

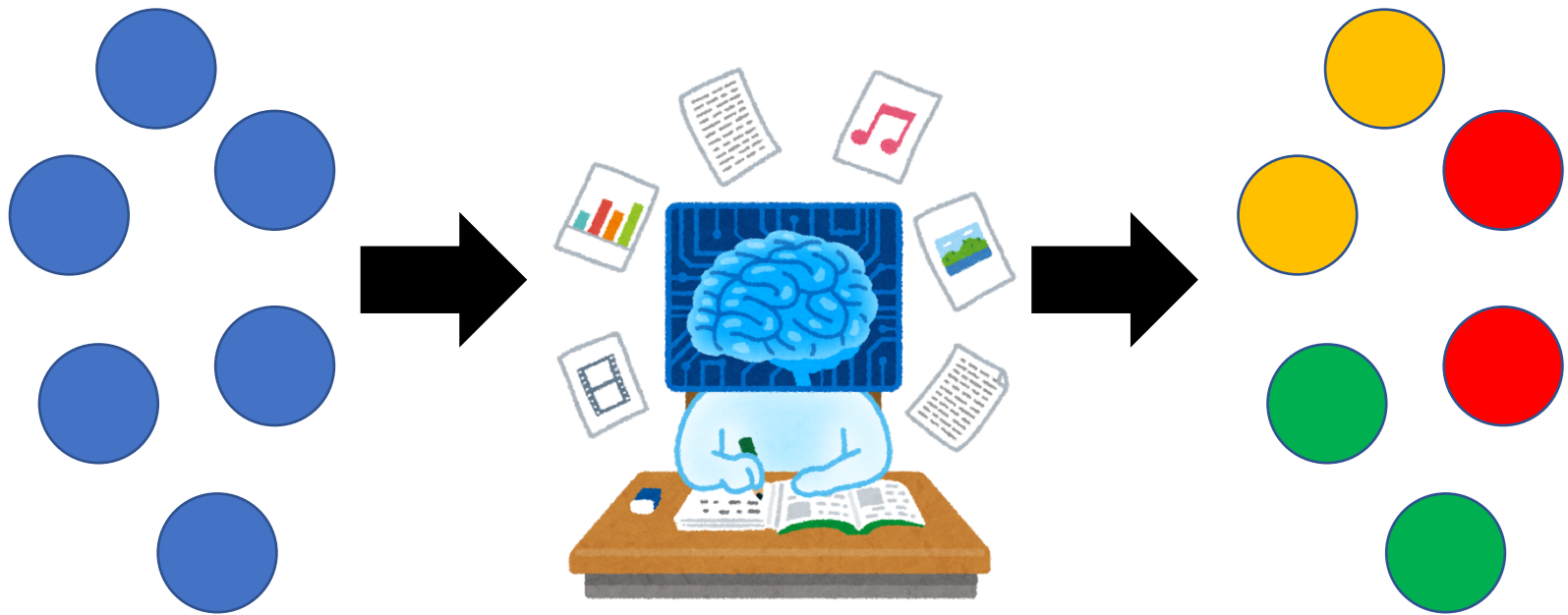
プロジェクト第6回

クラスタリング&ニューラルネット

クラスタリング

クラスタリング

- 多くのデータから類似性を見つけて自動で分類すること



クラスタリングの準備

- <https://word2vec.googlecode.com/svn/trunk/questions-words.txt> にアクセスする。
- 実行するPythonファイルと同じ位置に置く。

Ex6_1. “family” の抽出

```
with open("./family.txt","w")as output:  
    with open("./questions-words.txt","r")as data:
```

```
        section_flag=False  
        for line in data:  
            if section_flag:  
                if line.startswith(": "):  
                    print("Finish!")  
                    break  
                output.write(line)  
            elif line.startswith(": family"):  
                section_flag=True
```

演習1. 類似度計算

第5回プロジェクトのword2vecを使って、Ex6_1で作成したfamilyファイルの $\text{vec}(\text{2列目の単語}) - \text{vec}(\text{1列目の単語}) + \text{vec}(\text{3列目の単語})$ を計算し、そのベクトルと類似度が最も高い単語と、その類似度を求めよ。

