

## B-6) 単純ベイズでテキスト分類

---

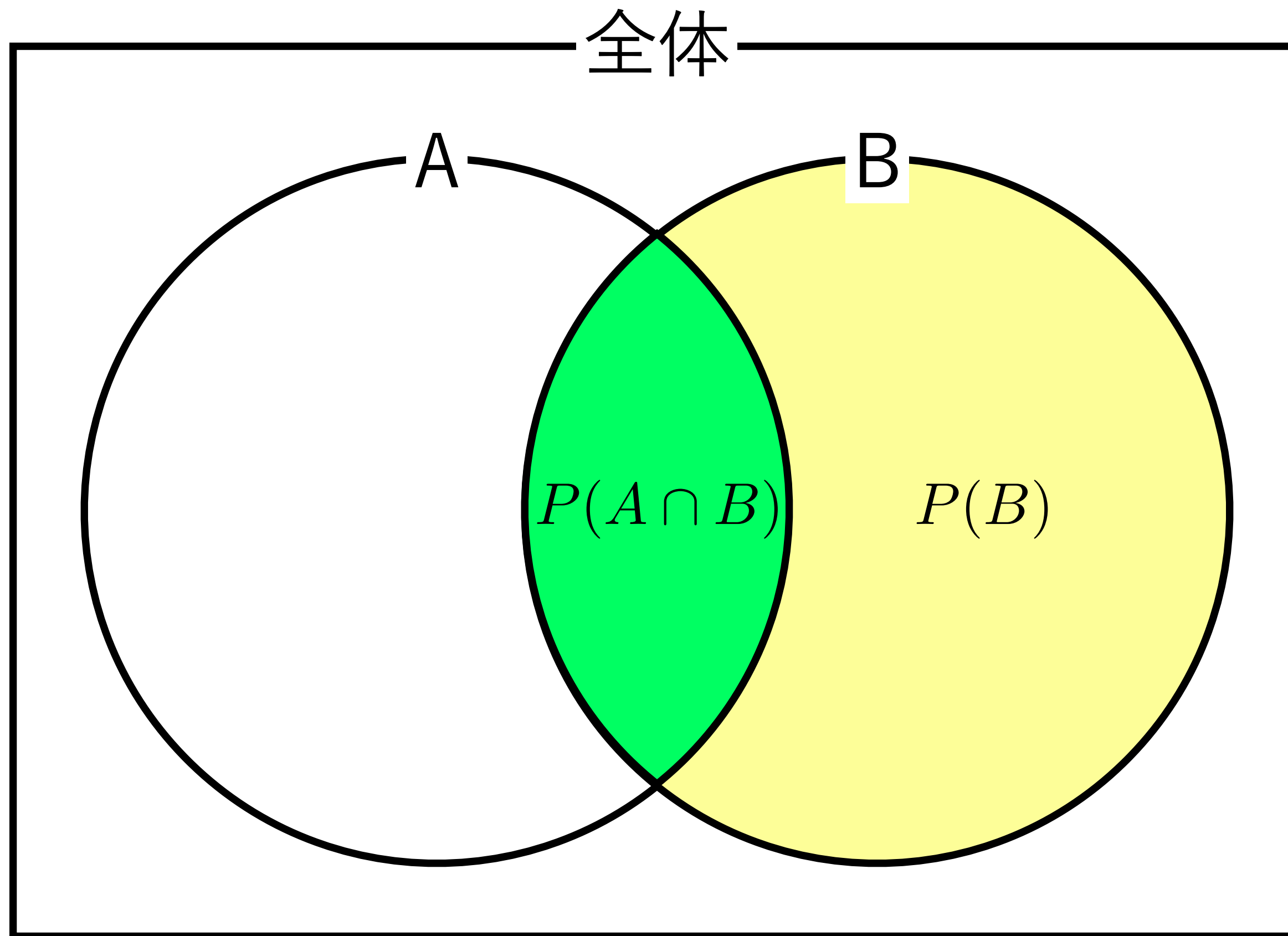
社会システム科学

# 単純ベイズ分類器

# 条件付き確率

- 事象Bが起こったという前提のもとで事象Aが起こる確率

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



$$P(A \cap B)$$

同時確率（事象Aと事象Bが同時に起こる確率）

※事象Aと事象Bが独立の場合は以下

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

# 条件付き確率

---

- 検査陽性になった人が実際に陽性である確率は？
  - 全体の1%がかかっている病気がある
  - 病気になっていない人でも10%は検査陽性（偽陽性）（
  - 病気になっている人の90%は検査陽性（10%は検査陰性（偽陰性））

