딥러닝을 위한 수학 I

김영록

한국외대
교육대학원

- 이론편의 마지막 장은 확률과 통계입니다.
- 분류를 하기 위한 지도학습 모델 중에서 딥러닝과 관련이 깊은 것으로 로지스틱 회귀가 있습니다. 로지스틱 회귀 모델은 확률을 빼 놓고 설명하기 힘든데 어떤 입력 데이터가 어느 클래스에 속하는지 예측하려면 그 클래스에 속할 '확률값'을 알아야 하기 때문입니다.
- 한편 측정값에서 만들어진 확률 모델로부터 가장 높은 확률을 끌어내기 위해 최적의 매개변수를 찾는 것을 '최대가능도 추정'이라고합니다. '최대가능도 추정'은 로지스틱 회귀 모델의 학습 과정에서 근간이 되는 개념입니다.
- 이번 장에서는 확률과 통계의 수많은 개념 중에서도 머신러닝과 딥러닝 모델에 관련 있는 것만 중점적으로 살펴보겠습니다.

- '확률'은 어떤 사건의 잠재적 가능성을 백분율로 표시한 것입니다.
- 확률을 표기할 때는 '$P(X)$'와 같이 쓰는데 이때 주의할 점이 있습니다. 일반적인 함수에서는 서로 다른 함수를 표기할 때 $f(x), g(x)$와 같이 앞의 글자로 구분하는 반면 확률에서는 서로 다른 확률을 쓸 때 $P(X), P(Y)$와 같이 뒤의 글자로 구분합니다. 그리고 이때의 X와 Y를 '확률변수'라고 합니다.
- 예를 하나 들어 봅시다.
  - X: 동전을 한 번 던졌을 때 나오는 동전의 면
  - Y: 주사위를 한 번 던졌을 때 나오는 주사위의 숫자
- X와 Y가 위와 같을 때 각 경우의 수는 다음과 같다는 것을 알 수 있습니다.
  - $X = \{\text{앞면}, \text{뒷면}\}$의 2개의 값
  - $Y = \{1, 2, 3, 4, 5, 6\}$의 6개의 값
- 이때 확률변수를 이용해 다음과 같이 확률을 표현할 수 있습니다.

$$P(X = \text{앞면}) = 1/2, \;\; P(Y = 2) = 1/6$$

- 확률의 표기법과 일반적인 함수의 표기법을 비교해 보면 표 6-1과 같은 차이가 있다는 것을 알 수 있습니다.

- 표 6-1 확률의 표기법

	전체를 표현하는 경우	특정한 값을 표현하는 경우
함수	$f(x), g(x)$	$f(2), g(-3)$
확률	$P(X), P(Y)$	$P(X = \text{앞면}),\ P(Y = 2)$

확률변수가 가질 수 있는 값과 그에 대한 확률을 표 형태로 정리한 것을 '확률분포(確率분포, probability distribution)'라고 합니다. 위의 예에서 확률변수 X와 Y 각각에 대한 확률분포를 정리하면 다음과 같습니다.

- 표 6-2 X의 확률분포

확률변수 X	앞면	뒷면
$P(X)$	$1/2$	$1/2$

- 표 6-3 Y의 확률분포

확률변수 Y	1	2	3	4	5	6
$P(Y)$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$

- 이러한 내용을 좀 더 확장하면 더 복잡한 확률변수도 생각해볼 수 있습니다. 동전의 예를 확장한다면 확률변수 X_n을 '동전을 n번 던졌을 때 앞면이 나오는 횟수'라고 정의할 수 있습니다.
- 이처럼 '결과가 성공(1)이나 실패(0)로 나오는 독립시행을 n번 했을 때 성공(1)이 나오는 횟수'를 확률변수라고 할 때 이에 대한 확률분포를 '이항분포(binomial distribution)'라고 합니다.

$n=1$인 경우		
확률변수 X_1	0	1
$P(X_1)$	$1/2$	$1/2$
동전	H	T

$n=2$인 경우			
확률변수 X_2	0	1	2
$P(X_2)$	$1/4$	$2/4$	$1/4$
동전	HH	HT, TH	TT

$n=3$인 경우				
확률변수 X_3	0	1	2	3
$P(X_3)$	$1/8$	$3/8$	$3/8$	$1/8$
동전	HHH	HHT, HTH, THH	HTT, THT, TTH	TTT

$n=4$인 경우					
확률변수 X_4	0	1	2	3	4
$P(X_4)$	$1/16$	$4/16$	$6/16$	$4/16$	$1/16$
동전	HHHH	HHHT, HHTH, HTHH, THHH	HHTT, ... , TTHH	HTTT, ... , TTTH	TTTT

