Data Mining

나이브 베이즈 (Naïve Bayes)

학습 목표

- 나이브 베이즈 알고리즘과 구현 코드를 살펴본다.

주요 내용

1. 베이즈 정리
2. 나이브 베이즈
3. 나이브 베이즈 구현
4. 스팸어쌔신 데이터셋 적용

베이즈정리 오덜봐

어떤 클래스인지 알려주고
확률하기 계산

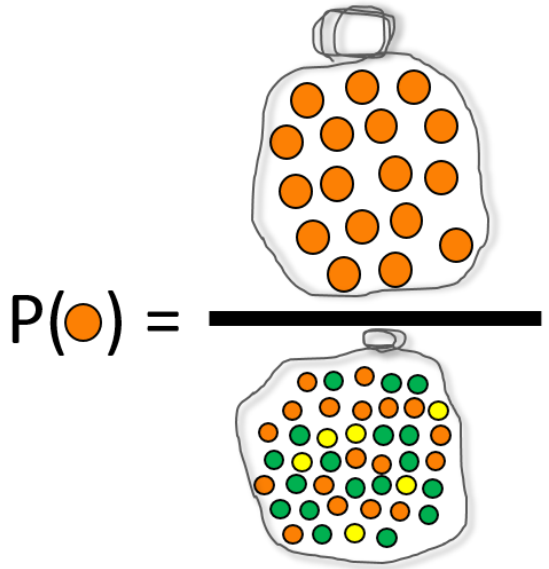
1. 베이즈 정리

- 6장. 확률 (Probability)

확률 (probability)

확률이란?

- 어떤 사건이 실제로 일어날 것인지 혹은 일어났는지에 대한 지식 혹은 믿음을 표현하는 방법
- 같은 원인에서 특정한 결과가 나타나는 비율



$$\mathrm{p}(\text{특정 사건}) = \frac{\text{특정 사건}}{\text{전체 사건}}$$

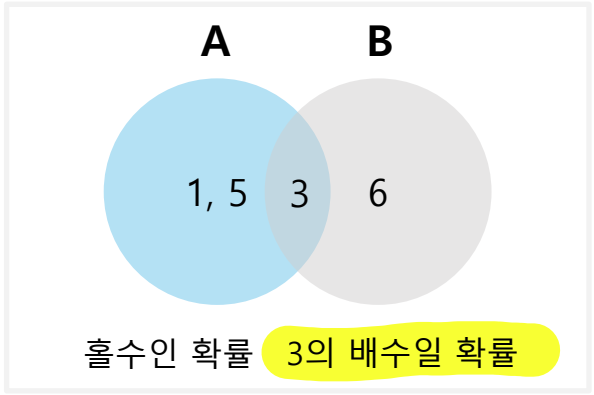
독립 사건 vs. 종속 사건 (Independent vs. dependent)

독립 사건 (Independent event)

- 한 사건의 결과가 다른 사건에 영향을 주지 않는 경우

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

따라서, $P(A, B) = P(A)P(B)$



주사위를 던졌을 때 홀수인 확률 3의 배수일 확률

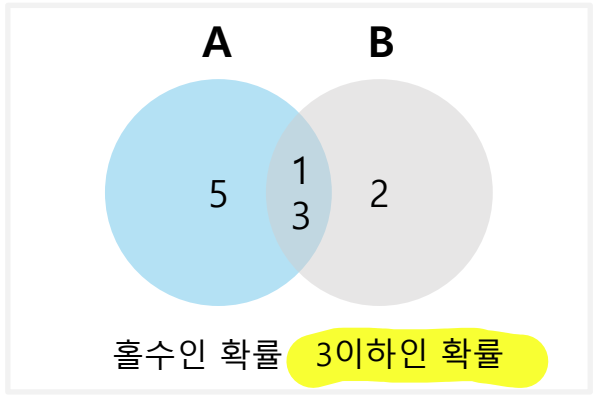
$P(A) = \frac{1}{2}$ 인데 $P(A|B) = \frac{1}{2}$ 이므로 독립

종속 사건 (Dependent event)

- 한 사건의 결과가 다른 사건에 영향을 주는 경우

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

따라서, $P(A, B) = P(A|B)P(B)$



주사위를 던졌을 때 홀수인 확률 3이하인 확률

$P(A) = \frac{1}{2}$ 인데 $P(A|B) = \frac{2}{3}$ 이므로 종속

독립 사건 vs. 종속 사건 (Independent vs. dependent)

가정사항

- **동전의 앞면이 나올 확률(H)과 뒷면이 나올 확률(T)은 같다.**
- **두 개의 동전이 있고 첫번째 동전을 던졌을 때 앞면이 나왔다.**

첫번째 동전

문제1 **두번째 동전이 앞면이 나올 확률은?**

독립 사건 vs. 종속 사건 (Independent vs. dependent)

가정사항

- **동전의 앞면이 나올 확률(H)과 뒷면이 나올 확률(T)은 같다.**
- **두 개의 동전이 있고 첫번째 동전을 던졌을 때 앞면이 나왔다.**

첫번째 동전

문제1 **두번째 동전이 앞면이 나올 확률은?** 50%

$$P(\text{동전2} = \text{H} \mid \text{동전1} = \text{H})$$

조건부확률

$$= P(\text{동전2} = \text{H}) = \frac{1}{2}$$

- 첫번째 동전의 결과로 두번째 동전을 던졌을 때 결과를 알 수 없기 때문에 두 사건은 독립 사건

독립 사건 vs. 종속 사건 (Independent vs. dependent)

가정사항

- **동전의 앞면이 나올 확률(H)과 뒷면이 나올 확률(T)은 같다.**
- **두 개의 동전이 있고 첫번째 동전을 던졌을 때 앞면이 나왔다.**

첫번째 동전

문제2 **두 동전이 모두 뒷면이 나올 확률은?**

독립 사건 vs. 종속 사건 (Independent vs. dependent)

가정사항

- **동전의 앞면이 나올 확률(H)과 뒷면이 나올 확률(T)은 같다.**
- **두 개의 동전이 있고 첫번째 동전을 던졌을 때 앞면이 나왔다.**

첫번째 동전

문제2 **두 동전이 모두 뒷면이 나올 확률은?** 0%

$$P(\text{동전}1 = \text{T},\ \text{동전}2 = \text{T} \mid \text{동전}1 = \text{H})$$

$$= \frac{P(\text{동전}1 = \text{T},\ \text{동전}2 = \text{T},\ \text{동전}1 = \text{H})}{\text{동전}1 = \text{H}}$$

$$= 0$$

동전1 = T 과 동전1 = H 는 배반 사건

결합 확률분포

- 첫번째 동전의 결과가 두 동전이 모두 뒷면이 나올 확률에 영향을 미치므로 종속 사건

조건부 확률 (conditional probability)

사건 B가 발생했을 때 A가 발생할 확률

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

조건부 확률 (conditional probability)

가정사항

- **아이가 딸이거나 아들일 확률은 같다.**
- **둘째의 성별은 첫째의 성별과 독립이다.**

첫째	둘째	확률
딸	딸	¼
	아들	¼
아들	딸	¼
	아들	¼

사건 **G** : 첫째가 딸인 경우 $\mathrm{P(G)} = \frac{1}{2}$

사건 **B** : 두 아이가 모두 딸인 경우 $\mathrm{P(B)} = \frac{1}{4}$

사건 **L** : 최소 한명이 딸인 경우 $\mathrm{P(L)} = \frac{3}{4}$

문제1 **첫째가 딸인 경우 두 아이가 모두 딸일 확률은?**

조건부 확률 (conditional probability)

가정사항

- **아이가 딸이거나 아들일 확률은 같다.**
- **둘째의 성별은 첫째의 성별과 독립이다.**

첫째	둘째	확률
딸	딸	¼
	아들	¼
아들	딸	¼
	아들	¼

사건 **G** : 첫째가 딸인 경우 $P(G) = \frac{1}{2}$

사건 **B** : 두 아이가 모두 딸인 경우 $P(B) = \frac{1}{4}$

사건 **L** : 최소 한명이 딸인 경우 $P(L) = \frac{3}{4}$

문제1 **첫째가 딸인 경우 두 아이가 모두 딸일 확률은?** 50%

$$P(B|G) = \frac{P(B,G)}{P(G)} = \frac{P(B)}{P(G)} = \frac{\frac{1}{4}}{\frac{1}{2}} = \frac{1}{2} \quad (P(B,G)\text{는 } P(B) \text{ 이므로})$$

given 으로 표현함.

