به نام خدا

تمرین پنجم

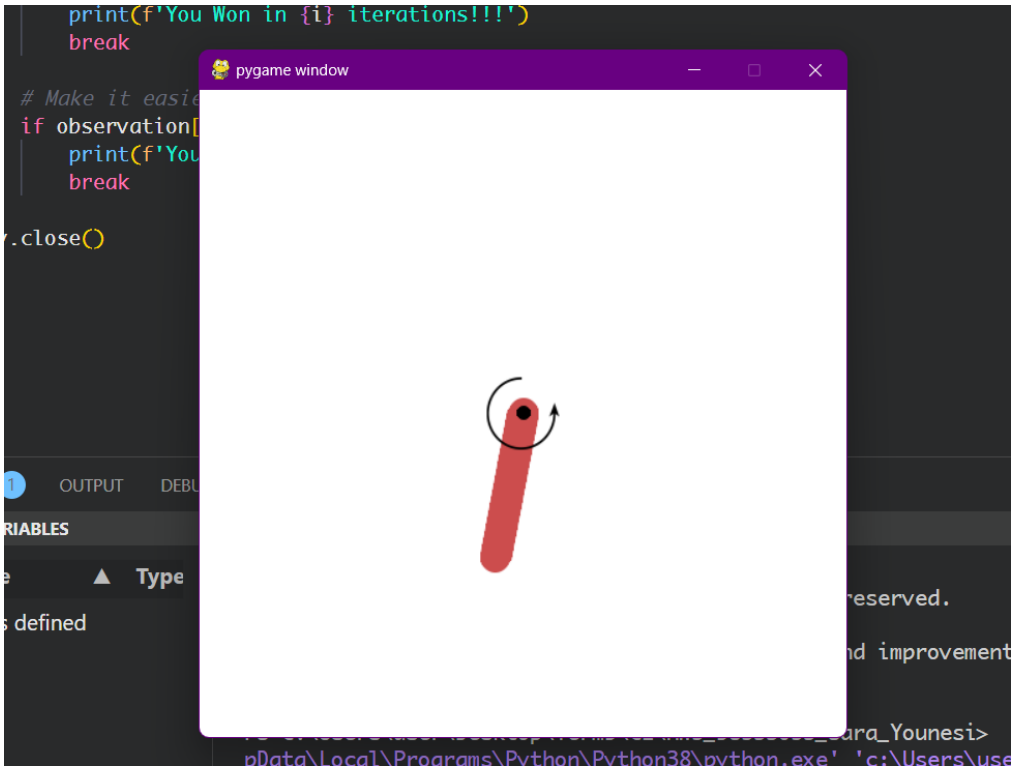
سارا سادات یونسی - ۹۸۵۳۳۰۵۳

فهرست

سوال ۱

مرحله اول

با ایمپورت کردن کتابخانه های فازی و جیم محیط گرافیکی را مشاهده کردیم که یک آونگ در حال حرکت می باشد. که می توانیم پوزیشن آونگ را مشاهده کنیم همچنان نحوه ی حرکت آونگ و نوسان آن را.



مرحله دوم

•ورودی شما در این مسئله عبارت است از:

[-1.0, 1.0] – (x = cos(angle)) مختصات x سر ازاد بازو

[-1.0, 1.0] – (y = sin(angle))مختصات y سر ازاد بازو

[-8.0 -8.0,] – سرعت زاویه ای بازو

• خروجی شما در اینجا گشتاور اعمال شده به انتهای آزاد بازو است که باید برای این خروجی نیز بازه هایی تعریف کنید و پیاده سازی کنید.

• قوانینی را طراحی کنید که بتواند بازو را به مکان هدف برساند . سپس این قوانین را با استفاده از کتابخانه پیادهسازی کنید.

• در نهایت از سیستم پیاده سازی شده برای انتخاب action برای کنترل بازو در محیط اجرا شده استفاده کنید

در اینجا دو شرط خاتمه داریم. اول اینکه به هدف مورد نظر رسیده باشیم. دوم اینکه تعداد مراحل به ۵۰۰ رسیده

باشد . یعنی سیستم طراحی شده توسط شما باید قبل از ۵۰۰ مرحله یا step بازو را به هدف خود برساند و اجازه ندارید که از تعداد مراحل بیشتر از ۵۰۰ استفاده کنید.