- 확률분포의 표는 막대 그래프로도 표현할 수 있습니다. 이 그래프를 '히스토그램(histogram)'이라 합니다.
- 위의 예에서 $n=2,3,4$일 때의 확률분포를 히스토그램으로 그리면 다음과 같습니다.

```
N=2
M=2**N
X=range(0,3)
plt.bar(X, [scm.comb(N,i)/M for i in X])
plt.show()
```

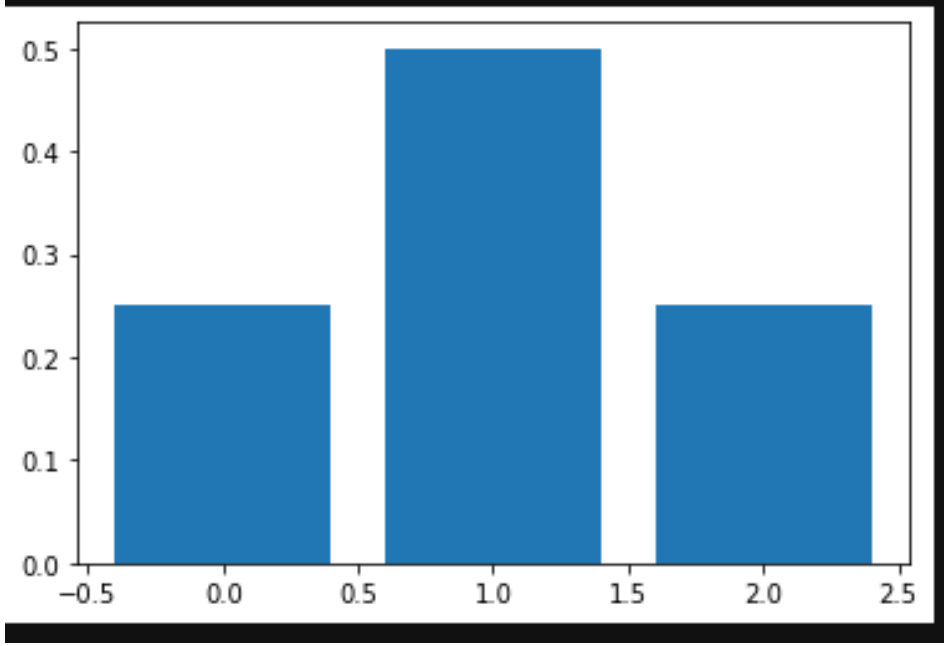

그림 6-1 히스토그램($n = 2$)

```
N=3
M=2**N
X=range(0,4)
plt.bar(X, [scm.comb(N,i)/M for i in X])
plt.show()
```

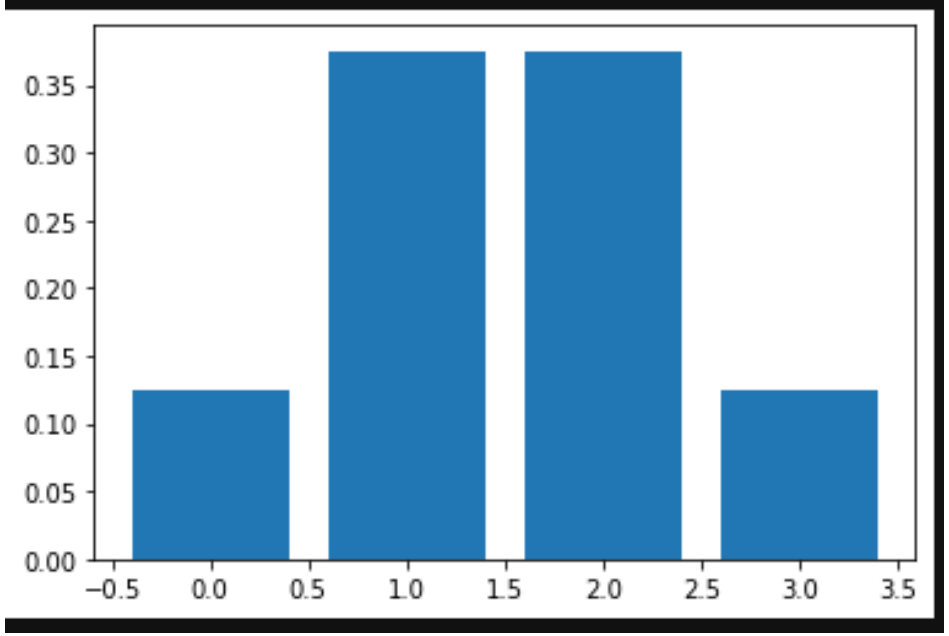

그림 6-2 히스토그램($n = 3$)

```
N=4
M=2**N
X=range(0,5)
plt.bar(X, [scm.comb(N,i)/M for i in X])
plt.show()
```