조건부 확률 (conditional probability)

가정사항

- **아이가 딸이거나 아들일 확률은 같다.**
- **둘째의 성별은 첫째의 성별과 독립이다.**

첫째	둘째	확률
딸	딸	¼
	아들	¼
아들	딸	¼
	아들	¼

사건 **G** : 첫째가 딸인 경우 $P(G) = \frac{1}{2}$

사건 **B** : 두 아이가 모두 딸인 경우 $P(B) = \frac{1}{4}$

사건 **L** : 최소 한명이 딸인 경우 $P(L) = \frac{3}{4}$

문제2 **자녀 중 최소 한명이 딸인 경우 두 딸이 모두 딸일 확률은?**

조건부 확률 (conditional probability)

가정사항

- **아이가 딸이거나 아들일 확률은 같다.**
- **둘째의 성별은 첫째의 성별과 독립이다.**

첫째	둘째	확률
딸	딸	¼
	아들	¼
아들	딸	¼
	아들	¼

사건 **G** : 첫째가 딸인 경우 $\mathrm{P(G)} = \frac{1}{2}$

사건 **B** : 두 아이가 모두 딸인 경우 $\mathrm{P(B)} = \frac{1}{4}$

사건 **L** : 최소 한명이 딸인 경우 $\mathrm{P(L)} = \frac{3}{4}$

문제2 **자녀 중 최소 한명이 딸인 경우 두 딸이 모두 딸일 확률은?** 1/3

$$\mathrm{P(B|L)} = \frac{\mathrm{P(B,L)}}{\mathrm{P(L)}} = \frac{\mathrm{P(B)}}{\mathrm{P(L)}} = \frac{\frac{1}{4}}{\frac{3}{4}} = \frac{1}{3} \qquad (\mathrm{P(B,L)}\text{는 } \mathrm{P(B)} \text{ 이므로})$$

조건부 확률 (conditional probability)

무작위 실행을 해서 확률을 검증해보자!

아들과 딸 정의

```
import enum, random

# An Enum is a typed set of enumerated values. We can use them
# to make our code more descriptive and readable.
class Kid(enum.Enum):
    BOY = 0
    GIRL = 1
```

- Enum : 열거형, 명명된 값의 집합을 정의하여 허용 가능한 값을 기술

아들과 딸일 확률을 균등 분포로 샘플링

```
def random_kid() -> Kid:
    return random.choice([Kid.BOY, Kid.GIRL])
```

조건부 확률 (conditional probability)

세 종류의 사건 정의

```
both_girls = 0
older_girl = 0
either_girl = 0
```

`older_girl` 사건 **G** : 첫째가 딸인 경우 $P(G) = \frac{1}{2}$

`both_girls` 사건 **B** : 두 아이가 모두 딸인 경우 $P(B) = \frac{1}{4}$

`either_girl` 사건 **L** : 최소 한명이 딸인 경우 $P(L) = \frac{3}{4}$

조건부 확률 (conditional probability)

만 번 샘플링을 해서 각 사건에 대해 개수를 셈

```
random.seed(0)

for _ in range(10000):
    younger = random_kid()
    older = random_kid()
    if older == Kid.GIRL:
        older_girl += 1
    if older == Kid.GIRL and younger == Kid.GIRL:
        both_girls += 1
    if older == Kid.GIRL or younger == Kid.GIRL:
        either_girl += 1
```

확률 추정

```
print("P(both | older):", both_girls / older_girl)     # 0.514 ~ 1/2
print("P(both | either): ", both_girls / either_girl)  # 0.342 ~ 1/3
```

첫째가 딸인 경우, 두 아이가 모두 딸일 확률

$$P(B|G) = \frac{1}{2}$$

딸이 최소 한명인 경우, 두 딸이 모두 딸일 확률

$$P(B|L) = \frac{1}{3}$$

베이즈 확률론

"동전의 앞면이 나올 확률이 50%이다."

빈도주의 확률론 (frequentism)

"100번 동전을 던졌을 때 50번은 앞면이 나온다"

베이즈 확률론 (bayesianism)

"동전의 앞면이 나왔다는 주장의 신뢰도가 50%이다."

사건의 믿음의 정도. 신뢰도를 확률에 빗대어 보는 관점.

베이즈 확률론

빈도주의 확률론 (frequentism)

- 확률은 "**전체 경우의 수 대비 특정 사건의 경우의 수** " 를 나타낸다고 보는 관점

베이즈 확률론 (bayesianism)

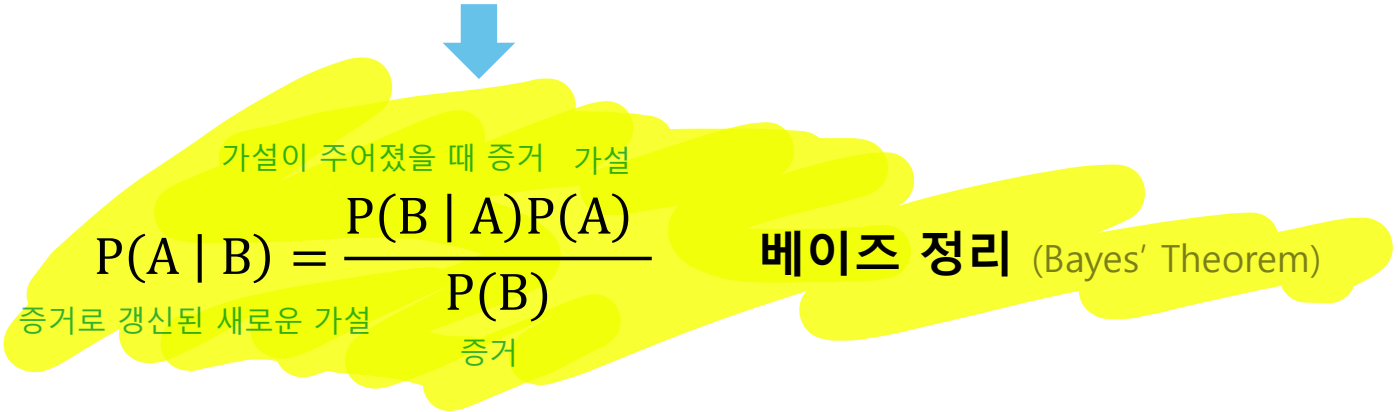
- 확률은 " **사건의 발생에 대한 믿음의 정도 (신뢰도)** " 를 나타낸다고 보는 관점
- 따라서, 어떠한 명제라도 확률을 부여할 수 있다.
- 새로운 증거를 토대로 지속적으로 " **사건의 발생에 대한 믿음의 정도** " 를 갱신해 나감
- 증거가 많아지면 빈도주의적 확률 값과 같아 짐

샘플(증거)가 적을 경우에는 빈도주의적 확률과가
베이즈 확률 값은 달라질 수 있다

베이즈 정리(Bayes' Theorem)

$$P(A \mid B)P(B) = P(A \cap B) = P(B \mid A)P(A)$$ **조건부 확률** (conditional distribution)

에 대해서 정리해보자!



