K-means Algorithm and Application in Object Segmentation using Pygame and Sklearn

氏名：NGUYEN HUU VU

千葉商科大学―2 年生

Email:
nguyenhuuvu286@gmail.com

I.導入

Kmeans algorithrm とは 基礎的なクラス分類アルゴリズムのひとつで、Unsupervised learning として知られています。 Kmeans アルゴリズムの中では、ラベルがないデータから分類するアルゴリズムです。

クラスリングには社会問題や経営戦略に多くのアプリケーションがあります。例では、オンラインショッピングからのデータをベクトルとして扱えば、顧客のセグメント化のシステムを作成できます。それから、顧客を感謝するために、それぞれの顧客のセグメントにおすすめ商品、お得なサービスパターンなどを齎します。

簡単な例は「x 、 y 」を ベクトルとしてすれば、座標 0XY 上でその点を決定できます。仮に、K = 4 。Kmeans アルゴリズムで図 1 のように 4 つのクラスを見つけられます。

図 1 。Kmean Input and Output with K = 4

II.数学分析

１．　記号

数学分析に行く前に One-Hot 表現について説明します。One-Hot 表現とは「１、０、０」、「０，１、０」のように、一つの部分が１で、残り、全部「０」で表せます

今回では One-Hot 表現扱ってそれぞれのラベル y を表します。

２．損失関数(Lost function)

仮に、N のデータがあるとし X = [x1, x2, x3,... ,xN]∈ RDxN と K（K<N） が求めている中心点数（Clusters）です.最後に m1,m2,m3...mK はそれぞれの中心点の位置です。それから , クラスターk に割り当てられたデータポイント xi の偏差は（x_i-m_K）です。

x_i 値ごとに、ラベルベクトル y_i の価値を取得できます。仮に、K＝２、クラスターグループは One-Hot で「１、０」、「０、１」のように表されます。各ラベルベクトルには、1 桁の数字 1 のみが含まれますので,$y_{ik} = 1$ と $y_{ij} = 0$ を取得できます。ですから、以下の式のように作成できます。

$$\sum_{K=1}^{K} yik = 1$$

目標は、クラスター内の二乗和（分散）を最小化することで、偏差二乗だと知られています。以上の仮で、K＝２、 y 1＝「１、０」、 y 2＝「０、１」の結果を取得できます。ですから、x_i から m_K の偏差二乗が以下のよって決定されています。

$$\|x_i - m_k\|^2 = yik\|x_i - m_k\|^2$$

また、以上に説明したように、$y_{ik} = 1$ と $y_{ij} = 0$ と組み合わせて

$$yik\|x_i - m_k\|^2 = \sum_{j=1}^{k} yij\left\|x_i - m_j\right\|^2$$

を取得できます。この問題では y と mk の二つの変数を見つけなればいけないので、A 式として、記述できます。

$$f(Y,M)=\sum_{i=1}^{N}\sum_{j=1}^{k}yij\left\|x_i-m_j\right\|^2\text{(A)}$$

最適化問題をするために、損失値が最小値であれば、よいです。言い換えると、ラベル y とクラスターMの最大値を与える引数（$argmin_{Y,M}$）を見つけばいいです。

$$\text{Y,M = argmin}_{Y,M}\sum_{i=1}^{N}\sum_{j=1}^{k}yij\left\|x_i-m_j\right\|^2\text{(A) (条件： } y_{ij}\in\{0,1\}\text{)}$$

3. 最適化問題解決

A 式では、各データのラベル y 、クラスターの位置 m の変数があります。従たって、最適化問題を解決するために、一つの引数を固定して、残りの引数の最適化をします。

最初、クラスターの位置 M を固定し、損失関数が最小値になるように各データのラベルを見つける問題になります。ですから、A 数式は以下のように狭められます。

$$y_i = argmin_{yi}\ \sum_{j=1}^{K}yij\left\|x_i-m_j\right\|^2$$

しかし、以上に説明したように、One-Hot 表示で yi = 1 の数は 1 つだけですから、

$$j = argmin_j\ \left\|x_i-m_j\right\|^2$$

$\left\|x_i-m_j\right\|^2$は xi から mj までの二乗和距離なので、各点 xi は、中心が最も近いクラスターに属すことがわかりました。したがって、各データのラベルを見つけることができます。