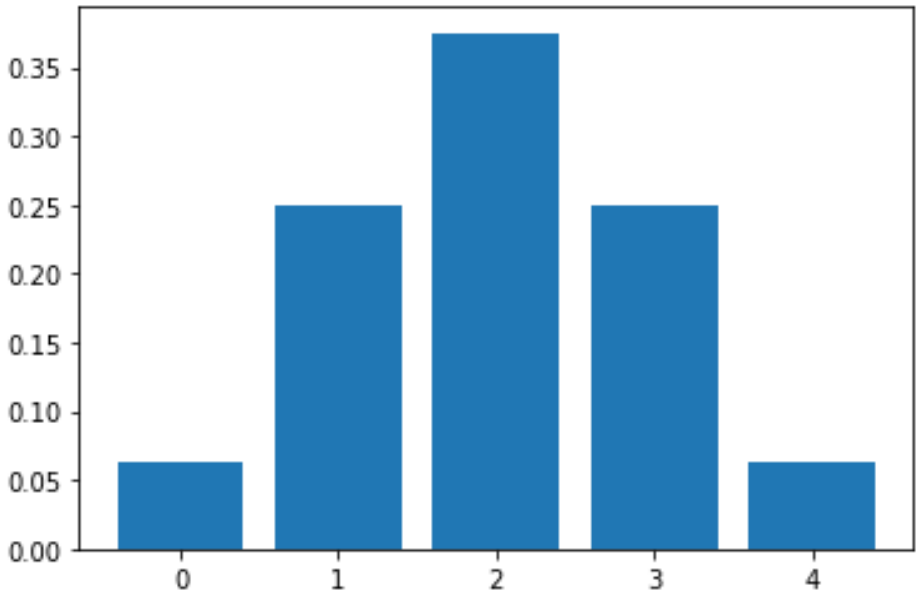

그림 6-3 히스토그램($n = 4$)

- 이 히스토그램에서 n의 수가 더 커지면 어떻게 될 지 그려보면 다음과 같습니다. $n = 10, 100, 1000$일 때의 그림을 파이썬으로 그려보면 다음과 같습니다.

```
N=11
M=2**N
X=range(0,11)
plt.bar(X, [scm.comb(N,i)/M for i in X])
plt.show()
```

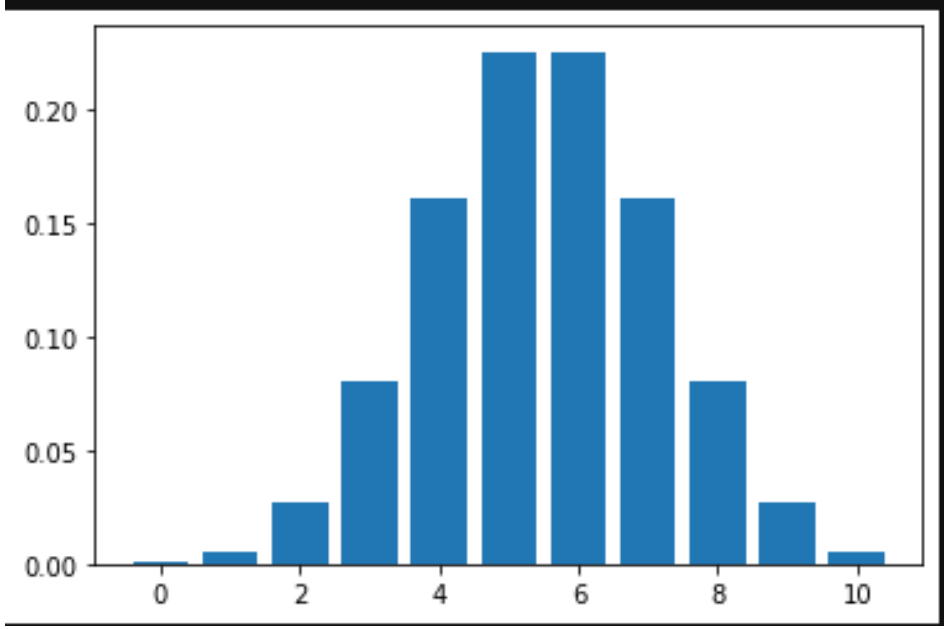

그림 6-4 히스토그램($n = 10$)

```
N=101
M=2**N
X=range(30,71)
plt.bar(X, [scm.comb(N,i)/M for i in X])
plt.show()
```

