준정규분포 $N(0, 1)$에서 추출되었다고 믿기는 어렵다. 가능도함수의 값의 수식인 (13.20)이나 로그가능도함수의 값의 수식인 (13.24)를 보면

$$\begin{aligned} L(0, 1, x_1, x_2, \ldots, x_9) &= \prod_{i=1}^{9} f(x_i) \\ &= \left(\frac{1}{\sqrt{2\pi}}\right)^9 \prod_{i=1}^{9} \exp\left(-\frac{x_i^2}{2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi}}\right)^9 \exp\left(-\sum_{i=1}^{9} \frac{x_i^2}{2}\right) \end{aligned} \tag{13.25}$$

의 값을 결정하는 우변의

$$\exp\left(-\sum_{i=1}^{9} \frac{x_i^2}{2}\right)$$

의 값은 x_i가 그래프상의 우측에 치우쳐 있으므로 그 값에 해당하는 가능도함수의 값은 작은 값을 갖는다. 예를 들어 그림 13.5에서 $x_5 = 1.3$에 대응하는 항의 값은 실제로 계산해보면 0.1714로 매우 작은 값이다. 두 개의 데이터 $x_4 = 0.4$와 $x_6 = 1.1$에서의 가능도함수의 값도 비교적 작으므로 이들 9개 데이터에 의해 계산되는 가능도함수의 값은 매우 작은 수이게 된다. 실제로 구해보면

$$\begin{aligned} L(0, 1, x_1, x_2, \ldots, x_9) &= (0.0224) \times (0.0656) \times (0.0136) \times (0.3683) \times (0.1714) \\ &\quad \times (0.2179) \times (0.1295) \times (0.1109) \times (0.0790) \\ &= 0.000000000311881015 \end{aligned} \tag{13.26}$$

로 거의 0에 가까운 수이다. 물론 로그가능도를 가지고 계산해도 마찬가지로 아주 작은 값이 된다. 따라서 이 데이터들이 추출되었을 거라고 생각할 수 있는 정규분포는 분명 평균이 오른쪽으로 더 이동되어야 한다는 것을 추측할 수 있다.

다음 예제는 데이터와 정규분포의 확률밀도함수를 겹쳐 그린다. 확률밀도함수의 평균은 0, 표준편차는 1이다. 그리고 데이터가 이 확률밀도함수에 따라 경우의 가능도를 찾아볼 수 있다.

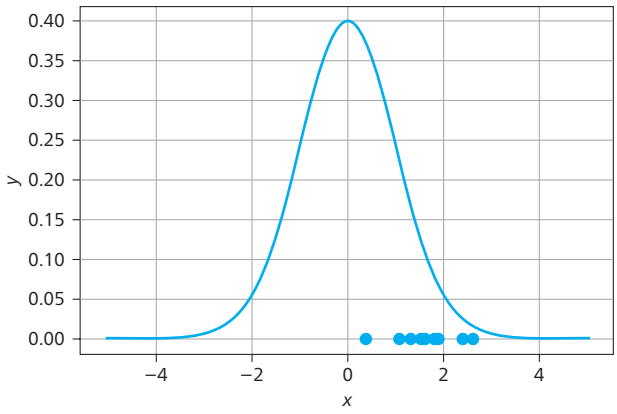
예제 13.2

다음 가능도가 작은 경우의 데이터와 확률밀도함수 코드를 실행해보자.

In

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

x_data = np.array([2.4, 1.9, 2.6, 0.4, 1.3, 1.1,1.5, 1.6, 1.8])  # 데이터
y_data = np.zeros(9)              # x_data를 산포도로 나타내기 위한 편의적인 데이터

mu = 0          # 평균치
sigma = 1       # 표준편차

# 확률밀도함수
def pdf(x, mu, sigma) :
   return 1 / (sigma * np.sqrt(2*np.pi)) * np.exp(-(x-mu)**2 /
                                                  (2*sigma**2))

x_pdf = np.linspace(-5, 5)
y_pdf = pdf(x_pdf, mu, sigma)

plt.scatter(x_data, y_data)
plt.plot(x_pdf, y_pdf)
plt.xlabel("x", size = 14)
plt.ylabel("y", size = 14)
plt.grid()

plt.show()

print("----가능도----")
print(np.prod(pdf(x_data, mu, sigma))) # 가능도 계산
```

Out




```
----가능도----
3.1124311189820244e-10
```

가능도가 큰 경우

확률밀도함수의 평균과 표준편차를 (0, 1)에서 (1.4, 1)로 변경해보자. 데이터와 정규분포 $N(1.4, 1)$의 확률밀도함수와 데이터를 겹쳐서 표현하면 다음 그림 13.6과 같다.

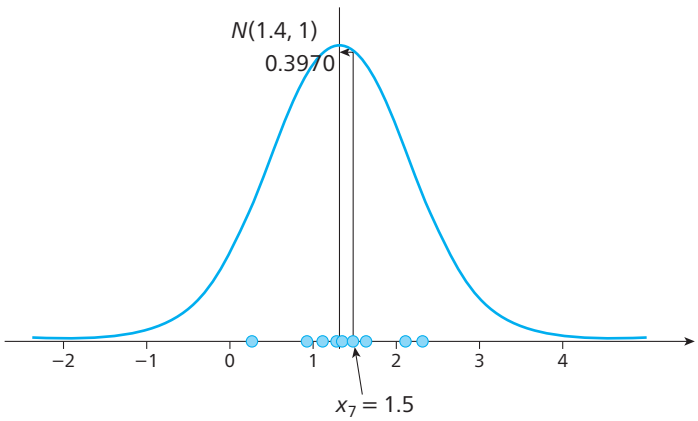


그림 13.6 가능도가 큰 경우

$N(1.4, 1)$인 정규분포 곡선이 데이터가 분포하는 정도를 비교적 잘 표현하고 있다. 그래서 이런 확률밀도함수를 가정하는 경우에 데이터가 적합하게 분포된 것으로 보인다. 즉, 데이터가 이 정규분포에서 추출되었을 거라는 믿음이 가게 된다. 실제로 가능도와 로그가능도를 계산해보면 그 값이 앞절의 수식 (13.26)에서 구했던 값보다 크다. 가능도함수의 값을 구해보면

$$\begin{aligned} L(0, 1, x_1, x_2, \ldots, x_9) &= (0.2420) \times (0.3521) \times (0.1942) \times (0.2420) \times (0.3970) \\ &\quad \times (0.3814) \times (0.3970) \times (0.3910) \times (0.3683) \\ &= 0.0000346646 \end{aligned} \tag{13.27}$$

이다. 수식 (13.27)에 계산된 값도 작은 값이지만 상대적으로 정규분포 $N(0, 1)$의 경우보다 $N(1.4, 1)$의 경우에서 가능도가 더 크다는 뜻이므로 정규분포와 같은 확률분포를 가정했을 때 데이터가 후자인 $N(1.4, 1)$인 정규분포에 더 어울리게 분포되며 적합해 보인다. 그러면 $N(1.4, 1)$보다 가능도가 더 크게 되는 매개변수 μ와 σ를 찾을 수 있지 않을까? 물론 그렇다. μ와 σ를 변수로 갖는 이변수함수인 수식 (13.25)를 각각 편미분하고 이변수함수의 극댓값을 구하는 방식으로 가능도가 최대가 되는 평균이 μ이고 표준편차가 σ인 $N(\mu, \sigma)$를 찾을 수 있다.

주어진 표본 데이터로부터 가능도가 최대가 되는 확률분포의 매개변수를 찾는 과정을 최대 가능도 추정이라고 한다. 이를 추정하는 데 사용하는 매개변수를 최대 가능도 추정량이라고 한다.

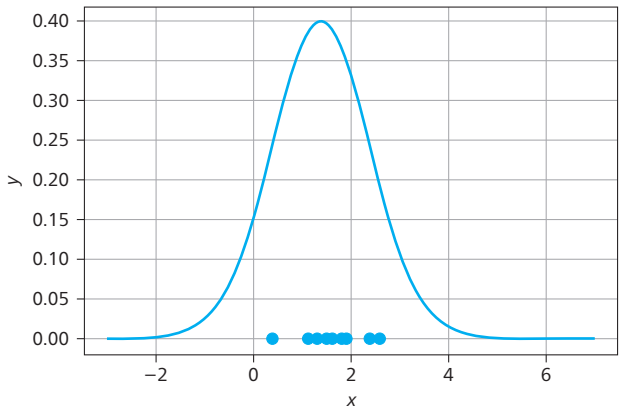
예제 13.3

다음 가능도가 큰 경우의 데이터와 확률밀도함수 코드를 실행해보자.

In

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

x_data = np.array([2.4, 1.9, 2.6, 0.4, 1.3, 1.1,1.5, 1.6, 1.8])  # 데이터
y_data = np.zeros(9)              # x_data를 산포도로 나타내기 위한 편의적인 데이터

mu = 1.4          # 데이터의 평균값
sigma = 1         # 데이터의 표준편차

# 확률밀도함수
def pdf(x, mu, sigma) :
   return 1 / (sigma * np.sqrt(2*np.pi)) * np.exp(-(x-mu)**2 /
                                                  (2*sigma**2))
x_pdf = np.linspace(-5, 5)
y_pdf = pdf(x_pdf, mu, sigma)

plt.scatter(x_data, y_data)
plt.plot(x_pdf, y_pdf)
plt.xlabel("x", size = 14)
plt.ylabel("y", size = 14)
plt.grid()

plt.show()

print("----가능도----")
print(np.prod(pdf(x_data, mu, sigma))) # 가능도 계산 후 출력
```

Out




```
----가능도----
3.464189400860282e-05
```

이와 같이 가능도가 크다는 뜻은 정규분포와 같은 확률분포를 가정했을 때 데이터가 정규분포에 어울리게 분포되며 적합해 보인다는 의미이다. 정규분포를 가정할 경우 확률밀도함수에 데이터의 평균과 표준편차의 값을 평균(μ)과 표준편차(σ)로 갖는 $N(\mu, \sigma)$의 확률밀도함수를 사용하면 가능도는 최댓값을 갖는다.

최대 가능도 추정량

데이터에서 가능도가 최대가 되는 확률분포의 매개변수를 찾는 과정을 최대 가능도 추정이라고 한다. 이를 추정하는 데 사용하는 매개변수를 최대 가능도 추정량이라고 한다.

최대가능도 추정(MLE: Maximum Likelihood Estimation)은 각 관측값에 대한 총 가능도(모든 가능도의 곱)가 최대가 되게 하는 분포를 찾는 것이다. 이는 기초 통계분석에서 회귀분석까지 거의 모든 통계분석에서 참의 값을 추정하는 원리이다.

예를 통해 최대가능도 추정을 이해하자. 앞에서 이미 3개의 공이 들어있는 주머니에서 1개의 공을 꺼내는 실험을 n번 실시하고 붉은색 공이 몇 번 나오는지 확인하는 예를 들었다. 그 과정은 이항분포 $B(n, p)$를 갖는다고 가정하였고 이를 통해 가능도를 설명했다.

이번에는 $n = 4$로서 4번 공을 꺼내서 붉은색 공이 나오는 횟수 x를 측정하는 실험을 10번 수행했을 때 결과로 얻는 표본 데이터가 다음 표와 같다고 해보자.

붉은색 공의 수	0개	1개	2개	3개	4개
횟수	0번	1번	3번	4번	2번

즉, 10개의 데이터는

$$x_1 = 1,\ x_2 = 2,\ x_3 = 2,\ x_4 = 2,\ x_5 = 3,\ x_6 = 3,\ x_7 = 3,\ x_8 = 3,\ x_9 = 4,\ x_{10} = 4$$

이다. 따라서 이 경우 가능도함수는 이들 모든 확률의 곱이므로

$$L(\mathrm{p}) = \prod_{i=1}^{10} p^{x_i}(1-p)^{4-x_i} = p^1(1-p)^3(p^2(1-p)^2)^3(p^3(1-p)^1)^4(p^4(1-p)^0)^2 \quad (13.28)$$

이 된다. 그리고 최대 가능도 추정량은 $B(4, p)$의 매개변수 p이다.

이 데이터들은 어떤 이항분포 $B(4, p)$에서 추출된 데이터일까? 즉, 0과 1 사이의 어떤 p값일 때 수식 (13.28)의 값이 최대가 될까? 수식 (13.28)을 미분해서 극댓값을 찾아보자. $L(\mathrm{p})$를 정리하면 $L(\mathrm{p}) = p^{27}(1-p)^{13}$가 되고 미분을 0으로 놓고 계산하면

$$\begin{aligned} L'(p) &= 27p^{26}(1-p)^{13} - 13p^{27}(1-p)^{12} \\ &= p^{26}(1-p)^{12}\{27(1-p) - 13p\} \\ &= p^{26}(1-p)^{12}(27 - 40p) = 0 \end{aligned}$$

이 되어

$$p = \frac{27}{40}$$

일 때 극댓값을 가지며 2차 미분을 통해서 이곳에서 최댓값을 갖는다는 사실을 알 수 있다. 따라서 위에서 얻은 표본 데이터들이 $B(4, 27/40)$인 이항분포에서 추출된 것이라고 볼 수 있다. 실험에서 얻은 표본 데이터(붉은색 점들)와 이항분포 그래프를 동일한 그래프에 그려보면 다음 그림 13.7과 같다.

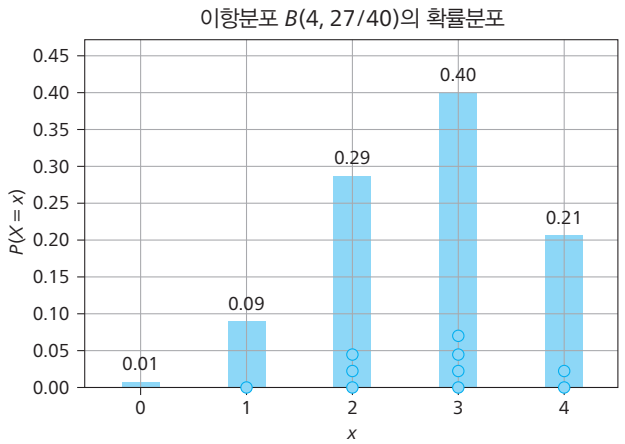


그림 13.7 이항분포 $B(4, 27/40)$의 확률분포와 표본 데이터

그림 13.7에 나타낸 그래프에서 볼 수 있듯이 표본 데이터의 분포를 $B(4, 27/40)$이 가장 잘 나타내주고 있다.

또 다른 예를 들어보자. 정규분포로부터 다음 3개의 표본 데이터를 얻었다고 해보자.

$$x_1 = 1, x_2 = 0, x_3 = -2$$

이 경우의 가능도함수는

$$\begin{aligned} L(\mu, \sigma) &= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^3 \prod_{i=1}^{3} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^3 \exp\left(-\sum_{k=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}\right) \\ &= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^3 \exp\left(-\frac{3\mu^2 + 2\mu + 5}{2\sigma^2}\right) \\ &= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^3 \exp\left(-\frac{3\left(\mu + \frac{1}{3}\right)^2 + \frac{14}{3}}{2\sigma^2}\right) \end{aligned}$$

가 된다. 이 가능도함수는 변수가 μ와 σ인 이변수 함수이므로 편미분을 이용해서 극댓값을 구할 수 있다. 하지만 그래프를 유추해보면 쉽게 최댓값의 위치를 구할 수 있다. σ값이 고정된 수라고 가정하면 가장 가능도를 높게 하는 매개변수 μ값은

$$-\frac{1}{3}$$

이다. 그러므로 이 표본 데이터들이 정규분포

$$N\left(-\frac{1}{3}, \sigma_2\right)$$

에서 얻었다고 볼 수 있다.

우리가 $\sigma = 1$이라고 가정해볼 때 이 표본 데이터는

$$N\left(-\frac{1}{3}, 1\right)$$

에서 얻은 것이며 이를 그래프로 그려보면 다음 그림 13.8과 같다.

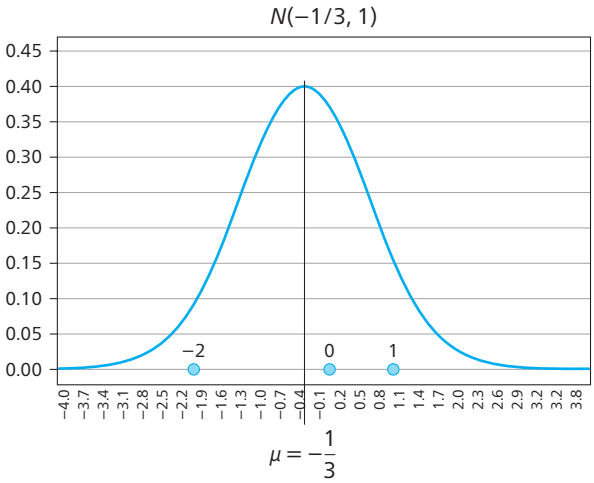


그림 13.8 표본 데이터와 정규분포 $N(-1/3, 1)$

표본 데이터(붉은색 점)를 이 정규분포에 추가해서 나타냈다. 표본 데이터가 3개뿐이지만 3개의 표본 데이터가 나올 수 있는 가장 적합한 정규분포가

$$N\left(-\frac{1}{3}, 1\right)$$

이라는 것을 알 수 있다.

가능도 구현

예제 13.4

주머니에서 공을 추출하는 실험을 총 5회 실시하기로 한다. $n = 5$로 공을 5번 꺼내서 붉은색 공이 나오는 횟수 x를 측정하는 실험을 20차례 수행했을 때 결과로 얻는 표본 데이터가 다음 표와 같다고 해보자.

붉은색 공의 수	0개	1개	2개	3개	4개	5개
횟수	1번	1번	2번	4번	8번	4번

즉, 20개의 데이터는

$$x_1 = 0,\ x_2 = 1,\ x_3 = 2,\ x_4 = 2,\ x_5 = 3,\ x_6 = 3,\ x_7 = 3,\ x_8 = 3,\ x_9 = 4\ x_{10} = 4,$$
$$x_{11} = 4,\ x_{12} = 4,\ x_{13} = 4,\ x_{14} = 4,\ x_{15} = 4,\ x_{16} = 4,\ x_{17} = 5,\ x_{18} = 5,\ x_{19} = 5,\ x_{20} = 5$$

이다.

이 경우 가능도함수는 이들 모든 확률의 곱이므로

$$
\begin{aligned}
L(p) &= \prod_{i=1}^{20} p^{x_i}(1-p)^{5-x_i} \\
&= p^0(1-p)^5 \cdot p^1(1-p)^4 \cdot (p^2(1-p)^3)^2 \cdot (p^3(1-p)^2)^4 \cdot (p^4(1-p)^1)^8 \cdot (p^5(1-p)^0)^4 \\
&= p^{69}(1-p)^{31} \qquad (13.29)
\end{aligned}
$$

이 된다. 그리고 최대 가능도 추정량은 $B(5, p)$의 매개변수 p이다.

이 데이터들은 어떤 이항분포 $B(5, p)$에서 추출된 데이터일까? 즉, 0과 1 사이의 어떤 p값일 때 수식 (13.29)의 값이 최대가 될까? 실제로 수식 (13.29)를 미분해서 극댓값을 찾아보자.

$L(p)$를 미분하고 이를 0으로 놓고 계산하면

$$
\begin{aligned}
L'(p) &= 69p^{68}(1-p)^{31} - 31p^{69}(1-p)^{30} \\
&= p^{68}(1-p)^{30}\{69(1-p) - 31p\} \\
&= p^{68}(1-p)^{30}(69-100p) = 0
\end{aligned}
$$

이 되어

$$
p = \frac{69}{100}
$$

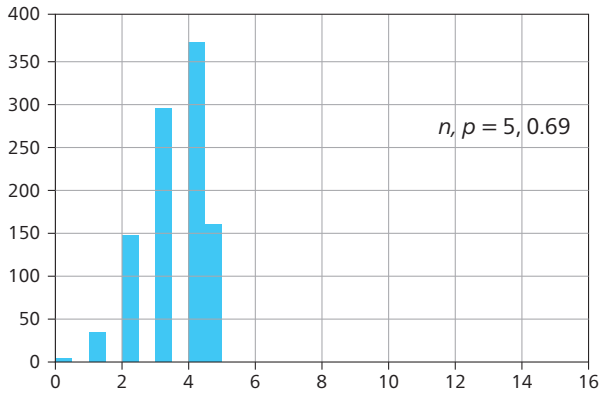
일 때 극댓값을 가지며 2차 미분을 통해서 이곳에서 최댓값을 갖는다는 사실을 알 수 있다. 따라서 위에서 얻은 표본 데이터들이 $B(5, 69/100)$인 이항분포에서 추출된 것이라고 볼 수 있다.

여기서 구한 이항분포 $B(5, 0.69)$의 확률함수를 파이썬과 넘파이, 맷플롯립을 이용해서 그래프로 나타내보자.

In

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
s = np.random.binomial(5, 0.69, 1000)  # 이항분포로부터 1000회 시행해서 얻는 샘플

plt.xlim(0, 16)      # x축 범위
plt.ylim(0, 400)     # y축 범위
plt.text(12.5, 250, 'n, p = 5, 0.69')
plt.hist(s, color='lightcoral')

plt.show()
```

Out



13.4 정보량

정보 이론(information theory)은 최대한 많은 데이터를 매체에 저장하거나 채널을 통해 통신하기 위해 데이터를 정량화하는 응용 수학의 한 분야이다. 정보 이론에서 핵심이 되는 측정 도구는 **엔트로피**(entropy)이다. 엔트로피로 확률변수의 값에 들어있는 불확실성 정도를 수량화하거나 랜덤 프로세스에서 산출되는 불확실성을 계량화할 수 있다.

예를 들어 동전 던지기에서 나올 수 있는 현상을 나타낼 때 필요한 정보와 주사위를 한 개 던질 때 나오는 정보를 나타낼 때 필요한 정보의 양은 분명히 다르다. 동전 던지기 현상을 표현하는 데 들어가는 엔트로피는 주사위 던지기 현상을 표현하는 데 들어가는 엔트로피보다 적다. 주사위 던지기에서는 적어도 6가지 현상(1부터 6까지의 수)을 표현해야 하지만, 동전 던지기에서는 2가지 현상(앞면과 뒷면)만 표현하면 된다. 2가지 현상을 표현하는 데는 1개의 비트만 있어도 할 수 있지만 6가지 현상을 표현하려면 적어도 3비트가 필요하다. 그래서 정보 엔트로피는 보통 저장 공간에 저장할 때 또는 데이터를 전송할 때 사용되는 평균 비트 수로 표현된다.

정보 이론의 또 다른 중요한 측정치로는 상호 정보, 채널 용량, 오류 지수와 상대 엔트로피가 있다. 정보 이론에서 다루는 중요한 영역으로는 소스 코딩, 알고리즘 복잡도 정리, 알고리즘 정보 이론 및 정보 이론적 보안 등이 있다. 또한 정보 이론은 추론 통계, 암호학, 신경생물학, 인지론, 언어학, 생물정보학, 열역학, 양자 컴퓨팅, 블랙홀, 정보 검색, 지능적 수집, 저작권 침해 탐지, 패턴인식, 비정상 탐지 등에서 응용된다.

정보량

정보 이론의 출발은 확률과 통계 이론이다. 정보 이론은 확률변수의 분포가 어떤지를 나타내는 데 필요한 정보를 측정하는 과정과 깊이 연관되어 있다. 정보 이론의 핵심은 잘 발생하지 않는 사건의 정보량이 자주 발생하는 사건의 정보량보다 크다는 사실이다. 확률적으로 발생할 가능성이 작은 사건이 가진 정보량이 확률적으로 발생할 가능성이 많은 사건의 정보량보다 크다는 뜻이다. 이러한 양을 계량화할 수 있을까? 답은 '그렇다'이다. 정보로부터 정보량을 계량화할 수 있는 방법이 있으며 이를 측정하거나 측정된 정보량을 비교할 수 있다.

예를 들어 직관적으로 정보량을 이해해보자. 권투 시합에서는 체급을 나누어 체중이 비슷한 선수끼리 대적하는 것이 정상이다. 그런데 체급이 낮은 플라이급 선수가 밴텀급 선수와 경기를 한다고 해보자. 일반적으로 플라이급 선수의 체중이 적게 나가기 때문에 승률은 매우 떨어질

것이다. 하지만 플라이급 선수가 게임에서 이길 확률이 10% 정도이고 질 확률은 70% 정도이며 무승부가 될 확률이 20% 정도라고 해보자. 만약 실제로 경기에서 플라이급 선수가 밴텀급 선수를 이겼다면 이 소식은 예상을 뒤엎은 정보이다. 낮은 확률을 가진 사건이 실제로 발생했기 때문에 밴텀급 선수가 승리한 소식보다 훨씬 파급력이 큰 정보이다. 정보량이라는 측면에서 볼 때 플라이급 선수가 밴텀급 선수를 이긴 경우의 정보량이 밴텀급 선수가 이기는 경우의 정보량이나 비기는 경우의 정보량보다 크다.

정보량 계산

정보량을 계산하는 식은 다음과 같다.

$$I(x_i) = -\log_a p(x_i) \tag{13.30}$$

수식 (13.30)에서 $p(x_i)$는 x_i가 발생할 확률이다. 즉, $0 \le p_i \le 1$인 실수이다. 로그의 밑수 a는 어떤 정보를 측정하느냐에 따라 결정되는 값이다. 주로 2를 많이 사용하는데, 그 이유는 밑이 2인 로그함수의 값이 해당 정보를 표현하는 데 필요한 비트 수를 계산하는 데 유용하기 때문이다. 이렇게 계산된 경우의 단위는 비트(bit)이다. 로그함수의 밑수로 자연대수 e를 사용할 경우의 단위는 *nat*(natural unit)이다. 상용대수 10을 사용할 경우의 단위는 *hartleys*이다. 즉, 발생 확률이 1/10이면 1 *hartley*라고 한다. 다시 말해서

$$1\ hartley = \log_2 10 \text{ 비트} = \ln 10\ nat$$

이다.

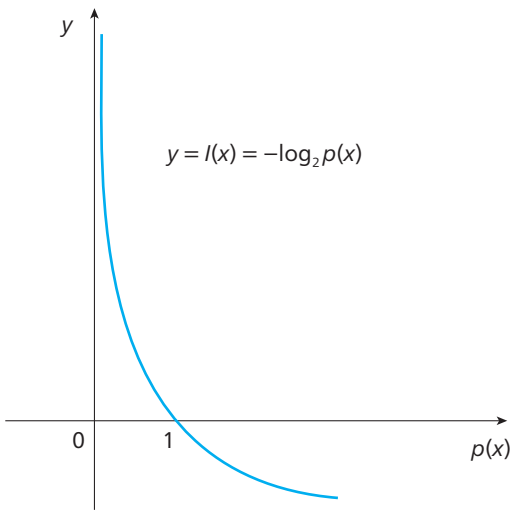


그림 13.9 정보량 계산식

정보량을 수식 13.30으로 나타내는 것이 의미가 있을까? 이 함수를 그래프로 그려보자. 아래 그래프 13.9에서 $p(x)$의 값이 [0, 1]에서 정의된다. $p(x)$축 아래에 나타난 그래프 부분이 정보량을 나타낸다. 그래서 $p(x)$가 0에 가까울수록 정보량 $y = I(x) = -\log_2 p(x)$는 무한히 커지며, $p(x)$가 1에 가까워질수록 정보량은 0에 가까워진다. 따라서 발생 확률이 적은 사건에 큰 정보량을 부여하고 발생 확률이 큰 사건에는 정보량을 적게 부여하는 식으로 정보량을 대응시킨다.

수식 (13.30)을 처음으로 소개한 사람은 정보 이론에 큰 업적을 남긴 랄프 하틀리(R.V. Hartley)라는 학자이다. 그는 해리 나이퀴스트(Harry Nyquist)와 클로드 섀넌(Slaude Shannon)과 함께 정보 이론 분야 발전에 공헌한 사람 중 한 명이다.

앞에서 예로 들었던 권투 선수 이야기를 다시 해보자. 플라이급 선수가 게임에서 이길 확률이 10% 정도이고 질 확률은 70% 정도이며 무승부가 될 확률이 20% 정도라고 해보자. 사건을 다음과 같이 정의하자.

$$A_1 = \text{플라이급 선수가 밴텀급 선수를 이긴다}$$
$$A_2 = \text{밴텀급 선수가 플라이급 선수를 이긴다}$$
$$A_3 = \text{플라이급 선수와 밴텀급 선수가 비긴다}$$

각각의 사건에 대한 정보량을 계산해보면 다음과 같다.

$$I(A_1) = -\log_2(0.1) = 3.3219$$
$$I(A_2) = -\log_2(0.7) = 0.5145$$
$$I(A_3) = -\log_2(0.2) = 2.3219$$

따라서 플라이급 선수가 밴텀급 선수를 이기는 경우가 나머지 경우에 비해 정보량이 훨씬 많다는 것을 알 수 있다.

정보량의 의미와 특징을 정리해보자.

(1) 정보량은 항상 양의 값을 갖는다.
로그함수는 양수함수이기 때문이다.

(2) 어떤 사건 E가 일어날 확률이 1이라면 $I(E) = 0$이다.
그 사건이 반드시 일어난다는 사실을 100% 확신할 수 있기 때문에 정보의 가치가 없다는 뜻이다.

(3) 두 개의 사건 E와 F가 $P(E) < P(F)$라면 $I(E) > I(F)$가 된다.
어떤 사건의 발생 확률이 낮을수록 그 사건이 일어나면 더 많은 정보를 얻을 수 있다.

(4) 두 개의 사건 E와 F가 독립사건이라면 $I(E \cap F) = I(E) + I(F)$이다.

두 개의 사건이 통계적으로 독립일 경우에 각각의 정보량을 더하면 된다. 발생 확률 $P(E) = 1/2$일 때, 정보량 $I(E) = 1$비트이므로 1비트 정보량 = 2개 사건 E와 F가 동일한 확률로 발생할 때의 정보량이다.

엔트로피

일반적으로 특정 결과와 관련된 정보량보다는 가능한 모든 결과의 평균정보량이 더 의미가 있다. 어느 확률변수 X의 모든 가능한 결과를 $x_1, x_2, \ldots, x_n$이라고 하자. 이러한 정보량의 기댓값을 **엔트로피**(entropy)라고 한다. 기댓값을 구하기 위해 확률변수 X의 확률분포를 $p(X)$라고 하고 확률질량함수를 $p(x) = P(X = x)$라고 하면 평균정보량을 다음 수식 (13.31)과 같이 정의하며 이를 엔트로피 또는 섀넌 엔트로피라고 부르고 다음과 같이 나타낸다.

$$H(p) = E[I(x_i)] = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) \tag{13.31}$$

X는 이산 확률변수이고 X가 가질 수 있는 값들이 $x_1, x_2, \ldots, x_n$이다. E는 기댓값을 나타낸다. 엔트로피는 전체 사건의 불확실성을 나타내는 값이라고 생각할 수 있다. 모든 결과가 비슷한 확률로 일어날 때 엔트로피는 가장 큰 값을 갖는다. 권투 시합에 대한 엔트로피를 계산해보자. 게임이 2번 벌어진다고 가정해보자.

게임 1: 플라이급 선수 A와 밴텀급 선수 B의 매치로서 플라이급 선수의 이길 확률이 0.05이고 질 확률은 0.85이며 비길 확률은 0.1이다.

x	이긴다	진다	비긴다
$p_1(x)$	0.05	0.85	0.1

게임 2: 플라이급 선수 A와 또 다른 플라이급 선수 C의 매치로서 플라이급 선수 A가 이길 확률은 0.35이고 질 확률은 0.37이고 비길 확률은 0.28이다.

x	이긴다	진다	비긴다
$p_2(x)$	0.35	0.37	0.28

두 게임의 엔트로피를 각각 계산해보자.

$$\begin{aligned}\textbf{게임 1의 엔트로피} = H(p_1) &= -\sum_x p_1(x)\log_2 p_1(x) \\ &= -\{(0.05)\log_2(0.05) + (0.85)\log_2(0.85) + (0.1)\log_2(0.1)\} \\ &= (0.05)(4.3219) + (0.85)(0.2344) + (0.1)(3.3219) \\ &= 0.74753\end{aligned}$$

$$\text{게임 2의 엔트로피} = H(p_2) = -\sum_x p_2(x)\log_2 p_2(x)$$

$$= -\{(0.35)\log_2(0.35) + (0.37)\log_2(0.37) + (0.28)\log_2(0.28)\}$$

$$= (0.35)(1.5145) + (0.37)(1.4344) + (0.28)(1.8365)$$

$$= 1.57502$$

두 엔트로피를 비교해보면 게임 2의 엔트로피가 게임 1의 엔트로피의 두 배가량이다. 따라서 게임 2가 게임 1보다 정보량이 더 큰 경기였다고 볼 수 있다. 결과가 뻔히 예상되는 경기인 게임 1의 엔트로피가 작고, 결과 예측이 어려운 게임 2의 엔트로피가 크다는 것을 알 수 있다.

예를 더 들어보자. 동전 던지기를 생각해보자. 앞면이 나올 확률을 α라고 하고 뒷면이 나올 확률은 $1 - \alpha$라고 하자. X는 베르누이(Bernoulli)분포를 갖는 확률변수이다. 그러므로 확률분포는 다음과 같다.

x	앞면	뒷면
$p(x)$	α	$1 - \alpha$

이의 평균정보량인 엔트로피는 수식 (13.31)에 의해

$$H(p) = -\alpha\log_2\alpha - (1-\alpha)\log_2(1-\alpha) \tag{13.32}$$

이다. 이를 이항 엔트로피함수라고 하고 보통 H_b(p)로 나타낸다. 첨자 b는 이항분포의 binomial에서 따온 첨자이다. 이를 그래프로 나타내면 다음 그림 13.10과 같다.[2]

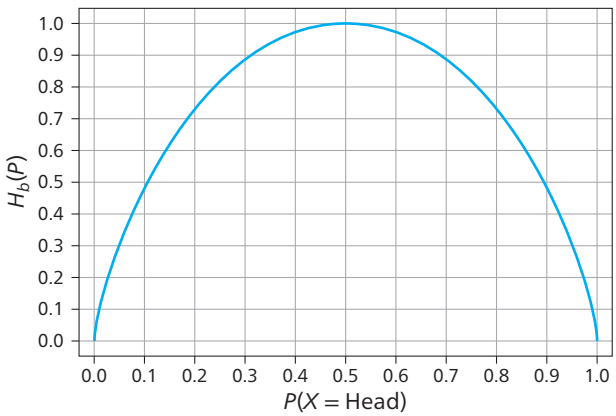


그림 13.10 이항 엔트로피함수

[2] 위키피디아 참조: By !Original: BronaVector: Alessio DamatoNewer version by Rubber Duck - Own work based on: Binary entropy plot.png by Brona, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid = 1984868

이항 엔트로피함수의 그래프를 보면 앞면이 나오는 확률이 $P(X = Head) = p = 0.5$일 때 최댓값 1을 갖는다.

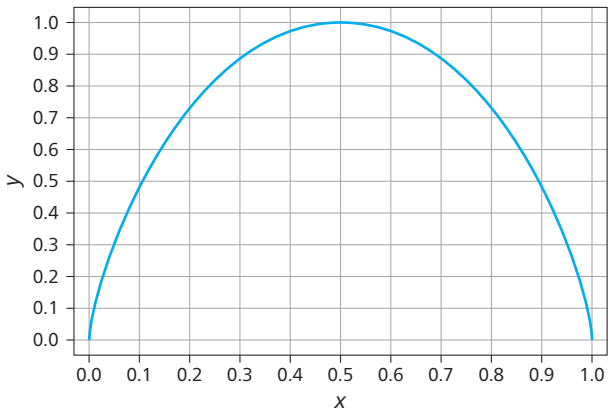
예제 13.5

다음 이항 엔트로피함수 그래프 코드를 실행해보자.

In

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.01, 0.99)
# 범위를 0.01에서 0.99까지 둔다

y = -x * np.log2(x) - (1-x) * np.log2(1-x)  # 평균정보량

plt.plot (x, y)
plt.xlabel("x", size = 15)
plt.ylabel("y", size = 15)
plt.grid()

plt.show()
```

Out



엔트로피를 이용한 확률분포의 차이

두 개의 확률분포가 가진 차이를 계산하는 데 다양한 방법이 있다. 엔트로피를 계산하여 이를 비교하는 방법으로 두 개의 확률분포가 얼마나 다른지 또는 유사한지를 알아보기로 한다. 어떤 이상적인 분포에 대해, 그 분포를 근사하는 다른 분포를 사용하여 샘플링한다고 할 때 발생할

수 있는 정보 엔트로피 차이를 계산하는 방식이다. 이를 **상대 엔트로피**(relative entropy)라고 하고 **정보 획득량**(information gain) 또는 **정보 확산**(information divergence)이라고도 한다.