次に、クラスターの位置 Y を固定し、クラスターをみつけます。A 数式は以下のように狭められます：

$$m_j= argmin_{mj}\sum_{i=1}^{N}yij\ \left\|x_i-m_j\right\|^2\text{(C)}$$

C 式は凸関数で i [1,N]ごとに異なる値を取ります.ですから,偏導関数を使えば、この問題を最適化できます。仮に、f (mj) = $\sum_{i=1}^{N} yij \left\| x_i - m_j \right\|^2$(C)を記入直し、f(mj)の偏導は以下のように記述されます：

$$\frac{\partial y(m_j)}{\partial m_j} = 2\sum_{i=1}^{N} yij \ (x_i - m_j) = 0$$

$$->2xi \sum_{i=1}^{N} yij = 2mj \sum_{i=1}^{N} yij$$

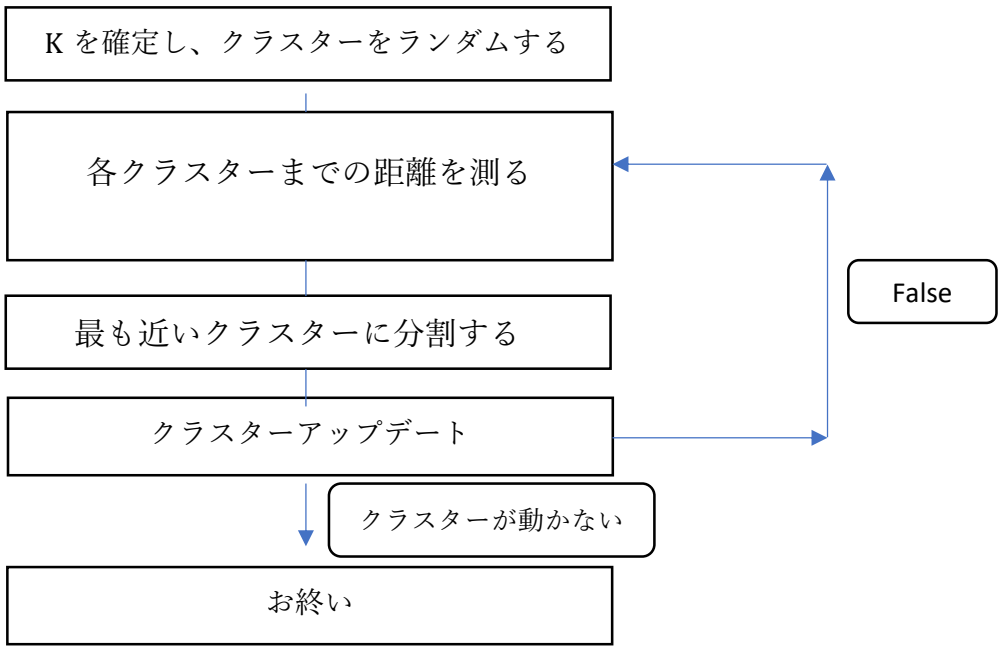
$$-> \quad m_j = \frac{\sum_{i=1}^{N} y_{ij} x_i}{\sum_{i=i}^{N} y_{ij}} \quad (*)$$

データポイントが m_j に属するときに、$y_{ij} = 1$ ですから、＊式により、結果は m_j に属する各データ x_i の平均です。分子は m_j に属する各データ x_i の和で、分母は m_j に属する各データ x_i の数です。

Ⅲ。アルゴリズム概略

1. K を決め、クラスターをランダムする。
2. 各データが各クラスターまでの距離を測る
3. 最も近いクラスターに分割する
4. m_j に属する各データ xi の平均を計算して、新しいクラスターを見つけてから、クラスターをアップデートする
5. ステップ 2 に戻る

止める条件：クラスターが変化しないまで繰り返し



IV:Pygame で Kmeans アルゴリズム表示。

アルゴリズム概略によって、Kmeans アルゴリズムを表示できる簡単な例を作成することができます。今回のレポートでは Pygame libray の Python プログラミングを扱って、ある程度アルゴリズムの効率を確認します。

1．まず、pygame などをインポートし、初期化のアルゴリズムの画面を設定します。

```
import pygame
from random import randint
import math
```

```
def crete_text_render(string):
    return font.render(string, True, WHITE)
```