"A를 **가설**이라고 하고, B를 **증거**라고 하면
베이즈 정리는 가설에 대한 믿음을 증거를 토대로 갱신하는 과정을 나타낸다."

베이즈 정리(Bayes' Theorem)

베이즈 정리 (Bayes' Theorem)

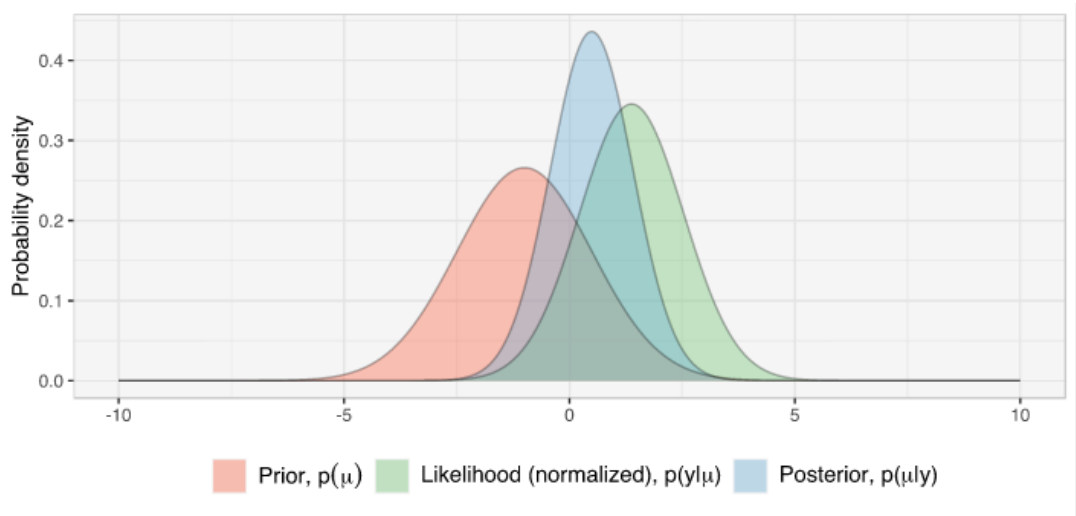
가능도 (likelihood) 사전 확률 (Prior)

사후 확률 (Posterior) $$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

증거 (Evidence)

A : 가설 (hypothesis), B : 증거 (evidence)

$$P(B) = P(B, A) + P(B, A^c)$$



- **사전 확률** (Prior) : 사전 지식을 통해 부여한 '가설 또는 새로운 주장'에 대한 확률
- **가능도** (likelihood) : 가설이 주어졌을 때 증거에 대한 확률
- **사후 확률** (Posterior) : 증거로 갱신된 '가설 또는 새로운 주장' 에 대한 새로운 확률

베이즈 정리(Bayes' Theorem)

10,000명 중 1명이 걸리는 질병이 있고 질병의 진단 정확도가 99%라고 하자.

양성 : 질병에 있다고 진단, 음성 : 질병이 없다고 진단

$$P(D) = \frac{1}{10{,}000}$$ D : 질병에 걸린 사건

T : 진단 결과가 양성인 경우

$$P(T|D) = 0.99 \qquad P(T|D^c) = 0.01$$
$$P(T^c|D) = 0.01 \qquad P(T^c|D^c) = 0.99$$

	Actual (D)	Actual (D^c)
Predicted (T)	0.99	0.01
Predicted (T^c)	0.01	0.99

양성 판정을 받았을 때 질병에 걸려있을 확률은?

$$P(D \mid T) = \frac{P(T|D)P(D)}{P(T)}$$

$$= \frac{P(T|D)P(D)}{P(T|D)P(D) + P(T|D^c)P(D^c)}$$

$$= \frac{\frac{99}{100}\frac{1}{10{,}000}}{\frac{99}{100}\frac{1}{10{,}000} + \frac{1}{100}\frac{9999}{10{,}000}}$$

$$= \frac{99}{99 + 9999}$$

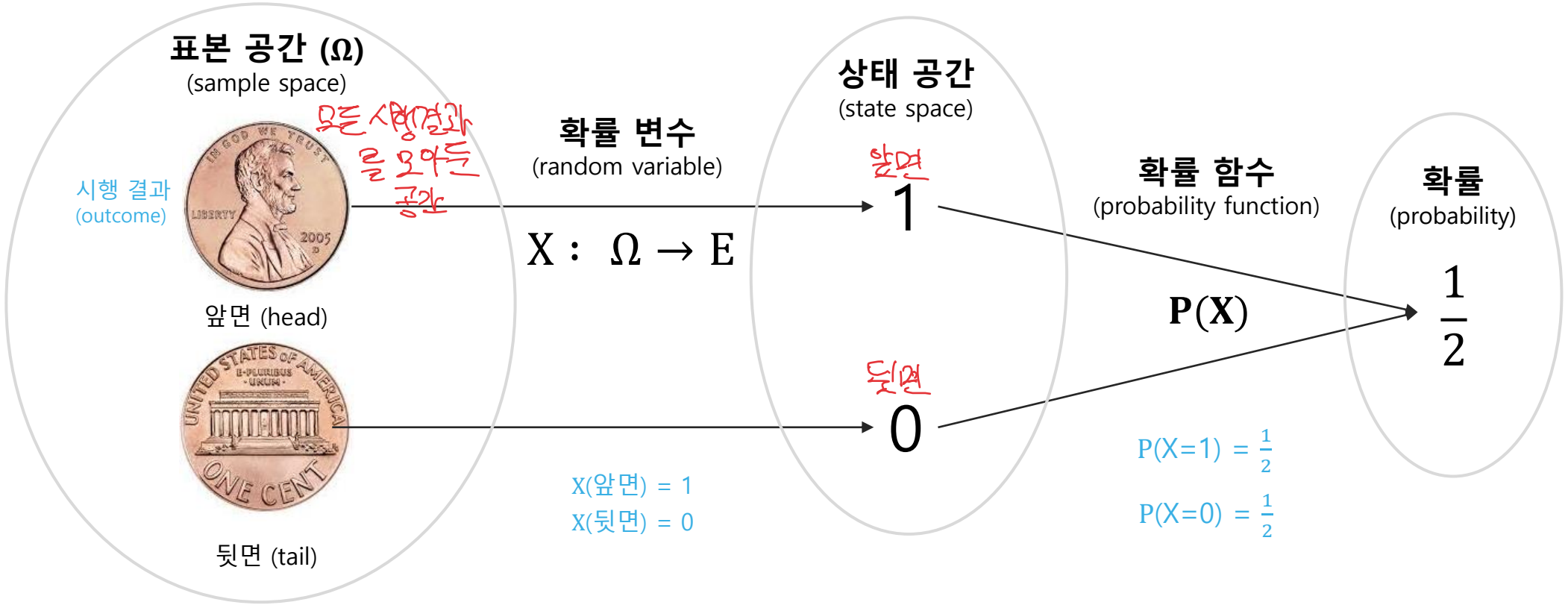
$$= \frac{99}{10098}$$

$$= 0.0098$$

0.98%

확률 변수 (Random variable)

확률 변수(random variable)는 임의의 현상에 의해 확률적으로 값이 결정되는 변수



사건 (event) : 표본 공간의 부분 집합

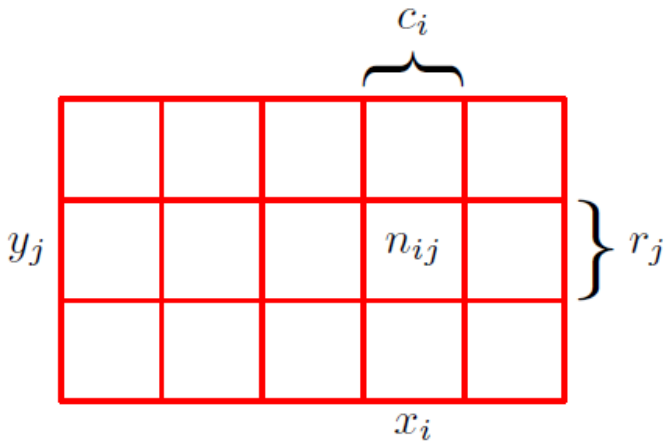
- 확률 변수는 모든 가능한 시행 결과를 실수로 맵핑하는 함수
- 확률 변수는 확률 질량 함수 또는 확률 밀도 함수에 사용됨

확률 공식 (Probability Rule)

= 확률의 법칙

확률 변수 X와 Y가 있다.