정보 엔트로피의 비교를 위해 동일 확률 공간에 정의되는 두 개의 유사한 확률분포 간의 정보 엔트로피 차이를 정의하기 위해 몇 가지 개념을 도입하기로 한다. 제일 먼저 결합 엔트로피(joint entropy)에 대해 알아보고, 조건부 엔트로피(conditional entropy), 상호 정보(mutual information), 쿨백-라이블러 확산(Kullback-Leibler divergence), 교차 엔트로피(cross entropy)에 대해 학습하기로 한다.

결합 엔트로피

두 개의 이산 확률변수 X와 Y에 대한 **결합 엔트로피**(joint entropy)는 단지 이들 쌍으로 이루어진 확률벡터 (X, Y)의 엔트로피이다. 만약 두 개의 확률변수가 독립이라면 이 확률벡터의 결합 엔트로피는 각각의 엔트로피의 합이다.

이산 확률변수 X가 한 표본 공간에 정의되고 다른 이산 확률변수 Y는 또 다른 표본 공간에 정의된다고 가정하자. 그러면 결합 확률변수 (X, Y)는 결합 확률분포 $p(X, Y)$가 결정된다. 이때 결합 엔트로피는 다음 수식 (13.33)과 같이 정의한다.

$$H(X, Y) = -\sum_{x}\sum_{y} p(x, y)\log_2[p(x, y)] \tag{13.33}$$

x와 y는 X와 Y가 각각 갖는 값이다. 그리고 $p(x, y) = P(X = x, Y = y)$로 결합 확률질량함수이며, $p(x, y) = 0$이면 $p(x, y)\log_2[p(x, y)] = 0$이다.

예를 들어보자. 만약 (X, Y)가 바둑판에서 바둑알의 위치를 나타낸다고 해보자. X는 가로축의 위치이고 Y는 세로축의 위치라고 하면 X는 1, 2, . . . , 19를 갖고 Y도 마찬가지로 1, 2, . . . , 19를 갖는 균등분포를 갖는 확률분포이다. 그래서

$$P(X = x) = \frac{1}{19},\ x = 1, 2, \ldots, 19$$

$$P(Y = y) = \frac{1}{19},\ y = 1, 2, \ldots, 19$$

이다. 또한 두 개의 확률변수 X와 Y는 서로 독립인 균등분포이므로

$$p(x, y) = P(X = x, Y = y) = P(X = x)p(Y = y) = \frac{1}{19}\frac{1}{19} = \frac{1}{361}$$

이다. 바둑알의 가로축 위치에 대한 엔트로피와 세로축 위치에 대한 엔트로피의 합이 바로 확

률벡터의 결합 엔트로피이다. 따라서

$$H(X, Y) = E_{X, Y}[-\log_2 p(x, y)] = -\sum_{x, y} p(x, y)\log_2 p(x, y)$$

이다. 이를 이용하여 실제 결합 엔트로피를 구해보면

$$H(X, Y) = -\sum_{1}^{19}\sum_{1}^{19}\frac{1}{361}\log_2\left(\frac{1}{361}\right) = -(361)\frac{1}{361}\log_2\left(\frac{1}{361}\right) = -\log_2\left(\frac{1}{361}\right) = 8.495898$$

이다. 이 의미는 바둑판의 교차점 개수가 361개라는 것이다. 따라서 이들을 정보로 구별하려면 필요한 비트 수가 8.49보다 큰 정수인 9이다. 실제로 교차점 수는 $256 < 361 < 512$이기 때문에 $2^8 = 256 < 361 < 512 = 2^9$으로서 엔트로피가 361개의 교차점을 구별해서 표시할 때 필요한 비트 수로 9를 얻게 된다.

이 정의를 뒤에 다루게 될 교차 엔트로피와 혼동하지 말기를 바란다. 결합 엔트로피와 교차 엔트로피를 구별하기 위해 결합 엔트로피에서는 $H(X, Y)$로 확률변수를 표시하였고, 뒤에 정의할 교차 엔트로피에서는 확률변수 자리에 해당 확률변수의 확률질량함수를 표기하여 넣었다. 그래서 $H(p, q)$라고 나타냈다.

조건부 엔트로피

주어진 확률변수 Y에 대하여 확률변수 X의 **조건부 엔트로피**(conditional entropy) 또는 조건부 불확실성은 Y상의 평균 조건부 엔트로피를 말하며 다음 수식 (13.34)와 같이 정의한다.

$$\begin{aligned} H(X|Y) &= E_Y[H(X|y)] \\ &= -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y)\log_2 p(x|y) \\ &= -\sum_{y \in Y} p(x, y)\log_2 p(x|y) \end{aligned} \tag{13.34}$$

수식 (13.34)에서 $p(x|y) = P(X = x|Y = y)$는 $Y = y$일 때 확률변수 $X = x$일 조건부 확률질량함수이다.

예를 들어보자. 하나의 주사위를 두 번 던지는 실험을 한다고 해보자. 표본 공간을 S라고 하면

$$S = \{(i, j)|i, j = 1, 2, 3, 4, 5, 6\}$$

이다. X를 첫 번째 주사위의 눈금이라고 하고 Y를 두 번째 주사위의 눈금이라고 한다. 그러면 결합 확률변수 (X, Y)의 확률분포 $p(X, Y)$는

$$\mathrm{p}(x, y) = P(X = x,\ Y = y) = \frac{1}{36},\ (x, y) \in S$$

이다. 조건부 확률질량함수 $p(x|y)$는 다음과 같다.

$$p(x|y) = P(X = x|\, Y = y) = \frac{P(X = x,\ Y = y)}{P(Y = y)} = \frac{\frac{1}{36}}{\frac{1}{6}} = \frac{1}{6}$$

확률변수 Y에 대한 확률변수 X의 조건부 엔트로피를 계산해보자.

$$H(X|\, Y) = -\sum_{x,\, y} p(x, y)\log_2 p(x|y)$$

$$= -\left\{ \sum_{(x,\, y) \in S} p(x, y)\log_2 p(x|y) \right\} = -36\frac{1}{36}\log_2\frac{1}{6} = -(-2.5849) = 2.5849$$

상호 정보

상호 정보(mutual information)는 두 개의 확률변수 중 한 확률변수를 관찰해서 나머지 다른 확률변수에 대해 얻을 수 있는 정보량을 측정하는 방법이다. 통신에서 매우 중요한 개념으로서 송신된 신호와 수신된 신호 사이에 공유되는 정보량을 최대화하는 데 사용된다. Y에 대한 X의 정보량인 상호 정보는 다음 수식 (13.35)와 같이 정의한다.

$$I(X;Y) = E_{X,\ Y}[SI(x, y)] = \sum_{x,\, y} p(x, y)\log_2\frac{p(x, y)}{p_X(x)p_Y(y)} \tag{13.35}$$

수식 (13.35)에서 SI(specific mutual information)는 특정 상호 정보로서 특정 값 (x, y)에 대한 상호 정보를 말한다.

상호 정보는 다음 수식 (13.36)에 나타낸 것과 같은 특성을 만족한다.

$$I(X;Y) = H(X) - H(X|\, Y) \tag{13.36}$$

즉, Y를 알고 있다면 Y를 모르고 있을 때와 비교해서 X를 인코딩할 때 평균적으로 $I(X;Y)$ 비트를 절약할 수 있다는 뜻이다. 상호 정보는 수식 (13.37)에 나타냈듯이 대칭적이다.

$$I(X;Y) = I(Y;X) = H(X) + H(Y) - H(X,\ Y) \tag{13.37}$$

상호 정보를 뒤에 배우게 될 KL-확산을 이용하여 나타낼 수도 있다. 즉, Y값이 주어진 뒤 X의

사후 확률 분산평균과 X의 사전분포 사이의 쿨백-라이블러 확산(Kullback-Leibler divergence)으로 표현이 가능하다.

$$I(X;Y) = E_{p(y)}[D_{KL}(p_X(X|Y=y) \| p_X(X))]$$

다시 말해서 Y값이 주어졌을 때, 평균적으로 얼마나 많이 X의 확률분포가 변할 수 있는지를 나타낸다. 이 값은 실제 결합분포에 주변분포를 곱하는 확산으로서 다시 계산할 수 있다. 즉,

$$I(X;Y) = D_{KL}(p(X,\ Y) \| p_X(X)p_Y(Y))$$

이다.

쿨백-라이블러 확산(정보 획득)

쿨백-라이블러 확산(Kullback-Leibler divergence)에 대해 알아보자. 다른 용어로 정보 확산, 정보 획득, 상대 엔트로피라고도 한다. **쿨백-라이블러** 확산을 KL 확산이라고 간단히 표기한다. 실제 확률분포인 $p(X)$와 임의의 확률분포인 $q(X)$ 두 개의 분포를 엔트로피를 이용하여 비교하는 방법이다. 데이터 분포가 $q(X)$라고 가정하고 데이터를 코딩한다고 해보자. 실제로 $p(X)$가 올바른 분포라고 한다면, **쿨백-라이블러** 확산은 코딩에 필요한 데이터 한 개마다 추가해야 하는 비트 수의 평균을 의미하는 값이다. 이를 다음 수식 (13.38)과 같이 정의한다.

$$\begin{aligned} D_{KL}(p(X)||q(X)) &= \sum_{x \in X} -p(x)\log_2 q(x) - \sum_{x \in X} -p(x)\log_2 p(x) \\ &= \sum_{x \in X} p(x)\log_2 \frac{p(x)}{q(x)} \end{aligned} \tag{13.38}$$

종종 이것을 **거리 측도**(distance metric)로 사용하기도 한다. 하지만 KL 확산은 실제 측도(Metric)는 아니다. 왜냐하면 대칭성도 가지고 있지 않고 삼각부등식도 만족하지 않기 때문이다.

KL 확산을 계산하는 예를 들어보자. 정상적인 주사위를 한 번 던졌을 때 나오는 눈금을 X라고 하자. 우리가 잘 알고 있듯이 주사위가 정상적이라면 모든 $x = 1, 2, 3, 4, 5, 6$에 대해

$$p(x) = P(X = x) = \frac{1}{6}$$

인 분포를 갖는다. 그래서 확률분포 $p(X)$는 다음 표와 같다.

x	1	2	3	4	5	6
$p(x)$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

그런데 해당 분포에 대한 정보를 정확히 알지 못하기 때문에 실험을 통해 분포를 추측해보게 된다. 주사위를 48번 던져서 눈금을 확인하였더니 다음 표와 같이 나왔다고 한다.

눈금	1	2	3	4	5	6
횟수	10	10	10	6	6	6

표본 데이터를 가지고 주사위의 확률분포 $q(X)$를 다음과 같이 가정하였다.

x	1	2	3	4	5	6
$q(x)$	$\frac{5}{24}$	$\frac{5}{24}$	$\frac{5}{24}$	$\frac{3}{24}$	$\frac{3}{24}$	$\frac{3}{24}$

분포가 $q(X)$라고 했을 때

$$
\begin{aligned}
&-\sum_x p(x)\log_2 q(x) \\
&= -\left\{\frac{1}{6}\log_2\left(\frac{5}{24}\right)+\frac{1}{6}\log_2\left(\frac{5}{24}\right)+\frac{1}{6}\log_2\left(\frac{5}{24}\right)+\frac{1}{6}\log_2\left(\frac{3}{24}\right)+\frac{1}{6}\log_2\left(\frac{3}{24}\right)+\frac{1}{6}\log_2\left(\frac{3}{24}\right)\right\} \\
&= -\frac{1}{2}\left\{\log_2\left(\frac{5}{24}\right)+\log_2\left(\frac{3}{24}\right)\right\} \\
&= \frac{1}{2}\{2.2630+3\} = 2.6315
\end{aligned}
$$

이고

$$
\begin{aligned}
&-\sum_x p(x)\log_2 p(x) \\
&= -\left\{\frac{1}{6}\log_2\left(\frac{1}{6}\right)+\frac{1}{6}\log_2\left(\frac{1}{6}\right)+\frac{1}{6}\log_2\left(\frac{1}{6}\right)+\frac{1}{6}\log_2\left(\frac{1}{6}\right)+\frac{1}{6}\log_2\left(\frac{1}{6}\right)+\frac{1}{6}\log_2\left(\frac{1}{6}\right)\right\} \\
&= -\left\{\log_2\left(\frac{1}{6}\right)\right\} \\
&= 2.5849
\end{aligned}
$$

이다. 따라서

$$
\begin{aligned}
D_{KL}(p(X)\|q(X)) &= \sum_{x\in X} -p(x)\log_2 q(x) - \sum_{x\in X} -p(x)\log_2 p(x) \\
&= \sum_{x\in X} p(x)\log_2\frac{p(x)}{q(x)} \\
&= 2.6315 - 2.5849 = 0.0466
\end{aligned}
$$

이 된다. 분포함수를 $q(X)$로 선택했기 때문에 발생하는 추가 비용에 해당하는 값이 0.0466이라는 의미이다. 이 수가 의미하는 것은 주사위 눈금의 종류가 6가지 상태이기 때문에 이를 비트를 이용해 구별하여 표현하고 통신망을 통해 전송하려면 $\log_2 6 = 2.5849$비트가 필요하다.

하지만 실제로 표본 데이터 수가 충분하지 않아서 모델링을 할 때 분포함수를 $p(X)$가 아닌 $q(X)$로 선택하였더니 6개의 정보를 구분해서 표현하기 위해서는 평균적으로 2.6315비트가 필요하다.

만약 제대로 모델링을 해서 확률분포가 $p(X)$였다면 주사위의 상태 정보를 1개 보내는 데 2.5849개의 비트가 필요하고 상태 정보 10000개를 보낸다면 25849비트가 필요하다. 하지만 $q(X)$를 분포함수로 선택하여 부정확한 모델링을 하였기 때문에 1개의 상태 정보를 보내는 데 실제로는 2.6315비트를 사용해야 하고 10000개의 상태 정보를 보내려면 26315비트를 사용해야 한다는 의미이다. 결국 통신에 사용되는 비트 수가 많아져서 통신 효율이 떨어진다.

일반적으로 *KL* 확산은 다음과 같은 특징이 있다.

(1) $D_{KL}(p(X) \| q(X)) \neq D_{KL}(q(X) \| p(X))$
(2) $D_{KL}(p(X) \| q(X)) \geq 0, (p = q)$

가장 이상적인 *KL* 확산 값은 0이다. 0일 경우 샘플링이 최적으로 되었다는 뜻이다. 또한 D_{KL} 값이 0일 때 교차 엔트로피값이 최소이며 그 값은 엔트로피이다. 실제로 교차 엔트로피 $H(p, q)$가 최소가 되는 때는 p와 q가 같을 때이다.

교차 엔트로피 계산

교차 엔트로피(cross entropy)는 예측 확률분포의 값과 실제 확률분포의 값이 얼마만큼 떨어져 있는지를 나타내는 척도이다.

동일한 사건집단에 분포하는 두 개의 확률분포 p와 q 사이의 교차 엔트로피란 해당 사건집단에 적용한 한 코딩 시스템이 실제 분포인 p가 아닌 추정 확률분포 q에 최적화되었다고 할 때, 이 사건집단에 속하는 한 사건을 찾는 데 필요한 평균 비트 수를 측정한다.

주어진 사건집단에 대한 분포 p에 상대적인 분포 q의 교차 엔트로피를 다음 수식 (13.39)와 같이 정의한다.

$$H(p, q) = -Ep[\log_2 q] \tag{13.39}$$

여기서 $E_p[\cdot]$은 분포 p에 대한 기댓값이다.

이 정의는 p와 q로부터 정의되는 쿨백-라이블러 확산인 $D_{KL}(p \| q)$를 이용하여 다음과 같이 나

타낼 수도 있다.

$$H(p, q) = H(p) - D_{KL}(p \parallel q)$$

$H(p)$는 p의 엔트로피이다. 두 개의 분포 p와 q가 동일한 사건집단에 정의되는 이산 확률분포라고 한다면, 교차 엔트로피는

$$H(p, \mathrm{q}) = -\sum_x p(x)\log_2 q(x)$$

이다. 교차 엔트로피를 최소화하는 것은 가능도를 최대화하는 것과 같다. 즉, 교차 엔트로피가 작다는 뜻은 선택한 확률분포가 원래 분포에 매우 가까워서 타당한 선택이라는 뜻이다.

예를 들어 교차 엔트로피를 이해해보자. 앞절의 쿨백-라이블러 확산에서 들었던 예제를 다시 생각해보자. 확률변수 X의 분포는 균등분포 $p(X)$로서 확률질량함수 $p(x)$는 다음과 같다.

x	1	2	3	4	5	6
$p(x)$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

확률변수 Y의 분포 $q(X)$는 다음과 같은 확률질량함수 $q(x)$를 가진다.

x	1	2	3	4	5	6
$q(x)$	$\frac{5}{24}$	$\frac{5}{24}$	$\frac{5}{24}$	$\frac{3}{24}$	$\frac{3}{24}$	$\frac{3}{24}$

교차 엔트로피 $H(p, q)$는

$$\begin{aligned} H(p, q) &= -\sum_x p(x)\log_2 q(x) \\ &= -\left\{\frac{1}{6}\log_2\frac{5}{24} + \frac{1}{6}\log_2\frac{5}{24} + \frac{1}{6}\log_2\frac{5}{24} + \frac{1}{6}\log_2\frac{3}{24} + \frac{1}{6}\log_2\frac{3}{24} + \frac{1}{6}\log_2\frac{3}{24}\right\} \\ &= \frac{1}{2}\{2.2630 + 3\} = 2.6315 \end{aligned}$$

이었다. 또한 $H(p)$와 $H(p, q)$ 그리고 $D_{KL}(p \parallel q)$를 각각 계산해보면

$$H(p) \equiv -\sum_x p(x)\log_2 p(x) = -6\left\{\frac{1}{6}\log_2\frac{1}{6}\right\} = 2.5849$$

$$H(p, q) = H(p) - D_{KL}(p \parallel q)$$

$$D_{KL}(p \parallel q) = 0.0466$$

이 성립한다는 것을 알 수 있다.

도전문제

문제 13.1

확률변수 X의 분포는 균등분포 $p(X)$로서 확률질량함수 $p(x)$가 다음과 같다.

x	1	2	3	4	5	6
$p(x)$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

확률변수 Y의 분포 $q(X)$는 다음과 같은 확률질량함수 $q(x)$를 가진다.

x	1	2	3	4	5	6
$q(x)$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

교차 엔트로피 $H(p, q)$를 구하시오.

풀이

교차 엔트로피의 정의로 구해보자.

$$\begin{aligned} H(p, q) &= -\sum_x p(x)\log_2 q(x) \\ &= -\left\{\frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{1}{6}\log_2\frac{1}{6}\right\} \\ &= 2.5847 \end{aligned}$$

교차 엔트로피 $H(p, q)$가 최소가 되는 때는 p와 q가 같을 때이다. 그러므로 교차 엔트로피인 2.5847이 가장 작은 교차 엔트로피이다.

정리

장에서는 조건부 확률의 개념을 설명하였다. 두 개의 확률이 동일한 표본 공간에 정의되었을 경우 한 사건이 발생한 사실이 다른 사건의 발생 확률에 영향을 미치게 될 경우 이러한 확률값을 조건부 확률이라고 한다.

한편 통계학에서는 기본적으로 빈도에 기반한 통계학이 있는 반면에 베이즈의 정리에 기반한 통계학이 있다. 빈도 기반 통계학에서는 확률변수와 분포함수가 미리 결정된 뒤에 특정 사건이 발생할 확률을 알아본다면, 베이즈의 정리 기반 통계학에서는 사건이 발생한 정도를 토대로 어떠한 분포로부터 해당 표본이 추출되었는지를 찾아간다. 따라서 베이지안 통계학은 사건이 발생한 후에 추론을 수정해나가는 형식의 통계학이다. 표본 자료가 많으면 많을수록 모수가 미지수인 특정 분포의 모수를 정확하게 근사시켜 나갈 수 있다. 이전의 경험과 현재의 증거를 토대로 어떤 사건의 확률을 추론하는데 이는 사전 정보를 바탕으로 어떤 사건이 일어날 확률을 토대로 의사결정을 할 때 활용된다. 베이즈의 정리를 도입하기 위하여 전확률의 개념을 설명하였고, 이를 통해 분할을 이용한 베이즈의 정리를 유도하였다.

가능도함수를 정의하였고 이론적으로 가장 가능성이 높은 모수를 찾는 방법인 최대 가능도 추정법에 대해 알아보았다. 최대 가능도 추정법은 모든 추정방법 중 가장 널리 사용되는 방법이다. 이를 위해 가능도함수의 개념을 직관적으로 이해할 수 있는 사례를 들었다. 베르누이분포, 이항분포, 정규분포 등 여러 기본 분포의 모수를 최대 가능도 추정법으로 추정하는 방법을 공부하였다. 최대 가능도 방법이란 어떤 확률변수에서 표집한 값들을 토대로 그 확률변수의 모수를 구하는 방법이다. 어떤 모수가 주어졌을 때, 원하는 값들이 나올 가능도를 최대로 만드는 모수를 선택하는 방법이다. 추정 통계학 관점에서 보면 이는 점추정 방식의 하나이다.

정보 이론의 기본적인 개념인 정보량을 소개하였다. 정보량은 인공지능에서 자주 등장하는 개념으로서 물리학에서는 에너지가 전체에 걸쳐 평평하게 흩어져가는 과정이 엔트로피의 증가과정이라고 본다. 즉, 엔트로피의 증가는 평형 상태로의 이동이며 에너지적으로 볼 때는 안정화되는 방향이다. 하지만 인공지능에서는 엔트로피가 감소하는 방향으로 학습을 최적화시킨다. 인공지능에서 정보 이론을 이용하여 인공지능 네트워크를 최적화해나간다. 정보를 확률의 로그 함수로 표현했다. 예를 들어 확률이 50%인 $P(X = x) = 1/2$이라면 정보량 $I = -\log_2(1/2) = 1$이 된다. 그 정보를 1비트의 2진수로 표현할 수 있다는 뜻이다.

정보량의 기댓값을 엔트로피라고 하고 이들을 어떻게 구하는지에 대해 다양한 예를 들어 설명하였다. 또한 상대 엔트로피, 결합 엔트로피, 조건부 엔트로피, 상호 정보 등에 대해 학습하였

으며, 쿨백-라이블러 확산과 교차 엔트로피의 개념을 설명하였다.

인공지능에서는 지도 학습과 비지도 학습을 하는데 지도학습의 경우 비용함수를 최소화하기 위해 인공지능 네트워크변수들을 정해나가는 학습을 한다. 이때 비용함수의 선택에 따라 학습의 속도나 정확성이 달라진다. 일반적으로 제곱오차함수를 비용함수로 사용하지만 정보 이론에서 제시하는 엔트로피함수를 비용함수로 사용하면 학습 속도가 더 빨라진다. 즉, 혼란을 최소화하고 분명한 결과를 내기 위해 엔트로피를 최소화하는 방식으로 알고리즘을 학습시킨다.

연습문제

01 어떤 병에 걸릴 확률이 0.4%이다. 즉, 1000명 중 4명이 걸린다. 병원에서 이 병에 걸렸는지 진단하는 키트를 개발했다. 키트의 검사 정확도는 실제 병에 걸렸고 병이 있다고 진단할 확률이 95%이고, 실제로 병에 걸리지 않았고 병이 없다고 진단할 확률은 98%라고 한다. 실제로 어떤 사람의 검사 결과 병이 있는 것으로 양성 판정을 받았다고 한다. 이 사람이 실제로 병에 걸렸을 확률은 얼마인가?[3]

02 문제 1번에서 키트의 오진 판정률을 낮추고 정확도를 높이려면 다음 표에 나타낸 2가지 확률의 값을 낮춰야 한다.

병에 걸리지 않았는데 양성으로 판정하는 확률	$P(B\|A^c)$
병에 걸렸는데 음성으로 판정하는 확률	$P(C\|A)$

(1) 1번 문제에서 $P(A) = 0.004$이고 $P(B|A^c) = 0.02$이었다. '병에 걸리지 않았는데 양성으로 판정하는 확률'인 $P(B|A^c)$을 10배 개선하여 $P(B|A^c) = 0.002$가 되었다면 '양성 판정을 받았을 때 실제로 병에 걸렸을 확률'인 $P(A|B)$의 값은 얼마인가?

(2) 1번 문제에서 $P(A) = 0.004$이고 $P(C|A) = 0.05$이었다. '병에 걸렸는데 음성으로 판정하는 확률'인 $P(C|A)$를 10배 개선하여 $P(C|A) = 0.005$가 되었다면 '양성 판정을 받았을 때 실제로 병에 걸렸을 확률'인 $P(A|B)$의 값은 얼마인가?

(3) 문제 (2)에서 '병에 걸렸는데 음성으로 판정하는 확률'인 $P(C|A)$를 100배 개선하여 $P(C|A) = 0.0005$가 되었다면 '양성 판정을 받았을 때 실제로 병에 걸렸을 확률'인 $P(A|B)$의 값은 얼마인가?

(4) 문제 (1), (2), (3)의 결과를 가지고 키트의 정확도인 $P(A|B)$를 높이기 위해서 '병에 걸리지 않았는데 양성으로 판정하는 확률'인 $P(B|A^c)$의 확률과 '병에 걸렸는데 음성으로 판정하는 확률'인 $P(C|A)$의 확률 중 어느 쪽을 개선하는 것이 중요한가?

03 문제 01번과 동일한 상황을 생각해보자. 문제 01번에서는 병에 걸릴 확률 $P(A) = 0.004$로 주어졌는데 이 확률이 1%, 2%, 5%, 10%로 높아질 경우에 키트의 정확도인 $P(A|B)$의 값이 어떻게 변화하는지 계산하고 결과를 가지고 무엇을 알 수 있는지 설명하시오.

[3] 문제 참조 자료: 병 진단 오진율과 조건부 확률(이승훈 교수의 실용수학;경문사); https://horizon.kias.re.kr/7536/

04 살인사건의 용의자 3명이 잡혔다. 3명 중 1명이 진범이 분명하지만, 3명 모두가 범행을 완강히 부인하고 있어 진범을 못 찾고 있는 가운데 국과수에서 연락이 왔다. 범죄 현장에서 살인자의 것으로 추정되는 *DNA*가 발견됐고 용의자 중 1명의 *DNA*와 일치한다고 하며, 이런 일치는 100만 명 중 1명꼴이라고 한다. 해당 용의자가 진범일 확률은 얼마인가?[4]

05 [몬티 홀 문제] 빨강, 노랑, 파랑 3개의 문이 있다. 이 중 하나의 문 뒤에만 자동차 상품이 있다. 당신은 단 하나의 문만 열 수 있고, 열어서 자동차가 있다면 당신 것이다. 당신이 빨간색 문을 가리켰다. 어느 문에 자동차 상품이 들어있는지 알고 있는 사회자가 노란색 문을 열어 보이며 자동차 상품이 없음을 보여준다. 당신은 선택을 바꿔 파랑색 문을 선택할 것인가? 아니면 그대로 빨강색 문을 고수할 것인가? 확률적으로 어느 쪽이 유리한지 설명하시오.

06 주머니 속에 붉은색 공과 파란색 공이 합해서 5개가 들어 있다고 한다. 이 주머니에서 1개의 공을 꺼내서 색을 확인하는 실험을 한다고 해보자. 붉은색 공을 꺼내면 성공이라고 하고 베르누이 확률변수 X가 1을 갖는다고 한다. 파란색 공을 꺼내면 실패라고 하고 X는 0을 갖는다고 한다. 성공 확률을 μ_0라고 하면 $X \sim B(1, \mu_0)$인 베르누이분포이다.

(1) 베르누이분포의 확률질량함수 $p(x, \mu_0)$을 구하시오.

(2) 공을 꺼내는 실험을 1회 실시하여 파란색 공을 꺼냈을 때 가능도함수 $L(\mu, 0)$을 구하시오.

(3) 공을 꺼내는 실험을 5회 실시한 결과가 다음과 같다.

$$x_1 = 1,\ x_2 = 1,\ x_3 = 1,\ x_4 = 0,\ x_5 = 0$$

가능도함수 $L(\mu, x_1, x_2, x_3, x_4, x_5)$를 구하시오.

(4) 문제 (3)의 결과를 이용하여 μ값을 무엇으로 추정할 수 있는가?

07 정규분포를 갖는 확률변수 $X \sim N(\mu_0, \sigma^2)$의 확률밀도함수는 1개의 변수 x의 함수로 아래와 같다.

$$f(x; \mu_0, \sigma_0^2) = \frac{1}{\sigma_0\sqrt{2\pi}}\exp\left(-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right) \qquad (*)$$

표본이 x_0일 때, 가능도함수 $L(\mu, \sigma^2\ ; x_0)$은 입력변수가 μ와 σ^2인 이변수함수이다.

[4] 문제 참조 자료:(샤이냅소프트 블로그);https://www.synapsoft.co.kr/blog/6002

(1) 가능도함수가 어떻게 표현되는지 나타내시오.

(2) 수식 (*)에 표현한 확률밀도함수와 문제 (1)에서 구한 가능도함수를 비교하고 어떤 차이가 있는지 설명하시오.

(3) 문제 (2)에서 $x_0 = 0$이고, $\sigma^2 = 1$일 때, 가능도함수를 나타내고 이를 변수 μ의 함수로 그래프를 그려보시오.

(4) 문제 (2)에서 $x_0 = 0$일 때, 가능도함수를 나타내고 이를 파이썬 코드로 변수 μ와 σ^2의 함수로 그래프를 그려보시오.

08 다음과 같은 확률변수 X가 있다.

$$P(X = -1) = 0.3,\ P(X = 0) = 0.2,\ P(X = 1) = 0.5$$

확률변수 X의 엔트로피를 계산하시오.

09 확률변수 X가 이산 확률변수로서 $x_1, \ldots, x_5$을 함숫값으로 갖고 $p(x_i) = i/15$일 때, 엔트로피를 구하시오.

10 데이터가 60개가 있고 그중 $X = 0$인 데이터가 20개, $X = 1$인 데이터가 40개 있는 경우 엔트로피를 계산하시오. 단, 이론적인 확률밀도함수가 없고 실제 표본 데이터만 주어진 경우에는 데이터에서 확률질량함수를 추정한 후, 이를 기반으로 엔트로피를 계산한다.

11 두 개의 확률변수 X와 Y의 확률질량함수가 다음 표와 같이 주어졌다.