توضیح کلی بر روند پیاده سازی شده :

اعداد مجموعه فازی از تعمیم مفهوم اعداد کلاسیک به وسیله منطق فازی حاصل میشوند. اعداد کلاسیک دارای مقدار قطعی و مشخص هستند، اما اعداد فازی دارای مقدار تقریبی و نامشخص هستند. اعداد فازی را میتوان به عنوان مجموعههای فازی با شرایط خاص در نظر گرفت.

- **تعریف متغیرهای زبانی و نحوهی بازه بندی آن ها**
  - **متغیر زبانی موقعیت با دو متغیر فازی مثبت و منفی(ورودی)**

این کد یک متغیر فازی به نام x_position را تعریف می کند که نشان دهنده موقعیت افقی آونگ معکوس روی ارابه است. این متغیر دارای دو مجموعه فازی است: negative و positive. مجموعه فازی negative با استفاده از تابع عضویت trapmf (تراپزی) از کتابخانه fuzz ایجاد شده است. این تابع چهار پارامتر می گیرد که نشان دهنده نقاط شروع، پایان، بالا و پایین تراپز هستند. در اینجا، مقادیر [-۱,۰, -۱,۰, -۰,۱, ۰,۱] به تابع داده شده اند که به این معنی است که مجموعه فازی negative در بازه [-۱,۰, -۰,۱] دارای عضویت ۱ است و در بازه [-۰,۱, ۰,۱] دارای عضویت خطی کاهنده است. مجموعه فازی positive نیز به همین شکل با مقادیر [-۰,۱, ۰,۱, ۱,۰, ۱,۰] تعریف شده است. تابع view() نمودار مجموعه های فازی را نشان می دهد.

با این مورد مشخص کردیم پوزیشن از جند تا چند می تواند باشد رنجش **طبق رنجی که خود مسئله تعین کرده بود در صورت سوال** و همچنین دقت این رنج تولیدی چقدر است. **و در اخر هم نام متغیر زبانی را می نویسیم.**

```
Antecedent()
```

- بازه [-۱,۰, ۱,۰] نشان دهنده حداکثر و حداقل مقادیر ممکن برای موقعیت ارابه است که بر اساس محدودیتهای فیزیکی و مکانیکی تعیین شده است.

- مقادیر -۰,۱۲۵ و ۰,۱۲۵ نشان دهنده نقاط تغییر شیب توابع عضویت هستند که بر اساس تحلیل حساسیت و بهینهسازی انتخاب شدهاند. همانگونه که در نمودار هم دیده می شود نقاطی هشتند که نمودار از خط صاف به یک خط با شیب مثبت یا منفی تبدیل شده است.
- توابع عضویت trapmf (تراپزی) برای توصیف اعداد فازی موقعیت انتخاب شدهاند، زیرا دارای شکل ساده و قابلیت تطبیق بالا با دادههای تجربی هستند.

و با این تابع ذوذنقه هر کدام از متغیر فازی هارا مشخص می کنیم.و ان ۴ نقطه نقاط ذوذنقه از چپ به راست می باشد.

```
x_pos= ctrl.Antecedent(np.arange(-1.0, 1.0, 0.001), 'x_pos')
```

```
x_pos['Negative'] = fuzz.trapmf(x_pos.universe, [-1.0, -1.0, -0.1, 0.1])
x_pos['Positive'] = fuzz.trapmf(x_pos.universe, [-0.1, 0.1, 1.0, 1.0])

x_pos.view()
1, 0.1])
```

و مقدار ممبر شیپ را در این نمودار تولید شده می بینیم که در برخی نقاط کاهش و در برخی نقاط افزایش داشته است.