X는 x_i $(i = 1, 2, \dots, M)$ 값을 가질 수 있고 Y는 y_j $(j = 1, 2, \dots, L)$ 값을 가질 수 있다.



전체 N번을 샘플링 했을 때

- X=x_i인 횟수가 c_i이고
- Y=y_j인 횟수를 r_j이고
- X=x_i이고 Y=y_j인 횟수를 n_{ij}라고 하자.

이때, 확률은 다음과 같이 정의된다.

$$P(X = x_i, Y = y_j) = \frac{n_{ij}}{N}$$

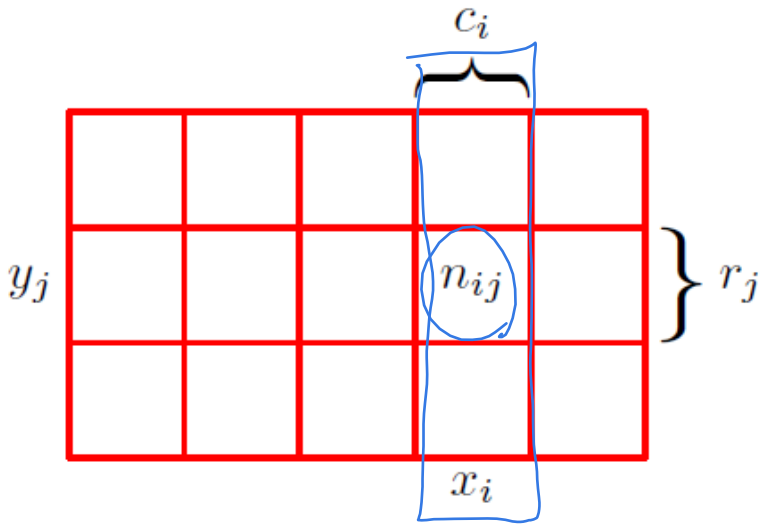
결합 확률 (joint probability)

$$P(X = x_i) = \frac{c_i}{N}$$

$$P(Y = y_j) = \frac{r_j}{N}$$

확률 공식 (Probability Rule)

= 확률의 법칙



$$\mathrm{P}(\mathrm{X}=x_i,\mathrm{Y}=y_j)=\frac{n_{ij}}{N}$$

$$\mathrm{P}(\mathrm{X}=x_i)=\frac{c_i}{N}$$

$$\mathrm{P}(\mathrm{Y}=y_j)=\frac{r_j}{N}$$

확률의 합 공식 (sum rule of probability)

$c_i=\sum_{j=1}^{L} n_{ij}$ 이므로

$$\mathrm{P}(\mathrm{X}=x_i)=\sum_{j=1}^{L}\mathrm{P}(\mathrm{X}=x_i,\mathrm{Y}=y_j)$$

주변 확률 (marginal probability)

marginalition 했다라고 표현

확률의 곱 공식 (product rule of probability)

$\mathrm{P}(\mathrm{Y}=y_j|\mathrm{X}=x_i)=\frac{n_{ij}}{c_i}$ 이므로

조건부 확률 (conditional probability)

nij 칸 / 열의 총 칸수

$$\mathrm{P}(\mathrm{X}=x_i,\mathrm{Y}=y_j)=\frac{n_{ij}}{N}=\frac{n_{ij}}{c_i}\cdot\frac{c_i}{N} \rightarrow P(X=x_i)$$

$$=\mathrm{P}(\mathrm{Y}=y_j|\mathrm{X}=x_i)\mathrm{P}(\mathrm{X}=x_i)$$

확률 공식 (Probability Rule)

확률의 합 공식 (sum rule of probability)

$$P(X) = \sum_{Y} P(X, Y)$$

확률의 곱 공식 (product rule of probability)

$$P(X, Y) = P(Y|X)P(X)$$

2. 나이브 베이즈

스팸 필터 (Spam Filter)

지금 도착한 메일이 스팸일까 스팸이 아닐까?

D1: “send us your password”	**Spam**
D2: “send us your review”	**Spam**
D3: “review your password”	**Ham**
D4: “review us”	**Ham**
D5: “send us password”	**Spam**
D6: “send us your account ”	**Spam**

스팸(spam)이 아닌 메일을 햄(ham)이라고 함

Password가 포함된 메시지가 스팸일까? 아닐까?

메일에 포함된 단어들이 스팸 메일에 자주 쓰이는 단어인지
햄 메일에 자주 쓰이는 단어인지로 판단해보자!

스팸 필터 비트 코인 스팸 필터

메일에 '비트 코인'이란 단어가 있을 때 메일이 스팸일 확률은?

S : 메일이 스팸인 경우

B : 메일에 단어 '비트 코인'이 포함된 경우

스팸 필터 비트 코인 스팸 필터

메일에 '비트 코인'이란 단어가 있을 때 메일이 스팸일 확률은?

베이즈 정리(Bayes' Theorem)**에 따라**

스팸일 때 '비트 코인' 단어를 포함 할 확률 스팸일 확률

$$P(S|B) = \frac{P(B|S)P(S)}{P(B)}$$

'비트 코인' 단어가 나타날 확률

$$P(B) = P(B|S)P(S) + P(B|S^c)P(S^c)$$

Spam 일 때 Ham 일 때

스팸 필터 일반화된 스팸 필터

메일에 포함된 모든 단어를 고려하여 메일이 스팸일 확률을 생각해보자.

w_{202} w_9 ...

w_{11} w_{32}

w_4 w_{74}

w_i : 메일에 포함될 단어 $(i = 1, 2, \dots, N)$

스팸 필터 일반화된 스팸 필터

단어 $w_1, w_2, \ldots, w_N$로 작성된 메일이 스팸이거나 햄일 확률은?

class

C_k $(k = 1, 2)$: 메일의 종류 (C_1: 스팸 메일, C_2 : 햄 메일)

W_i : 단어 w_i가 메일에 포함되어 있는지를 나타내는 확률 변수

Spam=1 Nam=2

$W = (W_1, W_2, \ldots, W_N)$: 모든 단어에 대한 확률 변수를 나타내는 벡터

스팸 필터 일반화된 스팸 필터

단어 $w_1, w_2, \dots, w_N$로 작성된 메일이 스팸이거나 햄일 확률은?