# ベイズの定理

---

- ・ 条件付き確率の式から：

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

- ・ これを変形して：

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

# ベイズの定理

---

- 傘を持っている場合に雨が降っている確率は？
  - 雨が降ったら傘を持っている確率 0.8
  - 雨が降る確率 0.4
  - 雨でも晴れでもとにかく傘を持っている確率 0.5

# 単純ベイズ分類器 (Naive Bayes Classifier)

---

- 目的：事例のクラス分類（クラスは既知）
  - 特徴の出現頻度（出現確率）からクラスへの分類確率を計算
  - クラスが既知の事例からパラメータを学習

# 単純ベイズ分類器：テキスト分類の場合

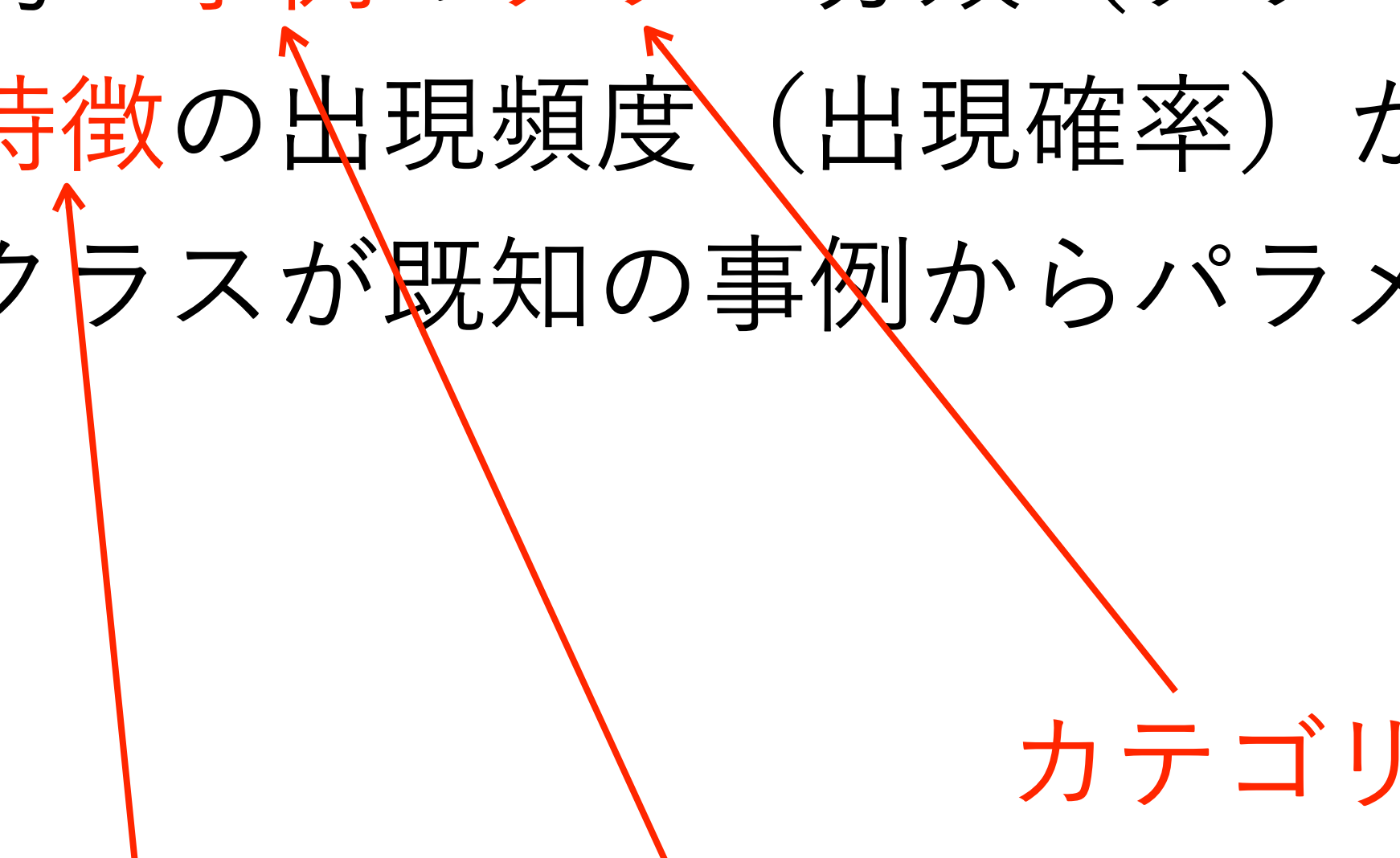
---

- 目的：事例のクラス分類（クラスは既知）
  - 特徴の出現頻度（出現確率）からクラスへの分類確率を計算
  - クラスが既知の事例からパラメータを学習

単語

文書

カテゴリ





# テキスト分類のための単純ベイズ分類器

---

- 目的

- 文書に対して  $P(\text{カテゴリ} | \text{文書})$  が最大となるカテゴリを求める
- ベイズの定理から以下のように変形：

$$P(\text{カテゴリ} | \text{文書}) = \frac{P(\text{文書} | \text{カテゴリ}) \cdot P(\text{カテゴリ})}{P(\text{文書})}$$

- 単一の文書に対して比較を行うならば**分子のみ**を比較すればOK

$$P(\text{カテゴリ} | \text{文書}) \propto P(\text{文書} | \text{カテゴリ}) \cdot P(\text{カテゴリ})$$

- 単語の出現確率を互いに独立として以下のように変形：

$$P(\text{文書} | \text{カテゴリ}) = \prod_i P(\text{単語}_i | \text{カテゴリ})$$

文書からこの確率分布を推定（学習）

# 単純ベイズ分類器の種類

---

- 単純ベイズ分類器の種類 = 確率分布の種類
- 特徴によって複数の種類がある
  - 特徴が連続値 → ガウス分布（正規分布）
  - 特徴が二値 → ベルヌーイ分布
  - 特徴が離散値 → 多項分布

# Google Colabで単純ベイズ分類器

# 単純ベイズ分類器に必要なパッケージ

---

- Python用パッケージ
