تمرین پنجم

هوش محاسباتی

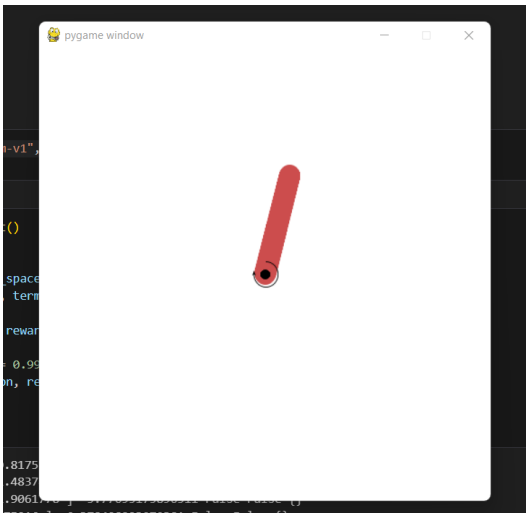
98522328

سوال 1)

قسمت اول)

نمایش آونگ در محیط گرافیکی:

```
env = gym.make("Pendulum-v1", render_mode="human")
observation  = env.reset()
env.render()
for i in range (10):
    action = env.action_space.sample()
    observation, reward, terminated, truncated, info = env.step(action)
    env.render()
    print( observation, reward, terminated, truncated, info)

    if observation[0] >= 0.99 and observation[2] <= 1.5:
        print(observation, reward, terminated, truncated, info)
        break
env.close()
```



قسمت ب) طراحی یک مدل فازی با قوانین مربوطه

همان طور که در صورت سوال اورده شده است، هدف ایستا کردن حرکت آونگ در حالت قائم است.

همان طور که توضیح داده شده است، سه ورودی داریم که هر سه ورودی را می توان در هر لوپ اجرا شونده environment gymاز obersvations می توان دریافت کرد. این سه ورودی به سادگی موقعیت x , y(در شکل دایروی sin , cos)

و همچنین سرعت زاویه ای بین -8و 8 است. (حرکت منفی در جهت ساعت و حرکت مثبت خلاف جهت ساعت است)

همچنین action ما در این اونگ گشتاور است که مقدار -2 تا +2 دارد. (مثبت:پادساعت، منفی: ساعتگرد)

در نتیجه ما با کمک این سه تعریف اولیه stateها و اکشن های خود را با کمک کتابخانه فازی تعیین می کنیم.

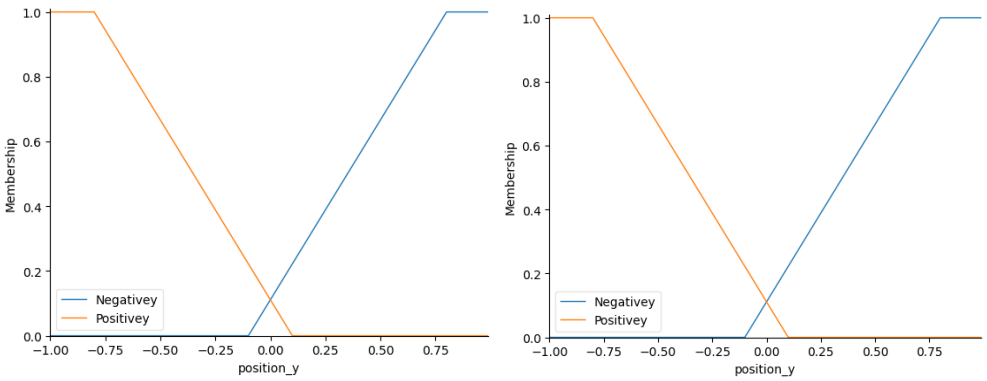
```
position_x = ctrl.Antecedent(np.arange(-1, 1, 0.01), 'position_x')
position_y = ctrl.Antecedent(np.arange(-1, 1, 0.01), 'position_y')
angular_speed = ctrl.Antecedent(np.arange(-8, +8, 0.01), 'angular_speed')
torque = ctrl.Consequent(np.arange(-2, 2, 0.01), 'torque')
```

حال یک نمودار فازی برای هر یک از این مقادیر در نظر میگیریم.

برای موقعیت مکانی :

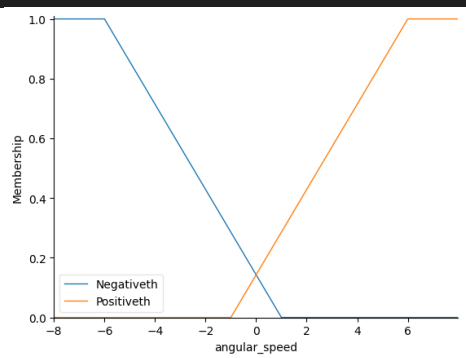
```
#Negative x and positive x
position_x['PositiveX'] = fuzz.trapmf(position_x.universe, [-0.01, 0.8, 1, 1])
position_x['NegetiveX'] = fuzz.trapmf(position_x.universe, [-1, -1, -0.8, 0.01])
position_x.view()

position_y['Negativey'] = fuzz.trapmf(position_y.universe, [-0.1, 0.8, 1, 1])
position_y['Positivey'] = fuzz.trapmf(position_y.universe, [-1, -1, -0.8, 0.1])
position_y.view()
```