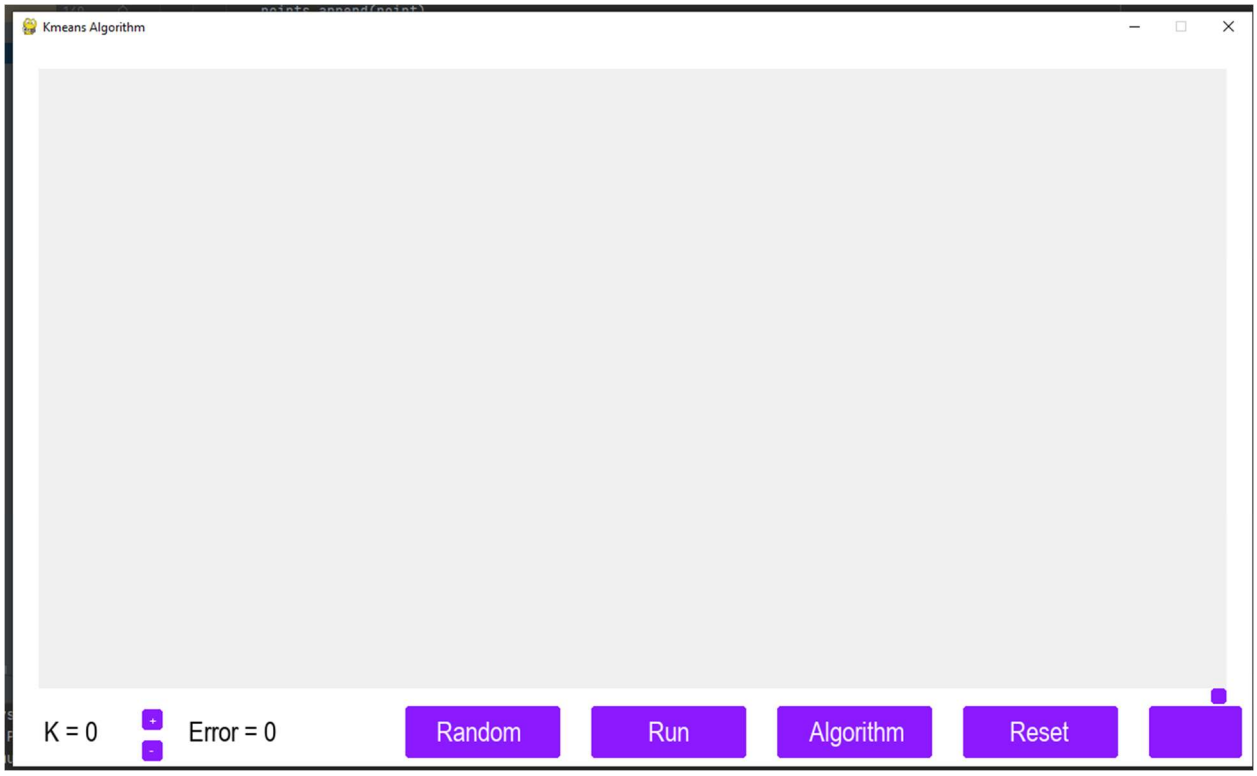
```
pygame.init()
screen = pygame.display.set_mode((1200,700))
pygame.display.set_caption("Kmeans Algorithm")


COLORS = [RED,BLUE,YELLOW,PURPLE,SKY,ORANGE,GRAPE,GRASS]
font = pygame.font.SysFont('sans', 27)
font_plus_mup = pygame.font.SysFont('sans',15)
text_random,text_run,text_alg,text_reset =
crete_text_render("Random"),crete_text_render("Run"),\

crete_text_render("Algorithm"),crete_text_render("Reset")
text_plus = font_plus_mup.render('+',True, WHITE)
text_mup = font_plus_mup.render('-',True, WHITE)
```

次でアルゴリズムのボタンなどのセットを設定してから、画面に塗ります。

```
while running:
    clock.tick(60)
    screen.fill(WHITE)
    mouse_x, mouse_y = pygame.mouse.get_pos()

    #draw panel
    pygame.draw.rect(screen, (240, 240, 240), (25, 25, 1150, 600))

    #k value
    k_value = font.render("K = "+ str(K), True,BLACK)
    screen.blit(k_value,(30,650))

    # Button +
    pygame.draw.rect(screen, GREEN, (125,645,20,20),border_radius=3)
    screen.blit(text_plus, (132,646))
```

```
 pygame.display.flip()
pygame.quit()
```