  - scikit-learn：機械学習用パッケージ

## [参考]

- scikit-learnに含まれる機械学習の種類と利用法について：  
[http://scikit-learn.org/stable/tutorial/machine\\_learning\\_map](http://scikit-learn.org/stable/tutorial/machine_learning_map)

# パッケージを読み込む

---

## 1. 必要なパッケージを読み込む

```
import numpy as np  
from sklearn.naive_bayes import GaussianNB
```

← ガウス分布の単純ベイズ分類器

# 学習セットを用意

## 2. 学習セット = 事例（特徴） + クラス

```
X = np.array([[1, 0, 1, 1, 2, 1, 1, 0, 0],  
              [1, 2, 0, 1, 0, 0, 3, 3, 1],  
              [2, 0, 1, 0, 3, 1, 0, 2, 1]])  
y = np.array([1, 2, 3])
```

← 特徴ベクトル

← それぞれの事例が分類されるクラス

# 特徴ベクトルの形式

特徴									
1	2	3	4	5	6	7	8	9	
[ 1, 0, 1, 1, 2, 1, 1, 0, 0],	1	事例							
[ 1, 2, 0, 1, 0, 0, 3, 3, 1],	2								
[ 2, 0, 1, 0, 3, 1, 0, 2, 1]]	3								

事例2の特徴3の値が0

# 単純ベイズ分類器の学習

---

## 3. 単純ベイズ分類器オブジェクトの生成

```
clf = GaussianNB()
```

## 4. 単純ベイズ分類器の学習

```
clf.fit(X, y)
```

## 5. 学習結果の確認（学習セットに対する正解率の表示）

```
clf.score(X, y)
```



# 学習済み単純ベイズ分類器を用いた推定

---

## 6. 適当な特徴ベクトルを入力してどのクラスに分類されるか確認

```
t = np.array([[1, 0, 1, 1, 0, 0, 1, 0, 0]])  
clf.predict(t)
```

# テキストの分類

# パッケージを読み込む

---

## 1. 必要なパッケージを読み込む

```
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.feature_extraction.text import CountVectorizer
```