演習1. 類似度計算

```
def compute_cosine(x, y):
```

```
    # コサイン類似度の計算
```

```
    # ここを埋める
```

```
    return similarity
```

$$similarity = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

Ex6_2. クラスタリング

```
import gensim
from gensim.models.word2vec import Word2Vec
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

```
importエラーが出る場合
pip install matplotlib
pip install pandas
pip install scikit-learn
```

```
wvmodel = Word2Vec.load("word2vec.gensim.model")
words = ["東京","沖縄","福岡","晴天","雨","曇","日本","アメリカ","中
国","みかん","ぶどう","りんご"]
```

```
wv = []
for w in words:
    wv.append(wvmodel[w])
```


Ex6_2. クラスタリング

```
model = KMeans(n_clusters=4).fit(wv)
labels = model.labels_
print(labels)
df = pd.DataFrame(wv)
df["word"] = words
df["cluster"] = labels
#PCAで2次元に圧縮
pca = PCA(n_components=2)
pca.fit(df.iloc[:, :-2])
feature = pca.transform(df.iloc[:, :-2])
```

Ex6_2. クラスタリング

#散布図プロット

```
color = {0:"green",1:"red",2:"yellow",3:"blue"}
```

```
colors = [color[x] for x in labels]
```

```
plt.figure(figsize=(6,6))
```

```
for x, y, name in zip(feature[:, 0], feature[:, 1], df.iloc[:, 2]):
```

```
plt.text(x, y, name)
```

```
plt.scatter(feature[:,0],feature[:,1],color=colors)
```

```
plt.show()
```

演習2. クラスタリング

Ex6_2のソースコードは、Kmeansを使ってクラスタリングを行なっている。Kmeansとは何か調査せよ。

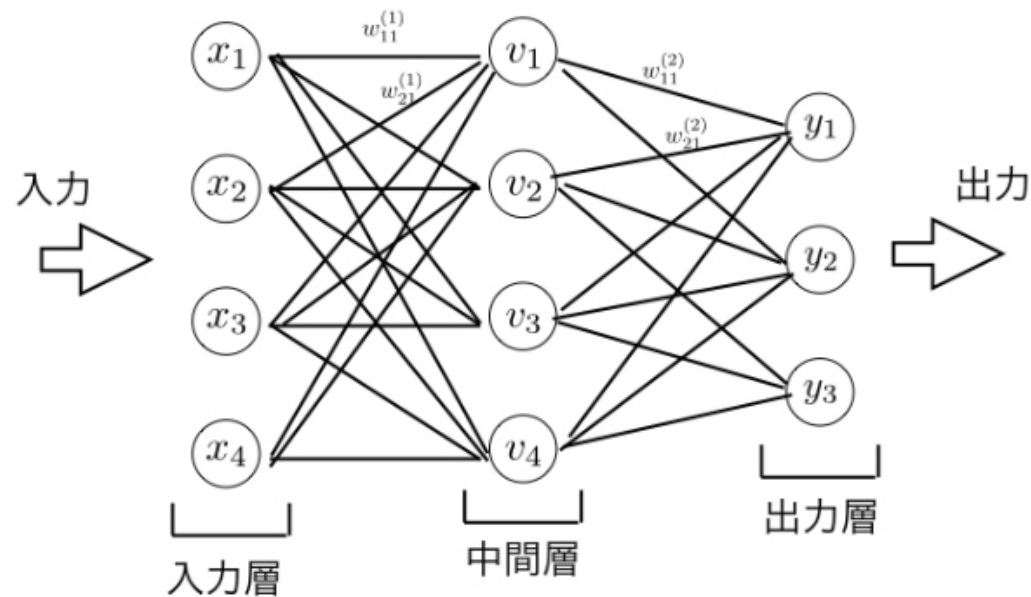
プロットで文字化けする場合

- <https://bit.ly/2Xi57QX> にアクセスして、サイトの手順に従う。
- 英語の場合、文字化けしない。

ニューラルネット


ニューラルネット

- 人間の脳機能に見られるいくつかの特性に類似した数理的モデル



データの準備

- <https://archive.ics.uci.edu/ml/datasets/News+Aggregator> にアクセスして Data Folder をクリック



UCI
Machine Learning Repository
Center for Machine Learning and Intelligent Systems

News Aggregator Data Set

Download [Data Folder](#) [Data Set Description](#)

Abstract: References to news pages collected from an web aggregator in the period from 10-March-2014 to 10-August-2014. The resources are grouped into clusters that represent pages discussing the same story.

Data Set Characteristics:	Multivariate	Number of Instances:	422937	Area:	N/A
Attribute Characteristics:	N/A	Number of Attributes:	5	Date Donated	2016-02-28
Associated Tasks:	Classification, Clustering	Missing Values?	N/A	Number of Web Hits:	84251

Source:

Provided by Artificial Intelligence Lab @ Faculty of Engineering, Roma Tre University - Italy

Contact: Fabio Gasparetti, Faculty of Engineering, Roma Tre University - Italy (gaspare.'@'dia.uniroma3.it)

データの準備

- NewsAggregatorDataset.zip をクリックしてダウンロードする

Index of /ml/machine-learning-databases/00359

- [Parent Directory](#)
- [NewsAggregatorDataset.zip](#)

Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips SVN/1.7.14 Phusion_Passenger/4.0.53 mod_perl/2.0.11 Perl/v5.16.3 Server at archive.ics.uci.edu Port 443

- ダウンロードした zip ファイルを解凍し、`readme.txt` の説明を読み、`newsCorpora.csv` を実行する Python ファイルと同じ位置におく。

データの作成

- 情報源（Publisher）が”Reuters”, “Huffington Post”, “Businessweek”, “Contactmusic.com”, “Daily Mail”の事例のみを抽出する
- 抽出された事例をランダムに並び替える
- 抽出された事例の80%を学習データ、残りの10%ずつを検証データと評価データに分割し、それぞれtrain.txt, valid.txt, test.txtで保存する

データの作成

importエラーが出る場合
pip install pandas
pip install scikit-learn