それから、以下のイメージのように設定できます：



それに、このコードの中では、ボタン処理のことを仕込みます。

```
for event in pygame.event.get():
```

２．　引数初期設定 とアルゴリズム処理

最初、引数の初期を設定します。

```
K = 0
ERORR = 0
points = []
labels = []
clusters = []
```

つぎに、必要な関数：

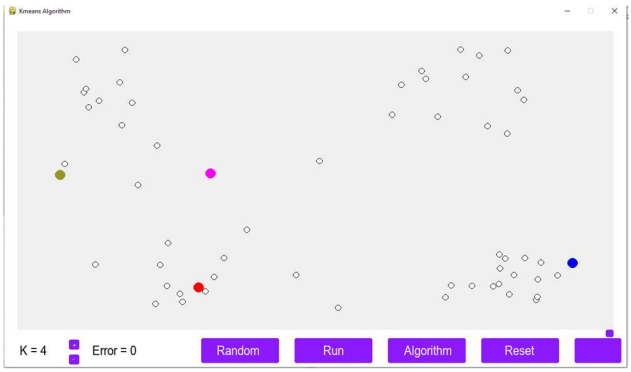
```
def distance(p1,p2):
    return math.sqrt((p1[0]-p2[0])**2 + (p1[1]-p2[1])**2)
```

この関数で空間内の 2 つのデータポイントの距離を計算する関数です

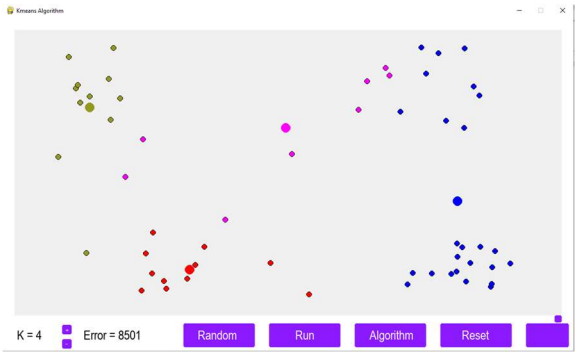
```
if 560 < mouse_x < 710 and 642 < mouse_y < 692:
    labels = []

    if clusters ==[]:
        continue
    #Assign points to closest cluster
    for p in points:
        distances_to_cluster = []
        for c in clusters:
            dis = distance(p,c)
            distances_to_cluster.append(dis)
        label = distances_to_cluster.index(min(distances_to_cluster))
        labels.append(label)

    #Update Cluseter
    for i in range(K):
        sumx = 0
        sumy = 0
        count = 0
        for j in range(len(points)):
            if labels[j] == i:
                sumx += points[j][0]
                sumy += points[j][1]
                count +=1
            if count != 0:
                new_cluster_x = int(sumx / count)
                new_cluster_y = int(sumy / count)
                clusters[i] = [new_cluster_x, new_cluster_y]
```

このコードでアルゴリズム概略の中では、ステップ 2 とステップ 3 の仕込みです。

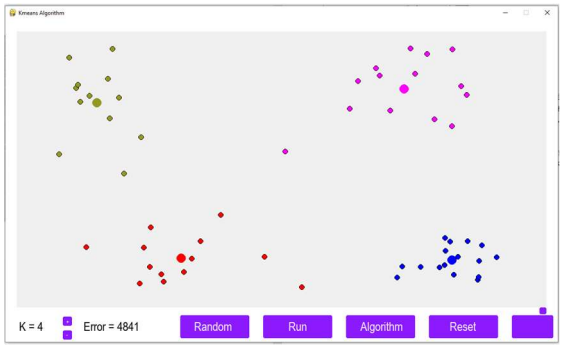
データ初期化



クラスターランダム



初めての仕組み



クラスターが動けない結果

このアプリケーションでは ERORR という文字を記入し、せっていしました。それはクラスターまでのクラスターに応じた点の距離の和です。各データまで毎回クラスターをランダムすれば、異なる ERORR 値を取得し、値が小さいほど、精度が高くなります。

V: 議論

Kmeans アルゴリズムではクラスタ数 k は最初に所与のものとして定めるため、クラスター数を選ぶことは問題になりますが、Elbow method を利用すれば、最適なクラスタ数を選べられます。しかし実際生活問題の中では、データには多くの次元があるので、K の値を確定できない場合もあります。

また、最初のクラスターのランダムな割り振りで、最後の結果を出す時間に影響を齎しますので、何回も繰り返して行って最良の結果を採用することが望ましいといえます。しかし、k-means++法でこの問題を解消できます。