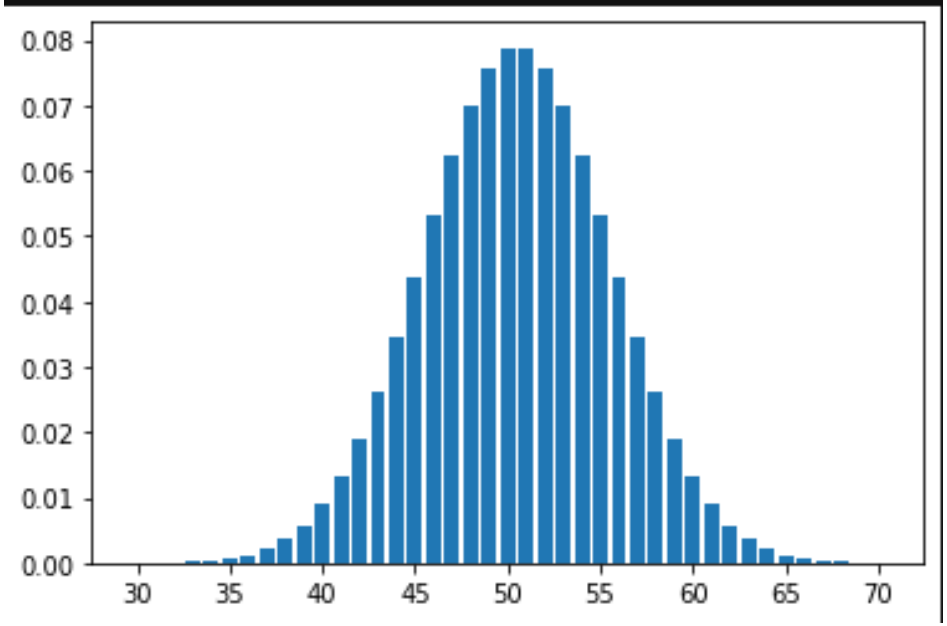

그림 6-5 히스토그램($n = 100$)

```
N=1001
M=2**N
X=range(440,561)
plt.bar(X, [scm.comb(N,i)/M for i in X])
plt.show()
```

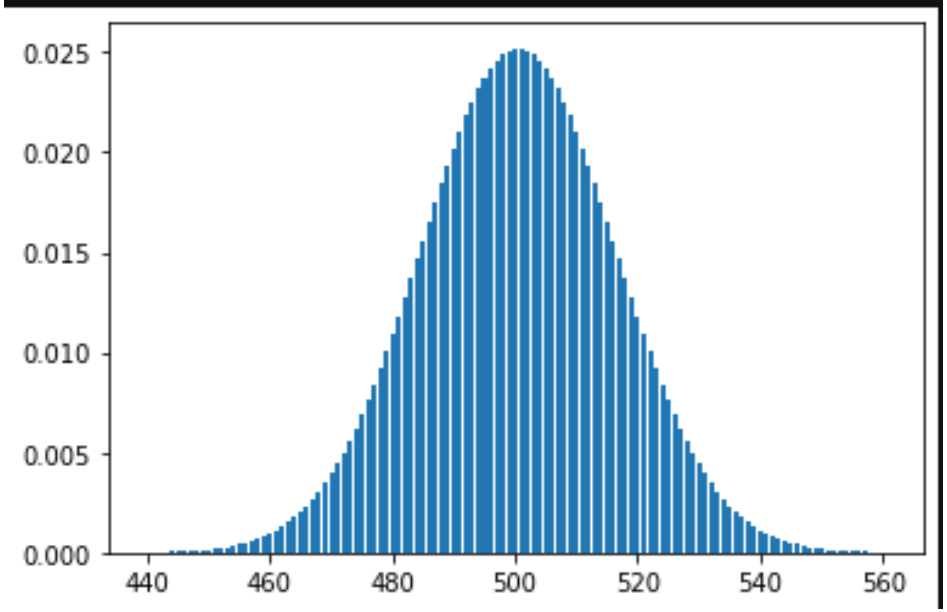

그림 6-6 히스토그램($n = 1000$)

- 앞 절의 그림 6-5와 그림 6-6을 보면 짐작할 수 있듯이 이항분포의 히스토그램은 n값이 커질수록 연속함수(continuous function)의 모양이 됩니다.
- 이 함수는 '정규분포(normal distribution)함수'라고 하며 다음과 같은 식으로 표현됩니다. 이때 $P(X_1 = 1) = p$라고 할 때 $\mu = np$이고 $\sigma^2 = np(1-p)$입니다.