- **متغیر زبانی سرعت زاویه ای با دو متغیر فازی مثبت و منفی (ورودی)**

در اینجا طبق مسئله سرعت زاویه ای بازه ی از ۸ تا -۸ را در بر میگیرد و ان را یک متغیر دیگر در نظر گرفتیم تا بدانیم سرعت زاویه ای اونگ مثبت است یا منفی در ان لحظه. تا در ادامه به ما کمک کند در حل مسئله . این متغیر زبانی برای کنترل سرعت زاویهای آونگ معکوس استفاده میشود. سرعت زاویهای یکی از پارامترهای مهم در تعیین وضعیت آونگ معکوس است و باید در حدود مقادیر مطلوب نگه داشته شود. این متغیر دارای دو مجموعه فازی است: Negative و Positive. مجموعه فازی Negative با استفاده از تابع عضویت trapmf (تراپزی) از کتابخانه fuzz ایجاد شده است. این تابع چهار پارامتر می گیرد که نشان دهنده نقاط شروع، پایان، بالا و پایین تراپز هستند. در اینجا، مقادیر [-۸,۰, -۸,۰, -۰,۹۵, ۰,۹۵] به تابع داده شده اند که به این معنی است که مجموعه فازی Negative در بازه [-۸,۰, -۰,۹۵] دارای عضویت ۱ است و در بازه [-۰,۹۵, ۰,۹۵] دارای عضویت خطی کاهنده است. مجموعه فازی Positive نیز به همین شکل با مقادیر [-۰,۹۵, ۰,۹۵, ۸,۰, ۸,۰] تعریف شده است. تابع view() نمودار مجموعه های فازی را نشان می دهد.

با این مورد مشخص کردیم سرعت زاویه ای از جند تا چند می تواند باشد طبق رنجی که خود مسئله تعین کرده بود در صورت سوال رنجش و همچنین دقت(۰,۰۰۱) این رنج تولیدی چقدر است. و در اخر هم نام متغیر زبانی را می نویسیم.

```
sorat_zavieh = ctrl.Antecedent(np.arange(-8.0, 8.0, 0.001), 'sorat_zavieh')

sorat_zavieh['Negative'] = fuzz.trapmf(sorat_zavieh.universe, [-8.0, -8.0, -0.95, 0.95])
sorat_zavieh['Positive'] = fuzz.trapmf(sorat_zavieh.universe, [-0.95, 0.95, 8.0, 8.0])

sorat_zavieh.view()
```

- **بازه [-۲,۰, ۲,۰] نشان دهنده حداکثر و حداقل مقادیر ممکن برای گشتاور است که بر اساس ظرفیت موتورهای dc تعیین شده است**
- **مقادیر -۰,۲۴ و ۰,۲۴ نشان دهنده نقاط تغییر شیب توابع عضویت هستند که بر اساس تحلیل حساسیت و بهینهسازی انتخاب شدهاند اعداد حدودی اند که می توانند طبق ازمون و خطا در همان حول پیدا شوند.**

- توابع عضویت **trapmf** (تراپزی) برای توصیف اعداد فازی گشتاور انتخاب شدهاند، زیرا دارای شکل ساده و قابلیت تطبیق بالا با دادههای تجربی هستند

و مقدار ممبر شیپ را در این نمودار تولید شده می بینیم که در برخی نقاط کاهش و در برخی نقاط افزایش داشته است.

- **متغیر زبانی گشتاور با دو متغیر فازی مثبت و منفی(خروجی)**

در اینجا طبق مسئله سرعت گشتاور بازه ی از ۲ تا -۲ را در بر میگیرد و ان را یک متغیر دیگر در نظر گرفتیم تا بدانیم سرعت زاویه ای اونگ مثبت است یا منفی در ان لحظه. تا در ادامه به ما کمک کند در حل مسئله . این متغیر زبانی برای کنترل گشتاور اعمال شده به آونگ معکوس دوار استفاده میشود. گشتاور یکی از عوامل مؤثر در تغییر سرعت زاویهای و وضعیت آونگ معکوس است و باید در حدود مقادیر مجاز تنظیم شود. این متغیر دارای دو مجموعه فازی است: **Negative** و **Positive**. مجموعه فازی **Negative** با استفاده از تابع عضویت **trapmf** (تراپزی) از کتابخانه **fuzz** ایجاد شده است. این تابع چهار پارامتر می گیرد که نشان دهنده نقاط شروع، پایان، بالا و پایین تراپز هستند. در اینجا، مقادیر [-۲,۰, -۲,۰, -۰,۲۴, ۰,۲۴] به تابع داده شده اند که به این معنی است که مجموعه فازی **Negative** در بازه [-۲,۰, -۰,۲۴] دارای عضویت ۱ است و در بازه [-۰,۲۴, ۰,۲۴] دارای عضویت خطی کاهنده است. مجموعه فازی **Positive** نیز به همین شکل با مقادیر [-۰,۲۴, ۰,۲۴, ۲,۰ ,۲,۰] تعریف شده است. تابع **view**() نمودار مجموعه های فازی را نشان می دهد.