베이즈 정리(Bayes' Theorem)**에 따라**

$$P(C_k|\mathbf{W}) = \frac{P(\mathbf{W}|C_k)P(C_k)}{P(\mathbf{W})}$$

C_k ($K=1,2$) : (C_1: 스팸메일, C_2: 햄메일)

$$\mathbf{W} = (W_1, W_2, \dots, W_N)$$

$$P(\mathbf{W}) = P(\mathbf{W}|C_1)P(C_1) + P(\mathbf{W}|C_2)P(C_2)$$

스팸 필터 일반화된 스팸 필터

나이브 베이즈 가정 **스팸 메일 또는 햄 메일에서 각 단어가 나타날 확률은 독립이다.**

조건부 독립 $W_i \perp W_j \mid C_k \quad (i \neq j)$ $\quad P(W_i, W_j|C_k) = P(W_i|C_k)P(W_j|C_k)$

이런 메일 왔을때 password word은 독립이라고 가정. (완전히 100% 독립이라고 생각x)

나이브 하게 생각한다는게 나이브 베이즈 가정임.

$$P(C_k|\mathbf{W}) = \frac{P(\mathbf{W}|C_k)P(C_k)}{P(\mathbf{W})}$$

있으면1 없으면0인 binary 데이터임.

$$= \frac{P(W_1, W_2, \ldots, W_N|C_k)\,P(C_k)}{P(\mathbf{W})}$$

(2^N의 경우에 대한 확률을 계산해야 함)

$$= \frac{P(W_1|C_k)P(W_2|C_k)\ldots P(W_N|C_k)\,P(C_k)}{P(\mathbf{W})}$$

(나이브 베이브 가정)

곱의 형태로 결합확률 분포로 구하고싶다는 뜻

스팸 필터 일반화된 스팸 필터

메시지가 스팸일 확률

$$P(C_1|\mathbf{W}) = \frac{P(\mathbf{W}|C_1)P(C_1)}{P(\mathbf{W})}$$

$$P(\mathbf{W}|C_1) = P(W_1|C_1)P(W_2|C_1)\dots P(W_N|C_1)$$

메시지가 스팸일 때 단어들의 확률

메시지가 햄일 확률

$$P(C_2|\mathbf{W}) = \frac{P(\mathbf{W}|C_2)P(C_2)}{P(\mathbf{W})}$$

$$P(\mathbf{W}|C_2) = P(W_1|C_2)P(W_2|C_2)\dots P(W_N|C_2)$$

메시지가 햄일 때 단어들의 확률

나이브 베이즈 (Naïve Bayes)

N 개의 특징 $x = (x_1, x_2, \dots, x_N)$을 특징이 조건부 독립이라는 가정을 하에
베이즈 정리를 이용해서 클래스 C_k $(k = 1, 2, \dots, K)$를 분류하는 모델

$$P(C_k|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_k)P(C_k)}{P(\boldsymbol{x})}$$

베이즈 정리 (Bayes' Theorem)

$$= \frac{P(C_k)P(x_1|C_k)P(x_2|C_k) \dots P(x_N|C_k)}{P(\boldsymbol{x})}$$

나이브 베이즈 가정

$$P(\boldsymbol{x}) = \sum_{C_k} P(\boldsymbol{x}|C_k)P(C_k)$$

참고 나이브 베이즈 (Naïve Bayes) 유도

확률의 곱 공식 (product rule of probability)

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

1 **분자의 결합 확률 분포를 연쇄 법칙으로 인수분해**

$$\mathrm{P}(x_1, x_2, \dots, x_N, \mathrm{C}_k) = \mathrm{P}(\mathrm{C}_k)P(x_1, x_2, \dots, x_N|\mathrm{C}_k)$$

$$= \mathrm{P}(\mathrm{C}_k)\mathrm{P}(x_1|\mathrm{C}_k)\mathrm{P}(x_2, \dots, x_N|\mathrm{C}_k, x_1)$$

$$= \mathrm{P}(\mathrm{C}_k)\mathrm{P}(x_1|\mathrm{C}_k)\mathrm{P}(x_2|\mathrm{C}_k, x_1)\mathrm{P}(x_3, \dots, x_N|\mathrm{C}_k, x_1, x_2)$$

$$= \mathrm{P}(\mathrm{C}_k)\mathrm{P}(x_1|\mathrm{C}_k)\mathrm{P}(x_2|\mathrm{C}_k, x_1) \dots \mathrm{P}(x_N|\mathrm{C}_k, x_1, x_2, \dots, x_{N-1})$$

2 **변수 x_i 와 x_j $(i \neq j)$가 조건부 독립이라고 가정**

$$\mathrm{P}(x_i|\mathrm{C}_k, x_j) = \mathrm{P}(x_i|\mathrm{C}_k) \quad (\mathrm{i} \neq \mathrm{j})$$

↳ C_k만 있어도 독립이므로 생략가능.

3 **다음과 같이 식이 정리됨**

$$\mathrm{P}(x_1, x_2, \dots, x_N, \mathrm{C}_k) = \mathrm{P}(\mathrm{C}_k)\mathrm{P}(x_1|\mathrm{C}_k)\mathrm{P}(x_2|\mathrm{C}_k) \dots \mathrm{P}(x_\mathrm{N}|\mathrm{C}_k)$$

나이브 베이즈 장점과 단점

장점

- 간단하고 빠르고 매우 효율적인 알고리즘
- 잡음과 누락 데이터를 잘 처리함
- 훈련에는 상대적으로 적은 데이터가 필요하지만, 대용량 데이터셋에도 매우 잘 작동함

단점

- 모든 특징이 동등하게 중요하고 독립이라는 가정이 잘 안 맞는 경우가 많음
- 수치 특징이 많은 데이터셋에는 이상적이지 않음
- 예측된 클래스에 비해 추정된 확률의 신뢰도가 떨어짐

가정이 잘못되도 잘 작동하므로 그게 장점임

응용

전체 확률을 추정하기 위해 동시에 여러 속성 정보를 고려해야 하는 문제에 적합

- 스팸 이메일 필터링과 같은 텍스트 분류
- 일련의 관찰된 증상에 대한 의학적 질병 진단
- 컴퓨터 네트워크에서 침입이나 비정상행위 탐지

3. 나이브 베이즈 구현

구현 팁 사전 확률 (Prior)

메시지가 스팸일 확률과 햄일 사전 확률이 같다고 가정해보자!

$$P(C_1) = P(C_2)$$

메시지가 스팸일 확률

$$P(C_1|\mathbf{W}) = \frac{P(\mathbf{W}|C_1)}{P(\mathbf{W}|C_1) + P(\mathbf{W}|C_2)}$$

메시지가 햄일 확률

$$P(C_2|\mathbf{W}) = \frac{P(\mathbf{W}|C_2)}{P(\mathbf{W}|C_1) + P(\mathbf{W}|C_2)}$$

구현 팁 가능도 (likelihood)

가능도 $\left\{ \begin{array}{l} \bullet \ \mathbf{P(W_i = 1 | C_1)} : \text{스팸 메일에 단어 } w_i \text{가 나타날 확률} \\ \bullet \ \mathbf{P(W_i = 1 | C_2)} : \text{햄 메일에 단어 } w_i \text{가 나타날 확률} \end{array} \right\}$ 를 어떻게 추정해야 할까?

스팸 메일 50개

햄 메일 100개

단어		스팸 (Spam)	햄 (Ham)
w_1	비트코인	12	21
w_2	롤렉스	4	2
…	…	…	…
w_n	계정	21	11

단어가 몇개의 햄 메일에 나타나는지

스팸 메일과 햄 메일의 개수를 센다.

각 단어가 나타난 스팸 메일과 햄 메일의 개수를 센다.

구현 팁 가능도 (likelihood)

단어가 스팸 메일과 햄 메일에 나타날 확률은 다음 식으로 계산한다.