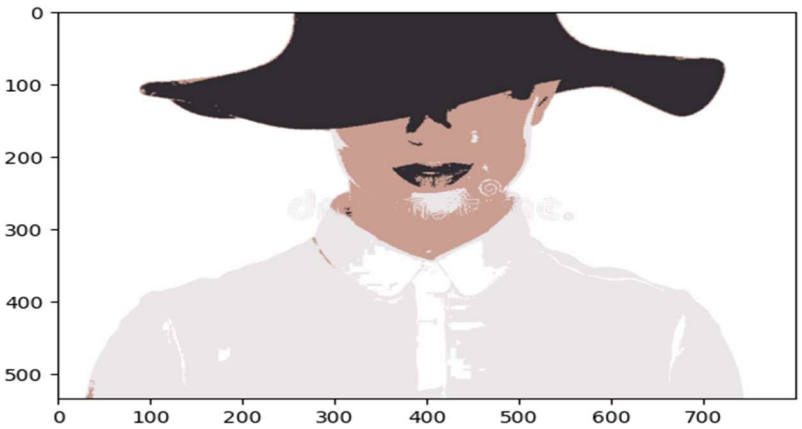
VI. オブジェクトセグメンテーション(Object segmentation)

Kmeans アルゴリズムのメリットにより、ある程度実際問題を解決することができます。その一つはオブジェクトセグメンテーション（Object segmentation）と知られます。今回のレポートでは、Sklearn というライブラリをインポートし、詳しく Kmeans アルゴリズムを表示するようにしました。

まず、一枚の写真は多くのピクセルで構成されていて、 カラー写真に対して 1 ピクセルは 3 つの RBG 値（0～255）で表されます（例えば、「255, 125, 126」）。 Kmean アルゴリズムを使用して K（cluster）を見つけ、オブジェクトを分離できます.

写真 1

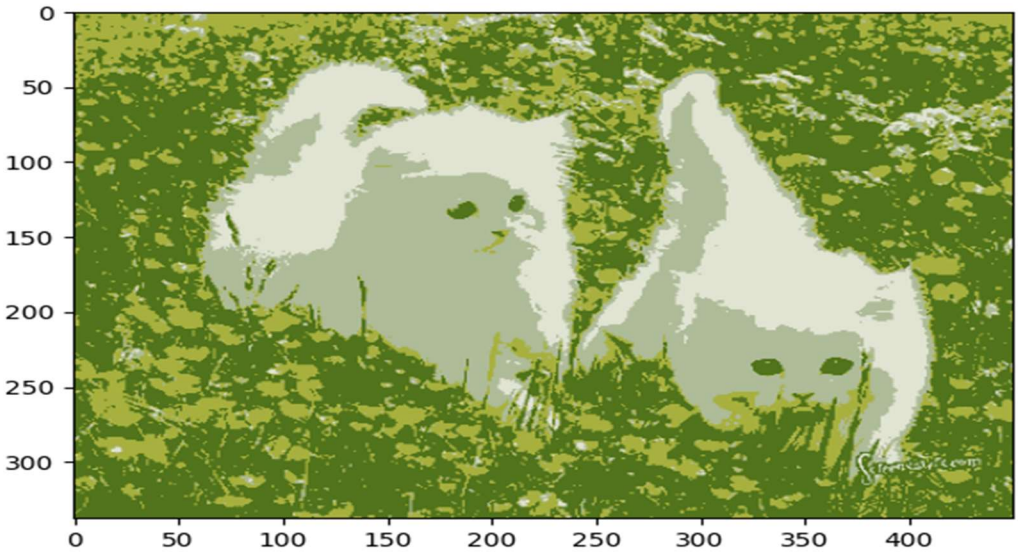
仮に、写真 1 の写真があって、顔を分離できるアルゴリズムが必要だとします。まず、この写真を見ると、主な 4 つの色を区別することができます。それは黒、ピンク、赤、白です。4 つのクラスターを見つけてから、応じたクラスターで各ピクセルを変えて、以下の結果ができました：



また、写真２もあって、二つの猫を分離できるアルゴリズムが必要だとします。

写真２

アルゴリズムを実行してから、以下のような結果が出られます：



Kmeans アルゴリズムによって、二つの例から望ましい結果を得られます。

VII.結論

Kmeans アルゴリズムは、分類と機械学習の問題のための単純なアルゴリズムです。比較的高い精度で実際に単純なクラスタリングの問題を解決するのに役立ちます。しかし、最初のクラスターのランダムな割り振りで、最後の結果を出す時間に影響を齎し、k（クラスター）を決定できない問題に適用するのは困難です。本レポートでは、アルゴリズムの基本のみを紹介しましたが。今後、最初のランダムクラスターについていきたいと思います。

VIII. 謝辞

VU KHAC TIEP さんには機械学習とアルゴリズムブログを提供してもらって、心より心より感謝申し上げます

Full Socre Code :

https://github.com/ngvu-neko/KemansObjectSegmentation/blob/main/main.py

参考文献：

[1] https://ja.wikipedia.org/wiki/One-hot