موقعیت مکانی به این صورت است که نسبت به حالت از بالا به پایین نمودار در نظر گرفته میشود و برای هر یک در دو قسمت بالای نمودار و پایین نمودار به عنوان قسمت مثبت و قسمت منفی در نظر می گیریم. هر چه به 1 نزدیک تر باشد در نمودار فازی مثبت بالا تر و در حالت منفی برعکس



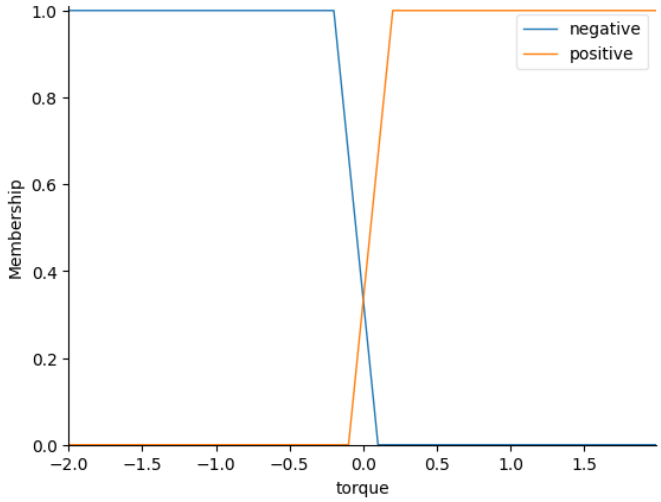
برای سرعت زاویه ای نیز دو حالت مثبت (حرکت در جهت ساعت گردو پاد ساعت گرد) و منفی در نظر میگیریم.

```
angular_speed['Negativeth'] = fuzz.trapmf(angular_speed.universe, [-8, -8, -6,
1])  # negative clockwise
angular_speed['Positiveth'] = fuzz.trapmf(angular_speed.universe, [-1, 6, +8,
+8])
angular_speed.view()
```



برای ACTION نیز همین دو حالت را در نظر خواهیم گرفت.

```
torque['negative'] = fuzz.trapmf(torque.universe, [-2, -2, -0.2, 0.1])
torque['positive'] = fuzz.trapmf(torque.universe, [-0.1, 0.2, 2, 2])
torque.view()
```



تعریف قوانین:

```
rule1 = ctrl.Rule(position_x['NegetiveX'] & angular_speed['Negativeth'],
torque['negative'])
rule2 = ctrl.Rule(position_x['NegetiveX'] & angular_speed['Positiveth'],
torque['positive'])
rule3 = ctrl.Rule(position_x['PositiveX'] & angular_speed['Negativeth'],
torque['positive'])
rule4 = ctrl.Rule(position_x['PositiveX'] & angular_speed['Positiveth'],
torque['negative'])
```

حال قوانینمان را تعریف می کنیم. حالتی که مد نظر ما است این است که زمانی که اونگ به حالت عمودی و مثبت نزدیک شود، (حالت قائم) سرعت آن کند شونده باشد و به نزدیکی صفر برسد. همچنین موقعیت Y را می توان در نظر نگرفت چرا که در هر حالتی که X تغییر کند Y به همان نسبت تغییر می کند. حال اینگونه در نظر میگیریم که در موقعیت منفی و در جهت عقربه های ساعت اونگ باید سرعت زیاد شونده یا شتاب مثبت داشته باشد در نتیجه سرعت و گشتاور در یک جهت باید باشمد. در حالتی که اونگ در قسمت مثبت محور است، سرعت زاویه ای و گشتاور باید در خلاف یک دیگر باشند تا شتاب اونگ کند شونده باشد.

حال میخواهیم نحوه ی اجرا را داشته باشیم.

```
controller = ctrl.ControlSystem([rule1, rule2, rule3, rule4])
simulator = ctrl.ControlSystemSimulation(controller)
print(simulator.input)
reward_history = []
env = gym.make("Pendulum-v1", render_mode="human")
observation, _ = env.reset()
for _ in range(500):
    simulator.input['position_x'] = observation[0]
    simulator.input['angular_speed'] = observation[2]
    simulator.compute()
    decision = simulator.output['torque']
    # print(   , decision = simulator.output['torque'])
    observation, reward, terminated, truncated, info = env.step([decision])
    reward_history.append(reward)
    # Make it easier
    if observation[0] >= 0.99 and -1.5<=observation[2] and observation[2] <= 1.5:
        terminated=True

    env.render()

    if terminated:
        print(f'You Win in {_} iteration!')
        break

env.close()
```

You Win in 63 iteration!

همان طور که در سوال گفته شده است، میخواهیم که اونگ در زیر 500 دور به حالت خواسته شده ما برسد.