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

- 또한 이항분포함수가 정규분포함수에 가까워지는 것을 '중심극한정리(central limit theorem)'라고 합니다.
- 앞의 예에서 동전을 한 번 던졌을 때 결과가 0(앞면)이 나올 확률이 $p = 1/2$이었으므로 $\mu = np = n/2$이고 $\sigma^2 = np(1-p) = n/4$이 됩니다. $n/2 = m$이라고 놓으면 근사식을 다음과 같이 쓸 수 있습니다.

$$P(X_n = x) \approx \frac{1}{\sqrt{m\pi}} \exp\left(-\frac{(x-m)^2}{m}\right)$$

- 실제로 이렇게 나오는지 파이썬으로 그래프를 그려 봅시다. 다음은 그림 6-6의 이항분포 그래프와 정규분포 그래프를 함께 그리는 코드입니다.

```python
import numpy as np
import scipy.special as scm
import matplotlib.pyplot as plt

#정규분포함수의 정의
def gauss(x,n):
    m=n/2
    return np.exp(-(x-m)**2 / m) / np.sqrt(m * np.pi)

#이항분포 그래프와 정규분포 그래프를 함께 그리기
N=1000
M=2**N
X=range(400,561)
plt.bar(X, [scm.comb(N,i)/M for i in X])
plt.plot(X, gauss(np.array(X), N), c='k', linewidth=2)
plt.show()
```

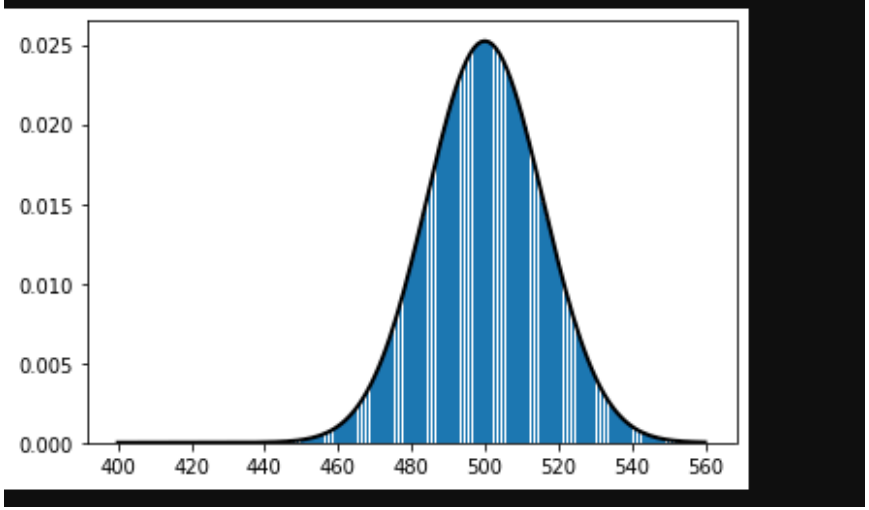

그림 6-7 이항분포 그래프와 정규분포 그래프를 그리는 코드
그림 6-8 이항분포 그래프와 정규분포함수 그래프를 함께 그린 모습

- 두 그래프가 정확하게 일치하는 것으로 보아 중심극한정리가 맞다는 것을 알 수 있습니다.
- 한편 정규분포함수처럼 확률변수가 연속적인 값을 가질 때 확률분포함수는 연속함수입니다. 이때의 확률분포함수를 '확률밀도함수(確率密度函數, probability density function)'라고 합니다.
- 말이 나온 김에 확률밀도함수에서 확률을 구해 봅시다. 6.1절의 이항분포에서 $n = 1000$인 그래프(그림 6-6)를 예로 들어 이 그래프가 정규분포를 따를 때의 확률값을 알아봅시다.