X \ Y	0	1
0	0.25	0.25
1	0.4	0.1

(1) 확률변수 X, Y의 결합 엔트로피 $H(X, Y)$를 구하시오.

(2) 확률변수 X가 주어졌을 때, Y의 엔트로피인 $H(Y|X)$를 구하시오.

(3) Y에 대한 X의 정보량인 상호 정보 $I(X; Y)$를 구하시오.

14 인공지능과 수학

학습목표

- 인공지능에 대해 개괄적으로 이해한다.
- 회귀 분석의 개념을 이해하고 인공지능과 연관성을 알아본다.
- 신경망을 모방한 인공신경망의 개념을 알아본다.
- 인공신경망의 활성화함수와 다층신경망으로의 발전과정을 알아본다.
- 딥러닝의 학습, 오차, 회귀, 분류 등의 연관된 개념을 알아본다.
- 인공지능과 수학의 역할에 대해 알아본다.
- 인공지능을 위해 효과적인 수학 교육 과정과 교육 방법이 무엇인지 알아본다.

학습목차

14.1 인공지능
14.2 회귀 분석
14.3 인공신경망
14.4 딥러닝
14.5 인공지능에서 수학의 역할

14.1 인공지능

인공지능을 한마디로 정의하기는 매우 어렵지만 많은 데이터를 분석하여 자료를 정리하고 그 속에서 의미를 찾아낸 뒤 스스로 결정하는 기술이라고 말할 수 있다. 다시 말해서 많은 자료를 통해 학습하고 이를 토대로 새로운 입력에 해당하는 결과를 도출하려는 노력을 인공지능이라고 할 수 있다. 따라서 인공지능에서는 기존에 확보한 많은 자료로부터 필요한 자료를 검색하는 일과 효율적으로 빠르게 학습하는 알고리즘이 핵심이다. 사람이 지식을 학습하는 방식을 모방하여 기계가 학습하도록 하는 기술이 머신러닝이다. 딥러닝은 심층 학습이라고도 하고 여러 가지 비선형 변환기법을 조합하여 높은 수준의 추상화를 시도하는 기계 학습 알고리즘의 집합이다. 머신러닝의 한 영역으로서 인간의 두뇌가 데이터를 처리하고 의사결정을 위해 패턴을 만들어내는 과정을 모방하는 기능이다. 딥러닝은 목록화되지 않았거나 구조화되지 않은 데이터로부터 비지도 방식의 학습능력을 가진 네트워크이다. 비지도 학습을 통해 스스로 사물을 구분하는 능력을 갖추기 위한 노력이다. 딥러닝은 *DNN*, *CNN*, *RNN* 등의 딥러닝 기법을 컴퓨터 비전, 음성 인식, 자연어 처리, 신호 처리 등의 분야에 적용하여 물체를 탐지하거나 대화를 인식하거나 언어를 번역하거나 미흡한 인간의 의사결정 과정을 보완하여 의사결정을 하는 인간 뇌 기능을 모방한다.

인공지능과 머신러닝 그리고 딥러닝의 관계를 다음 그림 14.1과 같이 나타낼 수 있다.

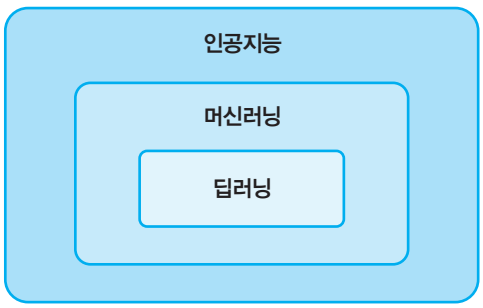


그림 14.1 인공지능과 머신러닝과 딥러닝

머신러닝

머신러닝은 경험을 통해 자동으로 개선해나갈 수 있는 컴퓨터 알고리즘을 탐구하는 연구영역이다. 이를 통해 훈련 데이터라고 알려진 샘플 데이터를 분석하여 자동으로 분석적 모델을 구

축할 수 있다. 프로그래밍 없이 시스템이 데이터를 학습할 수 있고, 패턴을 인식할 수 있고, 인간의 간섭을 최소화한 상태에서 의사결정 능력을 만들어낼 수 있다고 믿는 인공지능에 기반한 학문 분야이다. 단순한 프로세스가 아니고 알고리즘을 통해 학습 데이터를 수집한 후에 해당 데이터에 기반해서 더 정확한 모델을 구축한다. 머신러닝 모델은 데이터를 이용하여 머신러닝 알고리즘을 학습시키는 과정에 필요한 모델이다. 학습을 마친 후 모델에 알고 싶은 입력 내용을 제공하면 결과물을 출력으로 내보낸다.

사람이 컴퓨터와 체스 게임을 하거나 바둑을 두고, 자동차가 스스로 운전하고, 환자의 질환을 진단하고 최적의 치료법까지 제시하는 컴퓨터 등 다양한 분야에서 인공지능은 활용 영역을 확대해나가고 있다. 인공지능에서 머신러닝이란 주어진 조건 속에서 올바른 결정을 하기 위해 끊임없이 계산하는 과정이다. 인공지능이 결정한 다양한 선택은 데이터 분석을 통한 머신러닝의 결과이다.

머신러닝의 일부로서 밀접하게 연관된 분야가 컴퓨터를 이용한 예측이 목적인 계산 통계학이다. 하지만 머신러닝이 모두 통계학적 학습만 하지 않는다. 머신러닝에서는 수학적 최적화 방법이나 이론과 응용을 이용하여 학습한다. 대규모로 축적한 데이터 안에서 체계적이고 자동적으로 데이터를 탐색하고 분석하여 통계적 규칙이나 패턴을 찾는 방식을 데이터 마이닝이라고 한다. 비지도 학습을 통한 탐색 데이터 분석에 집중하는 데이터 마이닝도 머신러닝과 관계가 깊다. 비즈니스 문제해결에 필요한 예측 분석도 머신러닝의 중요한 일부분이다.

딥러닝

인공신경망(ANN: Artificial Neural Network)과 표현 학습에 기초한 머신러닝 방법을 보통 딥러닝이라고 한다. 학습은 지도 학습, 비지도 학습 및 부분 지도 학습 등으로 분류한다. 응용 분야로는 컴퓨터 비전, 기계 비전, 음성 인식, 자연어 처리, 소셜 네트워크 필터링, 머신 번역, 생물 정보학, 의학 영상 분석, 보드 게임 프로그램, 물질 검사, 정보보안 등의 영역이다. 딥러닝은 **딥신경망**(DNN: Deep Neural Networks), **딥 빌리프망**(DBN: Deep Belief Networks), **재귀신경망**(RNN: Recurrent Neural Networks), **합성곱신경망**(CNN: Convolutional Neural Networks) 등의 기법으로 구현한다.

초기 인공지능의 발달과정을 살펴보면 인공신경망이라는 말은 생물학적 신경전달 시스템인 뉴런 안의 분산통신 노드와 정보처리과정을 모방한 인공적인 네트워크라는 의미로 사용하기 시작했다. 초기 인공신경망인 퍼셉트론으로 해결할 수 없었던 문제를 해결하기 위해 인공신경망의 네트워크층을 여러 층으로 늘려나가면서 해결책을 찾을 수 있었다. 이렇게 만들어진 여러 개의 층으로 구성된 인공신경망의 이름을 자연스럽게 '다층인공신경망'이라고 해도 무방했지만, 당시 인공지능에 대한 여론이 좋지 못했고 인공신경망이라는 이름 자체도 인기가 없어서

기존 방식과는 다른 이름을 사용할 필요성이 있어 다층인공신경망이라는 용어 대신에 '딥러닝'이라는 용어를 사용하였다. 과거 실패한 인공지능 세대의 용어였던 '인공신경망' 대신에 필요했던 용어로서 여러 개의 층으로 구성되어 있다는 '복수(multiple)' 대신 '깊다(deep)'라는 용어를 사용하기 시작한 것이 딥러닝이라는 용어를 사용하기 시작하게 된 이유이다.

초기 딥러닝의 발달과정을 살펴보면 두뇌의 인지 능력을 모방하도록 만든 인위적인 선형적 네트워크인 퍼셉트론만 가지고는 *XOR* 논리 문제 같은 아주 간단한 문제조차 해결할 수 없다는 비판을 받았고 이를 계기로 인공지능 연구가 침체하는 계기가 되었다. 무제한의 폭을 가진 한 개의 은닉된 층에 비 다항적 활성화 기능을 갖춘 네트워크가 단일 선형 인공신경망이 가졌던 보편적 문제를 해결할 수 있을 거라고 예상했고 머신러닝을 그 방향으로 발전시켜가면서 *XOR* 문제뿐만 아니라 훨씬 복잡한 문제도 해결하면서 지금의 딥러닝 단계까지 발전해왔다. 적절한 조건하에 이론적으로는 보편성을 유지하면서도 실제 응용이나 최적화된 구현이 가능한 유한 크기의 무제한 층수를 가진 현대판 버전이 바로 딥러닝이다.

물론 딥러닝이 앞으로 또 어떤 난관에 부딪힐지 알 수 없다. 하지만 지금까지 인공지능의 많은 문제를 딥러닝 방식으로 해결해나가고 있으며 컴퓨팅 계산 능력의 발달과 함께 특정 영역에서는 매우 정확한 예측을 할 수 있는 성숙단계에 이르고 있다.

지도 학습과 비지도 학습

머신러닝에서 학습이 매우 중요하다고 했다. 머신러닝에서 활용하고 있는 학습 방법을 살펴보면 **지도 학습**(supervised learning)과 **비지도 학습**(unsupervised learning)과 **강화 학습**(reinforcement learning)이 있다. 지도 학습이란 인간이 학습하는 방식처럼 결과에 맞게 미리 분류된 데이터들을 기계에 제공하고 이를 통해 기계를 학습시키는 방법이다. 예를 들어 컴퓨터에 다양한 고양이 사진 데이터를 제공하고 '이것은 고양이 사진'이라고 정보를 주며 학습하는 방식이다. 반면에 고양이 사진, 개 사진, 새 사진 집합으로 구성된 데이터를 컴퓨터에 제공하고 컴퓨터가 스스로 이들을 분류하도록 학습시키는 방식을 비지도 학습이라고 한다. 학습한다는 의미는 각 사진의 특성을 파악하여 유사한 사진끼리 클러스터링하여 분류하는 능력을 갖추게 하는 것이다. 강화 학습이란 지도 학습이나 비지도 학습과는 조금 다른 개념이다. 분류할 수 있는 데이터가 아예 존재하지 않으며, 데이터가 있다고 해도 정답이 따로 정해져 있지 않다. 다만 기계 자신이 한 행동에 대해 보상(reward)을 제공하여 더 나은 결과를 예측하도록 학습시키는 방법이다. 강화 학습의 대표적 사례가 알파고(AlphaGo)이다.

머신러닝은 지도 학습과 비지도 학습을 통해 기계 학습을 하게 되는데 지도 학습은 다시 회귀(regression)와 분류(classification)로 구분한다고 했다. 반면에 비지도 학습은 군집화(clustering)와 차원축소(dimensionality deduction)으로 나눈다. 강화 학습은 실시간 결정, 로봇 네비

게이션, 인공지능 게임 등에 적용된다.

지도 학습에서 회귀란 연속된 값인 실수치를 예측하는 것이다. 예를 들어 어떤 사람의 교육 정도에 따라 연봉을 예측해보는 경우 연봉이 실숫값이고, 영양 상태에 따른 신장의 크기도 실숫값을 갖는 예측이므로 이는 회귀 문제라고 할 수 있다. 물론 연봉이나 키를 양의 정수 단위 숫자로 기록하지만 정수가 아닌 실숫값일 수도 있으므로 실수 전체 집합이라고 가정할 수 있다. 반면에 분류는 미리 정의된 가능성이 있는 여러 개의 클래스 레이블 중에서 하나에 속하는지 아닌지를 예측하는 것이다. 예를 들어 스팸메일을 분류한다고 할 때 해당 메일이 스팸메일이거나 아니거나 둘 중의 한 클래스에 속하게 된다. 이것도 저것도 아닌 메일은 없기 때문에 이는 2개의 클래스로 분류하는 비지도 학습 사례이다. 앞에서 예로 들었던 고양이와 개와 새를 구별해내는 경우는 3개의 클래스에 속하도록 분류하는 비지도 학습의 사례였다.

강화 학습에서는 환경으로부터 얻는 피드백을 통해 행위자(agent)의 행동을 분석하고 최적화한다. 최고의 보상을 획득하는 활동을 찾기 위해 시행착오 기법을 활용하며 규칙을 별도로 프로그래밍에 명시하지 않고 최고의 보상을 획득하는 방법을 찾아가는 방식으로 학습시킨다. 학습 방법은 어떤 행동의 학습 횟수를 초과 달성할 때마다 보상으로 높은 점수를 부여하는 전략이다.

14.2 회귀 분석

회귀 문제로 돌아가서 선형회귀 분석에 대해 말해보자. 예측하고 싶은 종속변수가 숫자라면 보통 회귀라는 머신러닝의 지도 학습 방법을 사용한다. 예를 들어 날씨와 아이스크림 판매량 간의 관계를 살펴보면 판매량이 실수 숫자로 나타나며 특정 값만 갖지 않고 연속적인 값을 갖고 있다고 말할 수 있다.

우리가 가장 간단하게 머신러닝을 이해할 수 있는 지도 학습에서 회귀 문제의 예는 통계학에서 구축해놓은 선형회귀 분석이다. 사실 **회귀**(regression)라는 말의 원래 뜻이 '돌아가다'는 의미이다. 왜 이런 의미의 용어를 사용했는지 직관적으로 와닿지 않는다. 왜 회귀라는 말이 나오게 되었는지 알아보자. 부모의 키와 자녀의 키 사이의 관계를 나타내는 자료를 연구하면서 자녀의 키가 부모의 키와 연관된다는 사실을 밝혀냈다. 이때 부모의 키가 양 극단치로 갈수록 자녀의 키는 인간의 평균키로 돌아가려는 성질이 있다고 결론을 내렸다. 따라서 회귀의 정확한 의미는 '평균으로의 회귀'를 의미한다. 예를 들어 영양상태와 신장 간의 관계를 파악하고자 할 때 영양상태와 신장으로 이루어진 순서쌍 형태로 이루어진 표본 데이터 집합을 대표하는 모델로서 직선을 하나 설정한다. 그리고 이 모델 직선상의 데이터와 표본 데이터 간의 차이를 오차라고 하자. 만약 모델 직선이 사용한 표본 데이터뿐만 아니라 미지의 데이터에 대한 예측까지 올바르게 할 수 있는 직선으로 설정되려면, 오차의 평균이 0이 돼야 한다. 그래서 '오차 평균이 0으로 회귀하다'라는 의미로서 '회귀'라는 용어를 사용한다. 통계학의 회귀 분석에서 사용해오던 '회귀'라는 용어를 인공지능 지도 학습에서 목표치로서 결과가 실숫값일 때를 지칭하는 용어로 사용하게 되었다.

회귀 분석의 분류

한 개의 설명변수를 이용하여 하나의 종속변수를 예측하기 위한 통계적 기법을 **단순회귀 분석**(simple linear regression)이라고 한다. 단순회귀 분석은 설명변수가 하나이고 사용하는 다항식의 차수에 따라 1차인 직선으로 회귀 모형을 만드는 **선형회귀 분석**(linear regression)과 이차 이상의 다항식을 이용한 **다항회귀 분석**(ploynomial regression)으로 분류할 수 있다.

복수개의 설명변수를 이용하여 하나의 종속변수를 예측하기 위한 통계적 기법을 다중회귀 분석(multiple regression)이라고 한다. 다중회귀 분석도 사용하는 다항식의 차수가 일차인 평면으로 회귀 모형을 만드는 **다변수 선형회귀 분석**(MLR: multivariate linear regression)과 이차

이상의 다항식을 이용한 **다변수 다중회귀 분석**(MMR: multivariate multiple regression)으로 분류할 수 있다.

위의 분류에서 단순회귀 분석이던 다중회귀 분석이던 선형회귀 분석이라는 의미는 모델이 직선(선형회귀 분석)이나 평면(이변수 선형회귀 분석) 또는 더 높은 차원의 공간에서 종속변수가 설명변수들의 선형 표현으로 정의되는 모델을 구하는 통계적 기법을 의미한다.

반면에 단순회귀 분석의 한 종류인 다항회귀 분석이나 다중회귀 분석의 한 종류인 다변수 다중회귀 분석은 사용되는 다항식이 차수가 이차 이상인 비선형 표현으로 정의되는 모델을 구하는 통계적 기법을 말한다.

예를 들어 이차 다항회귀 분석에서는 단일변수로 포물선 형태를 갖는 모델을 구축하지만 이변수 선형회귀 분석에서는 2개의 변수로 평면 형태를 갖는 모델을 구축한다.

선형회귀 분석

데이터 분석의 가장 기초가 되는 내용 중 하나로 다항식의 차수가 일차인 직선으로 회귀 분석을 하는 단순회귀 분석 중 하나인 **선형회귀 분석**(linear regression)에 대해 알아보자. 선형회귀 분석에서는 한 개의 확률변수 또는 복수개의 확률변수들과 이들이 실현되는 스칼라값 간의 관계를 모델링 하는 데 선형적인 방법을 사용한다. 모델링 목적은 현재까지 알고 있는 데이터를 분석하여 모르는 값을 예측하는 것이다. 기존에 알고 있는 데이터들의 특성을 반영한 모델을 만들고 그 모델에 미래에 발생할 현상을 입력하여 출력되는 미지의 값을 알아낸다. 다음 그림 14.2는 선형회귀 분석을 한 결과이다.[1]

선형회귀 분석의 경우 주어진 데이터 집합은 $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$으로서 종속변수 y는 독립변수 x와 선형적인 상관관계를 이루고 있다고 가정하고 이상적인 선형성을 알아내는 방법이다. x_i들을 설명변수라고 하고 y_i들을 응답변수라고 한다. 이들의 관계가 직선 형태를 갖고 있다고 가정하고 이를 모델링하여 알려지지 않은 새 설명변수 데이터로부터 응답변수를 예측하는 것이 목적이다.

쉽게 말해서 이미 알고 있는 좌표들의 집합인 $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$을 보고 이들 좌표점들을 최대한 가깝게 지나가는 하나의 직선을 그린다면 나머지 좌푯값도 직선에 가깝게 위치할 것이라는 가정을 하고 예측하는 방법이 바로 회귀 분석 방법이다(그림 14.2 참조).

회귀 분석은 모델이 데이터의 경향을 학습하기 위한 머신러닝의 일종이다. 가장 간단한 회귀 분석은 직선 $y = ax + b$를 데이터에 적용하는 방식이다. 차수를 하나 늘려서 이차 함수에 의한

[1] 위키피디아 참조: By Sewaqu - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid = 11967659

회귀 분석, 삼차 다항식에 의한 회귀 분석 등 다항식 회귀 분석을 하여 머신러닝을 실시할 수 있다. 분석하고자 하는 데이터의 설명변수와 응답변수가 선형적 관계가 아니라 곡선 형태로 나타나 있다면 선형성을 가지고 있을 거라고 가정하고 선형회귀 모델화한다면 분명 오차가 커지게 될 것이다.

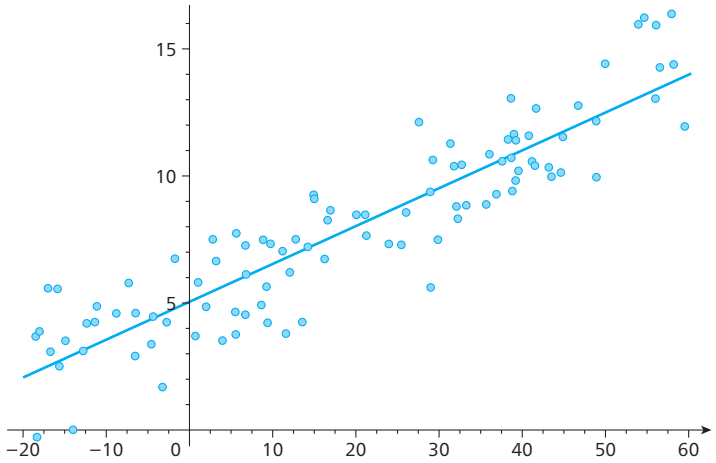


그림 14.2 선형회귀 분석 사례

분석하고자 하는 데이터를 그림으로 나타냈을 때 2차원 곡선인 포물선의 형태를 갖추고 있다면 2차원 곡선으로 모델화하는 것이 더 정확한 모델이 될 것이고, 3차원 곡선이 더 올바를 거라고 생각한다면 3차원 곡선으로 모델화하는 것이 적합하다. 주어진 데이터의 형태에 따라 해당 데이터를 다항식의 형태로 모델화할 수 있다.

다음 그림 14.3은 이차 다항식 $y = ax^2 + bx + c$를 이용하여 이차 다항회귀를 한 경우와 일차 다항식인 직선을 이용한 단순회귀를 비교한 그림이다.

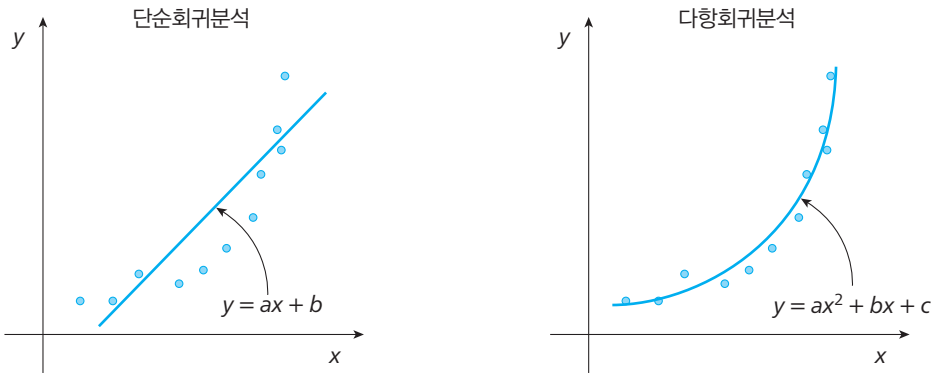


그림 14.3 단순회귀 분석과 다항회귀 분석

다중회귀 분석

다중회귀 분석이란 설명변수가 2개 이상인 회귀 모형을 분석 대상으로 삼는 회귀 분석이다. 예를 들어 설명변수가 2개이고 응답변수는 1개인 경우 이들이 선형관계에 있다고 가정하고 회귀분석을 하는 통계적 기법을 다중선형회귀 분석이라고 한다. 설명변수는 $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$이고, 응답변수는 y_i라고 표현할 때 다중선형회귀 분석 모델은 수식 (14.1)과 같이 나타낼 수 있다.

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \varepsilon_i = \boldsymbol{x}_i^T \boldsymbol{\beta} + \varepsilon_i,\ i = 1, \ldots, n \tag{14.1}$$

여기서 $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})$, $i = 1, \ldots, n$이다. T는 행과 열을 바꿔놓은 전환행렬을 의미한다. 따라서 수식 (14.1) $\boldsymbol{x}_i^T \boldsymbol{\beta}$는 벡터 $\boldsymbol{x}_i$와 $\boldsymbol{\beta}$의 내적이다. 실제로 수식 (14.1)은 n개의 방정식으로 이루어진

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{14.2}$$

이다. 수식 (14.2)에서

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{pmatrix} = \begin{pmatrix} 1 & x_{12} & \cdots & x_{1p} \\ 1 & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix},\ \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

이다. 다중회귀 분석을 위한 가정으로는 다음과 같은 조건이 필요하다.

- 독립변수 $\boldsymbol{x}_i^T = (x_{i1}, \ldots, x_{ip})$들과 종속변수 y_i들 사이에 선형관계가 있어야 한다.
- 독립변수 $\boldsymbol{x}_i^T = (x_{i1}, \ldots, x_{ip})$들끼리는 상관관계가 너무 커서는 안 된다.
- 모집단으로부터 종속변수인 y_i들을 샘플링 할 때는 독립적이며 무작위로 수행해야만 한다.
- $\boldsymbol{\beta}$는 $(p + 1)$차원 매개변수 벡터로서 회귀 계수이다.
- $\boldsymbol{\varepsilon}$은 오차를 나타내는 항으로서 잡음이라고도 한다. 오차항들 간의 상관관계가 선형회귀 모

델에서 매우 중요한 역할을 하는데 여기서 오차는 평균이 0이고 분산이 σ인 정규분포를 갖는다고 가정한다.

예를 들어 사람들의 월수입이 그 사람이 소지한 학위의 유형과 해당 업무에서 몇 년간 근무했는지에 따라 달라진다. 독립변수인 학위 유형과 근무년수 집합과 연속적 종속변수인 월수입 사이의 관계를 알아내는데 다중회귀 분석이 매우 유용하다. 또 다른 예로 자동차의 엔진 크기와 실린더 수에 따라 이산화탄소 배출량이 달라지는데 이를 다중회귀 분석 모델로 구성하여 배출량을 예측할 수 있다.

최소 제곱법

선형회귀 분석에서 이미 알고 있는 좌표의 집합인 $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$을 보고 이들 좌표점과 최대한 가깝게 지나가는 하나의 직선을 그리는 것이 목적이라고 했다. '최대한 가깝게'라는 말의 의미에 대해 생각해볼 필요가 있다. 이 말 자체가 매우 주관적이기 때문에 '가깝게'라는 말과 '최대한'이라는 말의 의미를 부여하기 위한 모종의 방법이 필요하다. 일반적으로는 **최소 제곱법**(least square method)을 사용해서 '최대한 가깝게'라는 말의 의미를 설명한다. 하지만 항상 이 방법이 유일한 것은 아니다. 인공지능에서는 이와 같이 '최대한 가깝게'에 해당하는 다양한 방법을 개발하고 지속적으로 연구하고 있다. 가장 빈번하게 사용하는 경사하강법도 오차를 최소화하는 방식 중 하나이다. 일반적으로 이들 함수를 **손실함수**(loss function)라고 하고 이를 최소화하는 방법으로 '최대한 가깝게'라는 말에 의미를 부여하는 것이다.

최소 제곱법의 개념을 이해하고 이를 어떻게 적용하는지 알아보자. 방법은 두 가지가 있다. 하나는 선형 또는 보편적 최소 제곱이고 다른 하나는 비선형 최소 제곱이다. 선형 최소 제곱 문제는 주로 통계의 회귀 분석에 사용된다. 목표는 주어진 데이터 집합에 가장 적합한 모델함수의 매개변수를 찾는 것이다. 주어진 데이터는 $\{(x_i, y_i) \mid i = 1, \ldots, n\}$이다. 여기서 x_i들은 독립변수들이고 y_i들은 종속변수로서 관찰을 통해 알아낸 값이다. 모델함수는 $f(x, \beta)$로서 벡터 β는 m개의 조정 매개변수로 구성된다. 다시 말해서 데이터 집합에 가장 적합한 벡터 β를 구성하는 매개변수들을 찾아내는 것이다. 데이터 집합에 대한 모델로 적합한지 아닌지를 나타내는 오차로서 **잉여**(residual)는 종속변수의 실제값과 모델로 계산된 값과의 차이로 정의된다. 잉여는 다음과 같다.

$$r_i = y_i - f(x_i, \boldsymbol{\beta})$$

최소 제곱법이란 다음에 나타낸 잉여의 제곱들을 합한 S를 최소화하는 매개변수의 값을 구하는 방법이다. 즉, 수식 (14.3)과 같이 표현된 S의 값을 최소화하는 매개변수의 값을 찾는 방법이다.

$$S = \sum_{i=1}^{n} \varepsilon_i^2 \tag{14.3}$$

2개의 매개변수 $(\beta_0, \beta_1) = \beta$를 갖는 모델함수 $f(x, \beta)$로 직선을 이용한다. 즉,

$$f(x, \beta) = \beta_0 + \beta_1 x \tag{14.4}$$

가 모델함수이고 이를 그래프로 나타내면 그림 14.4와 같다.

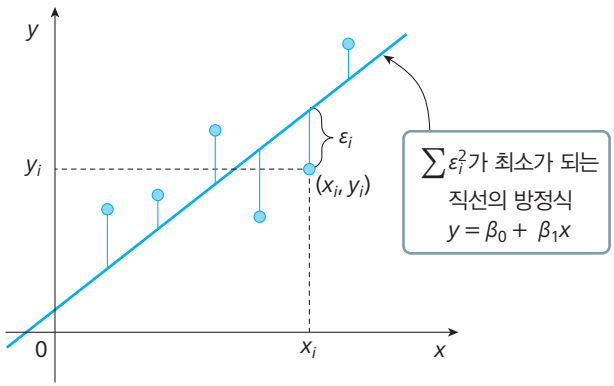


그림 14.4 선형회귀 분석

예를 들어보자. 5개의 데이터 (1, 2), (2, 2), (3, 3), (4, 3), (5, 5)가 주어졌다고 가정해보자. 5개의 데이터에 가장 적합한 직선의 방정식 $y = \beta_0 + \beta_1 x$를 구해보자. 즉, 다음과 같은 변수의 수가 식의 수보다 많은 선형 시스템의 근사해를 구하는 두 개의 수 β_0와 β_1을 구해보자.

$$\begin{cases} \beta_0 + 1\beta_1 + \varepsilon_1 = 2 \\ \beta_0 + 2\beta_1 + \varepsilon_2 = 2 \\ \beta_0 + 3\beta_1 + \varepsilon_3 = 3 \\ \beta_0 + 4\beta_1 + \varepsilon_4 = 3 \\ \beta_0 + 5\beta_1 + \varepsilon_5 = 5 \end{cases}$$

들은 잉여로서 데이터와 구하는 최적의 직선과의 y축과 평행한 거리를 나타낸다. 그래서

$$\begin{cases} \varepsilon_1 = 2 - (\beta_0 + 1\beta_1) \\ \varepsilon_2 = 2 - (\beta_0 + 2\beta_1) \\ \varepsilon_3 = 3 - (\beta_0 + 3\beta_1) \\ \varepsilon_4 = 3 - (\beta_0 + 4\beta_1) \\ \varepsilon_5 = 5 - (\beta_0 + 5\beta_1) \end{cases}$$

로 표현할 수 있다. 최소 제곱법은 잉여들의 제곱의 합을 최소로 하는 β_0와 β_1을 구하는 게 목

적이다. 즉, 다음 수식 (14.5) 함수의 최솟값을 구하면 된다.

$$\begin{aligned} S(\beta_0, \beta_1) &= \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 + \varepsilon_5^2 \\ &= [2 - (\beta_0 + 1\beta_1)]^2 + [2 - (\beta_0 + 2\beta_1)]^2 + [3 - (\beta_0 + 3\beta_1)]^2 \\ &+ [3 - (\beta_0 + 4\beta_1)]^2 + [5 - (\beta_0 + 5\,\beta_1)]^2 \\ &= 5\beta_0^2 + 55\beta_2^2 + 30\beta_0\beta_1 - 30\beta_0 - 104\beta_1 + 51 \end{aligned} \tag{14.5}$$

최솟값을 구하려면 $S(\beta_0, \beta_1)$을 β_0와 β_1의 이변수 함수로 보고 각각의 변수에 대한 편미분을 계산하고 이들을 0이라고 하고 해를 구해 임곗값이 무엇인지 확인해봐야 한다. 그리고 임곗값 중에서 극솟값을 갖는 점을 찾고 그 중 최소가 되는 점을 찾아야 한다. 편미분을 하면 다음과 같다.

$$\frac{\partial S}{\partial \beta_0} = 0 = 10\beta_0 + 30\beta_1 - 30$$

$$\frac{\partial S}{\partial \beta_1} = 0 = 30\beta_0 + 110\beta_1 - 104$$

이는 2개의 미지수와 2개의 방정식으로 구성되어 있으므로 해를 구하면

$$\beta_0 = 0.9$$

$$\beta_1 = 0.7$$

이다. 따라서 이들이 임곗값이 되며 이 점이 바로 최솟값을 갖는 점이 된다는 것을 보일 수 있다. 그래서 구하고자 하는 방정식은 수식 (14.6)이다.

$$y = 0.9 + 0.7x \tag{14.6}$$

각 점 (x_i, y_i)와 이 직선과의 잉여를 계산해보면 각각 1.4, −0.3, 0, −0.7, 0.6이다. 따라서

$$S(0.9, 0.7) = (1.4)^2 + (0.3)^2 + (0)^2 + (0.7)^2 + (0.6)^2 = 2.90$$

이다. 수식 (14.6)을 그림으로 나타내면 다음 그림 14.5와 같다.

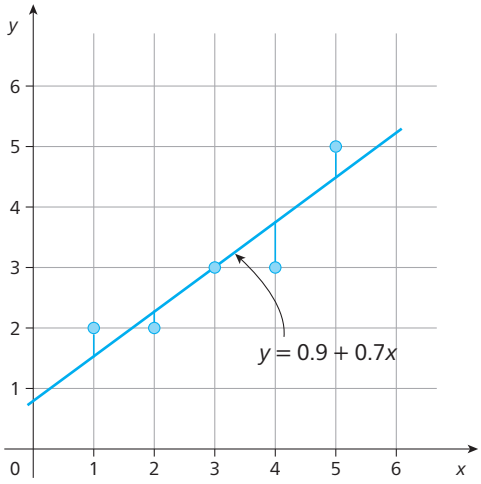


그림 14.5 $y = 0.9 + 0.7x$의 그래프

지금까지는 직선 형태의 모델을 찾아보았다. 좀 더 일반화하면 n개의 회귀변수 x_i에 대한 선형 모델을 다음 수식 (14.7)과 같이 표현할 수 있다.

