情報工学科　レポート

実験演習記録			判定・指示
	年月日時	共同作業者	
1	2019/06/17		
2			
3			
4			

レポート提出記録		
	提出年月日	期限年月日
初	2019/07/01	2019/07/01
再		

科目名	テーマ担当教員	学年	学期	単位
情報工学実験2	中川先生	3	前期	2

テーマ番号　　テーマ名	学籍番号　　氏　名
パターン認識	16268062 SALIC ERTUGRUL

1. はじめに

本レポートでは，情報工学実験 2 の第 8 回目のレポートとして，パターン認識についての課題に取り組み，作成するプログラムの設計や実装について述べる．

2. 目的

本実験では，パターン認識でよく用いられる「クラスタリング」を，簡単なプログラムの作成を通して理解することを目的とする．

3. 実験方法

本実験では，全ての課題を EDEN において，「**Python 言語**」で行う．

この実験では，k-means 法によってクラスタリングを行う．そのため，まず，いくつかのクラスタを持つ **2 次元特徴のパターン**を考える．そのパターンは〈https://archive.ics.uci.edu/ml/datasets/iris〉（閲覧日：2019 年 7 月 1 日）からダウンロードしたデータを 2 次元に変えたものである．（レポート提出時に添付する）データは txt ファイル（iris4d.txt）に保存されているため，プログラムでは最初にファイルからデータを読み込む処理を行う．そのコードを次に示す．

```
1. import numpy as np
2. X = np.loadtxt("./iris4d.txt")
3. (d, n) = X.shape  # 行数と列数を取得
```

クラスタリングを行う際は mahalanobis 距離を用いる．mahalanobis 距離は式（1）で定義できる．

$$d(x,y) = (x-y)^T \sum (x-y) \qquad (1)$$

式（1）を用いて，mahalanobis 距離を求める関数を定義する．共分散行列はデータ全体の共分散行列だと仮定する．そのコードを次に示す．

```
1. def dis(x, y, cov):  # Mahalanobis 距離の計算
2.     dist = np.dot(np.dot((x-y).T, cov), (x-y))
3.     return dist
```

k-means 法を以下の STEP で行う．

- STEP-1: 有限パターン集合 W をいくつかに分割するかを決める．これを N とし，指定できるようにする．
- STEP-2: N 分割する時，それぞれ代表パターンと思われる$c_1, c_2, c_3, \ldots c_N$を適当に選ぶ．
- STEP-3: 全てのパターン x について，もっとも近い代表パターンc_iを求め，x が属する S_i を決める．
- STEP-4: 各 S_i について，セントロイドを求め，代表パターンをそれに更新する．
- STEP-5: もし全ての代表パターンに変化がなければ終了．
- STEP-6: STEP-3 に戻る．

STEP-1 ではユーザに N を指定してもらう．そのコードは次のように書ける．

```
1. # STEP-1
2. print("N:")
3. N = int(input())
```

また，STEP-2 ではセントロイドの初期値を適当に定義する．今回はデータの最初の N 個のパターンを代表パターンとして決めておく．そのコードを次に示す．

```
1. # STEP-2
2. c = np.zeros((d, N))
3. for i in range(N):
4.     c[:, i] = X[:, i]
5. print("セントロイドの初期値：")
6. print(c)
```

STEP-3 では，全てのパターン x について，もっとも近い代表パターンc_iを求め，x が属する S_i を決める．そこでマハラノビス距離を用いて，x が一番近い代表パターンc_iを求める．そのコードを次に示す．

```
1. S = []
2. for i in range(N):
3.     S.append([])  # N 個の行列を要素とするリスト
4. d_mhlnbs = [0]*N
5.  for i in range(n):
6.     for j in range(N):
7.         d_mhlnbs[j] = dis(X[:, i], c[:, j], cov)  #マハラノビス距離
8.     min_d = d_mhlnbs.index(min(d_mhlnbs))    # 一番近い代表のクラスを取得
9.     S[min_d].append(X[:, i])  # 一番近い代表のクラスに入れる
```

