به نام خالق رنگین کمان

ستاره باباجانی – 99521109

سوال 1: سیستم کنترل فازی طراحی شده به شرح زیر است:

- صدا زدن کتابخانه های مورد نیاز:

Import Libraries

```
!pip install scikit-fuzzy
import gym
import math
import skfuzzy as fuzz
import matplotlib.pyplot as plt
import numpy as np
from skfuzzy import control as ctrl
```

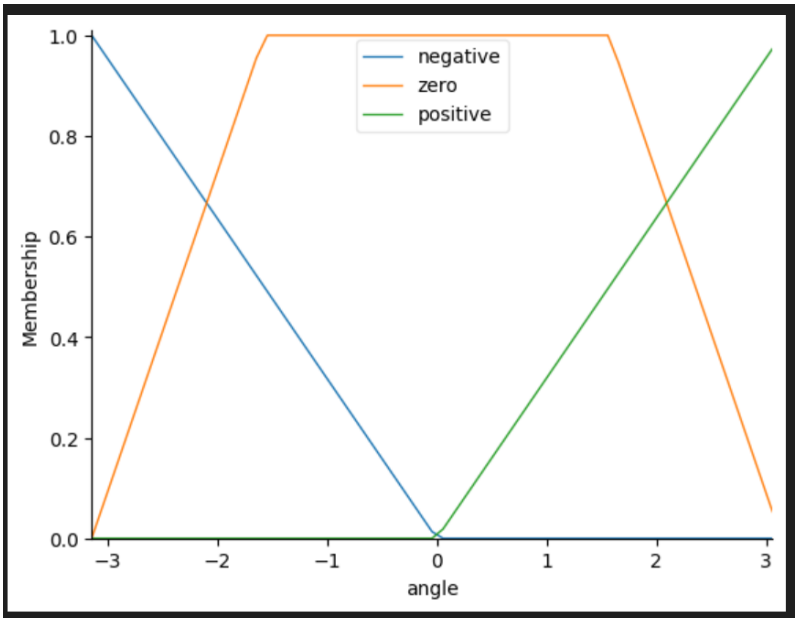
- نشان دادن محیط گرافیکی:

Showing the environment

```
env = gym.make('Pendulum-v1', render_mode='human') # making the env
observation = env.reset() # for starting at the initial state

for i in range(500):
    env.render()
    action = env.action_space.sample()
    step_result = env.step(action)
    observation, reward, done, info = step_result[:4]
    if done:
        break

env.close()
```

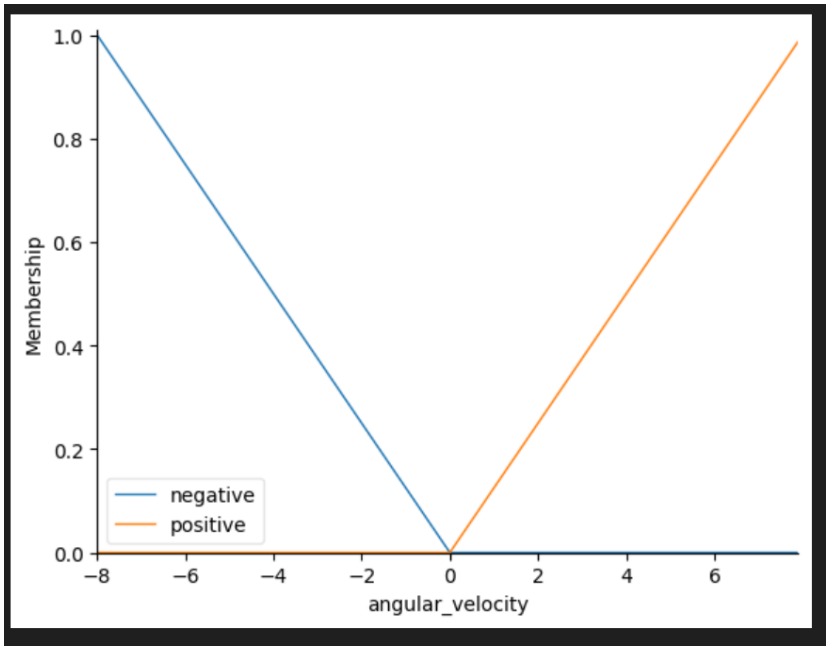
- تعریف متغیرهای زبانی خواسته شده(زاویه، سرعت زاویه ای و گشتاور):
همان طور که در کد ها مشاهده میشود، هرکدام از متغیرها را بازه بندی کردیم و سپس نمودار آن را برای درک بهتر رسم کردیم:
  - زاویه:

```
# angle
angle = ctrl.Antecedent(np.arange(-np.pi, np.pi, 0.1), 'angle')
angle['negative'] = fuzz.trimf(angle.universe, [-np.pi, -np.pi, 0])
angle['zero'] = fuzz.trapmf(angle.universe, [-np.pi, -np.pi / 2, np.pi / 2, np.pi])
angle['positive'] = fuzz.trimf(angle.universe, [0, np.pi, np.pi])

angle.view()
```