در هر دور پنج اطلاعات دریافت می شود که حالت اول متغیر های زبانی و ورودی ما هستند. آن ها را به شبیه ساز میدهیم و با محاسبه ی قوانین نتیجه را ذخیره می کینم. حال لازم است یک شرط برای تمام شدن بازی بگذاریم

```
observation, reward, terminated, truncated, info = env.step([decision]
if observation[0] >= 0.99 and -1.5<=observation[2] and observation[2] <= 1.5
    terminated=True
```

در این حالت در صورتی که موقعیت xنزدیک به 1 باشد و قدر مطلق سرعت از 1.5کم تر باشد آونگ ما به حالت مورد نظر رسیده است و بازی تمام می شود.در این مثال با 63 دور بازی برنده شده است.

-Rewards :

برای نمایش rewardsآن را در یک لیست در هنگام بازی ذخیره کرده و نمودار آن را رسم می کنیم.

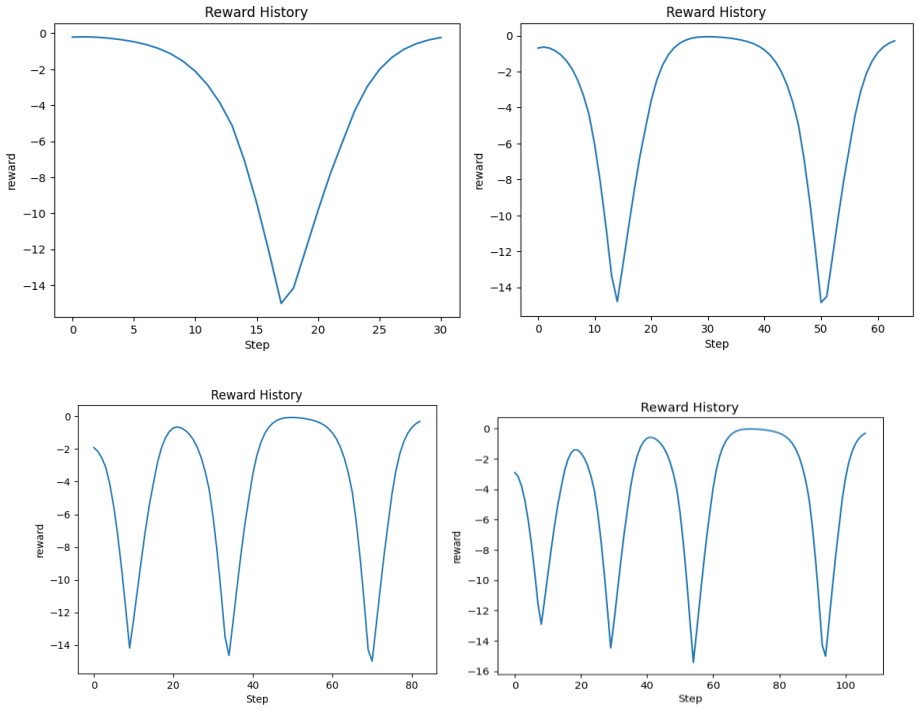
The reward function is defined as:

$r = -(theta^2 + 0.1 * theta\_dt^2 + 0.001 * torque^2)$

حال با توجه به فرمول، rewardsیک مقدار مین ($-(pi^2 + 0.1 * 8^2 + 0.001 * 2^2) = -16.2736044$,)

و مقدار ماکس آن صفر است یعنی زمانی که سرعت صفر و اونگ حالت عمود و در جهت مثبت داشته باشد.

حال میخواهیم برای چند دور مختلف نمودارهای مختلف را مشاهده کنیم.



حالت اول) 63 و حالت دوم 30 و حالت سوم 110 و حالت چهارم 82 .

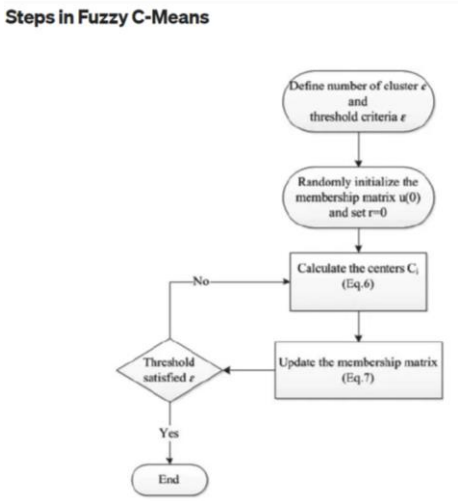
همان طور که مشخص است، در حالت نوسانی از مینیمم حالت به کم ترین حالت می رود و حالت نوسانی دارد و در هر دوره، زمان نزدیک بودن به عدد صفر نسبت به قله بعدی بیشتر است(قله در دوره های بیشتری روی صفر است) و زمانی که به حالت ثابت و سکون رسید از برنامه خارج میشود.)