STEP-4 では，各 S_i について，セントロイドを求め，代表パターンをそれに更新する．セントロイドを計算する際は，S に所属している x から S の各要素への距離の合計を求め，その合計値が一番小さくなるような x を計算する．そのコードを次に示す．

```
# セントロイドを更新
for i in range(N):
    for j in range((len(S[i]))):
        summ = 0
        for k in range((len(S[i]))):
            summ = summ + dis(S[i][j], S[i][k], cov)
        if j == 0:
            min_value = sum
            min_index = j
         elif summ < min_value:
            min_value = sum
            min_index = j
     c[:, i] = S[i][min_index]
print("新しいセントロイド：")
print(c)
```

STEP-5 では，全ての代表パターンに変化がなければプログラムを終了する．そこで前のセントロイドを保存しておく control 変数を用いる．もし更新したセントロイドが control と同じであればプログラムを終了する．でなければまた処理を繰り返す．そのコードを次に示す．

```
# 前のセントロイドと更新したものが同じであれば終了
for i in range(d):
    for j in range(N):
       if c[i, j] != control[i, j]:
            flag = 1
```

もし control（前のセントロイド）と更新したセントロイドが同じであれば flag が 0 のままになり，プログラムが終了する．一方，同じでなければ flag が経ち，処理が繰り返される．つまり STEP-3 に戻る．

また，クラスタリングしたデータをグラフで表示したい．そのために次にコードを用いる．

```
# グラフの用意
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
color_list = ["r", "g", "b", "c", "m", "y", "k", "w"]
for i in range(N):
    for j in range(len(S[i])):
        ax.scatter(S[i][j][0], S[i][j][1], c=color_list[i %
len(color_list)], marker='.', label='group1')
fig.show()
```