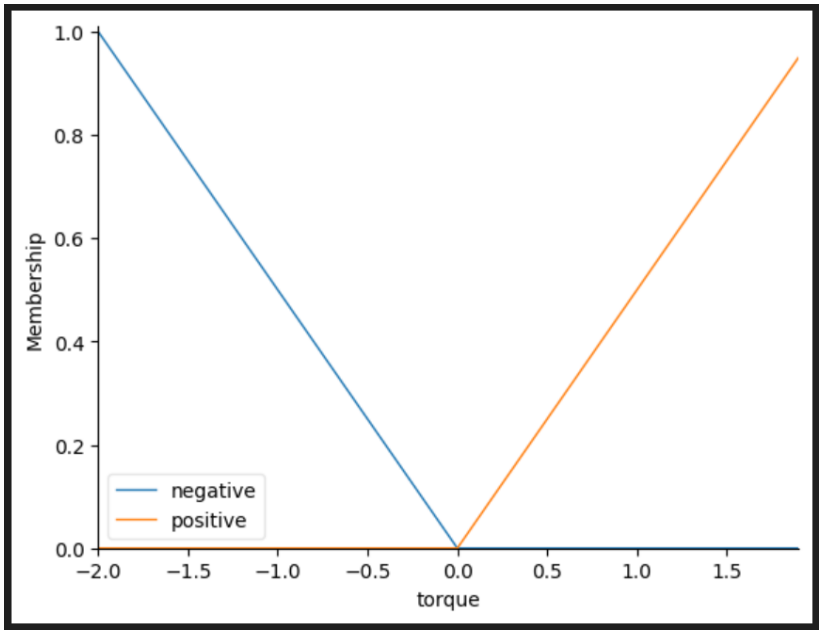
  - سرعت زاویه ای:

```
# angle velocity
angular_velocity = ctrl.Antecedent(np.arange(-8, 8, 0.1), 'angular_velocity')
angular_velocity['negative'] = fuzz.trimf(angular_velocity.universe, [-8, -8, 0])
angular_velocity['positive'] = fuzz.trimf(angular_velocity.universe, [0, 8, 8])

angular_velocity.view()
```



○ گشتاور:

```python
# torque(output)
torque = ctrl.Consequent(np.arange(-2, 2, 0.1), 'torque')
torque['negative'] = fuzz.trimf(torque.universe, [-2, -2, 0])
torque['positive'] = fuzz.trimf(torque.universe, [0, 2, 2])

torque.view()
```



- تعریف کردن قوانین و ساخت control system و simulator: این قوانین به شرح زیر هستند:
  - اگر زاویه مثبت و سرعت زاویه ای مثبت باشد، گشتاور مثبت می باشد.
  - اگر زاویه مثبت و سرعت زاویه ای منفی باشد، گشتاور منفی می باشد.
  - اگر زاویه منفی و سرعت زاویه ای مثبت باشد، گشتاور مثبت می باشد.
  - اگر زاویه منفی و سرعت زاویه ای منفی باشد، گشتاور منفی می باشد.
  - اگر زاویه صفر و سرعت زاویه ای مثبت باشد، گشتاور منفی می باشد.
  - اگر زاویه صفر و سرعت زاویه ای منفی باشد، گشتاور مثبت می باشد.

```
rule1 = ctrl.Rule(antecedent=(angle['positive'] & angular_velocity['negative']), consequent=torque['negative'])
rule2 = ctrl.Rule(antecedent=(angle['positive'] & angular_velocity['positive']), consequent=torque['positive'])
rule3 = ctrl.Rule(antecedent=(angle['negative'] & angular_velocity['positive']), consequent=torque['positive'])
rule4 = ctrl.Rule(antecedent=(angle['negative'] & angular_velocity['negative']), consequent=torque['negative'])
rule5 = ctrl.Rule(antecedent=(angle['zero'] & angular_velocity['positive']), consequent=torque['negative'])
rule6 = ctrl.Rule(antecedent=(angle['zero'] & angular_velocity['negative']), consequent=torque['positive'])
```

```
controller = ctrl.ControlSystem(rules=[rule1, rule2, rule3, rule4, rule5, rule6])
simulator = ctrl.ControlSystemSimulation(controller)
```

- تعریف تابع برای محاسبه زاویه: این تابع x, y را بعنوان ورودی گرفته و زاویه متناظر با (x,y) را محاسبه میکند(واحد آن رادیان است).

ابتدا در این تابع برای محاسبه زاویه، تانژانت معکوس x/y را میگیرد و سپس با استفاده از قوانین تعریف شده، زاویه را اصلاح می کند.(اگر y منفی و x مثبت باشد، به زاویه پی تا اضافه می شود. همچنین اگر y و x هر دو منفی باشند، از آن پی تا کم می شود)

```
def angle_finder(x, y):
    radian_degree = math.atan(x / y)
    if y < 0 and x > 0:
        radian_degree += np.pi
    if y < 0 and x < 0:
        radian_degree -= np.pi
```

در نهایت، زاویه با دو شرط چک میشود تا بین منفی پی دوم و پی دوم باشد:

```
result = 0
if -np.pi / 2 <= radian_degree:
    result = radian_degree - np.pi / 2
if -np.pi <= radian_degree <= -np.pi / 2:
    result = radian_degree + 3 * np.pi / 2
return result
```