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i \tag{14.7}$$

수식 (14.7)을 살펴보면 표본 데이터 $(\boldsymbol{x}_i, y_i) = (x_{i1}, \ldots, x_{in}, y_i)$, $i = 1, \ldots, k$가 결정된 값으로 주어지면 k개의 y_i는 $n + 1$개의 변수 $\beta_0, \beta_1, \ldots, \beta_n$으로 구성되는 함수로 볼 수 있다. 예를 들어 $n = 2$라고 하면 k개의 y_i, $i = 1, \ldots, k$는 모두 3개의 변수 $(\beta_0, \beta_1, \beta_2)$로 표현되는 식이고 다음과 같이 표현된다.

$$y_i = f(x_{i1}, x_{i2}, \beta_0, \beta_1, \beta_2) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2},\ i = 1, 2, \ldots, k$$

수식 (14.5)와 같은 방식으로 잉여를 구하면

$$\begin{aligned} S(\beta_0, \beta_1, \beta_2) &= \varepsilon_1^2 + \ldots + \varepsilon_k^2 \\ &= [y_1 - (\beta_0 + x_{11}\beta_1 + x_{12}\beta_2)]^2 + \ldots + [y_k - (\beta_0 + x_{k1}\beta_1 + x_{k2}\beta_2)]^2 \end{aligned}$$

이다. 이제 최소 제곱법이나 경사하강법 등을 이용하여 S를 최소화하는 $(\beta_0, \beta_1, \beta_2)$를 구한다. S가 $(\beta_0^*, \beta_1^*, \beta_2^*)$에서 최솟값을 갖는다고 해보자. 그러면

$$y = y(x_1, x_2) = \beta_0^* + \beta_1^* x_1 + \beta_2^* x_2$$

를 얻는다. 이는 변수가 (x_1, x_2)인 다중선형회귀 모델이고 평면의 방정식이며 변수가 한 개인 선형회귀 모델의 그림 14.5를 한 차수 높여서 나타낸 그래프에 대응된다.

다시 일반화한 수식 (14.7)을 중심으로 전개할 때로 돌아가서 오차함수 S를 생각해보자. 보통 머신러닝에서 사용하는 오차는 S에 1/2을 곱한 값인

$$E = \frac{1}{2}S = \frac{1}{2}\sum_{i=1}^{n}(f(\boldsymbol{x}_i, \boldsymbol{\beta}) - y_i)^2 \tag{14.8}$$

이다. 수식 (14.8)에서 1/2을 곱하는 이유는 미분할 때 상수에 분수가 나타나지 않게 만들고 싶기 때문이다.

선형회귀 분석

앞에서 배운 단순회귀 분석의 내용을 토대로 파이썬을 이용하여 선형회귀 분석과 이변수 선형회귀 분석을 해보자.

예제 14.1

어느 학생들 집단에서 주당 수학 학습 시간에 따른 수학 성적 자료이다. 앞에서 설명한 내용에 따라 다음과 같이 주어진 표본 데이터를 이용하여 선형회귀 직선을 구하라. 단, 최소 제곱법을 이용하라.

표 14.1 주당 수학 학습 시간에 따른 수학 성적 자료

학생	주당 수학 학습 시간	수학 성적
1	10	62
2	15	73
3	20	76
4	25	81
5	30	84
6	12	72
7	17	65
8	18	80
9	27	92
10	32	89

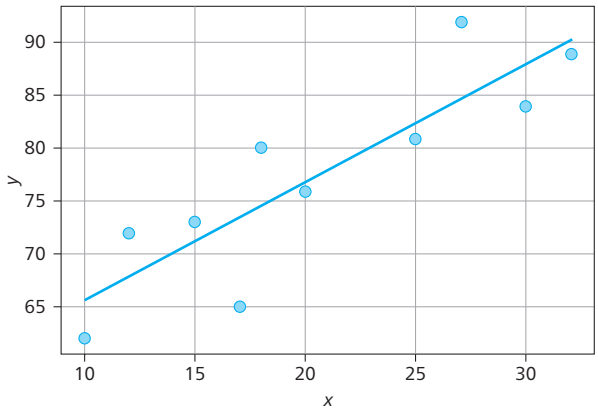
이 문제는 학생이 주당 수학을 얼마나 공부하느냐에 따라 수학 성적이 선형적으로 어떻게 되는지를 파악해보는 과정이다. 결과로 회귀직선의 방정식을 얻을 수 있다.

In

```
import numpy as np
import math
import matplotlib.pyplot as plt
matplotlib.rc('font', family='Malgun Gothic')

X = [10, 15, 20, 25, 30, 12, 17, 18, 27, 32]   # 평균은 20.6
Y = [62, 73, 76, 81, 84, 72, 65, 80, 92, 89]   # 평균은 77.4

x_mean = np.mean(X)
y_mean = np.mean(Y)

a = sum([(X[i] - x_mean ) * (Y[i] - y_mean) for i in range (len(X))]) /
sum([math.pow(i - x_mean, 2) for i in X])
b = y_mean - (x_mean * a)

print("기울기=", a )
print("절편=", b)

y_e = [(a * i + b) for i in X]

plt.xlabel("x", size=14)
plt.ylabel("y", size=14)
plt.plot(X, y_e, 'c')
plt.plot(X, Y, 'ro')

plt.show()
```

Out

```
기울기 = 1.1165762974438418
절편 = 54.39852827265686
```



<선형회귀 직선>

예제 14.2

다음 표본 데이터는 특정 학생들 집단에서 주당 총 학습 시간과 일평균 수면 시간에 따른 종합 성적 자료이다. 앞에서 설명한 내용에 따라 다음과 같이 주어진 표본 데이터를 이용하여 이변수 선형회귀 모델을 구하라. 단, 최소 제곱법을 이용하라.

표 14.2 주당 총 학습 시간과 일평균 수면 시간에 따른 종합 성적 자료

학생	주당 총 학습 시간(시간)	수면 시간(시간)	종합 성적
1	30	5	62
2	31	9	56
3	35	7	82
4	42	8	85
5	51	6	84
6	36	8	78
7	47	7	94
8	41	8	90
9	20	9	65
10	25	7	70
11	33	8	88
12	45	5	82
13	29	6	72
14	15	8	54
15	26	8	76

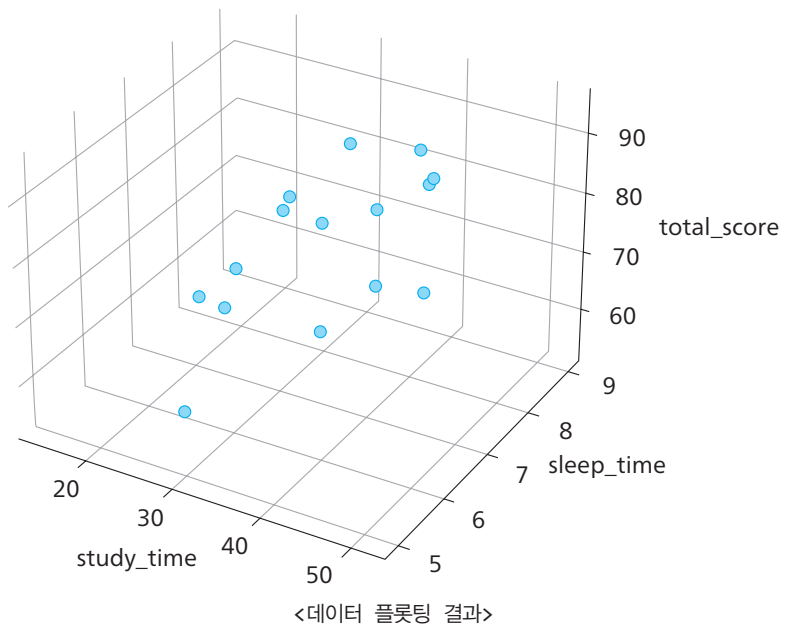
이 문제는 두 개의 변수를 갖는 실함수 형태로서 3차원 플롯이 가능하다. 먼저 3차원 공간상에 표 14.2에 있는 15개의 데이터를 플롯팅해보기로 한다.

In

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

study_time = [30, 31, 35, 42, 51, 36, 47, 41, 20, 25, 33, 45, 29, 15, 26]
sleep_time = [5, 9, 7, 8, 6, 8, 7, 8, 9, 7, 8, 5, 6, 8, 8]
total_score = [62, 56, 82, 85, 84, 78, 94, 90, 65, 70, 88, 82, 72, 54, 76]
x_data = np.array(study_time)
y_data = np.array(total_score)
```

```
ax = plt.axes(projection='3d')
ax.set_xlabel('study_time')
ax.set_ylabel('sleep_time')
ax.set_zlabel('total_score')
ax.scatter(study_time, sleep_time, total_score)
plt.show()
```

Out



<데이터 플롯팅 결과>

위의 데이터 플롯팅 결과 그래프를 보면 3차원 공간에 15개의 데이터가 플롯팅되어 있다. 이제 이 데이터에 최소 제곱법을 적용한 이변수 선형회귀모델인 평면의 방정식이 어떻게 되는지 구해보자.

In

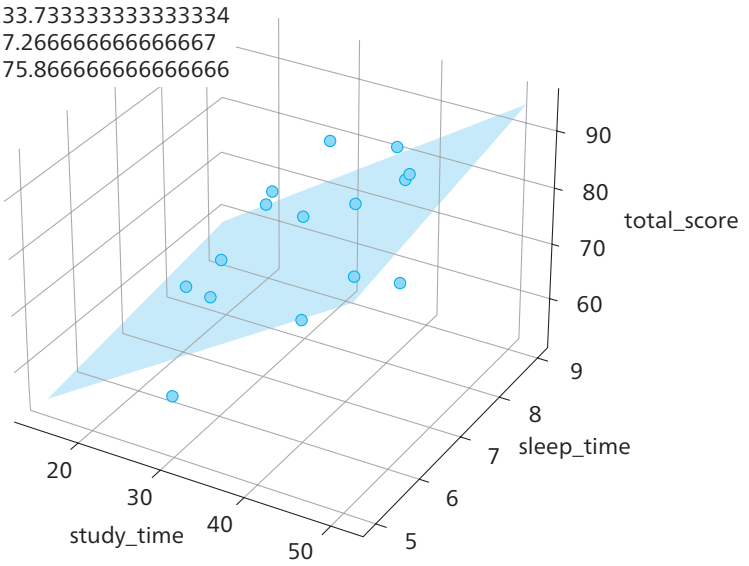
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

study_time = [30, 31, 35, 42, 51, 36, 47, 41, 20, 25, 33, 45, 29, 15, 26]
sleep_time = [5, 9, 7, 8, 6, 8, 7, 8, 9, 7, 8, 5, 6, 8, 8]
total_score = [62, 56, 82, 85, 84, 78, 94, 90, 65, 70, 88, 82, 72, 54, 76]
```

```
x1_data = np.array(study_time)
x2_data = np.array(sleep_time)
y_data = np.array(total_score)

x1_mean = np.mean(x1_data)
x2_mean = np.mean(x2_data)
y_mean = np.mean(y_data)

print("평균 학습 시간=", x1_mean)
print("평균 수면 시간=", x2_mean)
print("평균 종합 성적=", y_mean)

import statsmodels.api as statm
import statsmodels.formula.api as statfa

X = [[x1_data[idx], x2_data[idx]] for idx, value in enumerate
                                                        (study_time)]
Y = total_score

X_1=statm.add_constant(X)
results=statm.OLS(total_score,X_1).fit()

hour_class=pd.DataFrame(X,columns=['study_time','sleep_time'])
hour_class['total_score']=pd.Series(total_score)

model = statfa.ols(formula='total_score ~ study_time + sleep_time',
                                                    data=hour_class)

results_formula = model.fit()

a, b = np.meshgrid(np.linspace(hour_class.study_time.min(),hour_class.
study_time.max(),100), np.linspace(hour_class.sleep_time.min(),hour_
class.sleep_time.max(),100))

X_ax = pd.DataFrame({'study_time': a.ravel(), 'sleep_time': b.ravel()})
fittedY=results_formula.predict(exog=X_ax)

fig = plt.figure()
graph = fig.add_subplot(111, projection='3d')

graph.scatter(hour_class['study_time'],hour_class['sleep_time'],
              hour_class['total_score'], c='blue',marker='o', alpha=1)
```

```
graph.plot_surface(a,b,fittedY.values.reshape(a.shape), rstride=1,
                                        cstride=1, color='none', alpha=0.4)
graph.set_xlabel('study_time')
graph.set_ylabel('sleep_time')
graph.set_zlabel('total_score')
graph.dist = 11

plt.show()
```

Out

```
평균 학습 시간 = 33.733333333333334
평균 수면 시간 = 7.266666666666667
평균 종합 성적 = 75.866666666666666
```



<이변수 선형회귀 결과인 평면>

위의 이변수 선형회귀 결과인 평면 그래프를 보면 학습 시간이 많고 수면 시간이 많을수록 종합 성적이 증가하는 사실을 알 수 있다.

다항회귀 분석

앞에서 선형회귀 모델과 다중회귀 모델에 대해 학습하였다. 이번에는 다항회귀 분석에 대해 알아보자. 다항회귀 분석에서도 오차를 최소화하기 위하여 최소 제곱법을 이용한다. 선형회귀 분석을 응용에 적용해보면 대부분 선형관계가 잘 들어맞지 않는다. 예를 들어 화학물질을 합성하는 경우 온도에 따른 생산량의 변화를 모델화한다고 해보자. 각 단위에 따라 온도를 상승시켰을 경우 생산량이 증가하게 됨을 알 수 있다. 이때 표본 데이터를 살펴보면 다음 수식 (14.9)와 같은 이차 다항회귀 모델을 생각할 수 있다.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon \tag{14.9}$$

이 모델의 온도를 x에서 $x + 1$ 단위로 올리면 생산량은 $\beta_1 + \beta_2(2x + 1)$만큼 증가한다. 온도 증가를 단위별로 할 때 그 단위가 매우 작은 양이라면 이는 연속적으로 온도를 증가시키는 것과 마찬가지이다. 따라서 연속적으로 변하는 값은 미분 형태로 나타낼 수 있다.

주어진 데이터 $(x_1, y_1), \ldots, (x_n, y_n)$이 있다고 할 때 수식 (14.9)와 같이 이차 다항식 모델을 고려한다면 수식 (14.10)에 표현된 S가 최소가 되는 이차 다항식 모델에서

$$S = \sum_{i=1}^{n} \varepsilon_i^2 \tag{14.10}$$

매개변수 $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$를 결정하여

$$f(x, \boldsymbol{\beta}) = \beta_0 + \beta_1 x + \beta_2 x^2$$

를 구하는 게 목적이다. 수식 (14.10)의 S를 구체적으로 나타내면

$$S = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i + \beta_2 x_i^2))^2$$

이다. S가 어느 $(\beta_0, \beta_1, \beta_2)$에서 최솟값이 되는지 알고 싶으므로 S를 이 3개의 변수를 갖는 함수로 보고 편미분을 하고 이들을 0으로 놓고 해를 구하여 임계점을 구한다. 즉,

$$\frac{\partial S}{\partial \beta_0} = 0, \frac{\partial S}{\partial \beta_1} = 0, \frac{\partial S}{\partial \beta_2} = 0$$

이라고 하면 3개의 방정식과 3개의 미지수로 이루어진 연립방정식이 되고 이 연립방적식의 해를 다음과 같이 구한다.

$$-2\sum_{i=1}^{n} y_i + 2\beta_0 \sum_{i=1}^{n} 1 + 2\beta_1 \sum_{i=1}^{n} x_i + 2\beta_2 \sum_{i=1}^{n} x_i^2 = 0$$

$$-2\sum_{i=1}^{n} x_i y_i + 2\beta_0 \sum_{i=1}^{n} x_i + 2\beta_1 \sum_{i=1}^{n} x_i^2 + 2\beta_2 \sum_{i=1}^{n} x_i^3 = 0$$

$$-2\sum_{i=1}^{n} x_i^2 y_i + 2\beta_0 \sum_{i=1}^{n} x_i^2 + 2\beta_1 \sum_{i=1}^{n} x_i^3 + 2\beta_2 \sum_{i=1}^{n} x_i^4 = 0$$

이를 정리하면

$$n\beta_0 + \beta_1 \sum_{i=1}^{n} x_i + \beta_2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i$$

$$\beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2 + \beta_2 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i$$

$$\beta_0 \sum_{i=1}^{n} x_i^2 + \beta_1 \sum_{i=1}^{n} x_i^3 + \beta_2 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i$$

가 된다. 이 연립방정식을 β_0, β_1, β_2에 대하여 풀면, 주어진 데이터 $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$을 가장 적절하게 대표하는 포물선으로 수식 (14.11)과 같은 포물선의 방정식을 구할 수 있다.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 \tag{14.11}$$

이 수식 (14.11)을 그래프 형태로 나타내면 그림 14.6과 같은 형태로 나타난다. 일차 선형회귀에서는 결과가 직선이었지만 여기서는 포물선 형태의 2차 곡선이라는 점에 유의하기 바란다.

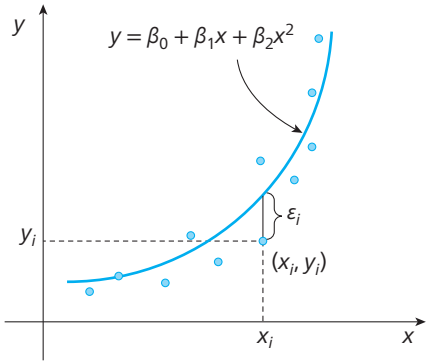


그림 14.6 이차 다항식 모델 회귀 분석

주어진 데이터의 형태가 이차 형태인 포물선이 아니라 삼차 큐빅 곡선에 적합해 보이면 삼차 다항식으로 회귀방정식을 세팅하고 구하면 된다. 방법은 위에서 했던 방식과 동일하다. 따라서 주어진 데이터 $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$으로부터 찾고자 하는 매개변수 $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)$를 계수로 갖는 k차 다항회귀 모델을

$$f(x, \boldsymbol{\beta}) = \sum_{j=1}^{k} \beta_j x^j$$

라고 하면 최소 제곱법을 이용한 오차는

$$E = S = \sum_{i=1}^{n} (f(x_i, \boldsymbol{\beta}) - y_i)^2$$

이 된다. 이 오차를 최소화하는 최솟값을 찾는 것이 다차 다항회귀 분석이다.

이차 다항회귀 분석

앞에서 배운 다항회귀 분석의 내용을 토대로 파이썬을 이용하여 이차 다항회귀 분석을 해보자.

예제 14.3

어느 학생들 집단에서 주당 수학 학습 시간에 따른 수학 성적 자료이다. 앞에서 설명한 내용에 따라 다음과 같이 주어진 표본 데이터를 이용하여 이차 다항회귀 곡선을 구하시오. 단, 최소 제곱법을 이용하시오.

표 14.3 주당 수학 학습 시간에 따른 수학 성적 자료

학생	주당 수학 학습 시간(시간)	수학 성적
1	5	42
2	8	53
3	10	67
4	13	78
5	15	87
6	20	92
7	23	95
8	25	80
9	28	76
10	30	72
11	32	70
12	35	68

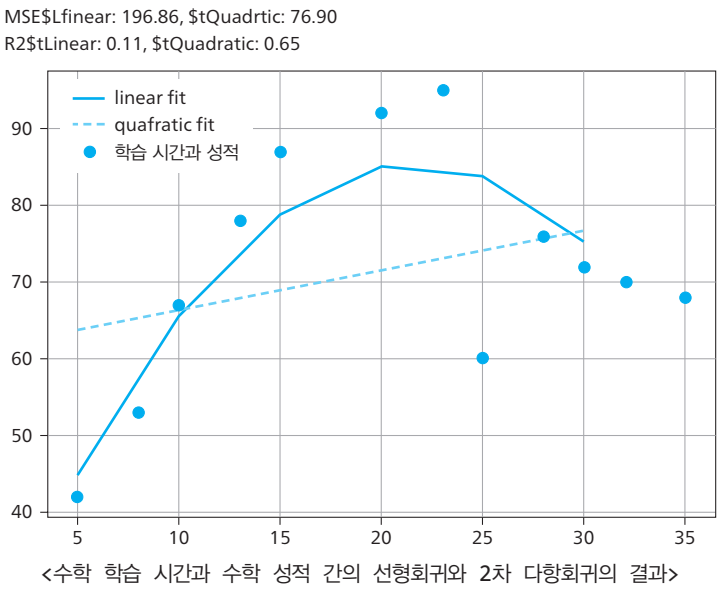
In

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import style
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.preprocessing import PolynomialFeatures
```

```
style.use('seaborn-talk')

krfont = {'family':'NanumGothic', 'weight':'bold', 'size':10}
matplotlib.rc('font', **krfont)
matplotlib.rcParams['axes.unicode_minus'] = False

X = np.array([5, 8, 10, 13, 15, 20, 23, 25, 28, 30, 32, 35])
                                                        [:, np.newaxis]
y = np.array([42, 53, 67, 78, 87, 92, 95, 60, 76, 72, 70, 68])

lr = LinearRegression()
pr = LinearRegression()

# 다항회귀를 위한 2차항 추가
quadratic = PolynomialFeatures(degree=2)
X_quad = quadratic.fit_transform(X)

# 비교를 위해 단순회귀 계산
lr.fit(X, y)
X_fit = np.arange(5, 35, 5)[:, np.newaxis]
y_lin_fit = lr.predict(X_fit)

# 다항회귀를 위해 변형된 모델에 다중회귀 모델 계산
pr.fit(X_quad, y)
y_quad_fit = pr.predict(quadratic.fit_transform(X_fit))

# 단순회귀 및 다항회귀 모델의 예측값 계산
y_lin_pred = lr.predict(X)
y_quad_pred = pr.predict(X_quad)

mse_lin = mean_squared_error(y, y_lin_pred)
mse_quad = mean_squared_error(y, y_quad_pred)

r2_lin = r2_score(y, y_lin_pred)
r2_quad = r2_score(y, y_quad_pred)

print('MSE$Linear: %.2f,$tQuadrtic: %.2f' %(mse_lin, mse_quad))
print('R2$tLinear: %.2f, $tQuadratic: %.2f' %(r2_lin, r2_quad))

plt.scatter(X, y, label = '학습 시간과 성적')
plt.plot(X_fit, y_lin_fit, label = 'linear fit', linestyle=' ')
plt.plot(X_fit, y_quad_fit, label = 'quadratic fit')
```

```
plt.legend(loc=2)