4. 実装結果

本節では，節 3 で説明したプログラムの実装結果について述べる．

プログラムを実装し，N＝3 にした結果を図 1 に示す．また，その時出力されたグラフを図 2 に示す．

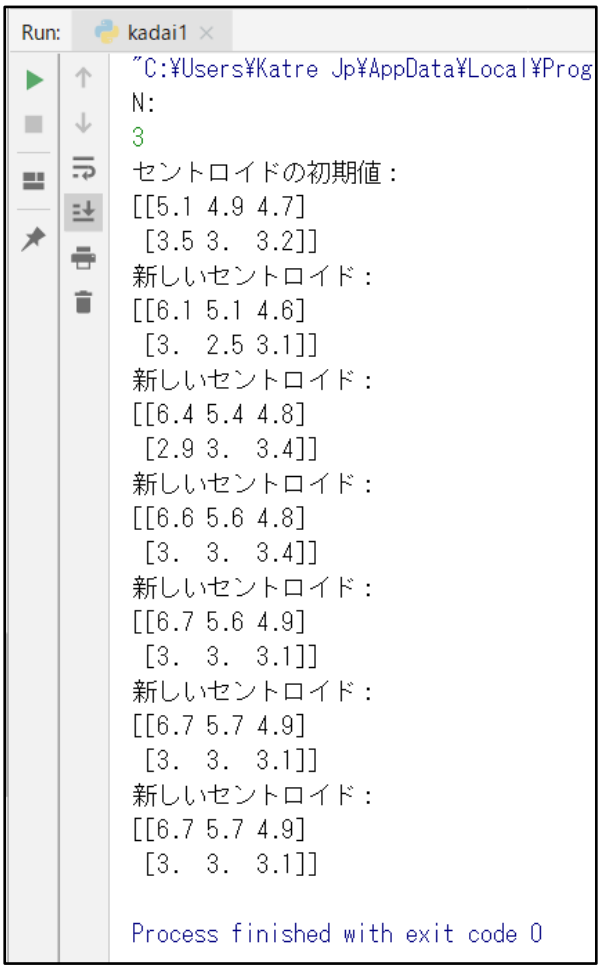

図 1　クラスタリング実装結果

図 1 ではセントロイドの状況や変化が見られる．図 2 ではクラスタリングの結果が確認できる．

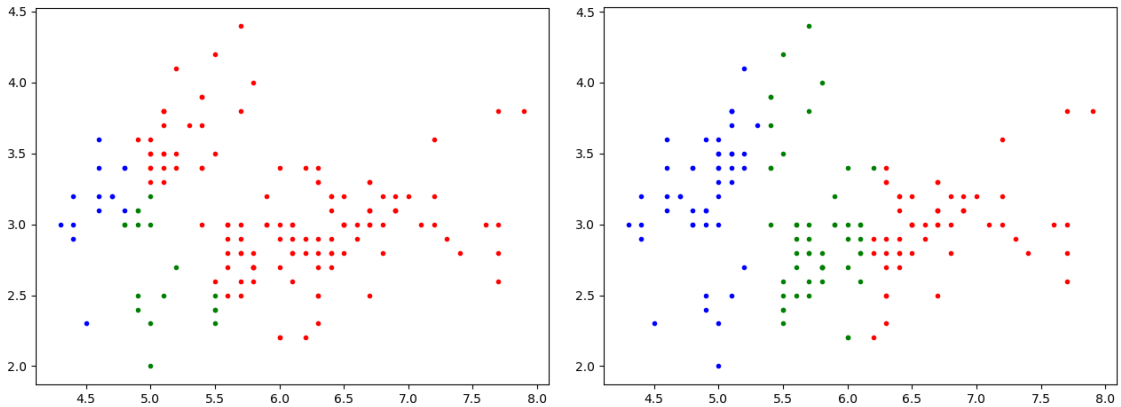

図 2　N=3 の時のクラスタリング結果

また，N=5 の時のクラスタリングを図 3 に示す．

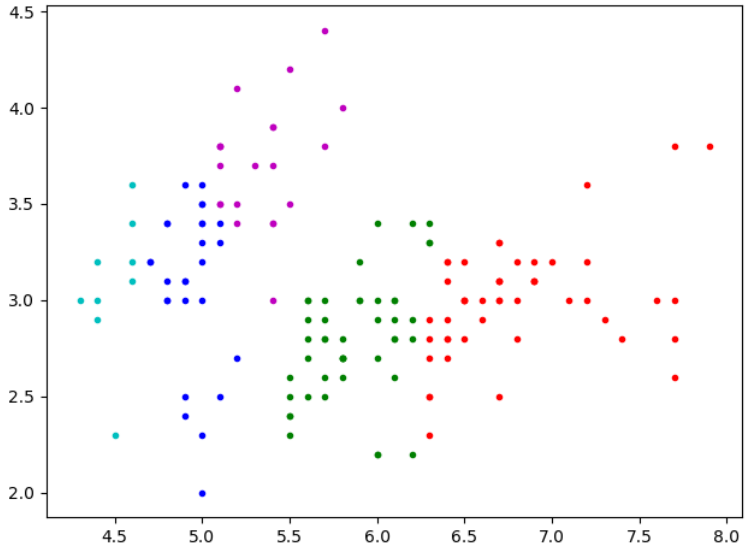

図 3　N=5 の時のクラスタリング結果

最後に，N=3 とし，セントロイドの初期値を変えてみた時の結果を出してみる．今までは，セントロイドの初期値をデータの最初の N 個の要素としていたが，それを最後の N 個の要素としてみる．その時の結果を図 4 と図 5 に示す．