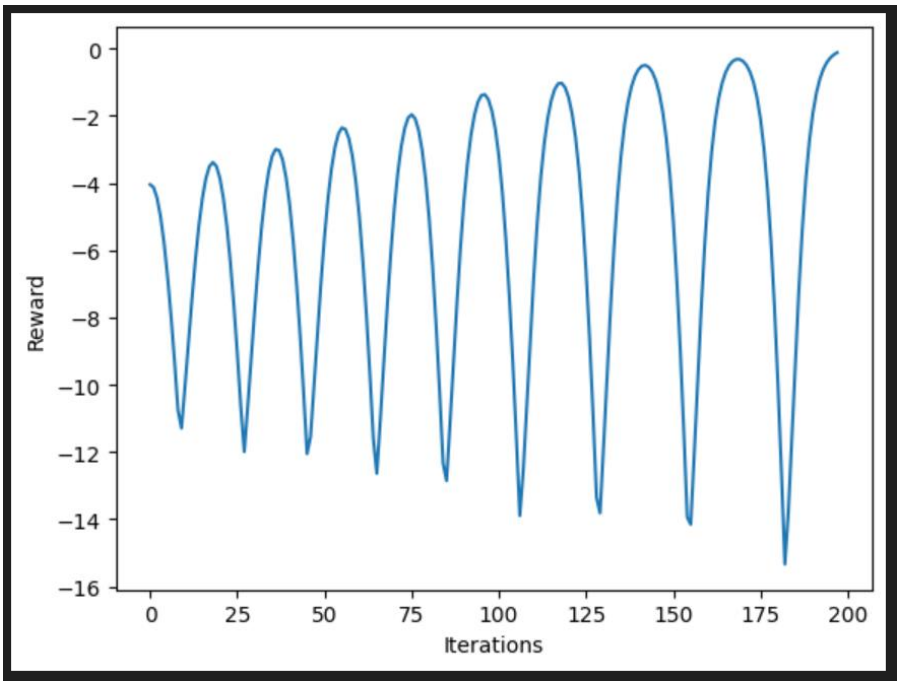
- Compute کردن: اگر pendulum از نقطه بالا فاصله داشته باشد، گشتاور به سمت جهت حرکت آن اعمال می شود تا سرعت افزایش یابد و همپنین به هنگام نزدیک شدن به اوج، گشتاور به جهت مخالف حرکت وارد می شود تا سرعت کاهش یابد و pendulum در حالت ثابت باقی بماند. (فرض کردیم اگر مختصات x را به بالای 0.99 برسد و قدر مطلق سرعت هم کمتر از 1.5 باشد، مسئله را حل شده است):

```python
env = gym.make('Pendulum-v1', render_mode='human') # making the env
observation = env.reset() # for starting at the initial state
rewards = []

for i in range(500):
    x = observation[0]
    y = observation[1]
    # give input to fuzzy lingustic variable:
    simulator.input['angle'] = angle_finder(x , y)
    simulator.input['angular_velocity'] = observation[2]

    simulator.compute() # compute result using fuzzy:
    decision = simulator.output['torque']
    observation, reward, terminated, info = env.step([decision])
    rewards.append(reward)
    if observation[0] >= 0.99 and math.fabs(observation[2]) <= 1.5:
        terminated = True

    env.render()
    # when is done:
    if terminated:
        print(f'we win in {i} iterations!')
        break

env.close()
```

```
we win in 197 iterations!
```

- در نهایت نمودار reward بر اساس iteration را رسم کردیم که همان طور که مشاهده میشود میزان پاداش دریافتی توسط مدل به شکل نوسانی تغییر میکند(چون ما سعی داریم نیروی مورد نیاز برای ایجاد حرکت پویا در پاندول را با تحریک نوسانی آن فراهم کنیم و این باعث فاصله گرفتن مدل بصورت نوسانی از اوج میشود. همچنین این تناوب در پاداش ها، نتیجه ای طبیعی از این تلاش برای ایجاد حرکت

دینامیک در مدل است).



سوال 2: الف) الگوریتم خوشه بندی C-means برای تقسیم بندی مجموعه داده ها به خوشه ها استفاده می شود. فازی C-means (FCM) توسعه ای از K-means است که امکان خوشه بندی فازی را فراهم می کند، جایی که نقاط داده می توانند به خوشه های متعدد با درجات مختلف عضویت تعلق داشته باشند. FCM از خوشه بندی نرم و K-Means از خوشه بندی سخت استفاده میکند بطوریکه در دومی هر داده به دقیقا یک خوشه تعلق گرفته و تخمین مرکز خوشه ها با استفاده از میانگین داده های هر خوشه انجام می شود. این باعث می شود که به داده های پرت و نویز حساس باشد. درصورتیکه درFCM، هر داده به هر خوشه با یک درصد عضویت تعلق می گیرد(که بین 0 و 1 است). که این درصد عضویت نشان میدهد که هر داده با چه احتمالی به هر خوشه تعلق دارد.

- K-Means:

1. خوشه بندی سخت: هر نقطه داده را به یک خوشه منفرد به طور واضح اختصاص می دهد.

2. تابع هدف: مجموع فواصل مجذور بین نقاط داده و مرکز خوشه آنها را به حداقل می رساند.

3. عضویت: نقاط داده به عنوان متعلق به یک خوشه یا خوشه دیگر تلقی می شوند.

4. Cluster Centroids: به عنوان میانگین تمام نقاط داده اختصاص داده شده به آن خوشه محاسبه می شود.

- FCM:

1. خوشه بندی فازی: درجاتی از عضویت را به هر نقطه داده برای همه خوشه ها بین 0 و 1 اختصاص می دهد که امکان عضویت جزئی را فراهم می کند.

2. عملکرد هدف: مجموع وزنی فواصل مجذور بین نقاط داده و مرکزهای خوشه را با در نظر گرفتن عضویت به حداقل می رساند.