単語の数を数える (BoW) を作成するパッケージ



# 学習セットを用意

---

## 2. テキストを用意

```
s = ['今日 は とても 天気 が いい',  
     '今日 は 晴れ です',  
     '天気 が いい 日 は 晴れ です']
```

## 3. テキストを特徴ベクトルに変換

```
vectorizer = CountVectorizer()  
vectorizer.fit(s)  
X = vectorizer.transform(s).toarray()
```

## 4. テキストごとのクラスを用意

```
y = np.array([1, 2, 3])
```

# 単純ベイズ分類器の学習

---

## 5. 単純ベイズ分類器オブジェクトの生成

```
clf = GaussianNB()
```

## 6. 単純ベイズ分類器の学習

```
clf.fit(X, y)
```

## 7. 学習結果の確認（学習セットに対する正解率の表示）

```
clf.score(X, y)
```

# 学習済み単純ベイズ分類器を用いた推定

---

## 8. 適当な文章を入力してどのクラスに分類されるか確認

```
t = vectorizer.transform(['明日 は 天気 が いい']).toarray()  
clf.predict(t)
```

もっと長いテキスト文書の分類

準備



# テキストファイルのダウンロード

---

## 1. 前回の資料からテキストファイルをダウンロード

- 34ro\_content.txt（夏目漱石「三四郎」）
- bot\_content.txt（夏目漱石「坊ちゃん」）
- sore\_content.txt（夏目漱石「それから」）
- jigokuhen\_content.txt（芥川竜之介「地獄変」）
- kappa\_content.txt（芥川竜之介「河童」）
- ningen\_content.txt（太宰治「人間失格」）
- shayou\_content.txt（太宰治「斜陽」）

## 2. セッションストレージ（左側のファイルのところ）にアップロード

# Janomeのインストール

## 3. Google Colabのコードセルで以下を実行

```
!pip install janome
```

## 4. 実行されてインストールされる。

▶ `!pip install janome`

↳ Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Collecting janome

Downloading Janome-0.4.2-py2.py3-none-any.whl (19.7 MB)

19.7/19.7 MB 63.1 MB/s eta 0:00:00

Installing collected packages: janome

Successfully installed janome-0.4.2

# 学習セットの生成

# パッケージを読み込む

---

## 5. 必要なパッケージを読み込む

```
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.feature_extraction.text import CountVectorizer
from janome.tokenizer import Tokenizer ← 形態素解析
```

# 学習用文書の読み込み

---

## 6. 学習用文書の読み込み

(7つアップしたうち4つを学習用に3つはテスト用として後で用います)

```
file_list = ['bot_content.txt', 'sore_content.txt', 'kappa_content.txt', 'shayou_content.txt']
text_list = []
for file in file_list:
    with open(file, 'r') as f:
        text_list.append(f.readlines())
```

# 学習用文書を分かち書きに変換

---

## 7. 分かち書きに変換

```
t = Tokenizer()
wakati_list = []
for text in text_list:
    words = []
    for line in text:
        for w in t.tokenize(line):
            words.append(w.base_form)
    wakati_list.append(' '.join(words))
```

## 8. 確認する

```
wakati_list
```

# 学習用セットの生成

---

## 9. BoW（単語の出現頻度）計算用オブジェクトの生成

```
vectorizer = CountVectorizer()
```

## 10. BoW計算用オブジェクトの学習（内部辞書の生成）

```
vectorizer.fit(wakati_list)
```

## 11. 分かち書き文書をBoWに変換

```
X = vectorizer.transform(wakati_list).toarray()
```

## 12. テキストごとのクラス（正解）を用意（1：夏目 / 2：芥川 / 3：太宰）

```
y = np.array([1, 1, 2, 3])
```

# 単純ベイズ分類器の学習



# 単純ベイズ分類器の学習

---

## 13. 単純ベイズ分類器オブジェクトの生成

```
clf = GaussianNB()
```

## 14. 単純ベイズ分類器の学習

```
clf.fit(X, y)
```

## 15. 学習結果の確認（学習セットに対する正解率の表示）

```
clf.score(X, y)
```

# 学習済分類器のテスト

# テスト用文書の読み込み

---

## 16. テスト用に残した文書を読み込む

```
test_file_list = ['34ro_content.txt', 'jigokuhen_content.txt', 'ningen_content.txt']
test_text_list = []
for file in test_file_list:
    with open(file, 'r') as f:
        test_text_list.append(f.readlines())
```

## 17. 分かち書きにする

```
test_wakati_list = []
for text in test_text_list:
    words = []
    for line in text:
        for w in t.tokenize(line):
            words.append(w.base_form)
    test_wakati_list.append(' '.join(words))
```

# テスト

---

## 18. BoWの計算（テスト用データの作成）

```
T = vectorizer.transform(test_wakati_list).toarray()
```

## 19. 学習済の分類器を用いてクラスの推定

```
clf.predict(T)
```