```
from sklearn.model_selection import train_test_split
import pandas as pd
newsCorpora = pd.read_table('./newsCorpora.csv', header=None)
newsCorpora.columns = ['ID', 'TITLE', 'URL', 'PUBLISHER', 'CATEGORY', 'STORY', 'HOSTNAME',
'TIMESTAMP']
newsCorpora = newsCorpora[newsCorpora['PUBLISHER'].isin(
['Reuters', 'Huffington Post', 'Businessweek', 'Contactmusic.com', 'Daily Mail'])].sample(frac=1,
random_state=0)
X = newsCorpora[['TITLE', 'CATEGORY']]
X['CATEGORY'] = X['CATEGORY'].map({'b': 0, 'e': 1, 't': 2, 'm': 3})
y = newsCorpora['CATEGORY']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=0)
X_valid, X_test, y_valid, y_test = train_test_split(X_test, y_test, test_size=0.5, stratify=y_test,
random_state=0)
X_train.to_csv('train.txt', sep='¥t', index=False, header=None)
X_valid.to_csv('valid.txt', sep='¥t', index=False, header=None)
X_test.to_csv('test.txt', sep='¥t', index=False, header=None)
```

単語ベクトルの和による特徴量

- 学習データ、検証データ、評価データを行列・ベクトルに変換する。例えば、学習データについて、全ての事例 x_i を並べた行列 X と、正解ラベルを並べた行列（ベクトル） Y を作成する。

$$X = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \dots \\ \mathbf{x}_n \end{pmatrix} \in \mathbb{R}^{n \times d}, Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} \in \mathbb{N}^n$$

n : 学習データの事例数

単語ベクトルの和による特徴量

- i 番目の事例の特徴ベクトル \mathbf{x}_i

$$\mathbf{x}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} \text{emb}(w_{i,t})$$

- i 番目の事例は T_i 個の単語列 $(w_{i,1}, w_{i,2} \dots w_{i,T_i})$ から構成され、 $\text{emb}(w_{i,t})$ は単語 w に対応する単語ベクトル

i 番目の事例のラベル y_i

$$y_i = \begin{cases} 0 & (\text{記事 } x_i \text{ が「ビジネス」カテゴリーの場合}) \\ 1 & (\text{記事 } x_i \text{ が「科学技術」カテゴリーの場合}) \\ 2 & (\text{記事 } x_i \text{ が「エンターテインメント」カテゴリーの場合}) \\ 3 & (\text{記事 } x_i \text{ が「健康」カテゴリーの場合}) \end{cases}$$

学習済み単語ベクトル

- <https://drive.google.com/file/d/0B7XkCwpl5KDYNINUTTISS21pQmM/edit?usp=sharing> にアクセスしてダウンロードする。
- ダウンロード後、解凍して実行するPythonファイルと同じ位置におく。

単語ベクトルの和による特徴量