plt.show()
```

Out



<수학 학습 시간과 수학 성적 간의 선형회귀와 2차 다항회귀의 결과>

위의 수학 학습 시간과 수학 성적 간의 선형회귀와 2차 다항회귀의 결과 그래프를 보면 단순회귀 모델로 계산된 것은 점선으로 된 직선이고, 2차 다항회귀를 적용한 회귀 모델은 곡선이다. 선형회귀직선을 보면 수학 학습 시간이 많아질수록 수학 성적이 증가한다는 결론에 도달할 수 있다. 하지만 2차 다항회귀의 결과로 분석해보면 주당 수학 학습 시간이 20시간에 도달할 때까지는 수학 성적이 점차적으로 증가하지만 20시간 이상 학습할 경우에는 오히려 수학 성적이 20시간 때보다 떨어진다는 결과에 도달한다. 선형회귀 분석보다 2차 다항회귀 분석이 데이터를 좀 더 잘 분석한다고 말할 수 있다.

2차 다항회귀 분석 결과가 원래의 데이터를 더 잘 설명해주고 있다는 근거로 제시할 수 있는 것이 평균 제곱 오차(*MSE*)값을 비교해보면 알 수 있다. 단순회귀 모델에 의해 계산된 *MSE*는 196.86으로 다소 크지만 2차 다항회귀를 적용한 다중회귀 모델의 *MSE*는 76.90으로 많이 줄어들었음을 알 수 있다. 즉, 데이터와 평균 사이의 차이가 작다는 의미로서 데이터가 평균으로부터 떨어진 정도가 작으므로 2차 다항회귀 포물선이 직선 형태인 선형회귀 모델보다 데이터를 좀 더 정밀하게 설명해주는 모델이라고 말할 수 있다.

딥러닝에서 *MSE*를 언급했던 이유를 다시 한번 상기해보면 딥러닝은 기본적으로 입력되는 데이터를 이용해서 현상에 최적인 모델을 만들고 그 모델을 이용하여 결과를 도출하는 연구 분

야이다. 따라서 가정한 모델이 데이터와 얼마나 차이가 나는지 알아볼 필요가 있고 이를 위해 *MSE*를 구해 판단해보는 것이다. 궁극적으로 모델을 잘 만들었다면 당연히 *MSE*값은 작은 값으로 나와야 한다.

또한 결정 계수는 보통 $R^2 = 1 - (RSS/TSS)$로 나타내는데, 여기서 *RSS*는 단순 오차로서 실제값과 예측값의 단순 오차 제곱 합을 말하고, *TSS*는 데이터 평균값과 실제값 차이의 제곱을 나타낸다. 결정 계수 R^2는 딥러닝에서 회귀 문제의 성능을 측정하는 도구로 사용하는데 1에 가까울수록 좋고 0에 가까워지면 성능이 좋지 않은 모델이라는 뜻이다. 위의 예제 14.3에서 구한 선형회귀 모델의 값이 0.11인 반면에 2차 다항회귀 모델의 결정 계수는 0.65로 1에 더 가까워져 적합도가 좋아졌음을 알 수 있다. 결론적으로 2차 다항회귀 모델이 주어진 데이터에 더 적합한 모델이라고 할 수 있다.

경사하강법을 이용한 오차 최소화

앞에서 최소 제곱법을 이용하여 오차를 최소화하는 방법을 알아보았다. 즉, 다항회귀 분석을 이용하여 주어진 데이터 $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$으로부터 최소 제곱법을 이용한 회귀 곡선을 구해보았다. 방법은 다음의 오차식 (14.12)와 같이 표현되는 오차

$$E = S = \sum_{i=1}^{n} (f(x_i, \boldsymbol{\beta}) - y_i)^2 \tag{14.12}$$

를 최소로 하는 다항회귀 분석함수인 $f(x, \boldsymbol{\beta})$를 구하는 것이다. 이렇게 구한 함수를 오차식 (14.12)에 대입해보자.

$$E = \sum_{i=1}^{n} \left\{ \sum_{j=1}^{k} \beta_j (x_i)^j - y_i \right\}^2$$

으로 표현할 수 있다. $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)$를 변수로 갖는 다변수 함수라고 생각할 수 있다. 다변수 함수의 최솟값을 구하는 것이 바로 오차를 최소화하는 것이기 때문에 다변수 함수에 앞에서 배운 경사하강법을 적용하여 최솟값을 구해보자. 오차식을

$$E(\boldsymbol{\beta}) = E(\beta_0, \beta_1, \ldots, \beta_k) \tag{14.13}$$

라고 표현하는 것이 경사하강법을 적용하는 데 더 편리하다. 먼저 $E(\boldsymbol{\beta})$ 함수가 정의되는 영역 안의 한 시작점 $\boldsymbol{\beta}_0$를 정한다. 현재 $\boldsymbol{\beta}_i$가 주어졌을 때, 그다음으로 이동할 점인 $\boldsymbol{\beta}_i + 1$은 다음과 같이 계산하면 된다.

$$\boldsymbol{\beta}_i + 1 = \boldsymbol{\beta}_i - \gamma_i \nabla E(\boldsymbol{\beta}_i),\ i = 1, 2, \ldots \tag{14.14}$$

수식 (14.14)에서 γ_i는 이동할 거리를 조절하는 매개변수이고 이는 인공지능의 학습 속도를 나타낸다. 수식 (14.13)의 그래디언트를 계산해보면

$$\nabla E(\boldsymbol{\beta}) = \left(\frac{\partial E}{\partial \beta_0}(\boldsymbol{\beta}), \frac{\partial E}{\partial \beta_1}(\boldsymbol{\beta}), \ldots, \frac{\partial E}{\partial \beta_k}(\boldsymbol{\beta}) \right) \tag{14.15}$$

이므로 수식 (14.15)에 나타낸 값을 실제로 계산한다. 편미분을 계산하기 위해 연쇄율(chain rule)을 적용해보자. 함수 E를 수식 (14.16)에 표현한 함수 g와의 합성함수로 볼 수 있다. 즉,

$$g_i(\boldsymbol{\beta}) = \sum_{j=0}^{k} \beta_j x_i^j - y_i \tag{14.16}$$

라고 하면

$$E(\boldsymbol{\beta}) = \sum_{i=1}^{n} (g_i(\boldsymbol{\beta}))^2$$

이므로 각각의 $i = 1, \ldots, k$에 대하여

$$\frac{\partial E}{\partial \beta_j} = \sum_{i=1}^{n} \frac{\partial g_i^2}{\partial g_i} \frac{\partial g_i}{\partial \beta_j}, \; j = 1, 2, \ldots, k$$

가 된다.

$$\frac{\partial g_i^2}{\partial g_i} = 2g_i \text{이고,} \; \frac{\partial g_i}{\partial \beta_j} = x_i^j$$

이므로

$$\frac{\partial E}{\partial \beta_j} = 2\sum_{i=1}^{n} \left(\sum_{j=1}^{k} \beta_j x_i^j - y_i \right) x_i^j = 2\sum_{i=1}^{n} \left(f(x_i, \boldsymbol{\beta}) - y_i \right) x_i^j, \; j = 1, 2, \ldots, k$$

가 된다. 이렇게 구한 값들을 수식 (14.15)에 적용한 뒤

$$\nabla E(\boldsymbol{\beta}) = \left(\frac{\partial E}{\partial \beta_0}(\boldsymbol{\beta}), \frac{\partial E}{\partial \beta_1}(\boldsymbol{\beta}), \ldots, \frac{\partial E}{\partial \beta_k}(\boldsymbol{\beta}) \right)$$

를 수식 (14.14)에 넣어 다음에 이동할 점 $\boldsymbol{\beta}_{i+1}$을 구한다.

오차를 계산하는 매개변수인 $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)$를 변수로 보고 편미분을 구한 기울기(그래디언트)를 구하는 방법이 인공지능의 딥러닝에서 사용하는 경사하강법 알고리즘의 핵심이다.

다항회귀용 데이터 생성과 다항회귀 모델 구축

다항식 회귀에서 필요한 데이터를 생성하기 위해 넘파이의 `random.randn()` 함수를 사용한다. 먼저 `np.random.rand()` 함수를 이용하여 이차 방정식으로 비선형 데이터를 생성하고 이렇게 구한 다항회귀용 비선형 데이터를 표본 데이터로 간주하고 경사하강법을 이용하여 다항회귀 모델을 할 수 있다.

이를 구체적으로 파이썬으로 구현하고 실제로 다항회귀 모델을 구축할 수 있다. 하지만 인공지능의 코딩 부분은 인공지능과 관련된 과목을 수강하면서 실습해볼 기회가 많아서 여기서는 원래 우리가 의도하였던 인공지능과 수학이 어떻게 연결되는지에 대한 이론적인 설명을 하는 것으로 마무리하려고 한다.

14.3 인공신경망

인공신경망(ANN: Artificial Neural Network)은 인간의 두뇌 기능을 시뮬레이션하는 인공지능의 한 부분으로서 머신러닝과 인지과학에서 동물의 중추신경계 중 뇌의 신경전달 메커니즘을 모방하여 만든 통계학적 학습 알고리즘이다. 인공신경망이란 시냅스의 결합으로 네트워크를 형성한 인공뉴런이 학습을 통해 시냅스의 결합 세기를 변화시켜 문제해결 능력이 있는 모델을 말한다. 입력되는 자료에 해당하는 단위에 가중치를 부여하여 다음 층으로 전달하는 방식의 구조를 가진 인공신경망을 구성하고 입력 데이터로부터 출력을 생성하는 형태이다. 입력되는 데이터는 훈련용 데이터와 검증용 데이터로 나누어 적용하여 허용할 수 있는 정도의 오차를 최소화하는 형태로 최적화한 모델을 만드는 것이 딥러닝의 목표이다. 실제로 알고 싶은 입력 데이터를 구성한 모델에 입력하여 출력을 낸다.

일반적으로 사용되는 기본적 인공신경망 알고리즘인 다층인공신경망(Multi-Layer Neural Network)의 경우 입력층(input layer), 은닉층(hidden layer), 출력층(output layer)으로 구성된다. 각층들은 노드들로 구성되어 있다.

인공신경망 발전

인공신경망 뉴런 모델은 생물학적 뉴런을 수학적으로 모델링한 것이다. 생물적 뉴런과 마찬가지로 여러 개의 뉴런에 입력되는 정보를 저장하고 있다가 일정 수준이 넘어서면 활성화되면서 출력값을 내보내는 구조를 갖추고 있다. 인공신경망은 뉴런들을 여러 층으로 쌓아서 만든다. 최초로 인공신경망은 인간의 신경 구조를 복잡하게 연결된 스위치 구조로 표현할 수 있다고 믿고 이를 모델화한 것으로부터 시작되었다.

처음으로 제안된 인공신경망은 퍼셉트론이라고 하는 선형 분류기였다. 입력과 가중치들의 곱을 합한 뒤 활성화함수를 적용하여 1과 1을 출력하는 선형 분류기 구조를 갖추고 있다. 하지만 퍼셉트론 개념만으로는 단순한 배타적 논리합인 XOR 논리조차 해결할 수 없다는 것을 증명하여 퍼셉트론을 이용한 문제해결이 인공지능이 나아가야 할 방향이 아니라고 단정하게 되었다.

하지만 현재의 딥러닝에 바탕을 둔 인공지능은 퍼셉트론에서 제시한 신경망을 다층화하여 단층 신경망에서 해결할 수 없었던 문제를 해결했다. 다층 퍼셉트론(MLP: Multi-Layer Peceptron)을 이용한 딥러닝 방식이 가능했던 이유는 컴퓨터의 단순한 계산을 대량으로 처리할 수 있는 그래픽카드를 응용한 계산 능력이 급속하게 성장했기 때문이다. 퍼셉트론을 여러 층으로

쌓아 원래의 퍼셉트론이 해결하지 못했던 문제인 배타적 논리합인 XOR 문제를 해결할 수 있었다. 다만 여러 층으로 구성된 퍼셉트론은 학습에 몇 가지 어려움을 겪게 되었다. 이를 해결하기 위해 제시된 방법이 역전파 알고리즘(backword algorithm)이다. 이를 오류 역전파 알고리즘이라고 말하기도 한다. 이 알고리즘은 순방향 연산을 수행한 후에 예측값과 실제값 사이의 오차를 뒤로 다시 전달하면서 학습하는 방식이다.

퍼셉트론이 하나의 뉴런을 모방한 것이라고 하면 다층 퍼셉트론은 여러 개의 뉴런을 표현한 모델이다. 그리고 이들에게 학습을 제공하는 방법으로 역전파 알고리즘을 사용하였다. 하지만 역전파 알고리즘 학습방법에도 문제점이 있었다. **경사도 소실**(vanishing gradiant)이라고 하는 데이터가 사라져서 학습이 제대로 되지 않는 문제였다. 이외에도 새로운 사실을 추론하는 데 취약하고 새 데이터를 제대로 처리하지 못하는 문제점도 가지고 있었다. 이러한 문제를 상당 기간 해결 못 하고 있었다. 인공신경망의 한계에 도달한 것이 아닌가 하는 의심이 드는 상황이었다. 이때가 1990년대 초였다.

2006년이 되기까지 인공신경망 분야는 난관을 극복하지 못하고 있었다. 그러다가 다층인공신경망을 학습시키기 전에 **사전훈련**(pretraining)을 통해 학습하면 경사도 소실 문제를 해결할 수 있다는 연구가 발표되었다. 또 다른 문제인 새로운 데이터 처리를 제대로 하지 못하는 문제도 2012년에 대량의 데이터 중 일부 데이터를 삭제하면서 처리하는 방식인 **데이터 누락**(drop-out) 방법으로 좋은 결과를 얻을 수 있다는 사실을 알게 되었다. 이렇게 하여 기존 인공신경망이 해결하지 못한 문제를 해결하게 되었다. 난관을 뛰어넘었다는 의미와 기존 인공신경망에 대한 안 좋은 인식을 벗어나기 위해 **딥러닝**(deep learning)이라는 용어가 인공신경망이라는 용어를 대체하게 되었다. '딥'의 원래 의미인 '깊다'의 의미는 입력층과 출력층 사이의 은닉층 수가 2개 이상으로 다층인공신경망을 의미한다.

뉴런 모델

그럼 이러한 인공신경망을 어떻게 모방하는지 퍼셉트론부터 층수를 늘려가면서 어떠한 수학적 모델로 구성되는지 살펴보자. 뇌에 존재하는 신경세포를 모델(그림 14.7 참조)로 하여 컴퓨터 상에 네트워크를 구축한 것을 인공신경망이라고 한다.[2]

뇌 속의 신경세포를 모델화하기 위해 그림 14.7과 같이 추상화한다. 초기 퍼셉트론을 나타내는 그림 14.8은 k번째 뉴런 한 개를 표현한 것이다. 후에 복수개의 뉴런을 표현할 때 이들을 구별하기 위해 이를 k번째 뉴런의 표현이라고 나타냈다. 입력은 $x_1, x_2, \ldots, x_m$이고 가중치는 $w_{k1}, w_{k2}, \ldots, w_{km}$이다.

[2] 위키피디아 참조: By Egm4313.s12 (Prof. Loc Vu-Quoc) - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid = 72816083

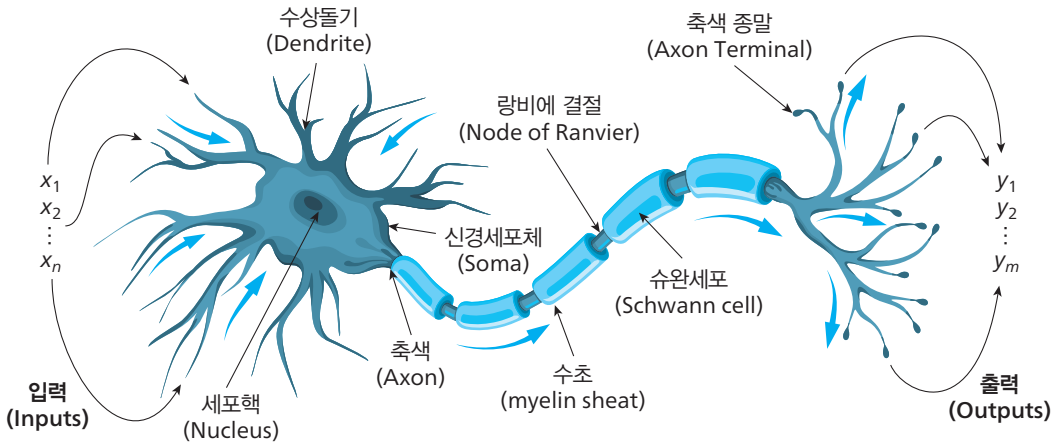


그림 14.7 신경세포 모형

입력값으로 나타내지는 않았지만 보통 입력값 중 맨 앞의 값을 $x_0 = 1$로 놓는다. 이는 기본적으로 가중치가 $w_{k0} = b_k$인 입력으로 '**편향**'이라고도 하는 **바이어스**(bias)는 하나의 뉴런에 입력되는 값에 가중치를 곱한 값들을 모두 더한 다음에 추가로 더해주는 상수이다. 이 상수는 한 뉴런에서 활성화함수(activation function)인 그림 14.8의 사각형 박스 속의 함수 φ를 거쳐 최종적으로 출력되는 값을 조절하는 역할을 한다. 뉴런의 측면에서 볼 때 바이어스란 신경세포에 가해지는 감도를 조정하는 값이다.

궁극적으로 실제는 m개의 입력만이 인공신경망에 작용한다고 보면 된다. 아래의 로젠블랫(Rosenblatt)이 제시한 퍼셉트론(그림 14.8)을 보면 복수개의 입력으로부터 하나의 출력을 내고 있으며 이는 생물학적 뉴런의 축색(axon)을 모방하고 있다.

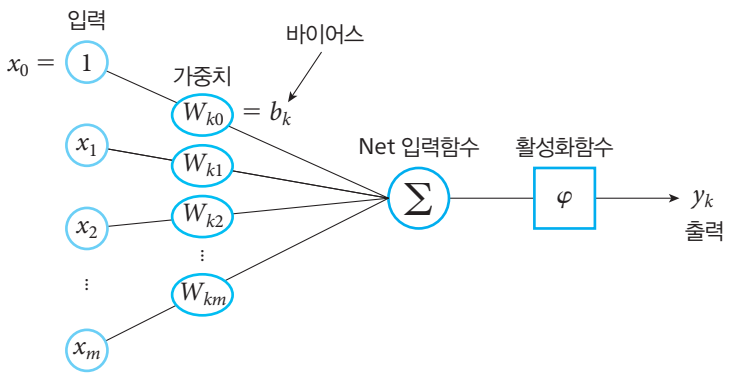


그림 14.8 퍼셉트론 모형

k번째 뉴런의 출력값을 수식으로 나타내면 수식 (14.17)이다.

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj}x_j\right) \tag{14.17}$$

활성화함수를 φ(Phi: '피'라고 읽는다)로 놓았다. 입력에 가중치를 곱하는데 가중치의 역할은 각 입력의 영향력을 조정하는 것이다. 영향력이 큰 입력에 가중치를 높인다. 입력과 가중치를 곱한 값들을 더한 뒤 바이어스($x_0 w_{k0} = b_k$)를 더하여 활성화함수의 입력으로 사용한다.

활성화함수

활성화함수(activation function)는 노드에 입력되는 값들을 비선형함수에 적용해서 다음 레이어로 전달한다. 활성화함수로 선형함수를 사용해서는 안 된다. 선형함수는 여러 개를 합성하여도 또 다른 하나의 선형함수이기 때문이다. 따라서 은닉층을 표현하는 데는 반드시 비선형함수인 활성화함수를 활용해야만 한다. 간단히 말해서 활성화함수란 입력을 받아서 다음 레이어도 전달할 때 활성화해서 전달할 것인지 비활성화해서 전달할 것인지를 결정하는 함수이다.

활성화함수는 입력(x_1부터 x_m)과 가중치(w_{k1}부터 w_{km})의 곱의 총합에 바이어스를 더한 값을 받아들여 k번째 뉴런의 출력 y_k를 낸다. 활성화함수는 특정 노드가 출력에 얼마나 많은 영향을 끼치는지를 나타내기 때문에 이를 얼마나 활성화할지 또는 말지를 결정하기 위한 함수이다. 따라서 입력값에 따라 뉴런의 자극 정도를 활성화함수로 조절할 수 있다. 활성화함수를 전이함수 또는 한계값함수라고도 한다. 활성화함수는 통상적으로 **단조 증가**(monotone increasing)하는 비선형함수를 사용한다. 활성화함수로 사용할 수 있는 함수는 여러 가지가 있지만 대표적으로 **시그모이드함수**(sigmoid function)가 있다. 최근에는 시그모이드함수보다 **렐루**(Rectified Linear Unitm ReLU)함수를 자주 사용한다. 이 함수는 양수에서 기울기가 1인 선형함수이며 음수는 0으로 버려버리는 함수로서 다음 수식 (14.18)과 같이 정의한다.

$$ReLU(x) = \max(0, x) \tag{14.18}$$

렐루함수의 그래프는 그림 14.9와 같다. 렐루함수를 사용하면 인공신경망의 레이어를 다층으로 쌓을 수 있는데, 시그모이드 함수보다 학습 속도를 높일 수 있다.

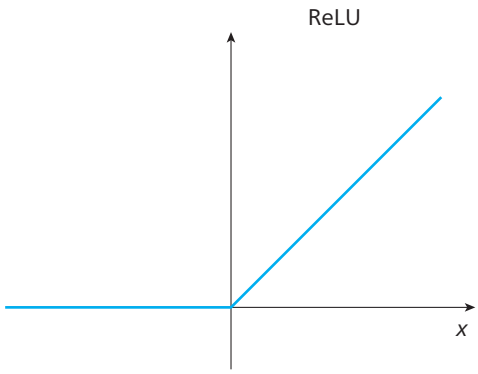


그림 14.9 렐루함수

로지스틱 회귀 분석 또는 인공신경망의 이진분류에서 마지막 레이어의 활성함수로 사용하는 함수가 시그모이드함수이다. 앞에서도 배웠지만 다시 한번 복습해보자. 다음 수식 (14.19)와 같은 함수

$$\varphi(u) = \frac{1}{1 + e^{-u}},\ u \in (-\infty, \infty) \tag{14.19}$$

가 시그모이드함수이다. 이 함수의 치역은 (0, 1)이다. 시그모이드함수의 그래프를 그려보면 다음 그림 14.10과 같다.

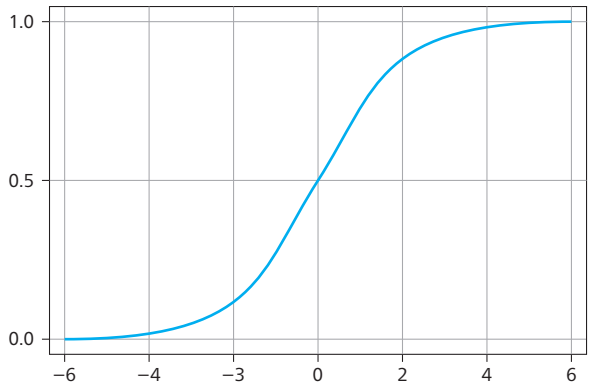


그림 14.10 시그모이드함수

또 다른 활성화함수의 예는 수식 (14.20)과 같이 정의되는 **하이퍼볼릭 탄젠트함수**(hyperbolic tangent function)이다.

$$\varphi(u) = \tanh(u),\ u \in (-\infty, \infty) \tag{14.20}$$

이 함수의 치역은 (−1, 1)이고 그래프는 그림 14.11과 같다.

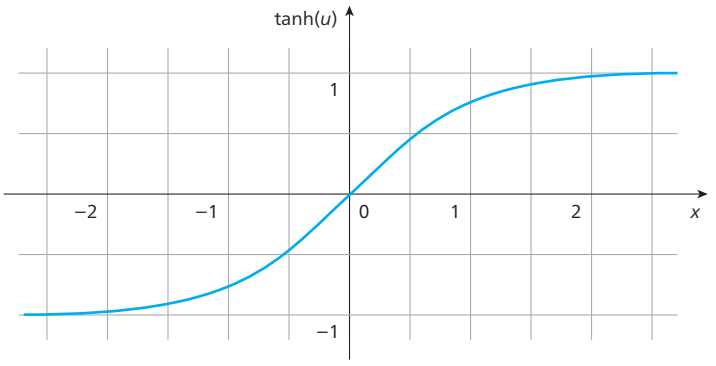


그림 14.11 하이퍼볼릭 탄젠트함수

앞에서 배웠던 시그모이드함수들 집합은 모두 다 활성화함수로 사용할 수 있다. 이들 함수의 특성을 살펴보면 입력치의 절대치가 커지면 함수의 증가 또는 감소가 현격하게 줄어든다. 반면에 0 근처에서는 함숫값이 매우 부드럽게 단조 증가한다는 것을 알 수 있다. 0에서 가장 빠른 증가세를 유지한다. 증가치가 크다는 것을 확인해보려면 주어진 활성화함수의 그래프를 보면 쉽게 알 수 있지만 직접 미분해보면 함수의 기울기가 가장 큰 곳이 0 근처라는 것을 확인해볼 수 있다. 이처럼 비선형으로 단조 증가하는 함수의 특징은 생물의 신경세포가 갖는 특성을 모델링하였더니 이러한 특성이 있다는 것이 알려졌기 때문에 시그모이드함수들을 활성화함수로 사용한다.

최근에는 시그모이드함수같이 부드러운 단조 증가의 특성을 갖는 활성화함수 대신 램프함수(ramp function)를 자주 사용하는데 선형함수를 수정하여 만든 것이다. 대표적으로 앞에서 언급한 렐루 계열의 함수들이다. 활성화함수로 사용하는 이유는 단순하면서 계산량이 줄어든다는 장점이 있기 때문이다. 따라서 사용 빈도가 높아지고 있다. 시그모이드함수를 사용할 때 입력값의 절댓값이 커지면 함숫값의 변화가 거의 없기 때문에 입력값 변동 폭에 주의해야 한다. 하지만 렐루함수에서는 그런 경우가 발생하지 않기 때문에 입력값 변동 폭에 신경 쓸 필요가 없다. 또한 시그모이드나 하이퍼볼릭 탄젠트함수보다 학습이 빠르고, 연산 비용이 적고, 구현이 매우 간단하다는 특징을 가지고 있다.

또한 은닉층에서 자주 사용되는 활성화함수로 렐루의 변형인 **리키 렐루**(Leaky ReLU, LReLU) **함수** $\varphi(u) = \max(au, u)$로서 a는 하이퍼파라미터로 0보다 작은 쪽에서 기울기가 약간 새는(leaky) 정도를 결정한다. 보통 $a = 0.01$을 사용하는데 이보다 더 작은 값이라면 무엇이든 사용할 수 있다. $a = 0.01$인 경우의 리키 렐루함수는 다음 수식 (14.21)과 같이 정의한다.

$$\varphi(u) = \begin{cases} 0.01u, & u < 0 \\ u, & u \geq 0 \end{cases} \tag{14.21}$$

리키 렐루함수는 u가 음수인 영역의 값에 대해 미분값이 양수가 된다는 점을 제외면 렐루함수가 갖는 특성을 그대로 유지한다.

활성화함수 φ에 의해 출력되는 값은 생물학적 뉴런의 축색(axon)에 대응되는 값이다. 이 값은 시냅스를 통해 확산되어 다음 층에서 입력으로 이용된다. 자주 사용하는 활성화함수를 비교해보기 위해 같은 좌표계에 그래프로 나타낸 것이 그림 14.12이다.

이외에도 사용할 수 있는 활성화함수들로는 $f(u) = u$인 기울기가 1인 직선이 있다. 생물 신경세포에서 활성화 정도는 비선형성함수로 모델화하지만 지역적으로 선형함수를 사용하기도 한다.

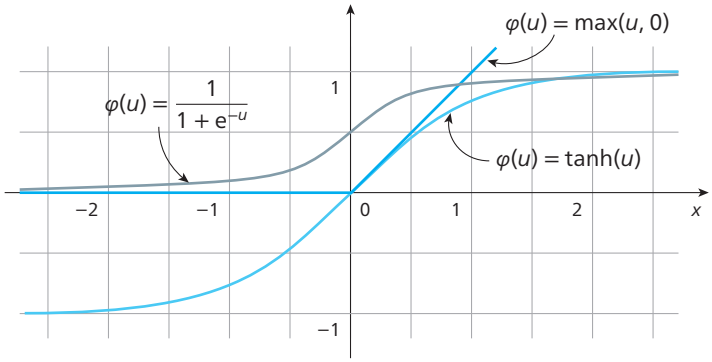


그림 14.12 활성화함수의 그래프 비교

신경망으로 해결하고자 하는 문제 중에서 지도 학습의 경우 회귀 문제나 분류 문제가 있다. 이를 풀기 위해 신경망 출력층에서 직선 형태의 활성화함수를 사용한다. 분류를 목적으로 하는 신경망에서는 출력층의 활성화함수로 대개 다음과 같은 소프트맥스(softmax)함수를 사용한다. 표준형 소트프맥스함수는 k차원 실수 공간에서 자기 자신으로 매핑되는 함수로서 수식 (14.22)와 같이 정의한다.

$$\varphi(\boldsymbol{u})_i = \frac{e^{u_i}}{\sum_{j=1}^{k} e^{u_j}},\ i = 1, \ldots, k \tag{14.22}$$

$\boldsymbol{u} = (u_1, \ldots, u_k) \in R^k$이다.

소프트맥스함수 $\varphi(\boldsymbol{u})$: $R^3 \rightarrow R^3$의 예를 들어보자. $k = 3$이라고 가정하고 입력값이

$$\boldsymbol{u} = \begin{pmatrix} 5 \\ 6 \\ 2 \end{pmatrix}$$

라고 하자. 먼저 다음을 계산하자.

$$e^{u_1} = e^5 = 148.413$$
$$e^{u_1} = e^6 = 403.429$$
$$e^{u_1} = e^2 = 7.389$$

그러므로

$$\sum_{i=1}^{3} e^{u_i} = e^{u_1} + e^{u_2} + e^{u_3} = 148.413 + 403.429 + 7.389 = 559.231$$

이고 정규화된 값은 다음과 같다.

$$\varphi(\boldsymbol{u})_1 = \frac{e^{u_i}}{\sum_{j=1}^{k} e^{u_j}} = \frac{148.413}{559.231} = 0.2654$$

$$\varphi(\boldsymbol{u})_1 = \frac{e^{u_i}}{\sum_{j=1}^{k} e^{u_j}} = \frac{408.429}{559.231} = 0.7304$$

$$\varphi(\boldsymbol{u})_1 = \frac{e^{u_i}}{\sum_{j=1}^{k} e^{u_j}} = \frac{7.389}{559.231} = 0.0132$$

그래서 함숫값은

$$\begin{pmatrix} 0.2654 \\ 0.7304 \\ 0.0132 \end{pmatrix}$$

이다.

인공신경망

인공뉴런에서 다층인공신경망까지 확장하는 과정을 살펴보자. 앞에서 로젠블랫 퍼셉트론을 이용하여 단 한 개의 인공뉴런을 설명하였다. 뉴런의 개수가 더 많아지면 출력되는 뉴런이 $y_1, y_2, \ldots, y_n$이 되어야 하고 이들의 층수를 늘리면 다층인공신경망이 구축되게 된다.

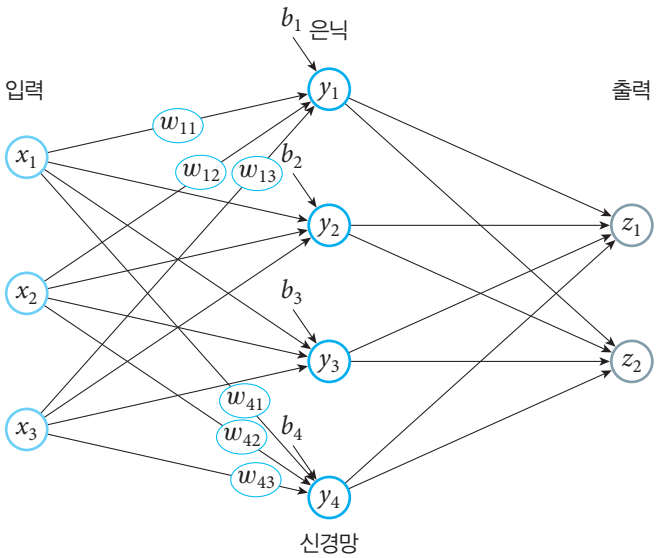


그림 14.13 하나의 은닉층을 가진 인공신경망

그림 14.13은 바이어스를 별개로 나타내면서 3개의 입력값 x_1, x_2, x_3가 있고, y_1, y_2, y_3, y_4가 은닉층에 출력으로 나타나며 이들을 입력값으로 이용하여 다시 출력되는 z_1, z_2, z_3가 최종 출력이 된다.

그림 14.13에 나타낸 신경망에서 하나의 **입력층**(input layer)과 하나의 **은닉층**(hidden layer)에 속한 유닛 간의 결합은 모두 $3 \times 4 = 12$개이며, 하나하나의 결합마다 각각 다른 가중치가 수식 (14.23)처럼 주어진다.

$$\boldsymbol{\omega} = \{w_{ji} | j = 1, 2, 3, 4, \ i = 1, 2, 3\} \tag{14.23}$$

은닉층에서의 출력값 y_1, y_2, y_3, y_4는

$$y_j = \varphi\left(\sum_{i=1}^{3} w_{ji} x_i + b_j\right), \ j = 1, 2, 3, 4 \tag{14.24}$$

가 된다. 수식 (14.24)에서 b_j는 바이어스이다. 그리고 출력층에는 2개의 뉴런 z_1과 z_2가 있다.

이를 벡터와 행렬을 이용하여 표기하면 수식 (14.25)와 같다.

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{W}_{\boldsymbol{x}} + \boldsymbol{b} \\ \boldsymbol{y} &= \varphi(\boldsymbol{u}) \end{aligned} \tag{14.25}$$

수식 (14.25) 표현에서

$$\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}, \ \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \ \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}, \ \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

이고

$$\boldsymbol{W} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{pmatrix}, \ \varphi(\boldsymbol{u}) = \begin{pmatrix} \varphi(u_1) \\ \varphi(u_2) \\ \varphi(u_3) \\ \varphi(u_4) \end{pmatrix}$$

이다. 또한 y_1, y_2, y_3, y_4를 새로운 입력으로 받아들여서 출력 z_1, z_2를 계산하려면 위와 같은 방법으로 수식을 적용하면 된다. 즉, 입력과 은닉층에 속한 유닛 간의 결합은 모두 $4 \times 2 = 8$개이며, 하나하나의 결합마다 각각 다른 가중치 $\boldsymbol{\omega}^* = \{w^*_{ji} | j = 1, 2, i = 1, 2, 3, 4\}$가 주어진다. 그래서

$$z_j = \varphi\left(\sum_{i=1}^{2} w_{ji}^* y_i + b_j^*\right), j = 1, 2 \tag{14.26}$$

가 된다. 수식 (14.26)에서도 b_j^*는 바이어스를 나타낸다.

수식 (14.26)으로 나타낸 값을 벡터와 행렬을 이용하여 표기하면 다음 수식 (14.27)과 같다.

$$\begin{aligned} \boldsymbol{u}^* &= \boldsymbol{W}^* \boldsymbol{y} + \boldsymbol{b}^* \\ \boldsymbol{z} &= \varphi(\boldsymbol{u}^*) \end{aligned} \tag{14.27}$$

수식 (14.27)에서

$$\boldsymbol{u}^* = \begin{pmatrix} u_1^* \\ u_2^* \end{pmatrix}, \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}, \boldsymbol{b}^* = \begin{pmatrix} b_1^* \\ b_2^* \end{pmatrix}, \boldsymbol{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

이고

$$\boldsymbol{W}^* = \begin{pmatrix} w_{11}^* & w_{12}^* & w_{13}^* & w_{14}^* \\ w_{21}^* & w_{22}^* & w_{23}^* & w_{24}^* \end{pmatrix}, \varphi(\boldsymbol{u}^*) = \begin{pmatrix} \varphi(u_1^*) \\ \varphi(u_2^*) \end{pmatrix}$$

이다. 여기서 첨자 '*'를 붙인 이유는 가중치나 바이어스의 값이 층별로 달라지기 때문에 구별하기 위해 사용하였다. 만약 은닉층의 수가 한 개가 아니라 점점 많아져서 층을 구분하고자 할 경우에는 '*'로 표현하면 '*'의 수를 늘려가며 표현해야 하는데, 그 모양이 매우 복잡해진다. 따라서 몇 번째 층수인지를 나타내는 첨자 '(n)'을 사용한다. 예를 들어 $\boldsymbol{u}^*$ 대신에 $\boldsymbol{u}^{(2)}$, $\boldsymbol{u}^{(3)}$ 등으로 2번째 은닉층, 3번째 은닉층 등으로 표현하면 훨씬 단순하게 수식 (14.27)을 표현할 수 있다.

실제 딥러닝에서는 입력되는 데이터가 수만 개에 이르고 층의 수도 수만 개가 되는 경우가 많아서 계산량이 매우 크다. 따라서 수식을 이용하여 모델화할 경우 첨자를 활용하여 표현한 수식만 잘 이해하고 있다면 모델의 형태를 파악하는 데 어려움이 없을 거라고 생각한다.

다층인공신경망

앞에서 은닉층이 단 한 개인 인공신경망을 표현해보았다. 이번에는 은닉층의 개수가 복수개인 인공신경망을 표현해보자. 다층인공신경망은 단일 신경망을 여러 개 조합하여 만든다.