단어		스팸 (Spam)	햄 (Ham)
w_1	비트코인	12/50	21/100
w_2	롤렉스	4/50	2/100
…	…	…	…
w_n	계정	21/50	11/100

$$P(W_i|C_1) = \frac{W_i \text{ 가 나타난 스팸 메일 수}}{\text{스팸 메일 수}}$$

나타나질 않을 경우 $i = 0$

$$P(W_i|C_2) = \frac{W_i \text{ 가 나타난 햄 메일 수}}{\text{햄 메일 수}}$$

구현 팁 라플라스 스무딩 (Laplace Smoothing)

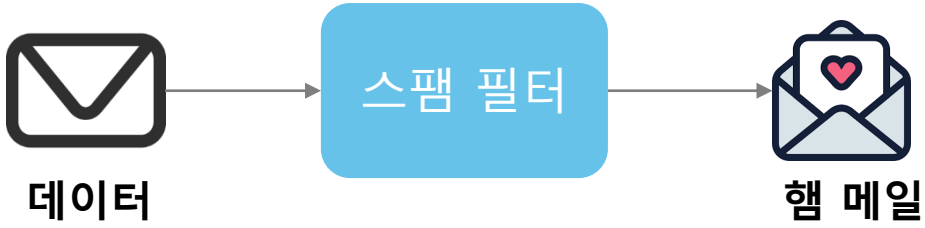
'데이터 ' 란 단어가 햄 메일에서만 나타난다면?

$$\mathbf{P}(\text{'데이터'}|C_1) = 0$$

스팸일 확률이 0이 됨

$$\mathrm{P}(\mathrm{C}_1|\mathbf{W}) = \frac{\mathrm{P}(\mathbf{W}|\mathrm{C}_1)\mathrm{P}(\mathrm{C}_1)}{\mathrm{P}(\mathbf{W})}$$

$$\mathrm{P}(\mathbf{W}|\mathrm{C}_1) = \mathrm{P}(\mathrm{W}_1|\mathrm{C}_1)\mathrm{P}(\mathrm{W}_2|\mathrm{C}_1)\ldots\mathrm{P}(\mathrm{W}_\mathrm{N}|\mathrm{C}_1)$$



'데이터'를 포함하는 메일은 무조건 햄 메일로 예측하게 됨

구현 팁 라플라스 스무딩 (Laplace Smoothing)

가짜 메시지 수인 k를 넣어 확률 값을 스무딩을 해주자!

$$P(W_i|C_1) = \frac{W_i \text{ 가 나타난 스팸 메일 수} + \boldsymbol{k}}{\text{스팸 메일 수} + \boldsymbol{2k}}$$

$$P(W_i|C_2) = \frac{W_i \text{ 가 나타난 햄 메일 수} + \boldsymbol{k}}{\text{햄 메일 수} + \boldsymbol{2k}}$$

가짜 빈도수 k의 의미는?

- 햄 메일과 스팸 메일 각각에
  - w_i 포함하는 메시지가 k 개 있고,
  - w_i 포함하지 않는 메시지가 k 개 있다고 가정한 것

K+K=2K

단어가 메일에 나타날 최소 확률이 존재한다고 가정하는 것

구현 팁 로그 합산 트릭 (Log-Sum-Exp Trick)

확률 값은 1보다 작기 때문에 여러 번 곱하면 언더플로(underflow)가 발생한다.

$$P(C_k|\mathbf{W}) = \frac{P(W_1|C_1)P(W_2|C_1)\dots P(W_N|C_1)\,P(C_k)}{P(\mathbf{W})}$$

로그 합산 트릭 (Log-Sum-Exp Trick)

$$e^{\log P(C_k) + \log P(W_1|C_k) + \log P(W_2|C_k) + \dots + \log P(W_N|C_k)}$$

- 작은 값들의 곱으로 표현된 식은 언더플로가 발생할 수 있으므로 log 를 적용하여 곱셈을 덧셈으로 변환해서 계산한 후 exp를 적용

log 취하고 Sum 하고 exp 를 적용

메세지 토큰화

메시지 단어 분리

```
from typing import Set
import re

def tokenize(text: str) -> Set[str]:
    text = text.lower()                         # Convert to lowercase,
    all_words = re.findall("[a-z0-9']+", text)  # extract the words, and
    return set(all_words)                       # remove duplicates.

assert tokenize("Data Science is science") == {"data", "science", "is"}
```

- 메시지를 소문자로 변환 (즉, 대소문자 구분하지 않음)
- 정규식을 이용해서 알파벳과 숫자, ' 로만 구성된 문자열 패턴을 모두 찾음
- Set으로 변환하여 중복을 제거한 후에 반환

메세지 클래스

메시지 클래스 정의

```
from typing import NamedTuple

class Message(NamedTuple):
    text: str
    is_spam: bool
```

- 메시지는 텍스트와 스팸 여부로 구성된다.

NaiveBayesClassifier 초기화

클래스 초기화

```python
from typing import List, Tuple, Dict, Iterable
import math
from collections import defaultdict

class NaiveBayesClassifier:
    def __init__(self, k: float = 0.5) -> None:
        self.k = k  # smoothing factor

        self.tokens: Set[str] = set()
        self.token_spam_counts: Dict[str, int] = defaultdict(int)
        self.token_ham_counts: Dict[str, int] = defaultdict(int)
        self.spam_messages = self.ham_messages = 0
```

- k : 스무딩 인수 0.5
- tokens : 전체 단어 집합
- token_spam_counts : 햄 메일 단어 별 빈도수
- token_ham_counts : 스팸 메일 단어 별 빈도수
- spam_messages : 스팸 메일 수
- Ham_messages : 햄 메일 수

초기화해야할 변수

token_spam_counts

단어		스팸 (Spam)	햄 (Ham)
w_1	비트코인	12	21
w_2	롤렉스	4	2
…	…	…	…
w_n	계정	21	11

token_ham_counts

tokens

NaiveBayesClassifier 모델 훈련

모델 훈련

```
    def train(self, messages: Iterable[Message]) -> None:
        for message in messages:
            # Increment message counts
            if message.is_spam:
                self.spam_messages += 1
            else:
                self.ham_messages += 1

            # Increment word counts
            for token in tokenize(message.text):
                self.tokens.add(token)
                if message.is_spam:
                    self.token_spam_counts[token] += 1
                else:
                    self.token_ham_counts[token] += 1
```

스팸 메일과 햄 메일의 개수를 센다.

각 단어가 나타난
스팸 메일과 햄 메일의 개수를 센다.

- 메시지 별로 for loop를 돌면서
  - 스팸 메일면 spam_messages 증가
  - 햄 메일면 ham_messages 증가
  - 각 메시지 별 텍스트에 대해서 단어 분리
    - tokens에 단어 추가 (집합 이므로 중복은 제거됨)
    - 현재 메시지가 스팸이면 token_spam_counts에 빈도수 증가
    - 현재 메시지가 햄이면 token_ham_counts에 빈도수 증가

NaiveBayesClassifier 단어 별 가능도 계산

$$P(W_i|C_1) = \frac{W_i \text{ 가 나타난 스팸 메일 수} + k}{\text{스팸 메일 수} + 2k}$$

$$P(W_i|C_2) = \frac{W_i \text{ 가 나타난 햄 메일 수} + k}{\text{햄 메일 수} + 2k}$$

단어 별 가능도 계산 (라플라스 스무딩 적용)

```
    def _probabilities(self, token: str) -> Tuple[float, float]:
        """returns P(token | spam) and P(token | not spam)"""
        spam = self.token_spam_counts[token]
        ham = self.token_ham_counts[token]

        p_token_spam = (spam + self.k) / (self.spam_messages + 2 * self.k)
        p_token_ham = (ham + self.k) / (self.ham_messages + 2 * self.k)

        return p_token_spam, p_token_ham
```

- p_token_spam :$P(W_i|C_1)$ 단어가 스팸 메일에 나타날 확률
- p_token_ham :$P(W_i|C_2)$ 단어가 햄 메일에 나타날 확률

NaiveBayesClassifier 예측

예측

```
    def predict(self, text: str) -> float:
        text_tokens = tokenize(text)
        log_prob_if_spam = log_prob_if_ham = 0.0
```

- 메시지를 토근화
- 스팸의 로그 확률, 햄의 로그 확률은 0으로 초기화

단어 별 로그 확률 계산

```
        # Iterate through each word in our vocabulary.
        for token in self.tokens:
            prob_if_spam, prob_if_ham = self._probabilities(token)
```