3. عضویت: نقاط داده می توانند به چندین خوشه به طور همزمان با درجات مختلف عضویت تعلق داشته باشند که نشان دهنده عدم قطعیت در طبقه بندی است.

4.Cluster Centroids : به عنوان میانگین وزنی محاسبه می شود، که در آن هر نقطه داده بر اساس عضویت آن در خوشه به مرکز کمک می کند.

- تفاوت ها:

1. درجه عضویت: FCM درجاتی از عضویت را امکان پذیر می کند، در حالی که K-means یک نقطه داده را صرفاً به یک خوشه اختصاص می دهد.

2. محاسبهCentroid : درFCM ، centroidها به عنوان میانگین وزنی با در نظر گرفتن درجه عضویت محاسبه می شوند، در حالی که K-means از میانگین ساده استفاده می کند.

3. استحکام نسبت بهOutliers: FCM می‌تواند با توجه به عضویت‌های جزئی بهتر با موارد پرت برخورد کند، در حالی که K-meansممکن است به دلیل تخصیص سختش بیشتر تحت تأثیر عوامل پرت قرار گیرد.

4. حساسیت به پارامترها: FCM شامل یک پارامتر اضافی (ضریب فازی) برای کنترل درجه مبهم بودن در خوشه بندی است ولی K-means چنین پارامتری ندارد.

ب) حال مراحل خواسته شده را به ترتیب برای داده اول و داده دوم انجام میدهیم:

1. خواندن فایل:

Reading File

```
1 df = pd.read_csv("/content/data1.csv")
2 print(df)
```

```
        X      Y  Class
0    5.50   7.00      1
1    9.40  13.00      1
2    6.00   6.80      1
3   12.50  13.00      0
4    5.50   5.60      1
..    ...    ...    ...
207 12.72  12.05      0
208 11.24   9.73      0
209 14.65  10.31      0
210 14.84  10.78      0
211 17.18  13.34      0

[212 rows x 3 columns]
```

2. نرمالسازی: (توضیحات مورد نیاز در کد آورده شده است)

Normalization

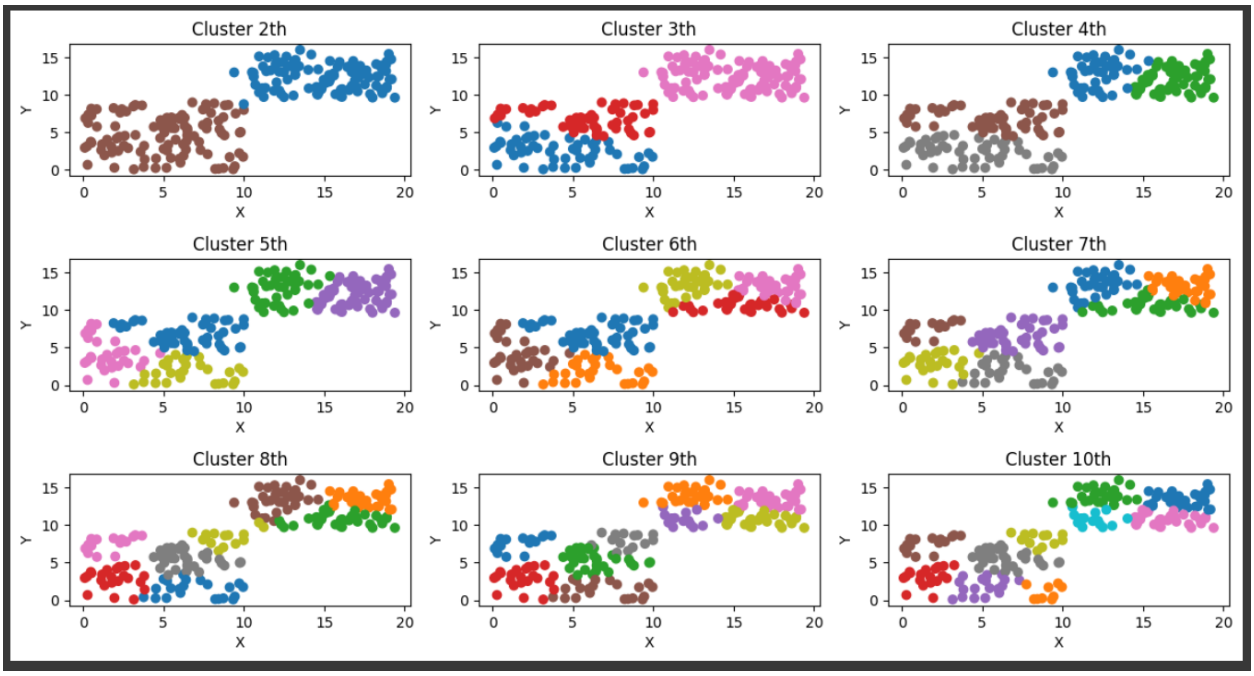
```
1 features_to_normalize =  ['X', 'Y'] # because we want these two to be normalized
2 data = df[features_to_normalize].values # separating the features from dataframe
3
4 normalizer = StandardScaler() # normalizer
5 normalized_data = normalizer.fit_transform(data)
6 df_normalized = pd.DataFrame(normalized_data, columns=features_to_normalize) # creating new DataFrames with normalized data
7
8 print(df_normalized)
```

```
           X         Y
0   -0.813747 -0.357511
1   -0.126986  0.998022
2   -0.725701 -0.402696
3    0.418901  0.998022
4   -0.813747 -0.673802
..        ...       ...
207  0.457641  0.783396
208  0.197024  0.259256
209  0.797500  0.390291
210  0.830958  0.496475
211  1.243014  1.074835

[212 rows x 2 columns]
```