سوال سوم)

الف)

از fuzzy-k-meansمی توان برای خوشه بندی داده های چند بعدی استفاده کرد و به هر نقطه عضویت در هر مرکز خوشه از 0 تا 100 درصد اختصاص داد. این می تواند در مقایسه با خوشه بندی آستانه سخت سنتی که در آن به هر نقطه یک برچسب واضح و دقیق اختصاص داده می شود بسیار قدرتمند باشد. این الگوریتم با اختصاص عضویت به هر نقطه داده مربوط به هر مرکز خوشه بر اساس فاصله بین مرکز خوشه و نقطه داده کار می کند. هر چه داده ها به مرکز خوشه نزدیک باشند، عضویت آن در مرکز خوشه ای خاص بیشتر است .

این الگوریتم بسته به اندازه ی کلاستری که میخواهیم داردو هر چقدر درجه فازی کم تر باشد درصد عضویت هر نقطه به هر گروه کم تر می شود و اندازه خوشه ها بزرگ تر است.



مراحل انجام . 1- در نظر گرفتن مقدار cluster k

2- initialization: برای هر نمونه داده، یک تابع عضویت فازی تعریف می شود که میزان تعلق آن نمونه به هر یک از خوشه ها را نشان می دهد به طور تصادفی هر μk مربوط به هر کلاستر ر آغاز کرده و احتمال اولیه ای را برای این که هر نقطه برای کدام خوشه است را آغاز می کند.

3- با تکرار ، مرکز هر خوشه را با توجه احتمال عضویت هر عنصر محاسبه می کند

4- همه ی این مراحل را تکرار می کند تا به همگرایی برسد.

نقاط مثبت : بهترین نتیجه را برای مجموعه داده های همپوشانی و نسبتاً بهتر از الگوریتم k-means می دهد.

برخلاف k-means که در آن نقطه داده باید منحصراً به یک مرکز خوشه تعلق داشته باشد، در اینجا نقطه داده به هر مرکز خوشه عضویت داده می شود، در نتیجه ممکن است نقطه داده به بیش از یک مرکز خوشه تعلق داشته باشد.

نقاط منفی :

با مقدار کمترβ ، نتیجه بهتری به دست می آوریم اما به قیمت تعداد تکرار بیشتر.

اندازه‌گیری‌های فاصله اقلیدسی می‌توانند عوامل زمینه‌ای را به طور نابرابر وزن کنند.

عملکرد الگوریتم FCM به انتخاب مرکز خوشه اولیه و/یا مقدار عضویت اولیه بستگی دارد.

لگوریتم FCM با نسخه کلاسیک آن یعنی K-Means در چند مورد تفاوت دارد:

درجه عضویت: درK-Means ، هر نمونه به یک خوشه خاص اختصاص داده می شود، در حالی که درFCM ، هر نمونه به چندین خوشه با درجات عضویت مختلف اختصاص داده می شود.

پارامتر m در FCM تعیین می کند که چگونه درجات عضویت نمونه ها محاسبه می شوند. درK-Means ، این پارامتر ثابت است و برابر با 1 است.

منطق فازی FCM :بر اساس منطق فازی کار می کند، در حالی که K-Means بر اساس منطق قطعی کار می کند.

در کل، الگوریتم FCM یک الگوریتم خوشه بندی انعطاف پذیر است که می تواند برای خوشه بندی داده های مختلف، از جمله داده های غیر خطی و پیچیده، استفاده شود.

ب)

-1

```
df = pd.read_csv('data1.csv')
print(df[0:10])
```

-2

```
features = df[['X', 'Y']]
target = df['Class']

# StandardScaler
scaler_standard = StandardScaler()
scaled_features_standard = pd.DataFrame(scaler_standard.fit_transform(features),
columns=['X', 'Y'])

# MinMaxScaler
scaler_minmax = MinMaxScaler()
scaled_features_minmax = pd.DataFrame(scaler_minmax.fit_transform(features),
columns=['X', 'Y'])
#concat different parts
df_standard = pd.concat([scaled_features_standard, target], axis=1)
df_minmax = pd.concat([scaled_features_minmax, target], axis=1)
```

```
StandardScaler
          X         Y  Class
0 -0.813747 -0.357511      1
1 -0.126986  0.998022      1
2 -0.725701 -0.402696      1
3  0.418901  0.998022      0
4 -0.813747 -0.673802      1  minmax.

          X         Y  Class
0  0.279419  0.434318      1
1  0.481597  0.811439      1
2  0.305340  0.421747      1
3  0.642302  0.811439      0
4  0.279419  0.346323      1
```

3-تابع برای خوشه بندی با کمک منطق فازی

```
def cluster(num_clusters):
    cntr, u, u0, d, jm, p, fpc = skfuzzy.cluster.cmeans(df_minmax[['X', 'Y']].T,
num_clusters, 2, error=0.005, maxiter=1000, init=None)

    cluster_membership = np.argmax(u, axis=0)
    return cntr, u, u0, d, jm, p, fpc , cluster_membership
```

ورود ی تابع منطق فازی ، نقاط مد نظر ، تعداد خوشه های مورد نظر، میزان ارور، ایتریشن و خروجی آن به ترتیب،

نقطه ی مرکزی هر خوشه ، Cntr ،

U0 ، تایع اولیه و اینیشیلایر شده برای پارتیشن بندی ماتریس است.