```
import joblib
import numpy as np
import pandas as pd
from gensim.models import KeyedVectors
from tqdm import tqdm
```

```
def culcSwem(row):
    global model
    swem = [model[w] if w in model.vocab else np.zeros(shape=(model.vector_size,)) for w in row['TITLE'].split()]
    swem = np.mean(np.array(swem), axis=0)
    return swem
```

```
X_train = pd.read_table('./train.txt', header=None)
X_valid = pd.read_table('./valid.txt', header=None)
X_test = pd.read_table('./test.txt', header=None)
use_cols = ['TITLE', 'CATEGORY']
n_train = len(X_train)
n_valid = len(X_valid)
n_test = len(X_test)
```

```
importエラーが出る場合
pip install joblib
pip install numpy
pip install pandas
pip install tqdm
```

単語ベクトルの和による特徴量

```
X_train.columns = use_cols
X_valid.columns = use_cols
X_test.columns = use_cols
data = pd.concat([X_train, X_valid, X_test]).reset_index(drop=True)
tqdm.pandas()
model = KeyedVectors.load_word2vec_format('ch07/GoogleNews-vectors-negative300.bin', binary=True)
swemVec = data.progress_apply(culcSwem, axis=1)
X_train = np.array(list(swemVec.values)[:n_train])
X_valid = np.array(list(swemVec.values)[n_train:n_train + n_valid])
X_test = np.array(list(swemVec.values)[n_train + n_valid:])
joblib.dump(X_train, './X_train.joblib')
joblib.dump(X_valid, './X_valid.joblib')
joblib.dump(X_test, './X_test.joblib')
y_data = data['CATEGORY'].map({'b': 0, 'e': 1, 't': 2, 'm': 3})
y_train = y_data.values[:n_train]
y_valid = y_data.values[n_train:n_train + n_valid]
y_test = y_data.values[n_train + n_valid:]
joblib.dump(y_train, './y_train.joblib')
joblib.dump(y_valid, './y_valid.joblib')
joblib.dump(y_test, './y_test.joblib')
```


Pytorch

- 機械学習ライブラリ
- 管理者権限でコマンドプロンプトを開き、以下のコマンドを入力する。

```
pip install torch==1.5.0+cpu torchvision==0.6.0+cpu -f https://download.pytorch.org/whl/torch_stable.html
```

正解率の計測

```
import joblib
import numpy as np
import torch
from torch import nn, optim

X_train = joblib.load('./X_train.joblib')
y_train = joblib.load('./y_train.joblib')
X_train = torch.from_numpy(X_train.astype(np.float32)).clone()
y_train = torch.from_numpy(y_train.astype(np.int64)).clone()

X_test = joblib.load('./X_test.joblib')
y_test = joblib.load('./y_test.joblib')
X_test = torch.from_numpy(X_test.astype(np.float32)).clone()
y_test = torch.from_numpy(y_test.astype(np.int64)).clone()

X = X_train[0:4]
y = y_train[0:4]

net = nn.Linear(X.size()[1], 4)
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.01)
```

importエラーが出る場合
pip install joblib
pip install numpy

正解率の計測

```
losses = []
```

```
for epoc in range(100):
```

```
    optimizer.zero_grad()
```

```
    y_pred = net(X)
```

```
    loss = loss_fn(y_pred, y)
```

```
    loss.backward()
```

```
    optimizer.step()
```

```
    losses.append(loss)
```

```
_, y_pred_train = torch.max(net(X), 1)
```

```
print((y_pred_train == y).sum().item() / len(y))
```

```
_, y_pred_test = torch.max(net(X_test), 1)
```

```
print((y_pred_test == y_test).sum().item() / len(y_test))
```

演習3. 学習回数

前ページのepochを100から別の値に変えて、正解率の変化を確認せよ。