3. پیاده سازی الگوریتم FCM و اعمال موارد خواسته شده:

```python
plt.figure(figsize=(12, 8))

AllFPCs = []
iteration = 1
for num_clusters in range(2 , 11):
    cntr, u, u0, d, jm, p, fpc = skfuzzy.cluster.cmeans(df_normalized.T, num_clusters, m=2, error=0.005, maxiter=1000)

    #determine the final class using argmax membership
    cluster_membership = u.argmax(axis=0)
    AllFPCs.append(fpc)

    # plot the result
    plt.subplot(4, 3, iteration)
    colors = plt.cm.tab10(cluster_membership / float(num_clusters))
    plt.scatter(df['X'], df['Y'], c=colors)
    plt.title(f'Cluster {num_clusters}th')
    plt.xlabel('X')
    plt.ylabel('Y')
    iteration+=1

plt.tight_layout()
plt.show()
```

خروجی نمودار ها به شرح زیر است:



4. معیار FCP: این معیار یک معیار ارزیابی برای خوشه بندی های انجام شده توسط FCM است.

این معیار بر اساس درصد عضویت داده ها در خوشه ها و درجه فازی محاسبه شده بطوریکهFPCها اغلب بر مبنای درصد عضویت بالاتر در

خوشه ها و کمترین درجه فازی محاسبه می شوند.(با هدف اینکه نشان دهد درصد بالاتری از داده ها به یک خوشه تعلق داشته و درجه فازی کمتری دارند.)

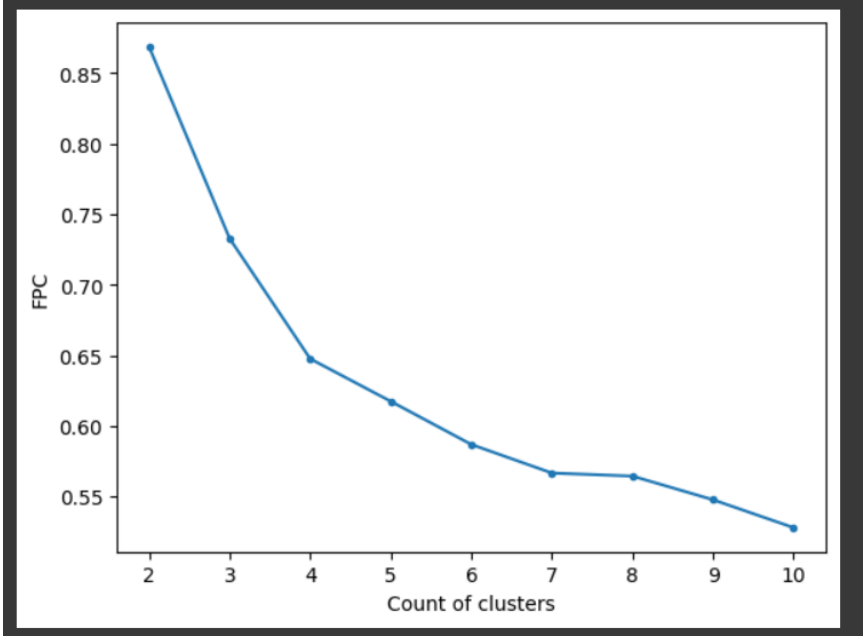
این معیار بیشترین مقدار ممکن(1) را دارد که نشان دهنده یک خوشه بندی است. فرمول آن به این صورت است: (در آن N تعداد داده، uij عضویت داده i در خوشه j است.)

$$FPC = \frac{\sum_{i=1}^{N}(\max_j(u_{ij})^2)}{N}$$

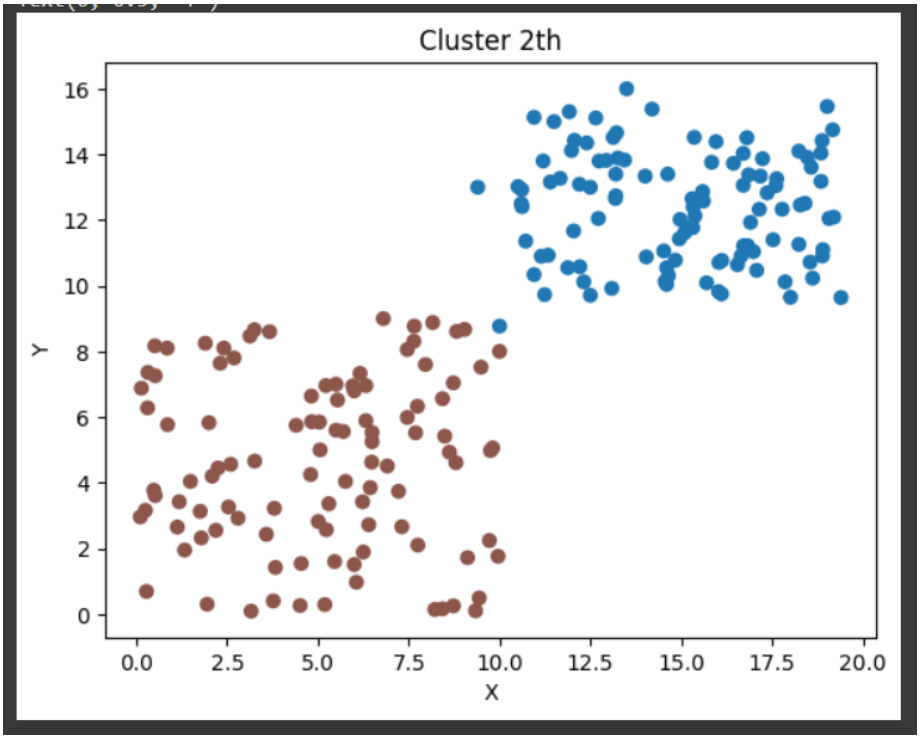
این فرمول مجموع مربعات درصد عضویت بیشترین در هر خوشه به ازای هر داده را محاسبه کرده و سپس میانگین این مقادیر را میدهد.