U ، تایع فازی برای پارتیشن بندی نقاط و مهم ترین آن fpc، ضریب تعلق خوشه بندی فازی

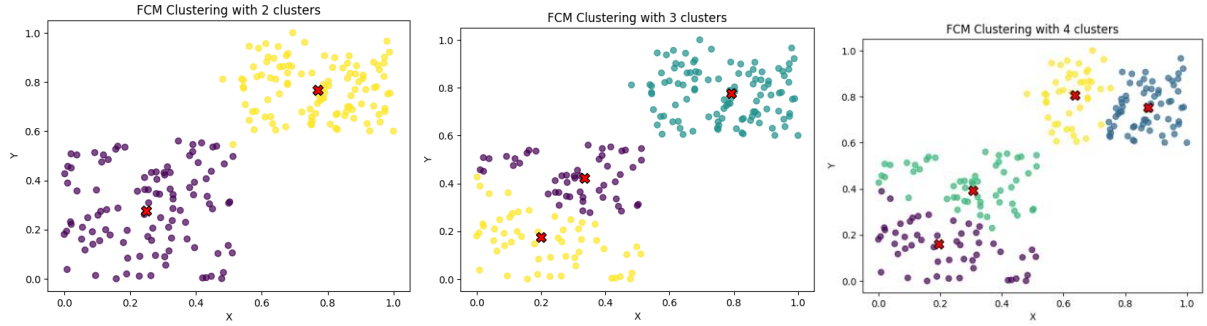
برای پیدا کردن بهترین حالت خوشه بندی فازی از آنجایی که در ماتریس uبری هر عنصر یک احتمال عضو در آن دسته مشخص می کند، یک argmaxمیگیریم تا بیشترین مقدار را براای هر عضو و آن خوشهه بندی نشان دهد.

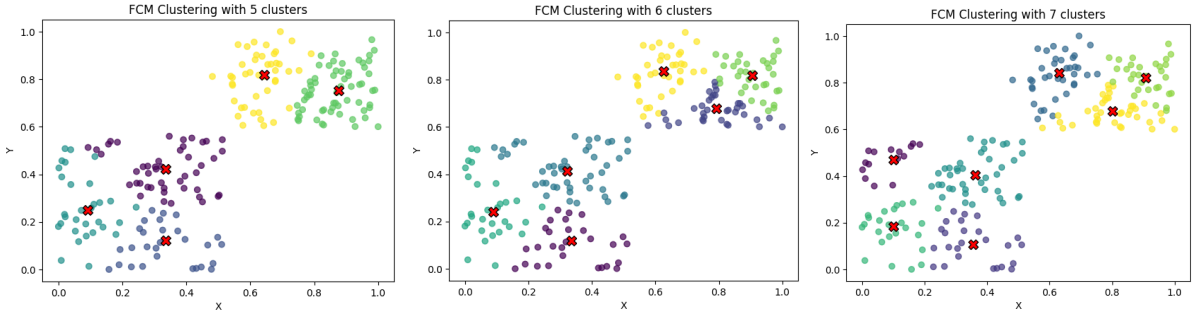
برای ویژوالایز کردن هر خوشه :

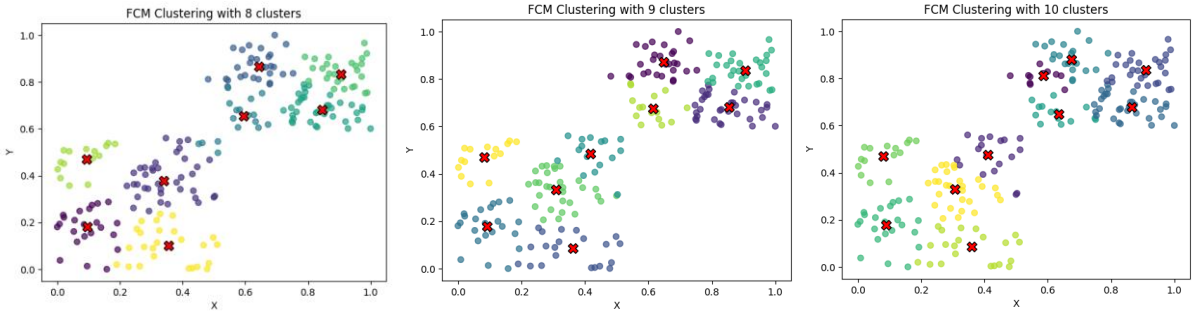
```
def visualize_fcm_clusters(num_clusters , df_minmax):
    cntr, u, u0, d, jm, p, fpc, cluster_membership = cluster(num_clusters)

    plt.scatter(df_minmax['X'], df_minmax['Y'], c=cluster_membership,
cmap='viridis', alpha=0.7)

    # Plot cluster centers
    for cluster_center in cntr:
        plt.scatter(cluster_center[0], cluster_center[1], marker='X', s=100,
c='red', edgecolors='black')

    plt.title(f'FCM Clustering with {num_clusters} clusters')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.show()
```