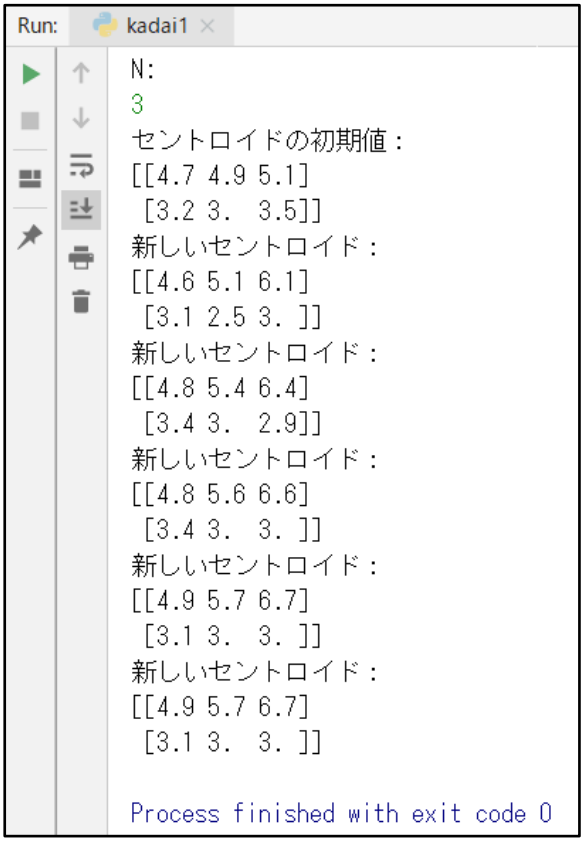

図 4　セントロイドの初期値を変えた時の結果（N=3）

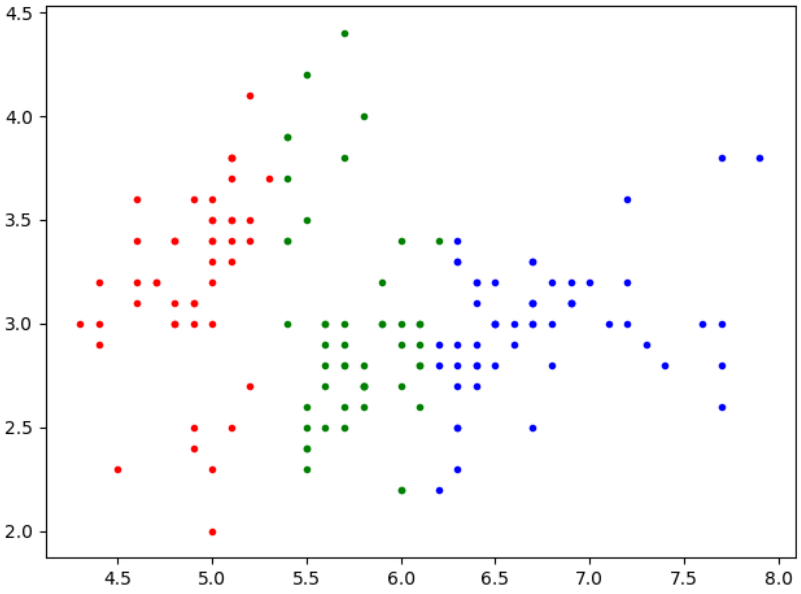

図 5　セントロイドの初期値を変えた時の結果（N=3）

5. 考察

本節では，節 4 の結果を基に，実験に関して考察を行うことを目的とする．

本実験では，k-means 法を用いてクラスタリングを行った．図 2（右）より，クラスタリングのプログラムが正常に動いていると考えられる．そこでセントロイドの初期値をパターン集合から適当に選んだので，図 2（左）が，まだ完璧な結果になっていないことも分かる．図 1 ではセントロイドの変化が見られる．セントロイドとしてプログラムが適切としているパターンの行列は

$$\begin{pmatrix} 6.7 & 5.7 & 4.9 \\ 3.0 & 3.0 & 3.1 \end{pmatrix}$$

だと分かる．その行列に格納されているセントロイドの値が，セントロイドの初期値を変えてプログラムを実行した時の結果と，つまり最終セントロイドの値と同じであることは，注意すべきだと考えられる．したがって，プログラムは正常に動いていることがこの点でも確認できている．またそれは，図 2（右）と図 5 のグラフが同じであることからも確認できる．