حال ابتدا نمودار FPC به ازای تعداد خوشه ها را رسم میکنیم:

```
plt.plot(range(2,11), AllFPCs , marker = '.')
plt.xlabel('Count of clusters')
plt.ylabel('FPC')
plt.show()
```



مشاهده میشود که در 2 تعداد خوشه بیشترین مقدار FPC وجود دارد که این خوشه به این شرح است:



حال تمامی مراحل ذکر شده را برای داده دوم طی میکنیم:

.1

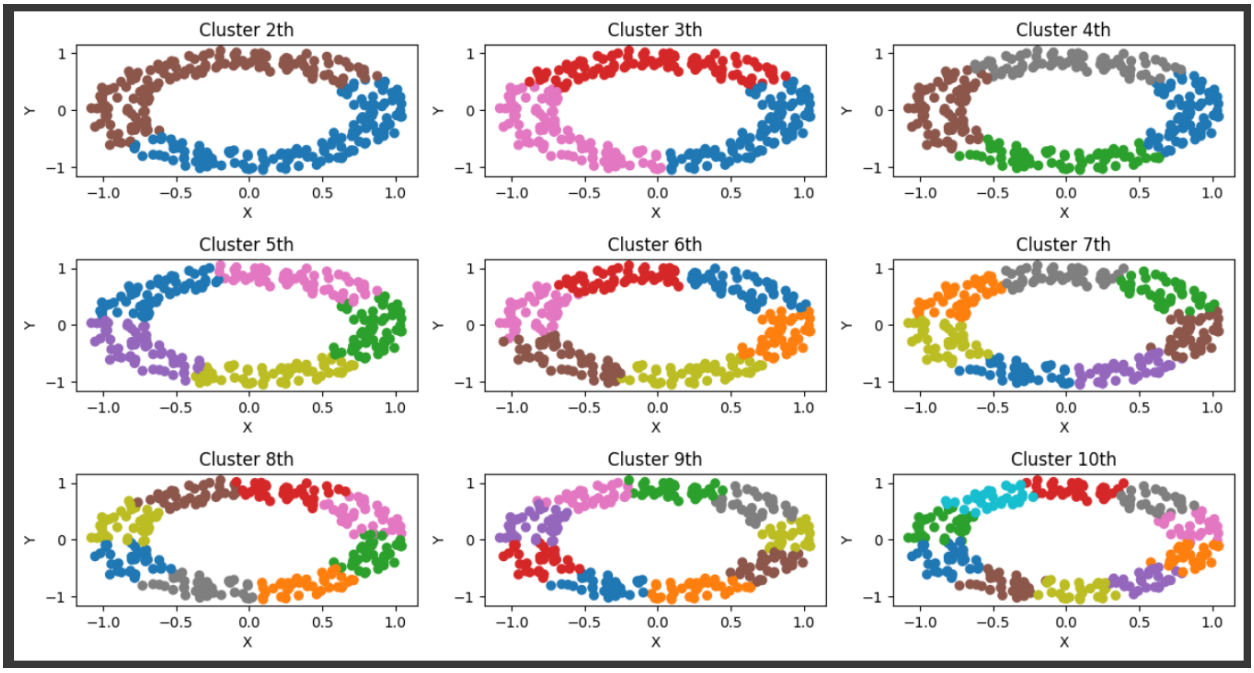
```
1 df = pd.read_csv("/content/data2.csv")
2 print(df)
```

```
            X         Y  Class
0   -0.842046  0.408155      0
1    0.096394 -0.852114      1
2   -0.964828  0.034454      0
3   -0.164699  0.817332      1
4    0.274231  0.756343      1
..        ...       ...    ...
295 -0.526197  0.815028      0
296  0.832952  0.086819      1
297 -0.900569 -0.555313      0
298 -0.534179 -0.513472      1
299 -0.949867 -0.258711      0

[300 rows x 3 columns]
```