نمایش :

```
for num_clusters in range(2, 11):
    visualize_fcm_clusters(num_clusters , df_minmax)
```







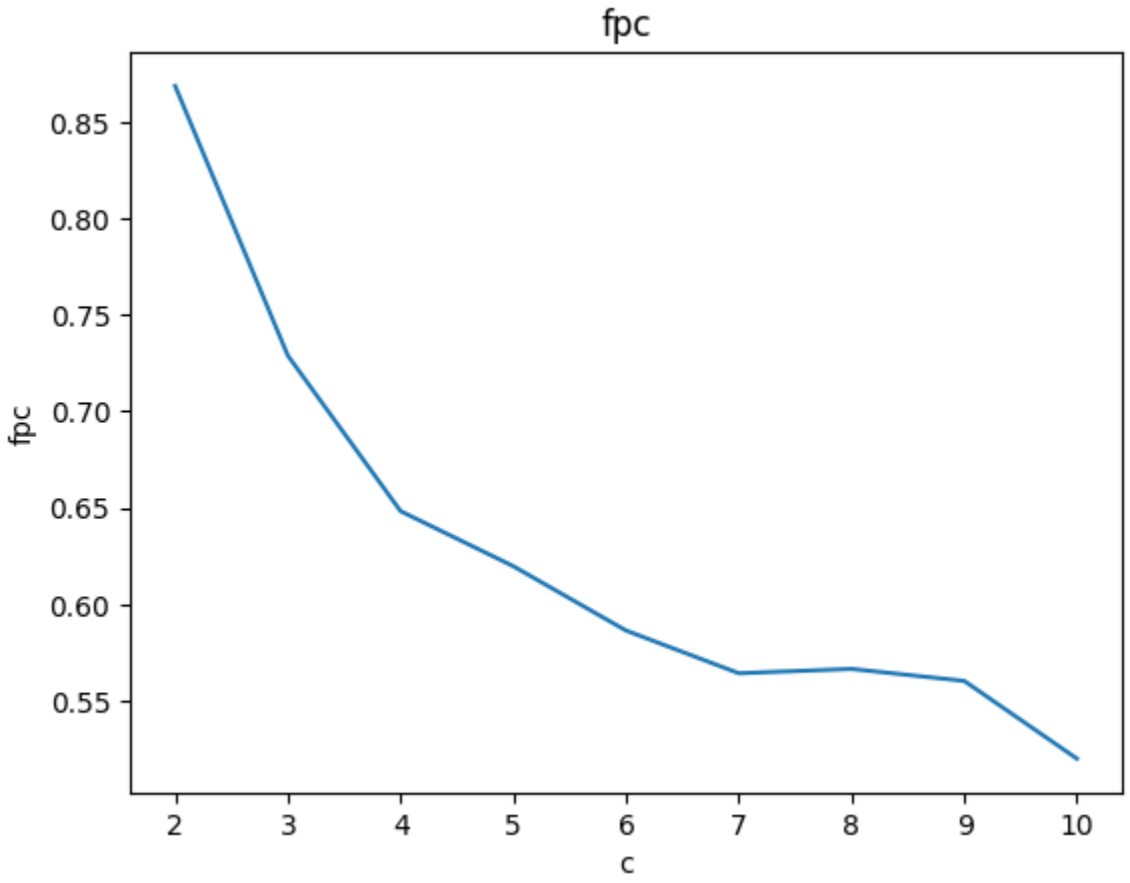
برای قسمت چهارم، برای این که ببینیم کدام مقدار خوشه ها بهترین دسته بندی بوده است، نمودار را در تمام این ده حالت Fpc رسم می کنیم و همان طور که لیبل مشخص کرده است و نمودار میگوید، بهترین دسته بندی متعلق به دسته بندی با دو خوشه است.

FPCمخفف Fuzzy Partition Coefficient است .این یک معیار خوشه بندی است که برای ارزیابی کیفیت خوشه بندی فازی استفاده می شود FPC .بر اساس ایده ای است که خوشه های خوب باید همگن باشند و دارای کمترین همپوشانی با یکدیگر باشند.

FP = 1 - (∑ (μik)^2) / c

این مقدار بین 0-1 است و هر چقدر به 1 نزدیک تر باشد خوشه بندی بهتر انجام شده است.

```
fpc_values = []
all_values = {}
for i in range(2,11):
    cntr, u, u0, d, jm, p, fpc , cluster_membership = cluster(i)
    fpc_values.append(fpc)
    all_values[i] = (cntr, u, u0, d, jm, p, fpc , cluster_membership)


max(fpc_values)
```



همان

طور که مشخص است هر چقدر تعداد خوشه ها بیشتر می شود، میزان دسته بندی نزدیک تر می شود و کم ترین مقدار آن برای حالت 10 است.