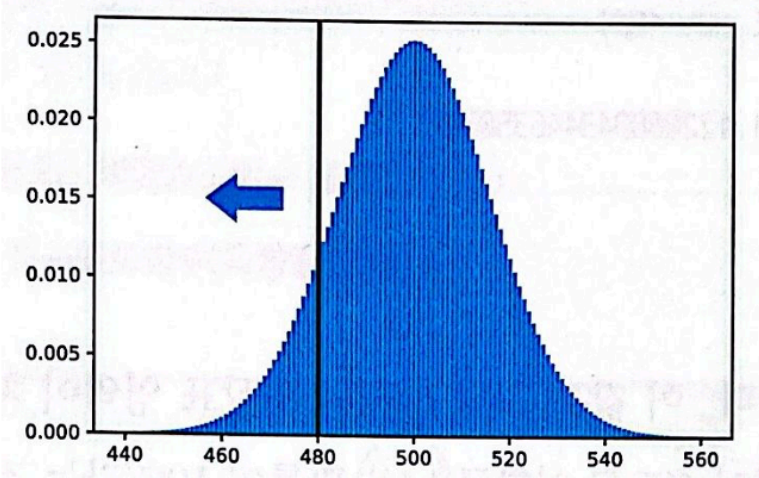

그림 6-9 n=1000일 때의 이항분포 히스토그램

- 이 그래프는 아주 작은 사각형으로 구성돼 있는데 모든 사각형의 면적을 합하면 1이 됩니다. 참고로 확률에서는 독립적인 사건의 확률들을 모두 더했을 때 1이 나옵니다. 감이 잘 오지 않는다면 앞 절의 그림 6-1, 6-2, 6-3의 히스토그램을 다시 살펴보기 바랍니다. 세 경우 모두 n의 크기가 제각각이지만 사각형의 면적을 모두 더했을 때 1이 되는 것은 $n = 1000$일 때도 마찬가지입니다.
- 확률을 알고 싶은 사건이 다음과 같은 조건이라 하겠습니다.

$$P(X_{1000} \leq 480)$$

- 이 말은 '동전을 1000번 던졌을 때 앞면이 480번 이하로 나올 확률'이라는 의미입니다.
- 이 확률은 그림 6-9에서 화살표로 표시된 영역의 면적과 같습니다. 연속함수의 면적은 곧 적분을 의미합니다(2.9절 참고). 그래서 다음과 같이 적분한 식을 사용하면 근사적인 확률값을 얻을 수 있습니다.

$$P(X_{1000} \leq 480) \approx \int_0^{480} f(x)dx$$

- 앞서 소개한 정규분포함수식에 $m = 1000/2 = 500$을 대입하면 다음과 같이 식을 쓸 수 있습니다

$$f(x) = P(X_n = x) \approx \frac{1}{\sqrt{500\pi}} \exp\left(-\frac{(x-500)^2}{500}\right)$$

- 이 식 $f(x)$의 적분을 파이썬으로 계산하면 다음과 같습니다.

```
import numpy as np
from scipy import integrate
def normal(x):
    return np.exp(-((x-500)**2)/500)/np.sqrt(500*np.pi)
integrate.quad(normal, 0, 480)
```

```
(0.10295160536603419, 1.1220689434463503e-13)
```

그림 6-10 적분한 결괏값

- 계산 결과로 약 0.1이 나왔습니다. 이 값은 그림 6-9의 화살표 영역이 차지하는 면적입니다. 결과적으로 '동전을 1000번 던질 때 앞면이 480회 이하로 나올 확률이 10%라는 것을 알 수 있습니다.
- 이 내용에서 알 수 있는 것은 어떤 사건의 누적된 확률을 계산하고 싶을 때는 확률밀도함수를 적분하면 된다는 것입니다. 이때 확률밀도함수를 적분해서 구한 함수를 '누적분포함수(cumulative distribution function)'라고 합니다.

정규분포 함수와 시그모이드 함수

- 지금까지 설명한 내용으로 눈치챘을지도 모르겠지만 어떤 실숫값에서 확률값을 구할 때 정규분포함수를 쓰는 것은 자연스러운 접근법입니다. 다만 머신러닝 모델에서는 확률값을 구할 때 일반적인 정규분포함수 대신 시그모이드 함수를 사용합니다.
- 가장 큰 이유로는 확률밀도함수가 정규분포일 때 적분 결과 (확률분포함수)를 함수식으로 표현하지 못하는 경우가 있기 때문입니다.
- 반대로 확률분포함수가 다음과 같은 시그모이드 함수일 때

$$f(x) = \frac{1}{1 + \exp(-x)}$$

- 이 식의 미분(확률밀도함수)은 원래의 함수식만으로도 구할 수 있습니다.

$$f'(x) = f(x)(1 - f(x))$$

- 이를 이번 장에서 설명한 확률 용어로 다시 풀어 써보면
  - 확률밀도함수: $f(x)(1-f(x))$
  - 누적분포함수: $f(x)$
- 가 되어 두 함수 모두 계산하기 쉽다는 것을 알 수 있습니다.
- 또한 시그모이드 함수와 정규분포함수는 그래프 모양도 상당히 비슷합니다.
- 실제로 그림 6-11은 다음 함수를 함께 그린 것입니다.
  - sig: 시그모이드 함수에서 계산한 확률밀도함수 $f(x)(1-f(x))$
  - std: 평균이 0, 분산이 1.6인 정규분포함수(확률밀도함수)