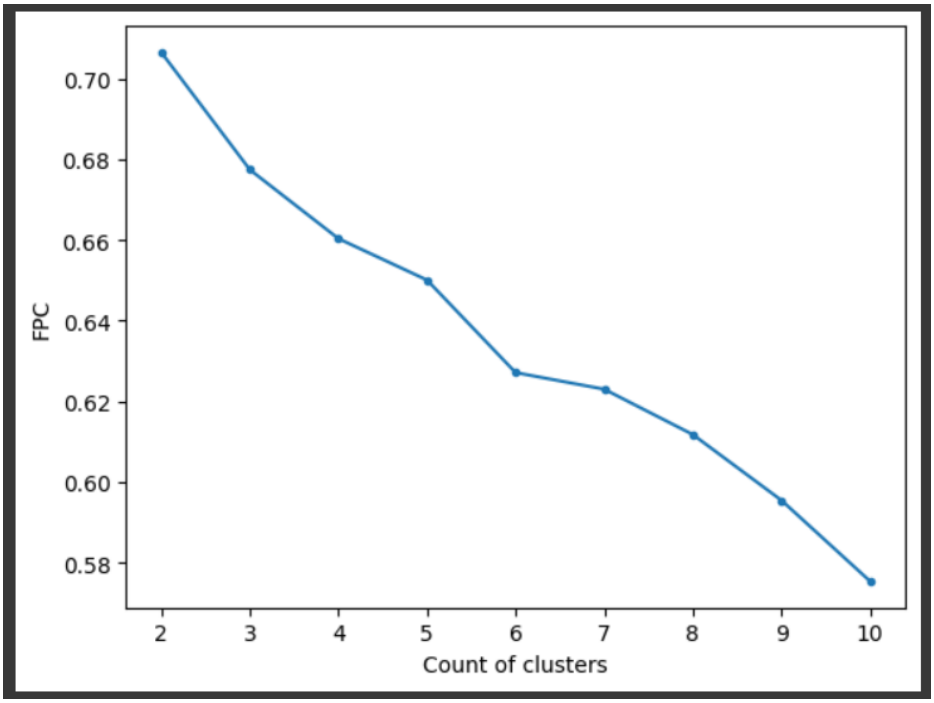
.2

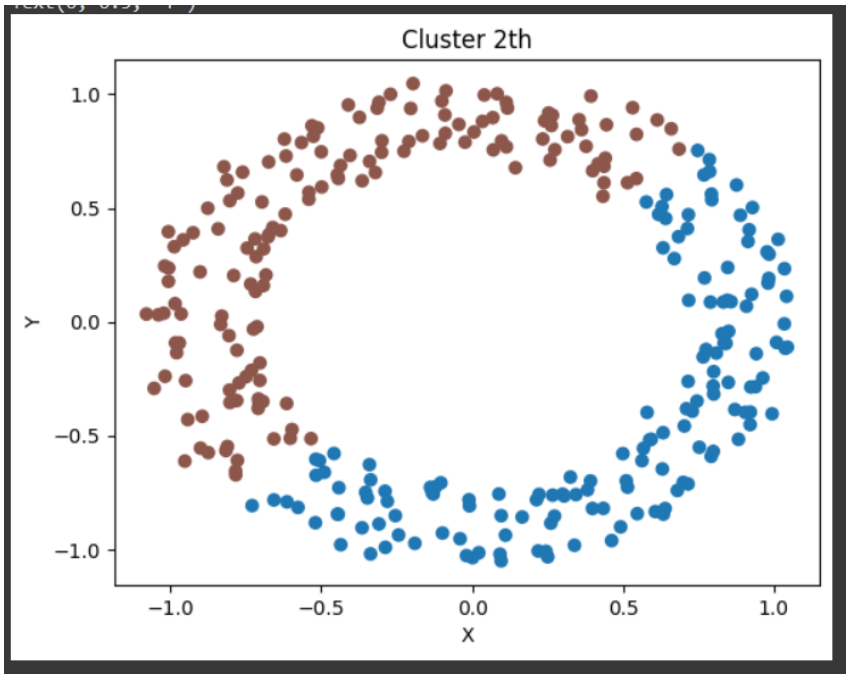
```
            X         Y
0   -1.315522  0.649213
1    0.141025 -1.323862
2   -1.506091  0.064147
3   -0.264216  1.289819
4    0.417045  1.194334
..        ...       ...
295 -0.825294  1.286212
296  1.284232  0.146129
297 -1.406355 -0.859190
298 -0.837684 -0.793684
299 -1.482871 -0.394832

[300 rows x 2 columns]
```