5) برای دیتا ست دوم:

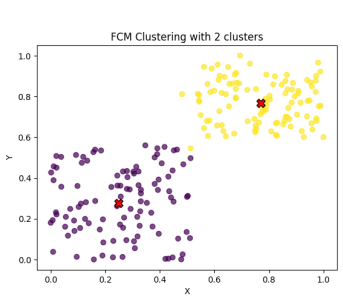
```
df = pd.read_csv('data2.csv')
features = df[['X', 'Y']]
target = df['Class']

# StandardScaler
scaler_standard = StandardScaler()
scaled_features_standard = pd.DataFrame(scaler_standard.fit_transform(features),
columns=['X', 'Y'])

# MinMaxScaler
scaler_minmax = MinMaxScaler()
scaled_features_minmax = pd.DataFrame(scaler_minmax.fit_transform(features),
columns=['X', 'Y'])
#concat different parts
df_standard = pd.concat([scaled_features_standard, target], axis=1)
df_minmax = pd.concat([scaled_features_minmax, target], axis=1)
```

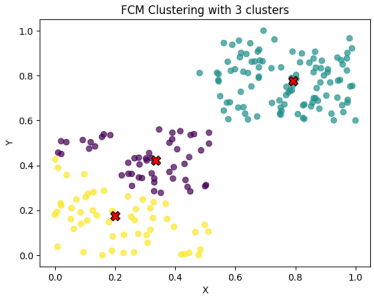
```
def cluster(num_clusters):
    cntr, u, u0, d, jm, p, fpc = skfuzzy.cluster.cmeans(df_minmax[['X', 'Y']].T,
num_clusters, 2, error=0.005, maxiter=1000, init=None)

    cluster_membership = np.argmax(u, axis=0)
    return cntr, u, u0, d, jm, p, fpc , cluster_membership
fpc_values = []

for i in range(2,11):
    cntr, u, u0, d, jm, p, fpc , cluster_membership = cluster(i)
    fpc_values.append(fpc)
```

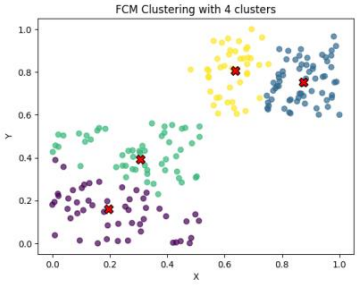
```
def visualize_fcm_clusters(num_clusters , df_minmax):
    cntr, u, u0, d, jm, p, fpc, cluster_membership = cluster(num_clusters)
    plt.scatter(df_minmax['X'], df_minmax['Y'], c=cluster_membership,
cmap='viridis', alpha=0.7)

    for cluster_center in cntr:
        plt.scatter(cluster_center[0], cluster_center[1], marker='X', s=100,
c='red', edgecolors='black')
   plt.title(f'FCM Clustering with {num_clusters} clusters')
```

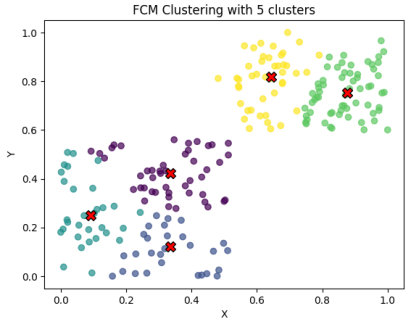
```
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.show()
for num_clusters in range(2, 11):
    visualize_fcm_clusters(num_clusters , df_minmax)
```

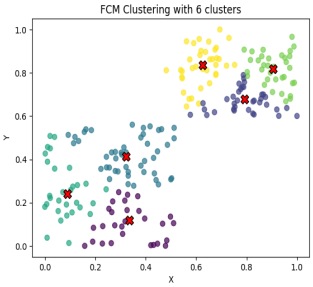
نمودار های به دست آمده برای هر کلاسترینگ :