다음 그림 14.14는 3개의 입력값을 갖는 입력층과 3개의 은닉층과 2개의 출력을 가진 다층신경망을 표현한 개념도이다. 층의 수는 5개인 신경망이다. 여기서 모든 은닉층이 모두 4개의 노드를 가지고 있지만 실제로는 층마다 노드의 수도 다르다. 편의상 층을 나타내기 위해 레벨(lev-

el)의 앞 글자를 사용하여 $l = 1, 2, 3, 4, 5$로 표기하자. 입력층이 $l = 1$이라는 의미는 1층을 의미하며, $l = 2, 3, 4$는 은닉층이다. 그리고 마지막 $l = 5$로 표기되는 층이 출력층이다.

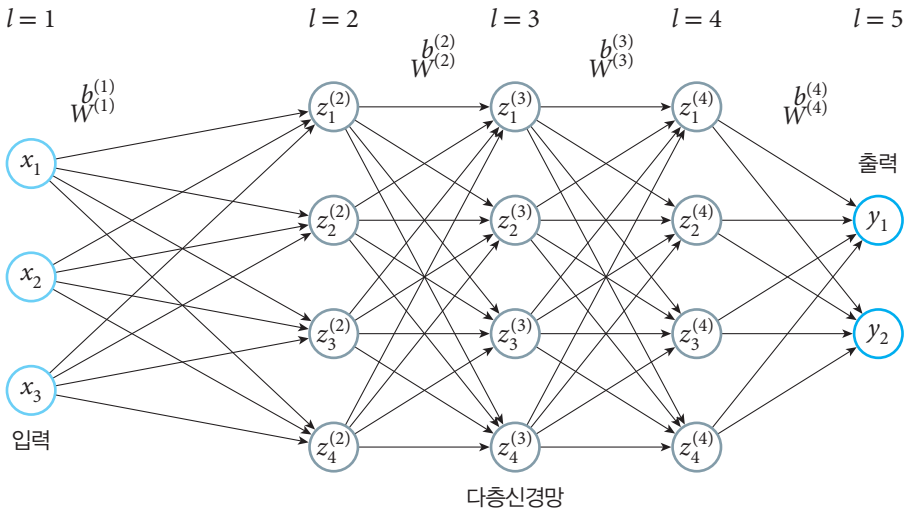


그림 14.14 5계층으로 구성된 다층신경망 예

2층에 해당하는 첫 번째 은닉층을 수식 (14.27)과 같이 다층신경망 모델로 나타내보면 다음 수식 (14.28)과 같다.

$$\begin{aligned} \boldsymbol{u}^{(2)} &= \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)} \\ \boldsymbol{z}^{(2)} &= \boldsymbol{\varphi}(\boldsymbol{u}^{(2)}) \end{aligned} \tag{14.28}$$

수식 (14.28)에서

$$\boldsymbol{u}^{(2)} = \begin{pmatrix} u_1^{(2)} \\ u_2^{(2)} \\ u_3^{(2)} \\ u_4^{(2)} \end{pmatrix}, \boldsymbol{z}^{(2)} = \begin{pmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \end{pmatrix}, \boldsymbol{b}^{(1)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{pmatrix}, \tag{14.29}$$

이고

$$\boldsymbol{W}^{(1)} = \begin{pmatrix} w_{11}^{(1)} w_{12}^{(1)} w_{13}^{(1)} w_{14}^{(1)} \\ w_{21}^{(1)} w_{22}^{(1)} w_{23}^{(1)} w_{24}^{(1)} \\ w_{31}^{(1)} w_{32}^{(1)} w_{33}^{(1)} w_{34}^{(1)} \\ w_{41}^{(1)} w_{42}^{(1)} w_{43}^{(1)} w_{44}^{(1)} \end{pmatrix}, \boldsymbol{\varphi}(\boldsymbol{u}^{(2)}) = \begin{pmatrix} \varphi(u_1^{(2)}) \\ \varphi(u_2^{(2)}) \\ \varphi(u_3^{(2)}) \\ \varphi(u_4^{(2)}) \end{pmatrix} \tag{14.30}$$

이다. $l = 2$인 첫 번째 은닉층을 표현할 수 있다면 나머지 은닉층도 쉽게 표현할 수 있다. 그래

서 $l = 3$일 때 수식 (14.28)에서 입력값 $\boldsymbol{x}$ 대신 $z^{(2)}$를 입력하고, $l = 4$일 때는 $z^{(2)}$ 대신 $z^{(3)}$를 대입하는 방식으로 수식 (14.28)을 변경하면

$$\begin{aligned} \boldsymbol{u}^{(l)} &= \boldsymbol{W}^{(l-1)}\boldsymbol{z}^{(l-1)} + \boldsymbol{b}^{(l-1)} \\ \boldsymbol{z}^{(l)} &= \boldsymbol{\varphi}(\boldsymbol{u}^{(l)}),\ l = 3,\ 4 \end{aligned} \tag{14.31}$$

라고 나타낼 수 있다. 각각의 $l = 3, 4$에 대한 $\boldsymbol{u}^{(l)}$, $\boldsymbol{z}^{(l)}$, $\boldsymbol{b}^{(l)}$, $\boldsymbol{W}^{(l)}$, $\boldsymbol{\varphi}(\boldsymbol{u}^{(l)})$도 수식 (14.29)와 (14.30)과 유사하게 나타낼 수 있다.

최종 출력은 다음 수식 (14.32)로 나타낸다.

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \tag{14.32}$$

수식 (14.32)에서 구체적인 값은 수식 (14.33)과 같다.

$$y_j = \varphi\left(\sum_{i=1}^{4} w_{ij}^{(4)} z_i^{(4)} + b_j^{(4)}\right),\ j = 1,\ 2 \tag{14.33}$$

수식 (14.33)에서 $\boldsymbol{W}^{(1)}$은 4×3 행렬이고, $\boldsymbol{W}^{(2)}$와 $\boldsymbol{W}^{(3)}$는 4×4 행렬이며, $\boldsymbol{W}^{(4)}$는 2×4 행렬이란 점에 유의하기를 바란다.

이 단계를 일반화하면 임의의 층수를 갖는 다층신경망을 표현할 수 있다. 층 $l + 1$에서 노드로 나타나는 출력 $z^{(l+1)}$은 바로 이전 층 l에 등장하는 모든 노드 $z^{(l)}$와 가중치를 나타내는 $\boldsymbol{W}^{(l)}$과 바이어스를 나타내는 $b^{(l)}$를 이용하여 다음 수식 (14.34)와 같이 계산할 수 있다.

$$\begin{aligned} \boldsymbol{u}^{(l+1)} &= \boldsymbol{W}^{(l)}\boldsymbol{z}^{(l)} + b^{(l)} \\ \boldsymbol{z}^{(l+1)} &= \boldsymbol{\varphi}(\boldsymbol{u}^{(l+1)}) \end{aligned} \tag{14.34}$$

수식 (14.34)에서 매개변수들은 다음 수식 (14.35)와 (14.36)과 같이 나타낸다.

$$\boldsymbol{u}^{(l)} = \begin{pmatrix} u_1^{(l)} \\ u_2^{(l)} \\ \vdots \\ u_{nl}^{(l)} \end{pmatrix},\ \boldsymbol{z}^{(l)} = \begin{pmatrix} z_1^{(l)} \\ z_2^{(l)} \\ \vdots \\ z_{nl}^{(l)} \end{pmatrix},\ \boldsymbol{b}^{(l)} = \begin{pmatrix} b_1^{(l-1)} \\ b_2^{(l-1)} \\ \vdots \\ b_{nl}^{(l-1)} \end{pmatrix}, \tag{14.35}$$

$$\boldsymbol{W}^{(l-1)} = \begin{pmatrix} w_{11}^{(l-1)} & w_{12}^{(l-1)} & \cdots & w_{1n_{l-2}}^{(l-1)} \\ w_{12}^{(l-1)} & w_{22}^{(l-1)} & \cdots & w_{2n_{l-2}}^{(l-1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_{l-1}1}^{(l-1)} & w_{n_{l-1}2}^{(l-1)} & \cdots & w_{n_{l-1}n_{l-2}}^{(l-1)} \end{pmatrix},\ \boldsymbol{\varphi}(\boldsymbol{u}^{(l)}) = \begin{pmatrix} \varphi(u_1^{(1)}) \\ \varphi(u_2^{(1)}) \\ \vdots \\ \varphi(u_{n_l}^{(2)}) \end{pmatrix} \tag{14.36}$$

n_j, $j = 1, 2, \ldots, L$는 l층을 구성하는 노드의 수이다.

활성화함수 φ를 모든 층에서 동일한 함수로 표기하였는데 실제 적용할 때는 층별로 다른 활성화함수를 적용하기도 한다. 특히 마지막 출력층 유닛의 활성화함수는 해결하고자 하는 문제 유형을 고려해서 낮은 층에서 사용하던 활성화함수와는 다른 별도의 활성화함수를 사용한다.

딥러닝에서 취급하는 다층인공신경망은 단순 신경망을 여러 개의 층으로 배열하여 구성한다. 한 개의 신경망은 인접한 모든 층의 모든 노드와 연결되어 있다. 같은 층에서 다른 노드들과는 연결되지 않는다. 은닉층의 각 노드의 출력은 가중치와 바이어스가 반영된 뒤 다음 층 노드의 입력으로 들어간다. 그림 14.14를 보면 좌측에서 우측으로 정보가 이동하고 있음을 알 수 있다. 이처럼 좌측에서 우측으로 정보가 이동하는 방향을 **순방향**(forward)이라고 한다. 순방향 전파에서는 입력값에 따라 출력값을 계산할 수 있다. 순방향만으로 딥러닝이 순조롭게 이뤄지면 좋겠지만 그렇지 못한 경우가 발생한다. 이 경우에는 역방향으로 데이터를 전달하여 출력과 올바른 값과의 오차가 최소가 되도록 학습시킨다. 이렇게 하면 오류를 피드백하여 학습 효과를 높이는 효과를 볼 수 있다. 순방향과 반대쪽으로 데이터를 전달하는 방식을 **역방향**(backword)이라고 한다. 입력과 출력을 알고 있는 상태에서 신경망을 학습시키는 지도 학습의 한 가지로서 역전파 알고리즘은 정보를 우측에서 좌측으로 이동시키는 역방향 전달을 활용하는 알고리즘이다.

우리가 고려하는 다층인공신경망의 층수가 L개라고 할 때, 주어진 입력 $\boldsymbol{x}$로부터 출력 방향쪽인 우측으로 이동하면서 출력 $\boldsymbol{y}$를 계산한다. $\boldsymbol{y}$가 궁극적으로 $\boldsymbol{x}$의 함수이므로 $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$라고 표현할 수 있다. 하지만 $\boldsymbol{y}$를 계산하는 동안에 입력되는 매개변수인 가중치와 바이어스가 있는데, 가중치 $\boldsymbol{W}^{(l)}$, $l = 2, 3, \ldots, L$과 바이어스 $\boldsymbol{b}^{(l)}$, $l = 2, 3, \ldots, L$이 함수 $\boldsymbol{y}$에 영향을 미친다. 매개변수의 값들이 각 계층별로 어떻게 주어지냐에 따라 다양한 함수를 표현할 수 있다. 또 한 가지 한 층에서 다른 층으로 함숫값이 계산될 때 활성화함수를 활용하고 있다는 점에도 유의해야 한다. 결국 층별로 사용되는 활성화함수 φ도 함수 $\boldsymbol{y}$에 영향을 미친다. $\boldsymbol{x}$를 입력값이라고 하면 $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$에 영향을 주는 입력값을 나열하여 다음 수식 (14.37)과 같이 좀 더 구체적으로 표현할 수 있다.

$$\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x};\ \boldsymbol{W}^{(l)},\ \boldsymbol{b}^{(l)},\ l = 2, 3, \ldots, L) \tag{14.37}$$

입력벡터 $\boldsymbol{x}$를 제외한 나머지 변수를 한 개의 벡터 $\mathbf{w} = (\boldsymbol{W}^{(l)},\ \boldsymbol{b}^{(l)},\ l = 1, 2, \ldots, L)$이라고 간단히 표기하면 수식 (14.37)을 더 간략히 표현할 수 있다. 그래서 모든 매개변수를 포함하는 벡터 $\mathbf{w}$를 이용하면 수식 (14.37)을 간단하게

$$\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x};\ \mathbf{w}) \tag{14.38}$$

라고 표기할 수 있다.

수식 (14.38)을 보고 전체 다층신경망을 떠올릴 수 있다면 다층인공신경망의 구조를 명확하게 이해하고 있다고 할 수 있다. 수식 (14.38)의 내부에 포함되어 있는 변수를 활용하여 전체 과정을 그림으로 나타내보면 그림 14.15와 같다. 그림 14.15에서 점선으로 나타낸 부분은 $z^{(i)}$, $i = 2, 3, \ldots, L-1$이 반복적으로 계산되는 과정이다. 각 계산 과정은 짙은 색으로 표현된 박스 내부에서 이뤄지는 계산 과정과 동일하다. 계산식의 첨자 중에서 우측 상단에 괄호로 표현된 첨자는 층을 나타낸다는 점에 주의하기 바란다. 그리고 실제 해당 우상단 첨자에 종속되어 나타나는 가중치와 바이어스의 성분에 나타나는 스칼라값들의 우측 하단에 붙는 첨자는 이 그림에 나타내지 않았다는 점에도 주의하기를 바란다.

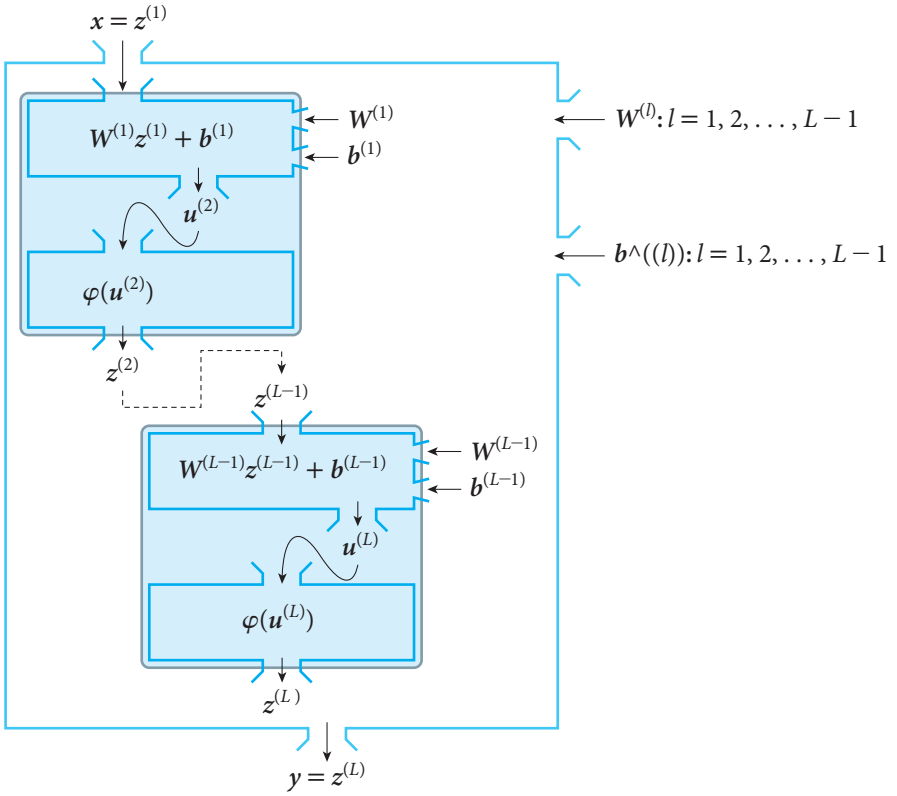


그림 14.15 인공신경망 수식 개념도

신경망이 올바른 값을 찾기 위해 어떻게 학습을 하는지 알아보자. 층의 수가 많은 인공신경망에서의 학습을 **딥러닝**(deep learning)이라고 한다. '딥(deep)이라는 말의 의미인 깊다는 뜻은 층의 수가 많다는 뜻이다. 기본적으로 층 수를 늘리고 각 층에 속하는 노드 수를 늘려나가면 인공신경망의 표현력을 향상시킬 수 있다.

14.4 딥러닝

인공신경망의 이름을 탈피하여 딥러닝이라는 새로운 이름으로 재탄생한 인공지능 머신러닝의 딥러닝은 기존에 해결하지 못했던 문제를 해결해줄 뿐만 아니라 인공지능의 다양한 문제를 효과적으로 해결할 수 있는 도구로 자리매김을 하였다. 인공지능 학자들 사이에서 환영받았으며 오차를 최소화하는 다양한 알고리즘이 개발되었고 새로운 알고리즘 개발을 위한 활발한 연구가 진행되고 있다. 특정 목적에 적합한 일부 딥러닝은 인간의 능력을 뛰어넘는 성과를 내고 있으며 컴퓨팅 성능이 좋아짐에 따라 학습 속도도 증가하고 있다. 이전의 학습 알고리즘과 딥러닝의 성능을 단순 비교해보면 그림 14.16과 같다.

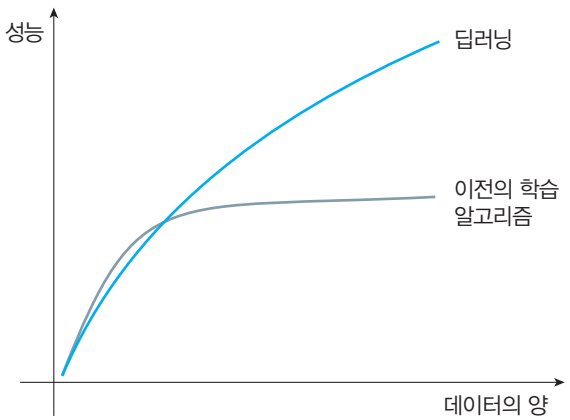


그림 14.16 기존 학습 알고리즘과 딥러닝 성능

그 원인은 인공신경망이 유행하던 시기와 크게 세 가지가 달라졌기 때문이다. 첫 번째는 인공신경망이 가진 한계를 극복할 수 있는 새로운 알고리즘이 개발되었다는 점이고, 두 번째는 빅데이터 분야의 발달로 인해 신경망 학습에 필요한 학습 데이터가 대량으로 확보되었다는 점이다. 마지막 세 번째는 막대한 데이터를 학습하는 데 필요한 계산량을 감당할 수 있는 하드웨어 장치인 **그래픽 처리 장치**(GPU: Graphics Processing Unit)가 발전하였기 때문이다.

간단히 왜 딥러닝이 지금에 와서 유행하게 되었는지를 말해보면 첫 번째는 목록화된 데이터의 양이 많아야 제대로 된 결과를 도출할 수 있는데 그동안 데이터 축적이 제대로 되어있지 못했고, 두 번째는 학습을 위해 엄청난 계산이 필요했는데 하드웨어적 컴퓨팅 능력이 이제 와서 가능해졌기 때문이다.

학습

딥러닝 모델을 어떻게 만들고 훈련은 어떻게 시키는지 알아보자. 딥 네트워크가 어떠한 학습도 되어있지 않은 초기 상태라고 가정하고 훈련시키는 방법을 생각해보자. 먼저 일반적으로 떠오르는 방법이 무엇인가? 제일 먼저 목록화된 데이터 집합을 수집하고 이를 토대로 특성들을 학습할 수 있는 네트워크 아키텍처를 설계하여 모델을 만든다. 대량의 출력 데이터를 가지고 있는 애플리케이션이거나 새롭게 만들려고 하는 애플리케이션이라면 이 방법이 수월하게 모델을 만들 수 있는 방법일 것이다. 하지만 이 방법은 모델을 훈련시키기에는 데이터양이 너무 많고 학습 속도에 따라 여러 날 또는 여러 주가 필요하기 때문에 잘 사용하지 않는다.

대부분의 딥러닝 애플리케이션에서는 **전이 학습**(transfer learning)이라고 하는 방법을 사용한다. 이 방법은 사전에 훈련된 모델을 세밀하게 수정하는 방법인데 이렇게 수정하는 과정을 **튜닝**(tuning)이라고 한다. 기존에 학습이 완료된 모델 네트워크에 이전에는 알지 못하는 유형의 새로운 데이터가 발견되면 이를 모델에 제공하고 다시 학습시킨다. 이렇게 하면 적은 양의 데이터를 가지고도 미세 조정을 할 수 있으며 계산 속도도 일 단위나 주 단위에서 분 단위 또는 시간 단위로 떨어뜨릴 수 있다.

다층신경망에서 입력되는 데이터 $\boldsymbol{x}$와 가중치와 바이어스를 합한 매개변수 벡터 $\mathbf{w}$가 입력되어 출력 $\boldsymbol{y}$가 출력되었다. 매개변수 $\mathbf{w}$값을 변화시켜 출력 $\boldsymbol{y}$값이 원하는 값이 되도록 해야 한다. 즉, 원하는 값과 출력 $y(\boldsymbol{x}; \mathbf{w})$의 차이가 최소가 되는 매개변수 $\mathbf{w}$를 찾아야 한다.

차이가 최소가 되도록 하는 과정인 다층신경망 학습은 어떻게 진행되는가? 또 다층신경망을 학습시키기 위해 필요한 데이터는 무엇인가? 현재 사용하는 모델로부터 이상적인 최종 모델을 만들기 위해서 복수의 **데이터 집합**(dataset)을 이용한다. 사용하는 데이터 집합은 3가지 종류가 있는데 각각의 용도에 따라 **훈련 데이터 집합**(traning dataset), **검증 데이터 집합**(varification dataset), **검사 데이터 집합**(test dataset)이라고 한다. 이들 모두를 사용할 때도 있고 훈련 데이터와 검증 데이터만 사용할 때도 있다.

훈련 데이터 집합은 모델의 적합한 매개변수 w를 찾기 위해 사용되는 입력벡터와 대응되는 출력벡터의 쌍으로 이루어진 표본 데이터 집합이다. 이 훈련 데이터 집합을 이용하여 지도 학습 방법으로 모델을 훈련시킨다. 이때 경사하강법이나 확률적 경사하강법 등을 활용하여 최적화한다. 바람직한 출력을 보통 **목표**(target) 또는 **레이블**(label)이라고 한다. 딥러닝 모델을 구축하기 위해 수많은 작업이 필요한데 그 중에서 학습에 필요한 정답을 만들어야 한다. 이 부분이 자동화되어 있으면 좋겠지만 수월치 않아서 수작업으로 한다. 그래서 인공지능 과정 중 가장 기본적이면서 노동력이 필요한 작업이 바로 데이터에 대한 정답을 만드는 과정이다. 이 과정을 **레이블링**(labeling)이라고 한다. 짐작하겠지만 사용할 데이터의 양이 많으면 많을수록 더 좋은 결과를 기대할 수 있다. 레이블링이 중요한 이유는 부정확한 레이블로 학습을 시키면 모델의

성능이 떨어진다. 특히 이미지 처리와 자연어 처리 등에서 레이블링을 정확히 하지 않으면 실제 처리 결과에 실망할 수밖에 없다.

목표를 $\boldsymbol{d}$라고 하자. 예를 들어 목표 $\boldsymbol{d}$는 회귀일 경우에는 구간 $[a, b]$나 실수 집합 R같은 연속적인 실수의 부분 집합의 원소로 나타날 것이고, 분류일 경우에는 0과 1 또는 이산개 집합으로 구성된 값으로 나타난다. 그리고 N개의 데이터로 이루어진 훈련 데이터 집합을

$\{(\boldsymbol{x}_1, \boldsymbol{d}_1), (\boldsymbol{x}_2, \boldsymbol{d}_2), \ldots, (\boldsymbol{x}_N, \boldsymbol{d}_N)\}$이라고 나타낸다. 즉, 입력이 $\boldsymbol{x}_i$일 때 출력이 $\boldsymbol{d}_i$이기를 희망하는 올바른 값들로 구성된 데이터 표본을 말하고 이 집합에 속하는 원소를 **훈련 샘플**(traning sample)이라고 한다.

우리가 비교하고자 하는 대상은 $\boldsymbol{x}_i$를 현재의 다층신경망 모델에 입력하여 얻는 출력값인 $\boldsymbol{y}(\boldsymbol{x}_i; \mathbf{w})$와 훈련 데이터 집합의 $(\boldsymbol{x}_i, \boldsymbol{d}_i)$이다. 즉, 두 값이 근사해지도록 모델을 만들기 위해 $\mathbf{w}$를 결정하는 과정이다. 즉,

$$\boldsymbol{y}(\boldsymbol{x}_i; \mathbf{w}) \rightarrow \boldsymbol{d}_i \tag{14.39}$$

가 되도록 $\mathbf{w}$를 조절한다. 이러한 과정을 학습이라고 한다.

궁극적으로 모델을 학습시킨다는 의미는 매개변수인 가중치와 바이어스의 값을 추정하는 것이다. 그래서 현재의 모델에 훈련 데이터 집합을 적용해서 각각의 입력벡터에 대하여 결과를 도출하고 도출된 결과와 목표를 비교하며, 비교 결과와 사용한 특정 학습 알고리즘에 근거하여 모델의 매개변수 $\mathbf{w}$를 수정하고, 적합한 모델로 만들기 위해 변수 선택 및 매개변수 추정하는 일련의 과정을 학습이라고 할 수 있다. 수정할 때 오차를 수식으로 나타낸 뒤 오차가 최소가 되도록 다양한 오차 최소화 방법을 활용한다. 오차를 최소화한다고 할 때 오차를 어떻게 측정하는지에 대한 명확한 의미를 부여해야 하는데, 이를 위해 다양한 오차에 대한 정의가 필요하다. 이에 대해 알아보자.

오차

위에서 훈련이 잘 되었는지 아닌지를 판단하는 기준은 수식 (14.39)를 얼마나 근사시킬 수 있느냐 하는 것이다. 근사한지 아닌지를 측정하기 위해서는 이들 간의 차이를 잴 수 있는 측정치가 있어야 한다. 이 측정치를 수식으로 표현한 것을 **오차함수**(error function)라고 한다. 오차를 작게 만들기 위해 가중치와 바이어스를 조정하는 과정을 학습이라고 하였다. 다층인공신경망에서 지도 학습의 경우에 회귀나 분류 문제를 다루지만 문제의 유형에 따라 오차함수 및 신경망의 출력층에 대한 디자인을 변경하여 사용할 수도 있다.

따라서 문제의 유형에 적합한 활성화함수와 오차함수를 선택해서 사용할 수 있고 은닉층과 출

력층에서도 서로 다른 활성화함수를 사용하기도 한다. 활성화함수와 오차함수를 사용 방법 또는 사용되는 층을 기준으로 나눌 수 있고 활성화함수 사용부터 살펴보면 다음 표 14.4과 같다.

표 14.4 층별 활성화함수와 용도

사용 층	활성화함수	용도
은닉층	렐루함수	기울기 소실 문제 완화 및 연산 속도 증가
	리키 렐루함수	
출력층	시그모이드 또는 로지스틱함수	이진분류
	소프트맥스	다중분류
	미사용	회귀

은닉층에서 렐루함수를 사용하면 입력값이 음수일 때 기울기도 0이 되는 문제가 발생한다. 그렇게 되면 해당 인공신경망은 다시 회생하기 어렵다. 이런 문제를 **죽은 렐루**(dying ReLU)라고 한다. 죽은 렐루 문제해결을 위해 사용하는 것이 리키 렐루이다. 실수의 음수 영역에서의 함숫값을 0이 아닌 기울기가 0에 가깝지만 양수가 되도록 만든 렐루의 변형된 형태이다.

오차함수를 문제 유형별로 정리해보면 다음 표 14.5와 같다.

표 14.5 문제 유형별 활용 오차함수

문제의 유형	오차함수
회귀	평균 제곱 오차(Mean Square Error)
이진분류	이진 교차 엔트로피(Binary Cross Entropy)
다중분류	범주형 교차 엔트로피(Categorical Cross Entropy)

회귀

인공지능 문제는 회귀와 분류로 나누어진다고 했다. **회귀**(regression)란 입력값에 관계없이 출력되는 값이 실수의 어떠한 값도 가질 수 있는 연속된 값을 출력하는 함수를 대상으로 훈련 데이터를 가장 잘 재현하는 함수를 찾는 과정이다. 예를 들어 집에서 특정 지역의 아파트 가격을 예측하는 문제를 생각해보자. 이 문제에 고려되는 요소는 학교까지의 거리, 관공서와의 거리, 마트의 수, 지하철역까지의 거리, 공원과의 근접성 등이 변수로 활용된다. 결과로 나오는 출력값은 4억 원, 4억 1천만 원, 4억 5천만 원 등과 같은 특정 실숫값으로 결정된다. 이 값들은 이산적인 것처럼 보이기는 하지만 실제로는 연속적인 값을 갖는다. 값을 반올림하거나 만 원 단위 이하는 절삭하는 등의 방법으로 가격을 표현하기 때문에 출력값이 연속된 실수인 문제라고 볼 수 있다. 주택마다 실제로는 천만 원이나 백만 원 단위로 끊어서 표현하지만 실제 가격은 실숫값으로 표현된다고 가정할 수 있다. 따라서 이 문제는 회귀 문제라고 할 수 있다.

활성화함수를 선택할 때는 목적으로 하는 함수의 치역이 어느 범위인지 확인하고 동일한 치역을 갖는 활성화함수를 선택해야 한다. 위의 예제에서 들은 회귀 문제인 아파트 가격은 (0, ∞) 범위라고 할 수 있으므로 시그모이드함수나 렐루함수와 같은 활성화함수를 활용해야 한다. 목표함수의 치역이 (−1, 1)이라면 하이퍼볼릭 탄젠트함수가 적합하고 목표함수가 치역으로 (−∞, ∞)를 갖고 있다면 기울기가 1인 실직선함수가 활성화함수로 적합하다.

활성화함수 선택이 끝났다면 다층인공신경망의 출력 $\boldsymbol{y}(\boldsymbol{x}_i; \mathbf{w})$가 훈련 데이터의 목표 출력인 $\boldsymbol{d}_i$에 가능하면 근사하는 방법을 모색해야 한다. 이때 필요한 것이 오차함수이다. 회귀 문제에서는 주로 다음 수식 (14.40)과 같이 정의하는 **평균 제곱 오차**(MSE: Mean Square Error)를 사용한다.

$$\frac{1}{N}\|\boldsymbol{d} - \boldsymbol{y}(\boldsymbol{x}; \boldsymbol{\omega})\|^2 = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{d}_i - \boldsymbol{y}(\boldsymbol{x}_i; \mathbf{w}))^2 \tag{14.40}$$

오차함수로 사용하려고 하는 평균 제곱 오차는 훈련 데이터의 모든 샘플 $i = 1, \ldots, N$에 대하여 제곱 오차를 전부 더한 다음에 이를 평균한 값이다. 이 함수의 변수를 매개변수 벡터인 $\mathbf{w}$라고 하고 이 함수가 최솟값을 갖게 되는 $\mathbf{w}_0$를 구하는 것이 회귀 문제를 해결하는 것이다.

사용하는 모든 데이터에서 순방향으로 전파를 실시하고 오차 합계를 이용하여 매개변수를 갱신하는 학습을 **배치 학습**(batch learning)이라고 한다. 배치 학습은 한 번에 훈련 데이터를 학습시키는 방법으로 시간과 자원 소요가 많아 주로 오프라인 환경에서 수행한다. 그래서 배치 학습을 **오프라인 학습**(offline learning)이라고도 한다. 제품으로 출시되는 모델이 배치 학습의 결과로 학습되었다면 해당 모델은 더 이상의 학습을 하지 않고 사용된다. 만약에 모델을 업데이트하고자 한다면 새로운 데이터를 원래 학습시켰던 훈련 데이터와 합한 뒤에 전체 데이터로 다시 학습시키고 학습된 모델을 제품으로 출시하고 출시된 제품은 더 이상 학습하지 않고 사용만 한다.

반면에 순방향 전파 1회마다 오차를 구하는 매개변수를 갱신하는 경우의 학습 방법을 **온라인 학습**(online learning)이라고 한다. 온라인 학습은 이미 학습이 완료되어 제품화된 모델을 활용하면서 얻게 되는 새로운 데이터 세트를 이용하여 업그레이드하는 방식으로 학습한다. 온라인 학습의 문제점으로는 올바르지 않은 데이터가 입력될 경우 모델의 성능이 떨어진다는 점이다. 예를 들어 도서 추천 알고리즘에서 악의를 가진 사용자가 특정 도서가 상위에 노출되도록 의도가 담긴 데이터를 대량으로 생성하고 그 데이터를 이용하여 온라인 학습이 되도록 한다면 문제가 발생할 수 있다. 따라서 온라인 학습이 적용된 모델에서는 지속적인 모니터링이 필요하며 성능 감소가 발생할 경우 즉시 학습을 중단하고 성능 감소 이전의 상태로 복구시켜야 한다.

온라인 학습과 배치 학습 및 기타 학습 방법에 대해서는 인공지능을 공부할 때 알고 있어야

하는 방법이므로 이에 대해서는 별도로 학습하기를 바란다. 또한 오차의 최솟값을 찾는 문제는 앞에서 배웠던 **경사하강법**(GD: Gradiant Descent)과 **확률적 경사 하강법**(SGD: Stochastic Gradiant Descent), **아담**(Adam: Adaptive moment estimation), **나담**(NAdam: Nesterov-Accelerated adam) 등을 이용하여 해결한다. 이들에 대해서도 본격적으로 인공지능을 공부할 때 대부분 자세히 다루므로 그때 학습하기를 바란다.

이진분류

결과가 2가지 중 하나로 분류하는 인공지능 문제인 이종분류(two-class classification) 또는 이진분류(binary classification)는 가장 널리 사용되는 머신러닝 문제이다.

예를 들어 정부 특정 정책에 대하여 찬성 댓글과 반대 댓글로 분류한 데이터를 기반으로 특성벡터 $\boldsymbol{x}$를 추출하고 그 특성 $\boldsymbol{x}$로부터 댓글 텍스트에 기반하여 찬성과 반대로 분류하는 방법을 학습한다고 할 때 이진분류를 사용한다.

다른 예로 사람의 얼굴 사진이 주어졌을 때 사진으로부터 어떤 특정한 방법으로 특성벡터 $\boldsymbol{x}$를 추출했을 때 그 특성 $\boldsymbol{x}$로부터 남녀를 구별하는 방법을 학습한다고 할 때도 이진분류를 사용한다.

첫 번째 예로 들은 문제를 가지고 이진분류를 설명해보자. 오직 2개의 값만 가지면 되기 때문에 목적값인 레이블은 1차 벡터로서 스칼라값을 가지면 된다. 즉, $d = 1$이면 찬성이고 $d = 0$이면 반대라고 할 수 있다. 종류를 이진변수 $d \in \{0, 1\}$로 나타낼 수 있다. 그래서 입력 $\boldsymbol{x}$로부터 d 값을 추정하는 것이 모델이 된다. 이 모델을 위해 훈련 데이터 집합과 검사 데이터 집합을 서로 다른 댓글들로 구성하는데 각각 50%는 찬성, 50%는 반대인 댓글로 이뤄지도록 선택한다. 훈련 데이터 집합과 검사 데이터 집합은 완전히 다른 집합으로 구성해야 한다. 그 이유는 머신러닝 모델을 동일 집합으로 훈련과 검사를 병행하면 절대 안 된다. 훈련 데이터로 훈련한 모델에 동일한 데이터인 검사 데이터를 적용해보면 당연히 좋은 결과를 출력하기 때문이다. 모델이 훈련 데이터 집합에서 잘 작동한다는 것이 새 데이터 집합에 대해서도 잘 작동한다는 것을 보장하지 않는다. 우리가 모델을 훈련시키는 이유는 새로운 댓글 데이터를 보고 찬반을 정확히 판단할 수 있는 능력을 갖추도록 모델을 만드는 것이 목적이기 때문이다. 이미 훈련 데이터 집합을 가지고 훈련한 모델은 동일한 데이터의 레이블을 모두 알고 있기 때문에 이를 예측하는 것 자체는 의미가 없다. 그래서 검사 데이터 집합은 훈련 데이터 집합에는 없는 다른 댓글들로 구성해야 한다.

이진분류 문제를 모델화하는 여러 가지 방법이 있는데, 여기서는 $\boldsymbol{x}$가 정해져 있을 때 $d = 1$이 되는 사후 확률 $P(d = 1|\boldsymbol{x})$를 모델화하는 방법을 사용한다. 주어진 $\boldsymbol{x}$에 대하여 d를 추정하는 모델을 사용하여 사후 확률을 계산한 뒤에 특정 값을 가지고 찬성과 반대를 구분하도록 한다.

이 특정 값을 임곗값이라고 하고 여기서는 임곗값을 0.5라고 하자. 그래서 사후 확률값이 0.5보다 크면 $d = 1$이라고 판단하여 찬성이라고 분류하고 그렇지 않으면 $d = 0$이라고 판단하여 반대라고 분류한다(그림 14.17 참조).