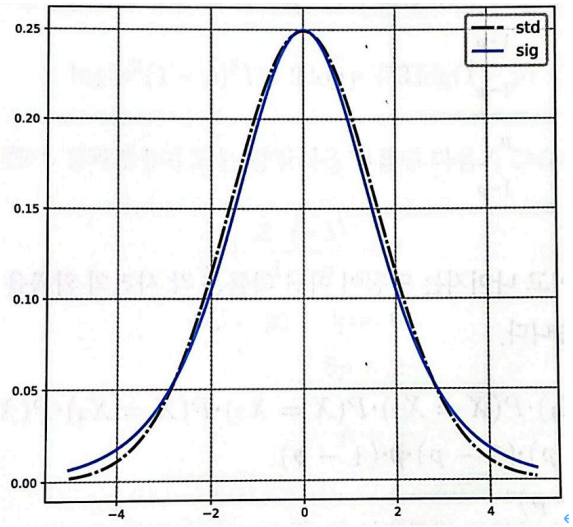

그림 6-11 시그모이드 함수(sig)와 정규분포함수(std)

- 이처럼 계산의 용이성과 정규분포함수와의 유사성 덕분에 머신러닝에서는 시그모이드 함수를 사용하는 것입니다.

- 다음과 같은 문제가 있다고 합시다.
- 제비뽑기 기계에서 당첨이 나을 확률은 항상 일정하다고 가정합니다. 이 기계에서 뽑기를 여러 번 할 때 각 시행은 이전 시행과는 관계없는 독립적인 시행입니다. 다섯 번 제비를 뽑아 보니 첫 번째와 네 번째에 당첨이 나오고 나머지 세 번은 당첨되지 않았습니다. 제비를 한 번 뽑았을 때 당첨될 확률을 p라고 할 때 가장 가능성이 높은 확률 p를 구하시오.
- 확률변수 X_i을 다음과 같이 정의합니다.

$$X_i = \begin{cases} 1 & (\text{당첨인 경우}) \\ 0 & (\text{당첨이 아닌 경우}) \end{cases}$$

- 당첨될 확률이 p일 때 당첨되지 않을 확률은 $(1-p)$이므로 위의 다섯 번의 시행 결과를 표로 정리하면 다음과 같습니다.

i	X_i	$P(X = X_i)$
1	1	p
2	0	$1-p$
3	0	$1-p$
4	1	p
5	0	$1-p$

- 첫 번째와 네 번째가 당첨이고 나머지는 당첨이 아닌 확률은 각 사건의 확률을 곱한 것과 같으며 다음과 같은 식으로 표현할 수 있습니다.

$$\begin{aligned} & P(X = X_1) \cdot P(X = X_2) \cdot P(X = X_3) \cdot P(X = X_4) \cdot P(X = X_5) \\ &= p \cdot (1-p) \cdot (1-p) \cdot p \cdot (1-p) \\ &= p^2 \cdot (1-p)^3 \end{aligned} \tag{1}$$

- 이렇게 구한 식은 제비를 한 번 뽑았을 때 당첨될 확률 p를 모르는 상태이므로 함수로 볼 수 있습니다.
- 이처럼 모델의 확률적 특징을 나타내는 변수를 포함한 식을 '가능도함수 (likelihood function)'라고 합니다.

- 그리고 가능도함수를 매개변수로 미분했을 때 그 값이 0이 되게 하는 매개변숫값을 구한 다음, 그 값을 가장 확률이 높은 매개변수로 추정하는, 알고리즘을 '최대가능도 추정(maximum likelihood estimation)'이라고 합니다.
- 최대가능도 추정을 사용할 때는 원래의 식 전체에 로그를 적용합니다. 왜냐하면 원래의 수식 (1)에는 곱셈이 많아 미분을 할 때 계산이 복잡해지기 때문입니다. 반면 로그를 적용하면 곱셈 대신 덧셈을 하면 되므로 계산이 한결 쉬워집니다.
- 로그를 쓰는 다른 이유는 너무 크거나 작은 수를 다루기가 용이하기 때문입니다. 예를 들어, 1만 건 정도의 대량 데이터로 확률을 곱하다 보면 결괏값이 너무 작아지기 때문에 계산하기 곤란한 상황(underflow)이 발생할 수 있습니다.
- 여기서는 로그함수가 단조증가함수이기 때문에 원래의 함수에서 최댓값이 나오는 매개변수와 로그를 적용한 후의 함수에서 최댓값이 나오는 매개변수가 똑같다고 전제합니다.

- 실제로 수식 (1)에 최대가능도 추정을 해 봅시다. 먼저 수식 (1)에 로그를 적용합니다.

$$\log(p^2(1-p)^3) = 2\log p + 3\log(1-p) \tag{2}$$

- 수식 (2)를 p로 미분해서 결과가 0이 되는 방정식을 만들면 다음과 같습니다.

$$\begin{aligned} \frac{2}{p} + \frac{3\cdot(-1)}{1-p} &= 0 \\ 2(1-p) - 3p &= 0 \\ 5p &= 2 \\ p &= \frac{2}{5} \end{aligned}$$

- 확인 차원에서 수식 (2)를 p에 대한 함수라고 할 때 그래프의 모양을 살펴봅시다. 확실히 가능도함수는 $p = 0.4$에서 최댓값인 것을 알 수 있습니다.