.3

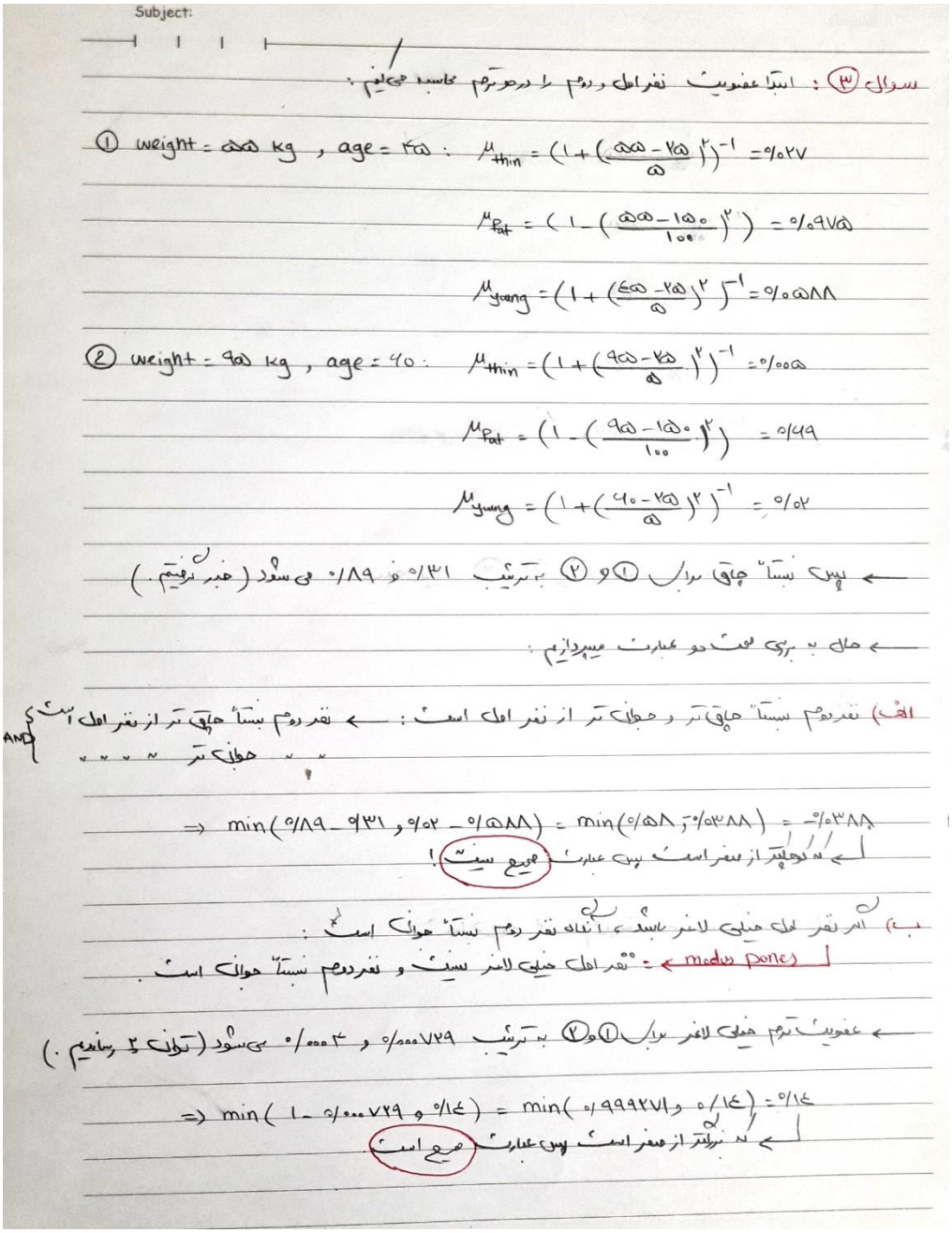

Cluster 2th
Cluster 3th
Cluster 4th
Cluster 5th
Cluster 6th
Cluster 7th
Cluster 8th
Cluster 9th
Cluster 10th
X
Y

.4


FPC
Count of clusters
0.70
0.68
0.66
0.64
0.62
0.60
0.58
2
3
4
5
6
7
8
9
10



سوال 3:


Subject:

سوال ③ : ابتدا عضویت نفر اول و دوم را در هر ترم محاسبه می‌کنیم :

① weight = ۵۵۵ kg , age = ۲۵ : $\mu_{thin} = \left(1+\left(\frac{۵۵۵-۲۵}{۵}\right)^{۲}\right)^{-1} = ۰/۰۲۷$

$\mu_{fat} = \left(1-\left(\frac{۵۵۵-۱۵۰}{۱۰۰۰}\right)^{۲}\right) = ۰/۰۹۷۵$

$\mu_{young} = \left(1+\left(\frac{۵۵۵-۲۵}{۵}\right)^{۲}\right)^{-1} = ۰/۰۰۵۸۸$

② weight = ۹۵۵ kg , age = ۴۰ : $\mu_{thin} = \left(1+\left(\frac{۹۵۵-۲۵}{۵}\right)^{۲}\right)^{-1} = ۰/۰۰۵$

$\mu_{fat} = \left(1-\left(\frac{۹۵۵-۱۵۰}{۱۰۰}\right)^{۲}\right) = ۰/۹۹$

$\mu_{young} = \left(1+\left(\frac{۴۰-۲۵}{۵}\right)^{۲}\right)^{-1} = ۰/۰۲$

← پس نسبتا چاق بودن ① و ② به ترتیب ۰/۳۱ و ۰/۸۹ می‌شود (خیر نوشتم .)

← حال به بررسی هفت دو عبارت میپردازیم :

الف) نفر دوم نسبتا چاق‌تر و جوان‌تر از نفر اول است : ← نفر دوم نسبتا چاق‌تر از نفر اول است AND جوان‌تر ← " " " " "

$\Rightarrow \min(۰/۸۹ - ۰/۳۱ , ۰/۰۲ - ۰/۰۵۸۸) = \min(۰/۵۸ , -۰/۰۳۸۸) = -۰/۰۳۸۸$

← نه دوبیتر از صفر است پس عبارت صحیح نیست !

ب) اگر نفر اول خیلی لاغر نباشد ، آنگاه نفر دوم نسبتا جوان است .

← modus ponens : "نفر اول خیلی لاغر نیست و نفر دوم نسبتا جوان است"

← عضویت ترم خیلی لاغر بودن ① و ② به ترتیب ۰/۰۰۰۷۲۹ و ۰/۰۰۰۳ می‌شود (توان ۲ رساندم .)

$\Rightarrow \min(۱ - ۰/۰۰۰۷۲۹ , ۰/۱۴) = \min(۰/۹۹۹۲۷۱ , ۰/۱۴) = ۰/۱۴$

← نه بزرگتر از صفر است پس عبارت صحیح است .

پایان