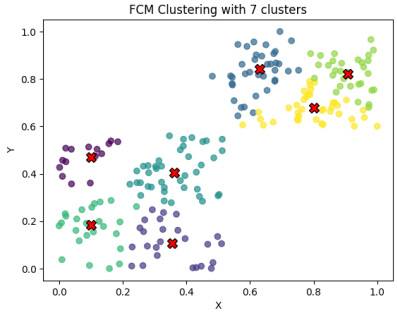


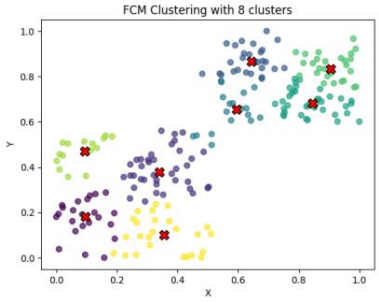


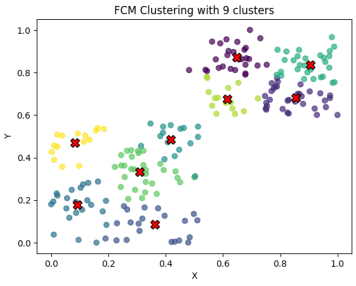


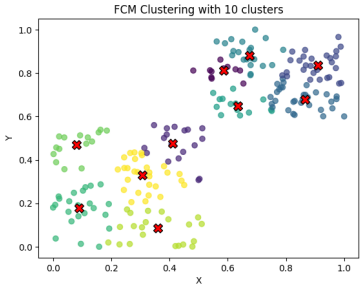




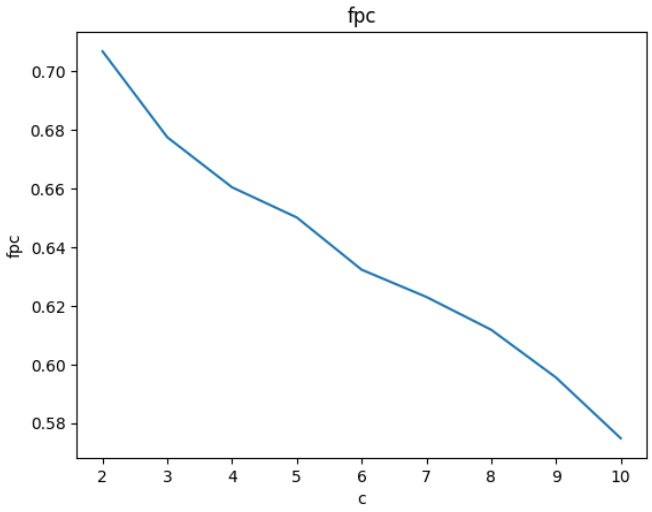








نمودار fpc



همان طور که مشخص است، بهترین کلاستر برای دسته بندی با دو خوشه است و همچنین تقریباا به صورت خطی شیب نمودار تغییر می کند و بد ترین کلاستر برای 10 خوشه است.

سوال سوم)

pA : age = 45 , weight : 55

pB: age :60 , weight: 95

$$\mu_{thin} = \begin{cases} 1 & \mu \in [0,25] \\ (1 + \left(\frac{u-25}{5}\right)^2)^{-1} & \mu \in [25,150] \end{cases}$$

$$\mu_{fat} = \begin{cases} 0 & \mu \in [0,50] \\ 1 - \left(\frac{u-150}{100}\right)^2 & \mu \in [50,150] \end{cases}$$

$$\mu_{young} = \begin{cases} 1 & \mu \in [0,25] \\ (1 + \left(\frac{u-25}{5}\right)^2)^{-1} & \mu \in [25,100] \end{cases}$$

Fuzzy logics :

Negation : 1 – a

Conjunction : min(a,b)

Disjunction : max(a,b)

Implication : min(1, 1+b-a)

Fair rule :

$$\mu_{true}(v) = v$$
$$\mu_{very\,true}(v) = (\mu_{true}(v))^2$$
$$\mu_{fairly\,true}(v) = (\mu_{true}(v))^{1/2}$$
$$\mu_{false}(v) = 1 - \mu_{true}(v)$$
$$\mu_{very\,false}(v) = (\mu_{false}(v))^2$$
$$\mu_{fairly\,false}(v) = (\mu_{false}(v))^{1/2}$$

A) Pb is fairly fatter than pa and pb is younger than pa

Fairly fat new rule :

$$\begin{cases} o\,if\ \mu \in [0,50] \\ \sqrt{1 - \left(\frac{u-150}{100}\right)^2} & \mu \in [50,150] \end{cases}$$

PA is fat value : 0.31

Pb is fat : 0.83 ➔ if pA is fatter the value is 0.31

pA young : 0.050

pB young: 0.005 ➔ if pA is younger the value is 0.050

min(0.31 , 0.050) = 0.050

از آنجایی که احتمال همچین حالتی بسیار کم است در نتیجه گذاره الف نمی تواند گذاره ی درستی باشد.

ب)

Very thin

$$1 \; if \; \mu \in [0,25]$$
$$\left(1 + \left(\frac{u-25}{5}\right)^2\right)^{-2} \quad \mu \in [25,150]$$

Fairly young

$$1 \; if \; \mu \in [0,25]$$
$$\left(1 + \left(\frac{u-25}{5}\right)^2\right)^{-\frac{1}{2}} \quad \mu \in [25,100]$$

pA is very thin: 0.16 ➔(1 + (30/5)^2)^-0.5 = (1/37)^0.5

pb is fairly young :(1 + (35/5)^2) ^-0.5 = (1/50)^0.5 = 0.14

min(1 , 1+b-a) = min(1 , 1+0.14-0.16) =min(1, 0.8) = 1

truth value : 1

از آنجایی که مقدار truth valueاین گزاره 1 است ، در نتیجه می تواند گزاره درستی باشد.