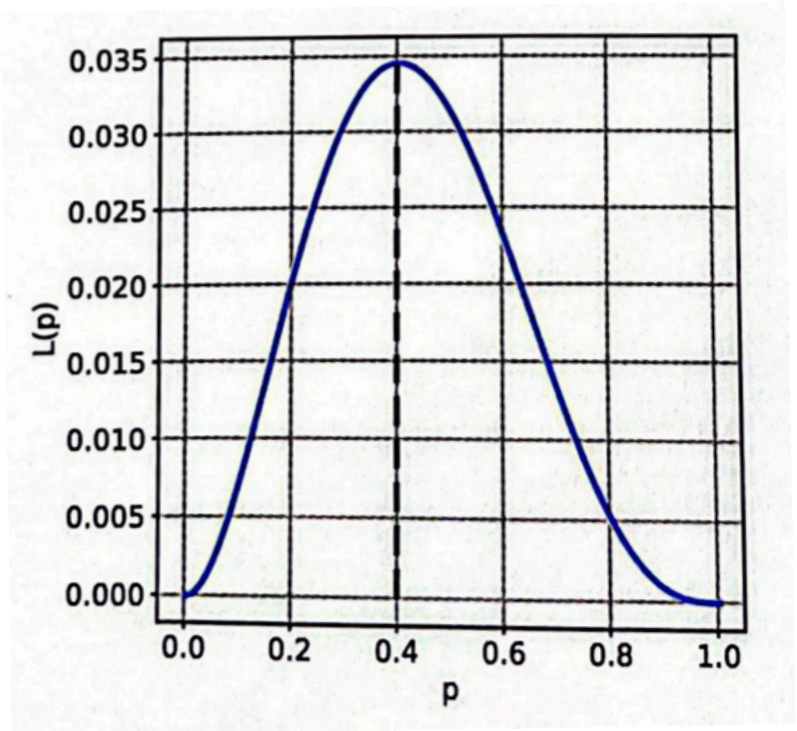

그림 6-12 변수가 p인 가능도함수의 그래프

- 최대가능도 추정의 결과는 '다섯 번의 시도에서 두 번이 성공했기 때문에 $2/5$의 확률'이라고 하는 상식적인 수준의 예측과 똑같이 나왔습니다. 지금까지 설명한 추정 방법을 간단히 요약하면 다음과 같습니다.

1. 측정값(X_i)과 매개변수(p)를 포함한 식을 만든다.
2. 확률식에 측정값(X_i의 실제 값)을 대입해서 매개변수만 있는 식으로 만든다.
3. 2의 식을 매개변수의 함수로 보고 로그를 적용한 다음, 매개변수로 미분한 결과가 0이 됐을 때의 매개변숫값을 구한다.

- 이 예는 아주 간단한 것이었지만 8장에 나오는 로지스틱 회귀는 더 복잡한 형태의 최대가능도 추정을 하게 됩니다. 다만 추정 방법 자체는 이 예와 같은 방법을 쓰기 때문에 이번 절을 통해 어떤 흐름으로 계산하면 되는지 파악해 두기 바랍니다.

가능도함수의 극값은 왜 최솟값이 아닌 최댓값을 가지는가?

- 2.4절에서 설명한 것처럼 함수의 미분값이 0이 되는 지점은 극댓값을 가지거나 극솟값을 가집니다.[1]
- 최대가능도 추정에서는 가능도함수를 미분했을 때 0이 나오는 값을 구하는데 왜 이 값은 극솟값이 아니라 극댓값을 가지는 것일까요?
- 가능도함수는 확률값의 곱셈으로 만들어지는데 각 확률값이 정답값에서 멀면 멀수록 함숫값이 0에 가까워지는 특징이 있습니다. 예를 들어, 입력변수가 2개인 가능도함수가 있다고 할 때 대부분의 점은 0에 가깝지만 정답값 주변에서만 그래프가 산처럼 솟아오르는 모양이 됩니다. 이런 모양을 상상해 보면 가능도함수를 미분했을 때 0인 지점은 곧 산의 정상 부분이 되며 결국 그지점이 극댓점이 된다는 것을 짐작할 수 있습니다.

[1] 엄밀하게는 극댓값이나 극솟값이 아닌 경우도 있습니다. (예: 경계값)

- https://github.com/sw-song/py_finance_practice?fbclid=IwAR1e_8X287nsN5H3glE-JC8_SEqpjf8aNKDUoqbhzU58AML2jSpFj3vlzxs
- Download code file.