- tokens에 있는 모든 단어에 대해 확률을 계산

$$\texttt{prob\_if\_spam}: \ P(W_i \mid C_1) \qquad \texttt{prob\_if\_ham}: \ P(W_i \mid C_2)$$

NaiveBayesClassifier 예측

로그 합산 트릭 (Log-Sum-Exp Trick)

$$\mathrm{P}(\mathrm{W}|C_k) = \mathrm{e}^{\log \mathrm{P}(C_k) + \log \mathrm{P}(\mathrm{W}_1|\, C_k) + \log \mathrm{P}(\mathrm{W}_2|C_k) + \ldots + \log \mathrm{P}(\mathrm{W}_\mathrm{N}|C_k)}$$

1) 단어가 메일에 포함된 경우

```
# If *token* appears in the message,
# add the log probability of seeing it;
if token in text_tokens:
    log_prob_if_spam += math.log(prob_if_spam)
    log_prob_if_ham += math.log(prob_if_ham)
```

$\log \mathrm{P}(\mathrm{W}_i = 1|C_1)$

$\log \mathrm{P}(\mathrm{W}_i = 1|C_2)$

2) 단어가 메일에 없는 경우

```
# otherwise add the log probability of _not_ seeing it
# which is log(1 - probability of seeing it)
else:
    log_prob_if_spam += math.log(1.0 - prob_if_spam)
    log_prob_if_ham += math.log(1.0 - prob_if_ham)
```

$\log \mathrm{P}(\mathrm{W}_i = 0|C_1) = \log 1 - \mathrm{P}(\mathrm{W}_i = 1|C_1)$

$\log \mathrm{P}(\mathrm{W}_i = 0|C_2) = \log 1 - \mathrm{P}(\mathrm{W}_i = 1|C_2)$

NaiveBayesClassifier 예측

예측

```
        prob_if_spam = math.exp(log_prob_if_spam)
        prob_if_ham = math.exp(log_prob_if_ham)
        return prob_if_spam / (prob_if_spam + prob_if_ham)
```

메일이 스팸일 확률 반환

$$P(C_1|\mathbf{W}) = \frac{P(\mathbf{W}|C_1)}{P(\mathbf{W}|C_1) + P(\mathbf{W}|C_2)}$$

$$P(\mathbf{W}|C_1) = P(W_1|C_1)P(W_2|C_1)\dots P(W_N|C_1)$$

$$P(\mathbf{W}|C_2) = P(W_1|C_2)P(W_2|C_2)\dots P(W_N|C_2)$$

모델 검증

메일 정의

```
messages = [Message("spam rules", is_spam=True),
            Message("ham rules", is_spam=False),
            Message("hello ham", is_spam=False)]
```

- 1개의 스팸 메일과 2개의 햄 메일 정의

모델 훈련

```
model = NaiveBayesClassifier(k=0.5)
model.train(messages)
```

훈련 결과 검증

```
assert model.tokens == {"spam", "ham", "rules", "hello"}
assert model.spam_messages == 1
assert model.ham_messages == 2
assert model.token_spam_counts == {"spam": 1, "rules": 1}
assert model.token_ham_counts == {"ham": 2, "rules": 1, "hello": 1}
```

모델 검증

테스트 메일

```
text = "hello spam"
```

스팸 메일 단어 별 확률

```
probs_if_spam = [
    (1 + 0.5) / (1 + 2 * 0.5),      # "spam"  (present)
    1 - (0 + 0.5) / (1 + 2 * 0.5),  # "ham"   (not present)
    1 - (1 + 0.5) / (1 + 2 * 0.5),  # "rules" (not present)
    (0 + 0.5) / (1 + 2 * 0.5)       # "hello" (present)
]
```

텍스트에 없으므로 1에서 뺀값

단어		스팸 (Spam)	햄 (Ham)
w_1	Spam	1	0
w_2	Ham	0	2
w_3	Rules	1	1
w_4	hello	0	1

$$P(W_i|C_1) = \frac{W_i \text{ 가 나타난 스팸 메일 수} + k}{\text{스팸 메일 수} + 2k}$$

- K = 0.5
- 스팸 메일 수 : 1
- 햄 메일 수 : 2

모델 검증

햄 메일 단어 별 확률

```
probs_if_ham = [
    (0 + 0.5) / (2 + 2 * 0.5),      # "spam"  (present)
    1 - (2 + 0.5) / (2 + 2 * 0.5),  # "ham"   (not present)
    1 - (1 + 0.5) / (2 + 2 * 0.5),  # "rules" (not present)
    (1 + 0.5) / (2 + 2 * 0.5),      # "hello" (present)
]
```

단어		스팸 (Spam)	햄 (Ham)
w_1	Spam	1	0
w_2	Ham	0	2
w_3	Rules	1	1
w_4	hello	0	1

$$P(W_i|C_2) = \frac{W_i \text{ 가 나타난 햄 메일 수 } + k}{\text{햄 메일 수 } + 2k}$$

- K = 0.5
- 스팸 메일 수 : 1
- 햄 메일 수 : 2

직접 계산한 결과와 예측 결과가 같은 지 검증

```
p_if_spam = math.exp(sum(math.log(p) for p in probs_if_spam))
p_if_ham = math.exp(sum(math.log(p) for p in probs_if_ham))

# Should be about 0.83
assert model.predict(text) == p_if_spam / (p_if_spam + p_if_ham)
```

3. 스팸어쌔신 데이터셋 적용

스팸어쌔신 데이터셋

https://spamassassin.apache.org/old/publiccorpus/

스팸어쌔신(SpamAssassin)

- 내용 일치 규칙 기반의 스팸 메일 필터링 프로그램
- 아파치 라이선스 2.0 공개 프로그램
- 현재 아파치 재단에 속함

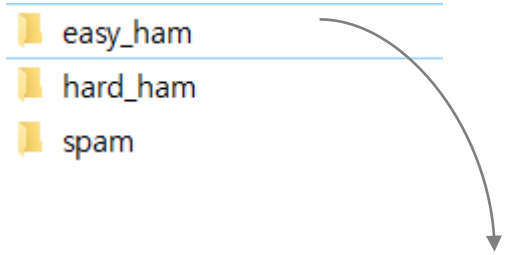
Index of /old/publiccorpus

Name	Last modified	Size	Description
Parent Directory		-	
20021010_easy_ham.tar.bz2	2004-06-29 03:26	1.6M	
20021010_hard_ham.tar.bz2	2004-12-16 19:49	1.0M	
20021010_spam.tar.bz2	2004-06-29 03:26	1.1M	
20030228_easy_ham.tar.bz2	2004-06-29 03:26	1.5M	
20030228_easy_ham_2.tar.bz2	2004-06-29 03:26	1.0M	
20030228_hard_ham.tar.bz2	2004-12-16 19:49	1.0M	
20030228_spam.tar.bz2	2004-06-29 03:26	1.1M	
20030228_spam_2.tar.bz2	2004-06-29 03:26	2.0M	
20050311_spam_2.tar.bz2	2005-03-11 23:55	2.0M	
obsolete/	2018-06-04 06:37	-	
readme.html	2006-01-31 20:30	4.5K	

- 20021010 버전 사용
- 2 버전의 햄 압축 파일과 1개의 스팸 압축 파일

데이터 다운로드

1 다운로드 해서 압축을 풀면 세 개의 디렉토리가 생김



2 각 디렉토리에는 email 형식의 메시지들이 들어있다.