사후 확률 $P(d = 1|\boldsymbol{x})$를 모델화하는 데 딥러닝의 인공신경망을 활용한다. 이 신경망은 출력층에 단 하나의 노드 d만 있으면 된다. 여기서 편의상 바이어스는 $b = 0$라고 하자.

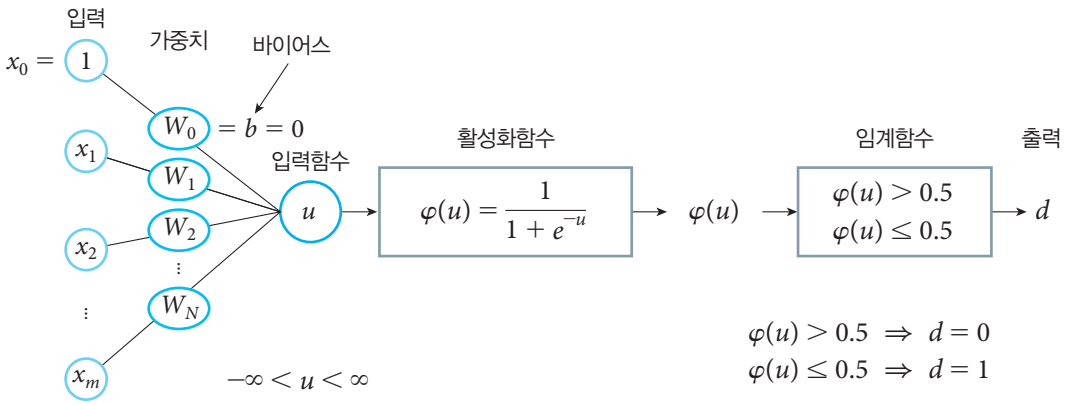


그림 14.17 이진분류 모델

사후 확률인 이 함수의 치역은 (0, 1)로 보아도 무방하기 때문에 활성화함수는 (0, 1) 사이의 값을 가지면서 S자 형태로 그려지는 조건을 충족하는 시그모이드함수 중 하나인 로지스틱함수

$$\varphi(u) = \frac{1}{1 + e^{-u}}$$

를 사용한다. 이 활성화함수를 사용하고 매개변수 중 하나인 바이어스 $b = 0$이라고 가정하면 매개변수 $\mathbf{w} = \boldsymbol{W}$이고 이를 사용하여 계산한 신경망 전체의 입출력함수 $d(\boldsymbol{x}; \mathbf{w})$를 사후 확률 모델이라고 하자. 그러면 이 인공신경망 모델은 그림 14.17과 같이 표현된다.

$$P(d = 1|\boldsymbol{x}) \approx d(\boldsymbol{x}; \mathbf{w}) \tag{14.41}$$

이고 한편

$$\boldsymbol{W} = \begin{pmatrix} w_1 w_2 \ldots w_N \end{pmatrix}, \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}, u = \boldsymbol{W}\boldsymbol{x} = \sum_{i=1}^{N} w_i x_i$$

이므로 수식 (14.41)에 의해

$$
\begin{aligned}
P(d=1|\boldsymbol{x}) &\approx d(\boldsymbol{x};\ \mathbf{w}) \\
&= \varphi(u) = \varphi(\boldsymbol{W}\boldsymbol{x}) \\
&= \frac{1}{1+\exp(-\boldsymbol{W}\boldsymbol{x})} \\
&= \frac{1}{1+\exp\left\{\sum_{i=1}^{N} w_i x_i\right\}}
\end{aligned}
\tag{14.42}
$$

이 출력값으로 나오게 된다. 인공신경망의 매개변수인 가중치 $\boldsymbol{W}$에 따라 다양한 사후 확률값을 얻을 수 있다. 여기에서 주어진 표본 데이터 $\boldsymbol{x}$에 가장 적합한 가중치 $\boldsymbol{W}$를 구하는 것이 목표이다. 인공신경망의 목적은 주어진 표본 데이터로부터 적합한 가중치 $\boldsymbol{W}$와 바이어스 b를 구하는 것인데 여기서는 바이어스 b를 0이라고 간주하였기 때문에 매개변수 $\boldsymbol{W}$만 구하면 된다.

매개변수 $\boldsymbol{W}$는 주어진 훈련 데이터 집합 $\{(\boldsymbol{x}_i,\ d_i)|\, i = 1,\ 2,\ \ldots,\ N\}$을 이용하여 모델로부터 얻은 사후 확률 $P(d=1|\boldsymbol{x})$가 데이터가 주는 사후 확률과 가장 잘 맞도록 정해야 한다. 이때 오차함수를 사용하며 오차가 최솟값이 되도록 매개변수 $\boldsymbol{W}$를 정하도록 한다. 그러면 어떤 오차함수를 사용하면 좋을까? 여기서는 최대 가능도함수를 사용하여 추정한 매개변수를 사용하자.

우리가 고려하고 있는 확률변수 d의 분포는 특정 값 이상이면 긍정 댓글이고 이하이면 부정 댓글이라고 판정하는 경우이므로 성공과 실패만 결과로 갖는 분포인 베르누이분포로서 $d \sim (\varphi(u))$이다. 다만 성공 확률이 $\varphi(u)$이고 실패 확률이 $1-\varphi(u)$인 실험을 실시했을 때 성공할 확률이라고 볼 수 있다. 확률변수 d는 0과 1만 값으로 갖는 베르누이 확률변수로서 확률분포는 다음과 같다.

$$
\begin{aligned}
P(d=1|\boldsymbol{x}) &= \varphi(u) = \varphi(\boldsymbol{W}\boldsymbol{x}) \\
P(d=0|\boldsymbol{x}) &= 1-\varphi(u) = 1-\varphi(\boldsymbol{W}\boldsymbol{x})
\end{aligned}
\tag{14.43}
$$

수식 (14.43)을 더 간단히 나타내면 확률 질량함수는 $d = 0,\ 1$일 때

$$
P(d|\boldsymbol{x}) = P(d=1|\boldsymbol{x})^d P(d=0|\boldsymbol{x})^{1-d} = (\varphi(\boldsymbol{W}\boldsymbol{x})^d (1-\varphi(\boldsymbol{W}\boldsymbol{x}))^{1-d}
$$

가 된다. $d \sim Bern(\varphi(u)) = Bern(\varphi(\boldsymbol{W}\boldsymbol{x}))$이고 매개변수는 $\mathbf{w} = \boldsymbol{W}$이다.

표본 데이터 $\boldsymbol{x}$의 개수가 M개로서 $\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \ldots,\ \boldsymbol{x}_M$인 경우의 확률변수 d의 가능도함수는

$$
L(\mathbf{w}) = \prod_{j=1}^{M} P(d_j|\boldsymbol{x}_j;\ \boldsymbol{W}) = \prod_{j=1}^{M} (\varphi(\boldsymbol{W}\boldsymbol{x}_j))^{d_j}(1-\varphi(\boldsymbol{W}\boldsymbol{x}_j))^{1-d_j} \tag{14.44}
$$

이다. 가능도함수 (14.44)의 최댓값 문제는 로그함수를 취해도 변화하지 않으므로 수식 (14.44)

에 로그를 취해보자.

$$L(\mathbf{w}) = \prod_{j=1}^{M}\left[d_j \log\{\varphi(\boldsymbol{W}\boldsymbol{x}_j)\} + (1-d_j)\log\{1-\varphi(\boldsymbol{W}\boldsymbol{x}_j)\}\right] \tag{14.45}$$

가 된다. 이 함수의 최댓값을 갖는 매개변수 $\boldsymbol{W}$를 찾는 것이 주어진 표본 데이터 $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$을 가장 잘 대표하는 베르누이분포를 찾는 것이다. 여기서 L의 최댓값을 찾는 것과 $-L$의 최솟값을 찾는 과정이 동일하므로 오차함수를

$$E(\mathbf{w}) = -L(\mathbf{w}) \tag{14.46}$$

라고 놓고 수식 (14.46)이 최솟값을 갖는 $\boldsymbol{W} = (w_1, w_2, \ldots, w_N)$을 구해보자. 즉, 이 오차함수 값이 최소가 되는 매개변수 $\boldsymbol{W} = (w_1, w_2, \ldots, w_N)$을 찾으면 된다. 이를 위해서 최소 제곱법이나 경사하강법 또는 확률적 하강법 등을 이용한다.

지금까지의 논의를 살펴보면 은닉층에서 출력값을 내기 위해 활성화함수로 로지스틱함수를 사용했다. 궁극적으로 다음과 같음을 알 수 있다. 사후 확률인 $P(d=1|\boldsymbol{x})$는 전확률 공식과 조건부확률 공식을 이용하면 다음과 같이 표현할 수 있다.

$$P(d=1|\boldsymbol{x}) = \frac{P(\boldsymbol{x}, d=1)}{P(\boldsymbol{x}, d=0) + P(\boldsymbol{x}, d=1)} \tag{14.47}$$

수식 (14.47)과 같이 정의하면 사후 확률 $P(d=1|\boldsymbol{x}) \approx d(\boldsymbol{x}; \mathbf{w}) = \varphi(u)$이므로

$$u \approx \log\frac{P(d=1, \boldsymbol{x})}{P(d=0, \boldsymbol{x})}$$

이다. 왜냐하면

$$\begin{aligned}
\log\frac{P(d=1, \boldsymbol{x})}{P(d=0, \boldsymbol{x})} &= \log\frac{P(d=1|\ \boldsymbol{x})\ \{P(\boldsymbol{x}, d=0) + P(\boldsymbol{x}, d=1)\}}{P(d=0|\ \boldsymbol{x})\ \{P(\boldsymbol{x}, d=0) + P(\boldsymbol{x}, d=1)\}} \\
&= \log P(d=1|\boldsymbol{x}) - \log P(d=0|\boldsymbol{x}) \\
&\approx \log\varphi(u) - \log(1-\varphi(u)) \\
&= \log\frac{\varphi(u)}{1-\varphi(u)} \\
&= \log\frac{\dfrac{1}{1+e^{-u}}}{1-\dfrac{1}{1+e^{-u}}} = u
\end{aligned}$$

이기 때문이다.

다중분류

지도 학습의 목표는 회귀와 분류로 나누어지는데 분류란 미리 정의된 가능성이 있는 여러 클래스 중 하나를 예측하는 것을 의미한다고 했다. 앞에서 여러 클래스 중 2개의 클래스 중에서 선택하는 형태를 이중분류라고 한다면 클래스의 종류가 여러 개인 경우를 **다중분류**(multiple classification)라고 한다. 예를 들어 특정 이미지가 사람인지 개인지 자동차인지로 분류하는 것이다. 또 다른 예로 주어진 필기체로 쓰여진 아라비아 숫자의 이미지를 보고 그 숫자가 0과 9까지의 수 중 어느 것인지 정확히 답하는 문제를 다중분류라고 할 수 있다.

따라서 입력 $\boldsymbol{x}$를 내용에 따라 유한개의 클래스로 분류하는 문제를 다중분류라고 한다. 포괄적으로 이진분류도 다중분류의 하나로 볼 수 있다. 이진분류와 다중분류를 구분하는 기준은 출력층 뉴런의 수이다. 출력층 뉴런이 오직 한 개만 있으면 이진분류 알고리즘이고 2개 이상이면 다중분류 알고리즘이다.

분류 종류에 따라 어떤 활성화함수와 오차함수를 사용하는지 살펴보자. 이진분류에서는 시그모이드함수를 활성화함수로 사용하였고 로지스틱함수를 오차함수로 사용하였는데, 다중분류에서는 활성화함수로 소프트맥스함수를 사용하며 손실함수로는 크로스엔트로피 오차함수를 사용한다.

다중분류기를 구현하는 방법은 크게 2가지로 나눌 수 있다. 하나는 여러 개의 클래스를 직접 처리하는 방식이고 다른 한 가지 방법은 이진분류기를 여러 개 사용하여 분류하는 방법이다. **랜덤 포레스트 분류기**(random forest classifier)나 **나이브 베이즈 분류기**(naive Bayes classifier)같은 알고리즘이 전자에 속한다. 이진분류기를 여러 개 사용해서 다중분류를 할 때는 각 분류기의 결정 점수 중 가장 높은 것을 클래스로 선택하는데 이러한 방법을 일대다 **전략**(OvA: One-versus-All approach)이라고 한다. 반면에 각 클래스를 2개씩 조합하여 이진분류기를 훈련시키는 방법이 있는데 이러한 방법을 **일대일 전략**(OvO: One-versus-One approach)이라고 한다. 일대일 전략을 사용하려면 클래스 종류가 N개일 때 $(N-1)N/2$개의 이진분류기가 필요하다.

여러 개의 클래스를 직접 처리하는 방식으로서 일대다 전략을 사용해보자. 가능한 솔루션이 N개인 분류 문제의 경우 일대다 솔루션은 가능한 결과 각각에 하나씩 총 N개의 이진분류기로 구성한다. 입력되는 이미지의 속성을 파악하여 해당 이미지가 N개의 클래스 중 어느 클래스에 속하는지를 나타내는 분류자를 학습한다. 문제를 단순화하기 위해 손글씨 인식의 예를 들어보기로 한다. 손으로 쓴 숫자의 흑백 이미지가 하나 주어졌다고 가정해보자. 그리고 이 숫자가 0부터 9까지의 수 중 어느 수인지를 결정하는 것이 알고리즘의 목표이다.

주어진 숫자 이미지의 모든 픽셀값은 0(흑색)과 1(흰색)로 구성된다. 따라서 픽셀값을 성분으

로 갖는 벡터를 입력 $\boldsymbol{x}$라고 하고 해당 이미지가 클래스 1부터 N 중 어느 클래스에 들어가는지 분류하는 것이 목표이다.

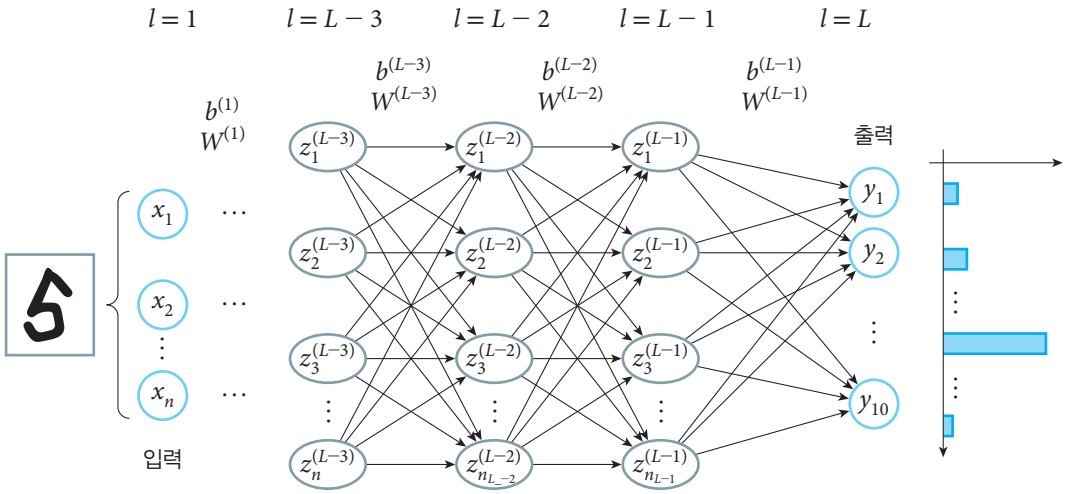


그림 14.18 숫자분류 다층인공신경망

그림 14.18에서 마지막 층인 층 L을 소프트맥스(softmax)층이라고 하고 입력되는 이미지에 의해 0부터 9까지의 숫자를 $y_1, \ldots, y_{10}$이라고 하고 이미지가 어느 것에 가까운지를 나타내는 확률값인 $p_1, \ldots, p_{10}$의 수치로 출력한다. 그러면

$$\boldsymbol{u}_k^{(L)} = \boldsymbol{W}^{(L)}\boldsymbol{z}^{(L-1)} + \boldsymbol{b}^{(L)},\ k = 1, 2, \ldots, 10 \tag{14.48}$$

이라고 표현할 수 있다.

소프트맥스함수(softmax function) σ는 K차원 실수 공간 $\boldsymbol{R}^K$에 정의되는 함수로서 σ: $\boldsymbol{R}^K \rightarrow \boldsymbol{R}^K$는 다음과 같이 정의한다. 정의역 $\boldsymbol{R}^K$의 독립변수 $\boldsymbol{z} = (z_1, \ldots, z_K)$에 대하여

$$\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}},\ i = 1, 2, \ldots, K \tag{14.49}$$

이다. 즉, 입력벡터 z의 각 성분 z_i에 표준 지수함수를 적용하고 이들 모두의 표준 지수함수를 더한 값으로 나누는 정규화한 값이다. 수식 (14.49)를 통해서 알 수 있는 것은

$$\sum_{i=1}^{K} \sigma(\boldsymbol{z})_i = 1 \tag{14.50}$$

이다. 우리의 예에서 $K = 10$이다.

수식 (14.48)에 활성화함수인 소프트맥스함수 σ를 적용하면, $\boldsymbol{u}^{(L)} = (u_1^{(L)}, \ldots, u_{10}^{(L)})$에 대하여

$$y_k \equiv z_k^{(L)} = \sigma(\boldsymbol{u}^{(L)})_k = \frac{\exp(u_k^{(L)})}{\sum_{i=1}^{10} \exp(u_i^{(L)})} \tag{14.51}$$

을 얻는다. 수식 (14.50)에 의해

$$\sum_{k=1}^{10} y_k = 1$$

이 됨을 알 수 있다. 위와 같은 다층인공신경망은 **교차 엔트로피**(cross entropy) 또는 **로그 손실**(log loss)를 이용하여 훈련하는 데 소프트맥스함수를 이용하면 다항 로지스틱 회귀를 비선형 방식으로 해결할 수 있다. 소프트맥스함수에서는 유닛 k의 출력인 z_k가 이 층 모든 유닛의 총 입력인 $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{10}$으로부터 결정된다. 이는 다른 활성화함수에서 유닛 k의 출력인 z_k가 해당 유닛으로 입력되는 총 입력 $\boldsymbol{u}_k$로부터만 결정되는 것과는 다르다.

수식 (14.51)를 보면 10개의 클래스인 이미지가 나타내는 숫자가 $i = 0, \ldots, 9$일 각각의 확률값 $p_1, \ldots, p_{10}$을 나타내고 있음을 알 수 있다. 즉,

$$p_i = \frac{\exp(\boldsymbol{u}_i^{(L)})}{\sum_{j=1}^{10} \exp(\boldsymbol{u}_j^{(L)})}, \; i = 1, \ldots, 10$$

은 이미지가 숫자 $i - 1$일 확률을 나타낸다. 그림 14.18에서 보면 표본 데이터는 n개의 독립변수를 갖는 n차원 벡터로 구성되어 있다. 하지만 소프트맥스함수는 10개의 클래스로 분류해야 하기 때문에 은닉층을 통과하여 최종적으로 10차원 벡터로 변화되어야 한다는 의미이다.

이제 오차에 대해 알아보자. 소프트맥스함수로 출력되는 값은 0과 1 사이의 값이고 각각은 특정 클래스가 정답일 확률을 나타낸다. 첫 번째 값인 p_1은 이미지가 표현하는 숫자가 0일 확률, p_2는 1일 확률, $\ldots$, p_{10}은 9일 확률이라고 생각하면 된다. 그렇다면 이 예측값과 비교할 수 있는 실제값의 표현 방법이 있어야 한다. 소프트맥스 회귀에서는 실제값을 아래와 같이 **원-핫벡터**(one-hot vector)로 나타낸다. 원-핫벡터란 입력되는 데이터의 유형이 달라서 처리하기 곤란할 경우에 모든 데이터를 한 가지 벡터 형태로 통일시키는 방법이다. 원래의 값 0부터 9까지 값이 입력값으로 활용될 경우에 모델은 이 값들의 크기가 다르기 때문에 값이 작은 것과 값이 큰 것에 어떠한 의미가 있다고 여겨 내부적 계산에 영향을 미칠 수 있다. 그래서 이러한 영향을 최

소화하기 위한 방법 중 하나로 원-핫벡터 방식의 표현을 활용한다. 이 예제에서 0부터 9를 원-핫벡터 형식으로 나타내면 다음과 같다.

$$0\text{의 원-핫벡터} = \begin{pmatrix}1\\0\\0\\\vdots\\0\\0\end{pmatrix},\ 1\text{의 원-핫벡터} = \begin{pmatrix}0\\1\\0\\\vdots\\0\\0\end{pmatrix},\ \ldots,\ 9\text{의 원-핫벡터} = \begin{pmatrix}0\\0\\0\\\vdots\\0\\1\end{pmatrix}$$

분류하고자 하는 클래스의 수가 10개이므로 이를 $C_1, \ldots, C_{10}$이라고 하자. 그러면 출력층의 유닛 k의 출력 $y_k = z_k^{(L)}$은 주어진 표본 데이터 $\boldsymbol{x} = (x_1, \ldots, x_n)$이 클래스 C_k에 속할 확률을 나타내는 것이라고 해석해도 무방하다. 즉,

$$P(C_k|\boldsymbol{x}) = y_k = z_k^{(L)} \tag{14.52}$$

이다. 그리고 $\boldsymbol{x}$는 이 확률값이 최대가 되는 클래스에 속한다고 판정한다. 다중분류에서도 이진분류와 마찬가지로 신경망에서 구현하는 함수를 각 클래스의 사후 확률에 대한 함수로 간주한다.

이 신경망의 목표 출력을 0과 1로 이루어진 10개의 원-핫벡터 대신에 다음과 같이

$$\boldsymbol{d}_n = \begin{pmatrix}d_{n_1}\\d_{n_2}\\\vdots\\d_{n_{10}}\end{pmatrix}$$

이라고 하자.

$$\text{여기서 } \boldsymbol{d}_{n_j} = \begin{cases}1, \text{ 대응하는 클래스가 정답 클래스일 때}\\0, \text{ 대응하는 클래스가 정답이 아닐 때}\end{cases}$$

예를 들어 그림 14.18에서 표본 데이터 이미지가 숫자 '5'를 나타내고 있는데, 입력 $\boldsymbol{x}_n$이 '5'를 나타내는 이미지의 픽셀값을 0과 1로 나타낸 데이터라고 하면 그에 따라 정답 클래스가 C_6일 경우에 이 입력의 목표 출력은 $\boldsymbol{d}_n = (0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)^T$이다. T는 열벡터를 행벡터로 변환하는 전환벡터(tranpose vector)를 의미한다.

이제 N개로 이루어진 훈련 데이터 집합 $\{(\boldsymbol{x}_i, \boldsymbol{d}_i)\,|\,i = 1, 2, \ldots, N\}$을 이용하여 모델로부터 얻은 사후 확률 $P(\boldsymbol{d}|\boldsymbol{x})$가 데이터가 주는 사후 확률과 가장 잘 맞도록 정해야 한다. 즉,

$$P(\boldsymbol{d}|\boldsymbol{x}) = \prod_{k=1}^{10} P(C_k|\boldsymbol{x})^{d_k} \tag{14.53}$$

이다. 다시 말해서 훈련 데이터 $(\boldsymbol{x}_i, \boldsymbol{d}_i)$가 숫자 5에 대한 픽셀 데이터값으로 이뤄진 벡터 $\boldsymbol{x}_i$와 $\boldsymbol{d}_i = (0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)^T$라고 가정해보면

$$P(\boldsymbol{d}_i = (0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)^T|\boldsymbol{x}_i) = P(C_6|\boldsymbol{x}_i)$$

라는 의미이다. 이 사후 확률의 수식 (14.53)에 위의 수식 (14.52)를 대체시키면 주어진 훈련 데이터 $\{(\boldsymbol{x}_i, \boldsymbol{d}_i)\,|\,i = 1, 2, \ldots, N$에 대한 $\mathbf{w}$의 가능도를 다음 수식으로 유도할 수 있다.

$$L(\mathbf{w}) = \prod_{i=1}^{N} P(\boldsymbol{d}_i|\boldsymbol{x}_i; \mathbf{w}) = \prod_{i=1}^{N}\prod_{k=1}^{10} P(C_k|\boldsymbol{x}_i)^{d_{ik}} = \prod_{i=1}^{N}\prod_{k=1}^{10} (y_k(\boldsymbol{x}; \mathbf{w}))^{d_{ik}} \tag{14.54}$$

이 가능도함수 (14.54)에 로그를 취하고 음의 값을 취한 것을 오차함수라고 하고 이 함수가 어떤 매개변수 $\mathbf{w}$에서 최솟값을 갖는지 알아보면 된다. 즉, 오차함수는

$$E(\mathbf{w}) = -\sum_{i=1}^{N}\sum_{k=1}^{10} d_{ik}\log_{y_k}(\boldsymbol{x}_i; \mathbf{w}) \tag{14.55}$$

이다. 함수 (14.55)를 자세히 살펴보면 예측값에 대한 분포와 목표 출력값 분포의 차이를 나타내고 있다. 즉, 예측값의 분포를 q라고 하고 목표 출력값의 분포를 p라고 할 때에 정의한 교차 엔트로피인

$$H(p, q) = -E_p[\log q] \tag{14.56}$$

와 동일한 값을 갖는다. 여기서 $E_p[\cdot]$은 분포 p에 대한 기댓값이다. 따라서 수식 (14.55)는 실제로 예측값과 목표 출력값의 확률분포가 얼마나 차이가 나는지를 나타내는 교차 엔트로피이다.

이번에는 클래스 C_j의 사후 확률을 다른 각도에서 살펴보자. 정의에 의하면 클래스 C_j의 사후 확률은

$$P(C_j|\boldsymbol{x}) = \frac{P(\boldsymbol{x}, C_j)}{\sum_{k=1}^{10} P(\boldsymbol{x}, C_k)} \tag{14.57}$$

이다. 여기서 $u_j = \log(P(\boldsymbol{x}, C_j))$라고 놓으면, 위의 사후 확률 (14.57)을 다음과 같이 나타낼 수 있다.

$$P(C_j|\boldsymbol{x}) = \frac{\exp(u_j)}{\sum_{k=1}^{10}\exp(u_k)} \tag{14.58}$$

수식 (14.58)은 입력 $\boldsymbol{u} = (u_1, \ldots, u_{10})$에 대한 소프트맥스함수와 같다는 것을 알 수 있다.

여기서 소프트맥스함수의 특성 몇 가지를 살펴보기로 하자. 소프트맥스함수는 다양한 다중분류 방법에 사용된다. 예를 들어 다중분포 로지스틱 회귀(소프트맥스 회귀)나 다중분류 선형 판별 분석, 나이브 베이즈 분류 및 인공신경망 등에 응용된다. 소프트맥스함수는 모든 성분에 상수 $\boldsymbol{c} = (c, \ldots, c)$를 입력값 $z = (z_1, z_2, \ldots, z_{10})$에 더해도 출력값에 변화가 없다. 즉,

$$\sigma(\boldsymbol{z} + \boldsymbol{c}) = \sigma(\boldsymbol{z})$$

가 성립한다. 왜냐하면 $e^{z_i + c} = e^{z_i} \cdot e^c$이기 때문이다. 그래서

$$\sigma(\boldsymbol{z} + \boldsymbol{c})_j = \frac{e^{z_i + c}}{\sum_{k=1}^{10} e^{z_k + c}} = \frac{e^{z_i} \cdot e^c}{\sum_{k=1}^{10} e^{z_k} \cdot e^c} = \sigma(\boldsymbol{z})_j$$

이다. 이러한 특성 때문에 신경망의 가중치를 학습할 때 출력층의 가중치 $\boldsymbol{W}^{(L)}$이 잘 수렴하지 않고 학습 속도가 느려지는 단점이 있다. 이를 극복하기 위해 출력층에 몇 가지 방법으로 제약을 주어 문제를 해결한다. 첫 번째 방법은 최적화를 실행할 때 가중치를 감쇠시키는 방법이다. 가중치에 제약이 가해지므로 수렴 문제가 해결된다. 두 번째 방법은 출력층의 유닛 중 하나를 선택하여 그 유닛의 입력을 강제로 0이라고 놓는 방법이다. 임의로 선택한 유닛 k의 입력 u_k를 0으로 선택하는 방법이 의미가 있는 이유는 바이어스 u_0가 값으로 0을 갖는 것과 동일하다고 볼 수 있기 때문이다. 이렇게 자의적으로 값을 0으로 선택해도 무방하다는 뜻이다.

딥러닝의 종류

딥러닝의 종류는 **딥신경망**(DNN: Deep Neural Network), **합성곱신경망**(CNN: Convolution Neural Network), **순환신경망**(RNN: Recurrent Neural Network) 등으로 분류할 수 있다. 이들을 간단히 설명해보자. 이 책에서는 부족하지만 이들이 무엇을 의미하는지 간단하게 서술하기로 한다. 이들에 대해 더 알아보려면 인공지능 전문 서적을 참조하길 바란다.

먼저 DNN에 대해 알아보자. ANN 기법의 여러 가지 문제가 해결되면서 모델 구축에 은닉층의 수를 늘려 학습 결과를 향상시키는 방법을 DNN이라고 한다. 컴퓨터가 스스로 분류 레이블을 작성하고 공간을 왜곡하고 데이터를 구분하는 과정을 반복하여 최적의 구분선을 찾아낸다. 많은 데이터와 반복 학습, 사전 학습과 오류 역전파 기법을 활용한다.

두 번째로 CNN에 대해 알아보자. 이미지를 인식하는 데 기존 방법들은 동일한 이미지라 하더라도 위치가 변동되거나 방향이 다르거나 이미지가 왜곡되었을 경우에 이미지를 분류하는 데 어려움이 많았다. 이러한 환경 변화에도 불구하고 이미지를 분류해낼 수 있는 방법으로 **합성곱**(convolution)과 **풀링**(pooling)이라는 기법을 활용하였다. 합성곱이란 데이터의 특성을 추출하는 과정으로서 데이터에 각 성분의 인접 성분들을 조사하여 특성을 파악하고 파악된 특성을 한 장으로 출력하는 과정이다. 반면에 합성곱 과정을 거친 층의 차원을 낮춰 크기를 줄여주고 잡음을 상쇄시키는 과정을 풀링이라고 한다. CNN은 합성곱층과 풀링층을 반복하는 복합적 구성을 통해 이미지로부터 추상화된 정보를 추출하는 기법이다.

세 번째로 RNN에 대해 알아보자. RNN 알고리즘은 반복되며 순차적인 데이터 학습에 특화된 딥러닝 알고리즘의 한 가지이다. 특성은 내부에 순환구조를 가지고 있어 이전 상태의 상태값이 다음 계산의 입력으로 들어가서 결과에 영향을 미친다는 점이다. 이를 이용하여 과거의 학습에 가중치를 주어 현재 학습에 반영한다. 이 방법으로 반복되며 순차적 특성을 가진 데이터 학습의 한계를 해결할 수 있는 알고리즘이다. 주로 음성 웨이브 형태를 파악하여 음성 인식을 하거나 단어나 글자 등의 앞뒤 성분을 파악하여 기계 번역을 하거나 이미지 설명 등을 할 때 RNN을 사용한다.

역전파 알고리즘

머신러닝에 사용하는 알고리즘은 경사하강법을 활용하여 주어진 매개변수 $\mathbf{w}$를 가진 목적함수 $E(\mathbf{w})$의 최솟값을 구한다. 이를 위해 매개변수의 값들을 지속적으로 업데이트하여 최솟값을 갖는 매개변수의 값이 무엇인지 알아내는 것이다. 경사하강법은 이 함수가 정의되는 영역 안의 한 시작점 $\mathbf{w}_0$를 먼저 정하고 $\mathbf{w}_1$으로 업데이트한다. 이를 반복해서 다음과 같이 수행한다. 현재 $\mathbf{w}_i$가 주어졌을 때 업데이트되는 다음 점 $\mathbf{w}_{i+1}$은 다음 수식 (14.59)와 같이 계산한다.

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma_i \nabla E(\mathbf{w}_i),\ i = 1, 2, \ldots \tag{14.59}$$

γ_i는 학습 속도를 조절하는 매개변수이다. 따라서 경사하강법을 수행하려면 목적함수의 미분값인 $\nabla E(\mathbf{w}_i),\ i = 1, 2, \ldots$를 구해야 한다. 기호 ∇는 '델'이라고 읽는다. 이들 미분을 계산하는 것이 은닉층 중에서 입력에 가까운 층의 매개변수일수록 까다롭다.

왜 어려운지 예를 들어 설명해본다. 한 개의 표본 데이터 $\boldsymbol{x}_n$에 대한 제곱 오차 $E_n(\mathbf{w}) = \|y(\boldsymbol{x}_n) - \boldsymbol{d}_n\|^2$을 l번째 층의 가중치 $\boldsymbol{W}^{(l)}$의 한 성분인 $w_{ji}^{(l)}$로 미분한다고 해보자. 그러면

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = (y(\boldsymbol{x}_n) - \boldsymbol{d}_n)^T \frac{\partial_y}{\partial w_{ji}^{(l)}} \tag{14.60}$$

이 된다. 그다음 수식 (14.60)의 우변에 있는 미분 $\partial y / \partial w_{ji}^{(l)}$을 구해야 한다. 하지만 함수 $y(\boldsymbol{x})$ 안

에 있는 $w_{ji}^{(l)}$이 다음 수식 (14.61)과 같이 활성화함수가 여러 겹으로 적용되어 있는 식의 깊은 곳에 있기 때문에 미분의 연쇄 법칙을 여러 차례 적용해야 하는 방식으로 문제를 풀어야 한다.