本実験では，また N=5 とし，クラスタリングを行ってみた．その結果が図 3 に示されている．図 3 より，クラス数が変わってもクラスタリングができることが確認できる．

本実験で使用したデータは 3 種類の花のデータである．これから，そのデータの本当のクラスについて紹介する．データの本当のクラスは図 6 に示す．

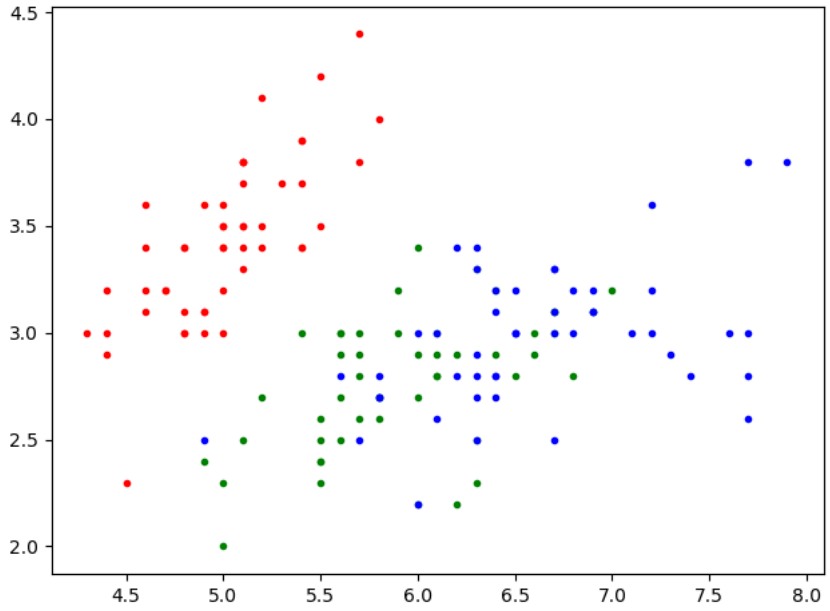

図 6　本実験で用いたデータの本当のクラスのグラフ

図 5 の N=3 の時のプログラム実装結果と図 6 の本当のクラスを比較してみたところ，プログラムが行ったクラスタリングははっきりとクラスを分けることができていることに気づいた．データの本当のクラスでは (図 6) 緑色のグループと青色のグループがはっきりと分かれていないことがわかるが，図 5 でははっきりと分けられている．よって，プログラムの分類精度が高いと思われ，しかしそれにしても，実際と誤差が生じていることがわかる．

6. おわりに

本実験では k-means 法を用いてクラスタリングを行った．それは情報工学科の学生である私にとって，とても貴重な経験・知識であった．なぜそうであるかを，このレポートを書く時に起こった出来事で説明させて頂きたい．

本レポートを書いているところに，東京の他の大学で国際関係について学んでいる友達が来た．私が k-means 法についてプログラミングをしているのを見て，「あっ！，これ最近使った方法．情報工学科の人は偉そうに IT 系の人だけやっているように思ってるかもしれないけど，僕たちもこういうのたまにやってるよ」と冗談付きで言った．私は同じ学年の文系の友達が，k-means 法を使って分類等を行ったことにびっくりしつつ，理系しかも情報系の勉強をしている私が今まで k-means 法の使い方を良く知らなかったことを恥ずかしく思った．本実験で非常に重要な知識を身についたと思う．(ありがとうございます..)

7. 参考文献

[1] 中川 正樹，東京農工大学，情報工学科，情報工学実験 2，2019 年度第 8 回講義資料

[2] <https://archive.ics.uci.edu/ml/datasets/iris> (閲覧日：2019 年 7 月 1 日)

[3] <https://octave.org/doc/v4.2.1/Two_002dDimensional-Plots.html>，閲覧日：2019 年 6 月 30 日

[4] <https://pythondatascience.plavox.info/matplotlib/pycharm >，閲覧日：2019 年 6 月 30 日