이름	수정한 날짜	유형	크기
0001.ea7e79d3153e7469e7a9c3e0af6a357e	2002-10-11 오전 5:52	EA7E79D3153E7469E7A9C3E0AF6A357E 파일	5KB
0002.b3120c4bcbf3101e661161ee7efcb8bf	2002-10-11 오전 5:52	B3120C4BCBF3101E661161EE7EFCB8BF 파일	4KB
0003.acfc5ad94bbd27118a0d8685d18c89dd	2002-10-11 오전 5:52	ACFC5AD94BBD27118A0D8685D18C89DD 파일	4KB
0004.e8d5727378ddde5c3be181df593f1712	2002-10-11 오전 5:52	E8D5727378DDDE5C3BE181DF593F1712 파일	4KB
0005.8c3b9e9c0f3f183ddaf7592a11b99957	2002-10-11 오전 5:52	8C3B9E9C0F3F183DDAF7592A11B99957 파일	5KB
0006.ee8b0dba12856155222be180ba122058	2002-10-11 오전 5:52	EE8B0DBA12856155222BE180BA122058 파일	4KB
0007.c75188382f64b090022fa3b095b020b0	2002-10-11 오전 5:52	C75188382F64B090022FA3B095B020B0 파일	4KB
0008.20bc0b4ba2d99aae1c7098069f611a9b	2002-10-11 오전 5:52	20BC0B4BA2D99AAE1C7098069F611A9B 파일	4KB
0009.435ae292d75abb1ca492dcc2d5cf1570	2002-10-11 오전 5:52	435AE292D75ABB1CA492DCC2D5CF1570 파일	4KB
0010.4996141de3f21e858c22f88231a9f463	2002-10-11 오전 5:52	4996141DE3F21E858C22F88231A9F463 파일	9KB
0011.07b11073b53634cff892a7988289a72e	2002-10-11 오전 5:52	07B11073B53634CFF892A7988289A72E 파일	6KB

데이터 다운로드

3 메시지를 중에 제목만을 추출해서 스팸 분류기의 훈련 데이터로 만들려고 함

```
....
From: Robert Elz <kre@munnari.OZ.AU>
To: Chris Garrigues <cwg-dated-1030377287.06fa6d@DeepEddy.Com>
Cc: exmh-workers@example.com
Subject: Re: New Sequences Window
In-Reply-To: <1029945287.4797.TMDA@deepeddy.vircio.com>
References: <1029945287.4797.TMDA@deepeddy.vircio.com>
    <1029882468.3116.TMDA@deepeddy.vircio.com> <9627.1029933001@munnari.OZ.AU>
    <1029943066.26919.TMDA@deepeddy.vircio.com>
    <1029944441.398.TMDA@deepeddy.vircio.com>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Message-Id: <13258.1030015585@munnari.OZ.AU>
X-Loop: exmh-workers@example.com
Sender: exmh-workers-admin@example.com
Errors-To: exmh-workers-admin@example.com
X-Beenthere: exmh-workers@example.com
X-Mailman-Version: 2.0.1
....
```

인터넷 메세지 포맷 : https://www.wikiwand.com/en/Email#/Internet_Message_Format

데이터 다운로드

데이터셋 URL 및 파일명

```
from io import BytesIO
import requests
import tarfile

BASE_URL = "https://spamassassin.apache.org/old/publiccorpus/"
FILES = ["20021010_easy_ham.tar.bz2",
         "20021010_hard_ham.tar.bz2",
         "20021010_spam.tar.bz2"]
OUTPUT_DIR = 'spam_data'
```

데이터 다운로드

파일 별로 다운로드 및 압축 풀기

```
for filename in FILES:
    content = requests.get(f"{BASE_URL}/{filename}").content

    fin = BytesIO(content)
                                        압축 풀기 과정
    with tarfile.open(fileobj=fin, mode='r:bz2') as tf:
        tf.extractall(OUTPUT_DIR)
```

- URL에서 파일 다운로드해서 ByteIO 객체 생성
  - ByteIO : 인 메모리 바이너리 스트림
- 압축 파일 열기
  - fileobj가 지정되면 바이너리 모드로 열린 파일 객체의 대안으로 사용됨
  - bzip2 포맷으로 읽기 모드로 열기 (mode='r:bz2')
- 모든 멤버를 지정된 경로에 추출

이메일 파일 읽기

경로에 있는 파일을 하나씩 읽음

```
import glob

# modify the path to wherever you've put the files
path = 'spam_data/*/*’
data: List[Message] = []

# glob.glob returns every filename that matches the wildcarded path
for filename in glob.glob(path):
    is_spam = "ham" not in filename
```

- 경로에 있는 파일들을 하나씩 읽음
- 파일 이름에 ham이 포함되어 있지 않으면 스팸 메일로 간주

이메일 파일 읽기

파일을 열어서 제목을 추출

```
    # There are some garbage characters in the emails, the errors='ignore'
    # skips them instead of raising an exception.
    with open(filename, errors='ignore') as email_file:
        for line in email_file:
            if line.startswith("Subject:"):
                subject = line.lstrip("Subject: ")
                data.append(Message(subject, is_spam))
                break  # done with this file
```

- 제목 추출 : “Subject: “로 시작하는 라인이 있으면 lstrip을 사용해서 제목 부분만 추출

데이터 분할 및 모델 훈련

데이터셋 분할

```
import random
from scratch.machine_learning import split_data

random.seed(0)      # just so you get the same answers as me
train_messages, test_messages = split_data(data, 0.75)
```

- 데이터의 75%는 훈련 데이터셋으로, 나머지 25%는 테스트 데이터셋으로 분할
- 모델 생성 및 훈련

모델 생성 및 훈련

```
model = NaiveBayesClassifier()
model.train(train_messages)
```

테스트

테스트

```
from collections import Counter

predictions = [(message, model.predict(message.text))
               for message in test_messages]
```

- 테스트 데이터셋으로 예측 (메시지, 스팸일 확률) 튜플 리스트 생성

테스트 및 성능 평가

혼동행렬 생성

```
# Assume that spam_probability > 0.5 corresponds to spam prediction
# and count the combinations of (actual is_spam, predicted is_spam)
confusion_matrix = Counter((message.is_spam, spam_probability > 0.5)
                           for message, spam_probability in predictions)

print(confusion_matrix)
```

확률이 0.5 이상이면 스팸이라고 가정하겠다

- 혼동행렬 생성 (실제 스팸 여부, 예측된 스팸 여부) 튜플을 생성해서 개수를 셈

Counter({(False, False): 678, (True, True): 89, (True, False): 37, (False, True): 21})

실제 클래스 (Actual Class)

예측 클래스 (Predictive Class)		스팸 (True)	햄 (False)
	스팸 (True)	89	21
	햄 (False)	37	678

실제 스팸인데 스팸으로 예측함

accuracy : 91%
precision : 74%
recall : 68%
f1_score : 71%

accuracy precision에 대한 조화평균

스팸일 확률이 높은 단어

단어가 스팸 메일에 나타날 확률 계산

```
def p_spam_given_token(token: str, model: NaiveBayesClassifier) -> float:
    # We probably shouldn't call private methods, but it's for a good cause.
    prob_if_spam, prob_if_ham = model._probabilities(token)

    return prob_if_spam / (prob_if_spam + prob_if_ham)
```

- 각 토큰에 대해서 스팸일 확률과, 햄일 확률을 구해서 사후 확률을 구함

스팸 메일에 나타날 확률 순으로 모든 단어를 정렬

```
words = sorted(model.tokens, key=lambda t: p_spam_given_token(t, model))

print("spammiest_words", words[-10:])
print("hammiest_words", words[:10])
```

- 스팸일 확률이 높은 순서대로 출력
- 햄일 확률이 높은 순서대로 출력

spammiest_words ['assistance', 'clearance', 'attn', 'per', 'sale', 'systemworks', 'rates', 'zzzz', 'money', 'adv'
hammiest_words ['spambayes', 'users', 'razor', 'sadev', 'zzzzteana', 'perl', 'apt', 'spamassassin', 'ouch', 'problem']

Thank you!