$$\begin{aligned} y(\boldsymbol{x}) &= \varphi(\boldsymbol{u}^{(L)}) \\ &= \varphi(\boldsymbol{W}^{(L)}\boldsymbol{z}^{(L)} + b^{(L)}) \\ &= \varphi(\boldsymbol{W}^{(L)}\varphi(\boldsymbol{W}^{(L-1)}\boldsymbol{z}^{(L-1)} + \boldsymbol{b}^{(L-1)}) + \boldsymbol{b}^{(L)}) \\ &= \varphi(\boldsymbol{W}^{(L)}\varphi(\boldsymbol{W}^{(L-1)}\varphi(\boldsymbol{W}^{(L-2)}\boldsymbol{z}^{(L-2)} + \boldsymbol{b}^{(L-2)}) + \boldsymbol{b}^{(L-1)}) + \boldsymbol{b}^{(L)}) \\ &= \varphi(\boldsymbol{W}^{(L)}\varphi(\boldsymbol{W}^{(L-1)}\varphi(\boldsymbol{W}^{(L-2)}\varphi(\ldots\varphi(\boldsymbol{W}^{(l)}\boldsymbol{z}^{(l)} + \boldsymbol{b}^{(l)})\ldots) + \boldsymbol{b}^{(L-2)}) + \boldsymbol{b}^{(L-1)}) + \boldsymbol{b}^{(L)}) \end{aligned} \tag{14.61}$$

따라서 계산 비용도 많이 증가할 뿐만 아니라 프로그래밍하기도 까다롭다. 게다가 층별로 서로 다른 활성화함수까지 사용하면 이 문제는 더 복잡해진다. 따라서 이런 문제를 해결할 때 **역전파 알고리즘**(backpropagation algorithm)을 활용한다. 역전파 알고리즘은 다음 두 단계로 수행된다.

1. 트레이닝 데이터를 입력하여 전방향 연산을 수행하고 그 결과로 나온 신경망의 예측치와 실제값과의 차이인 오차를 구한다.
2. 오차를 신경망을 구성하는 노드들로 역전파시킨다.

역전파 알고리즘은 오류를 역전파하여 매개변수 **w**를 업데이트하는 방식이다.

훈련 데이터 집합 $\{(\boldsymbol{x}_i, \boldsymbol{d}_i) \mid i = 1, 2, \ldots, N\}$에 대하여 모델의 오차함수 또는 손실함수는 예상값 $\boldsymbol{y}(\boldsymbol{x}_i; \mathbf{w})$와 목푯값인 $\boldsymbol{d}_i$ 간의 차이를 나타내는 데 여러 가지 종류가 사용되기 때문에 다음과 같이 일반적인 손실함수의 표현인

$$C(\boldsymbol{d}_i, \boldsymbol{y}(\boldsymbol{x}_i)) \tag{14.62}$$

라고 나타내기로 한다.

모델의 값을 계산할 때는 가중치를 고정된 값이라고 하고 입력되는 변수인 $\boldsymbol{x}_i$들이 독립적으로 변하면서 출력값이 계산된다. 그리고 목표 출력값은 알 수 없으며 네트워크는 오차함수를 포함하지 않는다. 하지만 모델을 훈련시킬 때는 표본 데이터인 입력값과 출력값의 쌍이 고정되며, 가중치는 변수로 간주되고 네트워크는 오차함수 형태로 끝난다.

역전파에서는 고정된 입력값-출력값 쌍인 $(\boldsymbol{x}_i, \boldsymbol{d}_i)$들에 대해 기울기를 계산한다. 여기서 변수는 가중치 $w_{jk}^{(l)}$와 바이어스 $b_j^{(l)}$이다. 기울기 $\partial C/\partial w_{jk}^{(l)}$의 개별 성분은 연쇄 법칙을 이용하여 계산할 수 있지만 수식 (14.61)에서 언급한 바와 같이 매우 비효율적이다.

역전파 알고리즘에서는 중복되는 계산을 피하고 불필요한 중간층의 값들을 계산하지 않으면서 기울기를 계산한다. 이는 각 층의 기울기를 계산하여 수행할 수 있다. 특히 각 층의 가중치가 부여된 입력의 기울기를 계산하여 수행한다. 뒤에서 앞으로 가는 역방향의 기울기를 $\delta^{(l)}$이라고 나타낸다.

역전파 알고리즘에 대해 더 자세히 알고 싶다면 인공지능 전문 서적을 참조하길 바란다.

14.5 인공지능에서 수학의 역할

인공지능을 설명할 때 인공지능을 어떠한 목적에서 사용하느냐에 따라 사용자별로 다양하게 설명한다. 일부에서는 수학의 깊은 지식 없이도 인공지능을 이해하고 사용하는 데 큰 문제가 없다고 말하는 사람도 있다. 반면에 수학적 지식이 없이는 인공지능을 이해할 수 없다고 말하는 사람들도 있다. 두 가지 다 맞는 말이기도 하고 틀린 말이기도 하다.

전자의 경우 수학 지식이 없어도 쉽게 응용할 수 있도록 개발된 인공지능 툴이 다양하게 제공되고 있기 때문에 이러한 툴만 잘 활용하면 훌륭한 인공지능 애플리케이션을 만들어낼 수 있다. 인공지능에 대한 수학 지식이 풍부한 개발자들이 쉽게 활용할 수 있도록 만들어 놓았기 때문이다. 자동차에 대한 전문 지식이 없어도 일반인이 멋진 차를 구매해서 운전할 수 있는 논리이다.

하지만 지금까지 이 책에서 공부해온 기초 수학 내용과 이 장에서 제시한 다양한 인공지능 기법을 살펴보면 수학적 개념이 핵심을 이루고 있다는 것을 알 수 있다. 따라서 인공지능 응용을 명확히 하고 발전시키기 위해서는 기초 수학 지식이 탄탄해야 한다. 어떠한 인공지능 모델이 어떠한 수학적인 알고리즘으로 구성되는지를 안다면 적용해야 할 상황이 일부 변경되었을 때 모델을 적합하게 수정하고 상황에 맞게 조정할 수 있다.

이 절에서 언급한 수식들이 복잡해보이지만 13장까지 전개된 기초 수학 지식을 충분히 소화하였다면 그 내용을 이해하는 데 어려움이 없을 거라고 생각된다. 어떤 문제를 해결하기 위해 인공지능을 활용한다는 의미는 수집된 데이터로부터 올바른 데이터 셋을 만들고 이를 이용한 알고리즘을 만들어서 훈련시키고 훈련된 알고리즘에 테스트 데이터를 적용하여 충분히 성능을 발휘하도록 지속적으로 알고리즘을 수정하는 것이다. 이렇게 만들어진 모델에 실제 데이터를 입력하고 출력되는 결과를 해당 입력 데이터에 대한 예측이라고 판정하는 과정이 인공지능 응용이라고 할 수 있다. 알고리즘을 설계하는 데에는 수학적인 이론이 필요하다. 그리고 이를 코딩하는 문제는 별개의 문제일 수 있다.

인공지능에서 수학의 역할을 이해하기 위해 지금까지 기초 수학 개념과 이들을 좀 더 코딩과 연관 지어 이해하기 위해 파이썬으로 어떻게 코딩을 하는지 많은 사례를 들어 설명하였다. 제시된 다양한 기초 수학 내용에 좀 더 친숙해지기 바라며 인공지능에서의 수학 역할이 중요함을 인식하는 계기가 되길 바란다.

이 절에서는 인공지능과 수학의 관계를 설명하고 앞에서 배운 인공지능에 필요한 수학 영역을 정리하여 설명하고 끝으로 인공지능을 체계적으로 학습하기 위해 대학 과정에서 뿐만 아니라 초중등 교과 과정에서부터 간단한 인공지능의 개념과 기초코딩에 대한 학습이 선행되어야 한다는 점을 강조하고 싶다.

인공지능과 수학

인공지능 이론을 응용하여 다양하면서도 놀라울 정도의 서비스를 제공하고 있다. 그러면 이러한 인공지능 툴은 어떻게 만들어지는가? 인공지능 툴을 사용자에게 편리하고 쉽게 개발하려면 반드시 수학적 지식이 필요하다. 인공지능의 문제를 해석하고 학습하고 오차를 줄이고 최적의 답을 구해내는 방식은 모두 다 수학적인 지식에 배경을 두고 있는 이론들이기 때문이다. 물론 이론을 알고리즘으로 구현하고 구현된 알고리즘의 계산은 성능 좋은 컴퓨터에 의존하고 있다. 하지만 인공지능이 수학 지식 없이는 절대로 발전할 수 없다. 많은 인공지능 학자들이 실제 인간이 사고하는 방식을 모방하고 인공지능을 이용한 사고의 결과가 인간의 사고를 넘어설 수 있도록 보다 정확하고 효율성 있는 알고리즘을 개발하기 위하여 다양한 기존의 수학 지식을 활용하고 있으며 새로운 수학적 이론들을 생각해내고 있다.

그런 의미에서 우리가 이 책에서 배우는 내용은 인공지능의 최첨단 영역에서 다뤄지는 수학적 지식에는 미치지 못하지만 최소한 인공지능에서 활용하는 기본적인 수학 지식에 대한 개괄적이고 보편적인 내용을 소개하고 있다. 인공지능을 앞으로 계속 학습하고 이 분야에 종사하고자 한다면 다른 사람이 만들어놓은 인공지능 툴만 사용할 줄 아는 것만으로는 부족하다고 생각된다. 해당 인공지능 툴이나 서비스가 어떠한 수학적 지식을 바탕으로 개발되었는지를 알고 활용한다면 해당 툴이나 서비스가 가진 장점과 단점을 쉽게 파악할 수 있을 뿐만 아니라 좀 더 개선시킬 수 있는 아이디어도 가질 수 있게 된다.

인공지능과 수학을 간단히 정리해보면 인공지능의 기초적 수학 지식, 취급하는 자료의 표현 문제, 분류와 예측 문제, 그리고 최적화 문제로 나눌 수 있다. 각각의 영역에서 기초 수학 지식은 그렇게 높은 수준의 수학 지식이면서도 인공지능을 비교적 쉽게 이해할 수 있게 도움을 준다. 미분적분에서 함수의 극한, 이차 함수의 미분, 극댓값이나 최솟값을 구하는 기법이 인공지능의 오차를 최소화 하는 방법인 경사하강법의 핵심 기법이기 때문이다. 고등학교 수준의 집합론 개념, 논리회로, 확률 통계 개념 등이 인공지능을 응용하는 데 수시로 활용되고 있다. 물론 대학교 수준의 수학으로서 행렬과 벡터, 수준 높은 해석학과 확률 통계 개념, 회귀분석, 조건부 확률, 손실함수 등 다양한 수학적 지식도 활용되고 있다. 또한 최적화 문제로 들어가게 되면 다양한 학습방법으로 경사하강법, 확률적 경사하강법, Adam, Nadam 등 좀 더 깊이 있는 수학적 이론이 필요하다. 하지만 이들 이론이 엄두가 나지 않을 정도의 높은 수학적 지식을 필요로 하

는 내용이 아니기 때문에 인공지능을 공부하면서 필요한 지식은 수시로 찾아서 익히는 방식으로 학습하면 된다.

인공지능에서는 개발된 알고리즘을 훈련시켜 오차를 최소화하는 최적화 단계가 반드시 필요하다. 따라서 최적화에 대한 학습 방법으로 지도 및 비지도 학습과 강화학습을 소개할 필요가 있다. 이를 통해 인공지능 작동 목표 중 하나가 손실함수를 최소화하는 것임을 이해하도록 해야 한다. 이를 위해 손실함수라고 하는 수학적 함수의 최솟값을 찾는 알고리즘의 중요성을 인식하게 된다.

인공지능에서 가장 자주 사례로 등장하는 자율주행차, 스마트 헬스 케어, 스마트 냉장고나 TV 같은 가전제품, 재난구호 로봇, 스마트 팩토리 등에서 어떠한 수학적 알고리즘이 활용되고 있는지 깊이 알아보고자 하는 호기심을 자극하도록 해야 한다. 인공지능에 흥미를 느낄 수 있게 하는 방식은 실제 적용되는 인공지능 서비스를 사용해보는 것이다. 사실 다양한 인공지능 기술이 이미 우리 생활의 깊숙한 곳까지 들어와 있다. 스마트폰을 사용하면서 느낄 수 있는 추천 시스템, 자동번역, 음성인식 비서 등은 이제 인공지능이라고 여겨지지 않을 정도로 보편화된 기술이 되었다.

인공지능 교육과 관련하여 다양한 계층에서 관심을 갖고 있다고 하는데 이에 대해 알아보자. 최근에 인공지능에 대한 관심이 높아지면서 인공지능 교육을 초등학교부터 도입해야 한다는 주장에 맞춰 다양한 방식으로 교과과정에 편입하기 위한 노력이 진행되고 있다. 벌써부터 인공지능을 교과과정에 편성했을 때 어떠한 내용을 포함시켜야 하는지에 대한 토론이 다양하게 진행되고 있으며 표준교과과정에 담아야 할 내용에 대한 논란도 있어 보인다. 절대로 내용이 어려워서는 안된다는 교육관을 가진 집단과 내용이 부실한 핵심 없는 보여주기식 교육이 되어서는 안된다는 교육관을 가진 집단 간의 논란이 있어 보인다. 어떠한 결론이 나든 인공지능이 결코 어려운 내용이 아니라는 인식을 갖도록 교과과정이 편성되어 흥미를 끌 수 있도록 해야 하며 학생들이 친숙해질 수 있도록 접근해야 한다는 원리에 충실해야 할 것이라고 생각된다.

인공지능 교육이 너무 응용사례에만 편중되어도 안 되지만 너무 인공지능의 도구기능인 수학만을 강조해서도 안된다. 인공지능으로 구현할 수 있는 사례를 통해 흥미를 유발하고 이러한 응용 서비스를 개발하기 위해 어떠한 지식이 필요한지 알도록 해주어야 한다. 그래야 인공지능을 개발하고 사용자에게 편리한 서비스를 만들기 위한 노력을 하면서 점차적으로 수학이 인공지능의 골격을 이루는 핵심적인 지식이라는 것을 알고 필요한 수학 지식을 쌓아나갈 수 있을 거라고 생각되기 때문이다.

응용단계에서는 반드시 필요한 것이 인공지능을 구현하기 위한 컴퓨터 프로그램 언어에 익숙해지기이다. 파이썬 같은 프로그램 언어에 쉽게 접근할 수 있게 어렵지 않으면서도 흥미를 일

으키게 하는 사례 중심의 프로그래밍 과정을 초보 단계부터 잘 접목할 필요가 있다. 실제로 인공지능의 전 과정과 절차를 모방해볼 수 있는 간단한 응용 프로그래밍 과정이 소개될 수 있으면 좋을거라고 생각한다. 실제로 데이터 수집, 데이터 가공 및 분석, 훈련 데이터와 검증 데이터 처리, 훈련과정, 오차 최소화 알고리즘, 최종 알고리즘까지의 전 과정을 쉽게 이해해볼 수 있는 다양한 파일롯 프로젝트들이 개발되어야 할 것이라고 생각된다.

인공지능에 필요한 수학 분야

이 책에서는 인공지능에서 머신러닝 분야의 딥러닝을 이해하는 데 최소한으로 필요한 수학 내용을 정리하였다. 따라서 필요한 기초 수학 내용을 도입으로 선형대수학, 미분적분학, 그리고 확률과 통계 내용 중 인공지능에서 필요한 핵심내용만 골라 비교적 쉽게 이해할 수 있는 내용만으로 구성하였다. 데이터 표현 방법과 행렬과 벡터에 대한 이해는 인공지능 시대에 가장 기초적이고 핵심적인 수학적 표현 방법을 이해하기 위한 출발이다. 특히 행렬은 인공지능에서 데이터의 공간 변환, 인공지능망의 최적 설계, 확률의 추출 과정 등에서 필수적 도구다.

한편 딥러닝은 학습 과정에서 내부 연결망과 그 매개변수들을 최적화한다. 이 과정을 통해 완성된 인공지능 모델의 성능은 사물 인식 분야에서 인간의 능력을 넘어선다. 또한 인공지능의 최적화 과정에서 수학의 '미분' 개념이 사용된다. 특히 편미분이 많이 사용된다. 마지막으로 인공지능이 판단의 근거로 삼는 값은 계산된 확률이다. 인공지능 모델은 가장 확률이 높은 답을 정답으로 제시한다. 확률값의 정확성에 따라 모델의 성능이 결정된다. 또한 확률과 통계뿐만 아니라 다양한 인공지능 모델에서 정보 이론, 게임 이론, 이산수학 등 고급수학의 개념에 대한 다양한 지식이 활용된다. 이 책에서 언급하지 못한 다양한 수학적 지식이 필요하다. 다른 인공지능 문서를 이해하기 위해 필요한 수학 지식이 있다면 필요할 때마다 연관된 수학 분야의 전문서적을 공부해서 고난도 인공지능 지식을 습득해나가길 바란다.

예를 들어 자율주행에서 중요하게 여기는 사항으로 물체를 어떻게 인식시키는가이다. 따라서 카메라가 제공하는 2차원 영상자료를 토대로 특성을 찾아내는 작업을 해야 한다. 이러한 2차원 영상자료는 픽셀의 크기로 결정되는 행렬로 표현된다. 이러한 행렬의 분석을 통해 보행자와 도로 주변의 사물을 인식하고 주행을 해나간다. 또한 인공지능 모델이 인식하고 안 하고는 대상에 대한 계산된 확률값에 의존해서 결정한다. 이러한 확률값은 학습 데이터의 양이 증가하면 정확도가 높아지므로 데이터의 양을 늘려서 정확도를 높일 수 있다.

인공지능과 수학과 직업

갈릴레오는 "수학은 우주를 설명하는 언어다"라고 말하였다. 인공지능이 전례 없이 우리의 생활 곳곳에 스며들고 있다. 인공지능이 마치 새로운 세계로 인도하는 도구처럼 과장되거나 범접

하기 어렵거나 신비한 형태로 다가오기도 한다. 하지만 간단히 말해서 인공지능은 신비로울 게 하나도 없는 순수하게 수학으로 이루어진 학문이라고 말할 수 있다. 인공지능과 수학은 뿌리가 하나인 나무의 서로 다른 가지일 뿐이다. 생각하는 기계와 인간의 행동을 모방하는 가능성은 모두 수학적 개념에서 출발하는 최종 목적이다. 인공지능 분야에서 직업을 갖기를 희망하거나 이 분야를 이해하고 응용력을 높이려면 인공지능을 사이언스 픽션처럼 보아서는 절대 안 된다. 대신에 수학 지식을 쌓아야 한다. 그래야 수학 지식의 창을 통해 인공지능의 진면목을 살펴볼 수 있다.

선진국에서의 수학 교육은 보편적인 면에서는 우리나라 교육보다 수준이 낮아 보이지만 실제 전문성을 필요로 하는 고등 교육 기관에서의 수학 교육은 한국보다 심도가 있으며 다양성도 풍부하다. 인공지능에서 필요하다고 해서 인공지능에 특화시켜 수학을 가르치기도 하지만 대체로는 미적분학 및 함수해석 등을 취급하는 해석학, 집합론이나 정수론 선형대수학 및 추상대수학을 취급하는 대수학, 도형과 유클리드 및 비유클리드 기하에 집중하는 기하학, 기초 통계부터 고급 확률론까지 전 분야에 걸쳐 다양한 학과에서 심도 있는 교육 과정을 제공하는 방식으로 인공지능과 수학의 연결고리를 만들어나가고 있다. 물론 대학원 과정에서 인공지능에 필요한 고급수학에 대한 활발한 연구가 진행되고 있으며 엄청나게 많은 인공지능 관련 논문이 출판되고 있다. 대학에서는 이들 분야를 융합한 수학 지식을 제공하고 이러한 수학 지식으로 훈련된 인재는 인공지능 분야에서 필요한 고급수학 지식으로 무장하고 창의적인 응용력을 발휘할 수 있다. 수학은 통해서 인공지능의 핵심이 되는 논리적 합리성을 만들어나갈 수 있고 정밀하게 목표를 달성할 수 있게 된다는 의미이다.

세계 경제 포럼(WEF)에서는 인공지능 분야에서 대단위의 새로운 일자리를 창출하고, 많은 기존 일자리를 대체할 것이라고 예측했다. 결국 앞으로는 누구든 인공지능 지식에 대한 상식이나 지식이 없으면 일자리를 찾는 데 어려움을 겪을 수밖에 없는 환경이 다가오고 있다는 의미이다. 좀 더 지엽적으로 말해서 인공지능을 이해하는 툴인 기초 수학 지식 없이는 취업이 어렵다는 전망이다. 인문계열, 자연계열, 경상계열, 예체능계열 등 분야와 상관없이 우리나라 모든 고등학교 학생을 대상으로 인공지능에 필요한 기초 수학 개념을 교육해야 한다. 더구나 대학생은 전문적인 인공지능에 대한 이해도를 높여나가야 한다. 그 중에서도 보편적 인공지능 활용자가 아닌 머신러닝 엔지니어, 데이터과학자, 로봇 과학자가 되고자 한다면 좀 더 수준 높은 수학 지식을 갖출 수 있도록 집중해서 학습해야 한다.

지금은 4차 산업혁명 시대이며 이 시대의 우리는 급격한 산업변화를 경험하고 있다. 따라서 미래세대의 경쟁력 확보라는 의미에서 우리나라 수학 교육의 커리큘럼 내용과 교육 방법 및 교육 목적이 재편되어야 한다. 특히 인공지능과 연관된 기초 수학 지식에 대한 이해도를 높여나가야 한다. 물론 인공지능에서 활용되는 수학적 개념을 중고등학교 교실에서 교육하고 평가하기가

어려운 점이 있는 게 분명하다. 하지만 최소한의 개념 이해와 활용의 예를 설명할 수 있는 다양한 교육 자료를 제공하여 학생이 수학을 공부하고자 하는 동기를 유발시켜야 한다. 이제 수학이 추상적인 개념을 설명하고 거기에 기반한 구름에 떠있는 듯한 반복적 문제 풀이나 하는 수준에 머물러서는 안 된다.

인공지능으로 구현되는 놀랄만한 다양한 응용 서비스를 더 이상 신비롭게 볼 필요가 없다. 이러한 인공지능 기법들이 우리 눈앞에서는 마술과 같이 전개된다 하더라도 이는 그렇게 보이기만 할 뿐 실제로 뒤에서는 수학적 이론에 기반해 설계된 알고리즘을 이용하여 컴퓨터가 엄청난 양의 계산을 해나가고 있다는 점을 간과해서는 안 된다.

정리

이 장에서는 인공지능에서 실제로 문제를 해결하는 데 어떠한 절차로 모델을 만들어 나가는지 살펴보았다. 이러한 절차를 전개하는 데 다양한 수학적 지식을 사용한다는 사실을 보여주고 있다. 머신러닝의 한 분야로서 딥러닝의 핵심은 바로 수학이다. 인공신경망을 구성하고 표현하는 방식이 기존의 선형대수학에서 다루는 행렬 지식을 활용하고 있다는 점을 알 수 있다. 인공지능에서의 지도 학습과 비지도 학습을 통해 얻으려고 하는 회귀와 분류에 활용하는 다양한 지식도 기초 통계와 확률론이라는 사실을 확인할 수 있었다.

회귀 분석에서는 다양한 회귀 분석 방법을 설명하였는데 표본 데이터를 직교좌표계에 나타낸 뒤에 이를 대표하는 직선으로 표현되는 선형회귀 분석과 변수가 여러 개인 경우의 다중회귀 분석에 대해 설명하였다. 회귀 분석에서 가장 중요한 내용은 오차를 정의하는 방법이고 이때 사용하는 최소 제곱 오차의 개념과 이 값을 최소화하기 위한 경사하강법에 대해 정리하면서 그 의미를 이해하였다. 인공지능의 학습 속도를 조절해가면서 최소 오차를 찾는 경사하강법에서 활용되는 편미분 기법은 이전에 학습하였기 때문에 자세한 내용은 언급하지 않고 그 개념만 정리하여 설명하였다. 파이썬 라이브러리인 넘파이 랜덤함수로 시뮬레이션 데이터를 생성하였고 이를 이용하여 다항식 회귀를 구현해보았다.

인공신경망이 딥러닝으로 발전해오게 된 내용을 간단히 정리하여 보여주었으며, 딥러닝이란 용어가 층의 개수가 많은 인공신경망이라는 사실을 설명하였고 한 개의 뉴런을 모델화한 퍼셉트론 모형을 자세히 설명하였다. 이를 통해 정보가 순방향으로 전달되는 과정을 행렬을 이용하여 수학적으로 표현해보았으며 이를 이용하여 최적의 모델을 찾아가는 방법인 최소 경사하강법을 설명하였다. 또, 퍼셉트론에서 해결할 수 없었던 문제 해결의 핵심이었던 다층인공신경망을 구성하는 데 가장 중요한 비선형함수인 활성화함수로 어떤 함수를 사용하는지 알아보았는데, 그중에서 가장 빈번하게 사용되는 활성화함수로서 렐루 계열의 함수와 시그모이드함수 계열의 함수를 그래프를 통해 비교해보았고 장단점도 간단히 알아보았다.

딥러닝의 학습이란 훈련 데이터를 통해 모델의 매개변수를 최적화하는 과정이며 학습된 모델을 검증하기 위해 검사 데이터를 별도로 준비하여 평가하고 오차를 최소화하는 방식으로 모델을 확정해간다. 즉, 매개변수를 결정하는 과정이 학습 과정이다. 딥러닝에서 회귀와 분류 문제를 설명하였는데, 이진분류란 특성을 뽑아내어 두 가지 중의 하나라고 결정하는 학습 방법이고 다중분류란 복수개의 특성을 규명하는 분류 방법이다. 최근까지 알려진 다양한 딥러닝 기법인 DNN, ANN, CNN, RNN 방법에 대해 간단히 설명하였다. 그리고 역전파 알고리즘의 의미와 간단한 절차에 대해서도 알아보았다.

끝으로 인공지능에서 수학의 역할에 대해 알아보았다. 인공지능과 수학의 관계를 설명하면서 인공지능이 절대로 신비한 것이 아니며 수학 지식의 결과물이라는 점을 강조하였다. 그리고 인공지능에서 필요하다고 여겨지는 수학 분야를 해석학, 대수학, 기하학 그리고 확률통계 분야로 나누어서 살펴보았다. 인공지능 교육에 있어서 우리나라 중고등학교에서 교과 과정을 어떻게 편성해야 하는지와 교육 방법에 대해 언급했다. 지금까지의 수학 교육을 개편하여 인공지능 지식을 갖춘 보편적 지식인을 배출하는 것도 중요하지만 인공지능 전문가를 양성하기 위한 수학 교육의 과정을 어떻게 편성해야 할지도 언급하였다. 특히 인공지능과 연관된 머신러닝 엔지니어, 데이터과학자, 로봇 과학자가 되려면 반드시 기초 수학 지식을 충분히 가지고 있어야 하며 고급수학을 이해하는 능력이 있어야 한다는 점을 강조하였다.

이 책을 통해서 우리가 언급하고자 하는 내용은 인공지능을 학습하기 위해 수학적 지식을 반드시 갖춰야 한다는 점을 강조하고 있으며, 어떠한 분야의 어떤 수학 지식이 필요한지 언급했고 대학생이 인공지능 학습을 하는 데 필요하다고 생각되는 기초 수학 내용 중 중요한 내용만 모아 설명하였다.

연습문제

01 인공지능과 연관하여 설명한 내용 중 적합하지 않은 것은?

❶ 인공지능이란 데이터를 분석하여 자료를 정리하고 그 속에서 의미를 찾아낸 뒤 스스로 결정하는 기술이다.

❷ 딥러닝에서는 DNN, CNN, RNN 등의 딥러닝 기법을 활용한다.

❸ 인공지능에서는 머신러닝을 활용하는데, 머신러닝은 딥러닝의 일부로 비지도 방식의 학습능력을 가진 네트워크이다.

❹ 사람이 지식을 학습하는 방식을 모방하여 기계가 학습하도록 한 것이 머신러닝의 핵심이다.

❺ 딥러닝은 여러 가지 비선형 변환기법을 조합하여 높은 수준의 추상화를 시도하는 기계학습 알고리즘의 집합이다.

02 머신러닝에 대한 설명으로 적합하지 않은 것은?

❶ 경험을 통해 자동으로 개선해나갈 수 있는 컴퓨터 알고리즘을 탐구하는 연구 영역이다.

❷ 훈련 데이터라고 알려진 샘플 데이터를 분석하여 자동으로 분석적 모델을 구축한다.

❸ 알고리즘을 통해 학습 데이터를 수집한 후 해당 데이터에 기반하여 더 정확한 모델을 구축한다.

❹ 인간이 특정 분야에 대해 가지고 있는 전문적인 지식을 정리하고 표현하여 컴퓨터에 기억시킴으로써, 일반인도 이 전문 지식을 이용할 수 있도록 하는 시스템이다.

❺ 학습을 마친 후 모델에 알고 싶은 입력 내용을 제공하면 결과물을 출력으로 내보낸다.

03 딥러닝에 대한 설명으로 적합하지 않은 것은?

❶ 초기의 딥러닝 발달 과정에서 두뇌의 인지 능력을 모방하도록 만든 인위적인 선형적 네트워크를 퍼셉트론이라고 한다.

❷ 퍼셉트론만 가지고는 *XOR* 논리 문제 같은 아주 간단한 문제조차 해결할 수 없다는 비판을 받았고 인공지능이 침체되는 계기가 되었다.

❸ 퍼셉트론으로 해결할 수 없었던 문제를 해결하기 위해 인공신경망의 네트워크층을 여러 층으로 늘려나가면서 해결책을 찾아나가서 현재의 딥러닝으로 발전했다.

❹ 머신러닝은 딥러닝이 진화한 것으로 기계가 사람의 도움 없이도 정확한 결정을 내릴 수 있도록 해주는 프로그래밍 가능한 신경망을 사용한다.

❺ 적절한 조건하에 이론적으로는 보편성을 유지하면서도 실제 응용이나 최적화된 구현이 가능한 유한 크기의 무제한 층수를 가진 인공신경망의 현대판 버전이 딥러닝이다.

04 딥러닝에서 사용하는 기법으로만 묶인 것은?

❶ DNN, DBN, RNN, CNN

❷ RNN, RSA, CNN, DNN

❸ AES, DBN, RNN, CNN

❹ ECC, DNN, DBN, RNN

❺ ARP, DBN, RNN, CNN

05 머신러닝에서 사용하는 학습 방법으로만 묶인 것은?

❶ 지도 학습, 비지도 학습, 강화 학습

❷ 지도 학습, 전문가 학습, 비전문가 학습

❸ 강화 학습, 회귀 학습, 군집 학습

❹ 강화 학습, 분류 학습, 실시간 학습

❺ 보상 학습, 전문가 학습, 군집 학습

06 머신러닝에서 사용하는 학습 방법 중에서 지도 학습을 분류한 것으로 적합한 것은?

❶ 강화와 군집

❷ 강화와 보상

❸ 회귀와 분류

❹ 회귀와 보상

❺ 분류와 군집

07 머신러닝에서 사용하는 학습 방법 중에서 비지도 학습을 분류한 것으로 적합한 것은?

❶ 강화와 실시간화

❷ 회귀와 강화

❸ 회귀와 분류화

❹ 군집화와 차원 축소

❺ 분류와 군집화

08 머신러닝의 지도 학습 중 회귀 문제의 예로 적합하지 않은 것은?

❶ 교육 정도에 따라 연봉을 예측하는 문제

❷ 영양 상태에 따른 신장의 크기를 예측하는 문제

❸ 보험 가입자에 대한 보험료 책정 문제

❹ 일반 메일과 스팸 메일을 구분하는 문제

❺ 날씨에 따르는 아이스크림 판매량 예측 문제

09 머신러닝의 지도 학습 중 분류 문제의 예로 적합한 것은?

❶ 교통 상황에 따른 출근 시간을 예측하는 문제

❷ 전기 자동차 생산량을 예측하는 문제

❸ 학생의 국어 성적과 영어 성적의 상관관계를 예측하는 문제

❹ 커뮤니티 웹 사이트에 올라오는 광고성 게시물 필터링 문제

❺ 인공지능 기술을 활용하는 기업의 생산성 증대 문제

10 한 개의 확률변수 또는 복수개의 확률변수들과 이들이 실현되는 스칼라값 간의 관계를 직선으로 모델링 하는 데 사용하는 회귀 분석으로만 묶인 것은?

❶ 선형회귀 분석, 다항회귀 분석

❷ 선형회귀 분석, 다중회귀 분석

❸ 다중회귀 분석, 다항회귀 분석

❹ 다항회귀 분석, 다변수회귀 분석

❺ 다변수회귀 분석, 다중회귀 분석

11 다중회귀 분석을 위해 가정해야 하는 조건으로 적합하지 않은 것은?

❶ 독립변수들과 종속변수들 사이에 선형관계가 있어야 한다.

❷ 독립변수들끼리의 상관관계가 너무 커서는 안 된다.

❸ 모집단으로부터 종속변수들을 샘플링 할 때 독립적이며 무작위로 수행해야만 한다.

❹ 오차를 나타내는 오차항들 간의 상관관계는 고려하지 않는다.

❺ 설명변수의 수가 2개 이상이 되어야 한다.

12 일반적으로 사용되는 기본적 인공신경망 알고리즘인 다층인공신경망을 구성하는 요소들로만 묶인 것은?

❶ 바이어스, 은닉층, 출력층

❷ 바이어스, 결합층, 출력층

❸ 입력층, 결합층, 은닉층

❹ 입력층, 훈련층, 결합층

❺ 입력층, 은닉층, 출력층

13 역전파 알고리즘 학습 방법의 문제점으로 지적되는 것은?

❶ 경사도 소실 문제

❷ *XOR* 문제

❸ 데이터 누락 문제

❹ 다층 퍼셉트론 문제

❺ 최소 제곱법 문제

14 인공신경망에서 편향 또는 바이어스라고 하는 입력값의 역할을 설명한 것으로 적합한 것은?

❶ 복수 개의 뉴런을 표현하는 데 필요한 매개변수이다.

❷ 입력변수들의 가중치를 나타내는 상수이다.

❸ 누락되는 데이터를 보상하기 위해 사용하는 값이다.

❹ 신경망의 측면에서 볼 때 신경세포에 가해지는 감도를 조정하는 값이다.

❺ 활성화함수에 영향을 주지 않도록 독립적인 값으로 설정된다.

15 인공신경망에서 활성화함수에 입력되는 값을 설명한 것으로 적합한 것은?

❶ 입력과 가중치의 합에 바이어스를 곱한 값

❷ 입력과 가중치를 곱한 값들을 더한 뒤 바이어스를 더한 값

❸ 입력값과 가중치를 곱한 값의 최소 제곱값

❹ 비중이 낮은 입력값을 제외하고 가중치가 높은 입력값만 더한 값

❺ 오차가 작은 입력값은 제외하고 가중치가 높은 입력값에 바이어스를 곱한 값

16 인공신경망에서 활성화함수로 사용하는 함수로 적합하지 않은 함수는?

❶ 시그모이드함수

❷ 렐루함수

❸ 하이퍼볼릭 탄젠트함수

❹ 소프트맥스함수

❺ 로그함수

17 딥러닝에서 취급하는 다층인공신경망에 대한 설명으로 적합하지 않은 것은?

❶ 단순 신경망을 여러 개의 층으로 배열하여 구성한다.

❷ 한 개의 신경망은 인접한 모든 층의 모든 노드와 연결되어 있다.

❸ 동일 층의 노드들도 상호 연결된다.

❹ 은닉층의 각 노드의 출력은 가중치와 바이어스가 반영된 뒤 다음 층 노드에 입력된다.

❺ 순방향 전파에서는 입력값에 따라 출력값을 계산하고 필요할 경우에는 역방향으로 데이터를 전달하여 출력과 올바른 값과의 오차가 최소가 되도록 한다.

18 다층신경망을 학습시키기 위해 필요한 데이터 종류를 열거한 것으로 적합한 것은?

❶ 빅데이터 집합, 오차 데이터 집합, 목표 데이터 집합

❷ 출력 데이터 집합, 입력 데이터 집합, 오차 데이터 집합

❸ 예비 데이터 집합, 훈련 데이터 집합, 목표 데이터 집합

❹ 훈련 데이터 집합, 검증 데이터 집합, 검사 데이터 집합

❺ 검증 데이터 집합, 검사 데이터 집합, 결과 데이터 집합

19 인공지능 과정 중에서 학습에 필요한 데이터에 대한 정답을 만드는 과정을 무엇이라고 하는가?

❶ 인덱싱(indexing)
❷ 레이블링(labeling)
❸ 필터링(filtering)
❹ 클리어링(clearing)
❺ 매칭(matching)

20 인공지능의 회귀 문제에서 사용하는 모든 데이터에서 순방향으로 전파를 실시하고 오차 합계를 이용해 매개변수를 갱신하는 학습을 무엇이라고 하는가?

❶ 지도 학습(supervised learnig)
❷ 비지도 학습(unsupervised learnig)
❸ 배치 학습(batch learning)
❹ 훈련 학습(traning learning)
❺ 경험 학습(experience learning)

21 인공지능의 지도 학습에서 분류 중 다중분류를 할 경우 활용하는 활성화함수와 오차함수를 올바르게 나타낸 것은?

❶ 하이퍼볼릭 탄젠트함수-크로스엔트로피
❷ 소프트맥스함수-크로스엔트로피
❸ 시그모이드함수-로지스틱함수
❹ 렐루함수-로지스틱함수
❺ 소프트맥스함수-로지스틱함수

Mathematics for Artificial Intelligence