با این مورد مشخص کردیم گشتاور از جند تا چند می تواند باشد طبق رنجی که خود مسئله تعین کرده بود در صورت سوال رنجش و همچنین دقت(۰,۰۰۱) این رنج تولیدی چقدر است. و در اخر هم نام متغیر زبانی را می نویسیم.

```
gashtavar = ctrl.Consequent(np.arange(-2.0, 2.0, 0.001), 'gashtavar')

gashtavar['Negative'] = fuzz.trapmf(gashtavar.universe, [-2.0, -2.0, -0.22, 0.22])
gashtavar['Positive'] = fuzz.trapmf(gashtavar.universe, [-0.22, 0.22, 2.0, 2.0])

gashtavar.view()
```

و مقدار ممبر شیپ را در این نمودار تولید شده می بینیم که در برخی نقاط کاهش و در برخی نقاط افزایش داشته است.

- **نحوه تعریف قوانین**

حال برای طراحی قوانین بازی این مورد را در نظر میگیریم که اگر جای **agent** بازی باشیم چه کار می کنیم. و برای این فضای پیوسته که پیچیده است به راحتی با تعریف قوانین به مورد مورد نظر می رسیم.و ۴ قانون طبق اند کردن فرض های متغیر زبانی و حکم متغیر زبانی در نظر میگیریم و در بخش **rule** ها می گذاریم.

حال باید در قسمت کنترلر قوانین خود را به ان بدهیم که بداند طبق چه قوانی باید کار کند و سپس شبیه سازی می کنیم ان را و به عنوان ابجکت ورودی کنترلر را به ان می دهیم. و از ان به بعد با سیمولیتر کار می کنیم.

```
control_system = ctrl.ControlSystem(rules)
simulation = ctrl.ControlSystemSimulation(control_system)
```

این قوانین فازی برای کنترل آونگ معکوس دوار استفاده میشوند. آونگ معکوس دوار یک سیستم غیرخطی و ناپایدار است که نیاز به کنترل دقیق و مقاوم دارد. این قوانین فازی بر اساس متغیرهای فازی موقعیت، سرعت زاویهای و گشتاور تعریف شدهاند که به ترتیب نشان دهنده موقعیت افقی ارابه، سرعت زاویهای آونگ و گشتاور اعمال شده به آونگ هستند. این متغیرها دارای دو مجموعه فازی هستند که به آنها نامهای **Positive** و **Negative** داده شدهاند. این نامها نشان دهنده جهت متغیرها نسبت به مرکز مختصات هستند. برای مثال، موقعیت **Positive** به معنی این است که ارابه در سمت راست مرکز قرار دارد و موقعیت **Negative** به معنی این است که ارابه در سمت چپ مرکز قرار دارد.

هدف از کنترل آونگ معکوس دوار این است که آونگ را در حالت عمودی نگه دارد و ارابه را در موقعیت مطلوب قرار دهد. برای این منظور، از قوانین فازی زیر استفاده میشود:

- اگر موقعیت **Positive** و سرعت زاویهای **Positive** باشد، گشتاور **Negative** اعمال میشود. این قانون برای جلوگیری از افزایش زیاد سرعت زاویهای و از دست رفتن تعادل آونگ طراحی شده است. با اعمال گشتاور **Negative**، سرعت زاویهای کاهش مییابد و آونگ به سمت چپ گرایش پیدا میکند.
- اگر موقعیت **Positive** و سرعت زاویهای **Negative** باشد، گشتاور **Positive** اعمال میشود. این قانون برای حفظ تعادل آونگ و رسیدن به موقعیت مطلوب طراحی شده است. با اعمال گشتاور **Positive**، سرعت زاویهای افزایش مییابد و آونگ به سمت راست گرایش پیدا میکند.
- اگر موقعیت **Negative** و سرعت زاویهای **Negative** باشد، گشتاور **Negative** اعمال میشود. این قانون برای جلوگیری از کاهش زیاد سرعت زاویهای و از دست رفتن تعادل آونگ طراحی شده است. با اعمال گشتاور **Negative**، سرعت زاویهای افزایش مییابد و آونگ به سمت چپ گرایش پیدا میکند.
- اگر موقعیت **Negative** و سرعت زاویهای **Positive** باشد، گشتاور **Positive** اعمال میشود. این قانون برای حفظ تعادل آونگ و رسیدن به موقعیت مطلوب طراحی شده است. با اعمال گشتاور **Positive**، سرعت زاویهای کاهش مییابد و آونگ به سمت راست گرایش پیدا میکند.

این قوانین فازی با استفاده از روش (**Mamdani**) اجرا میشوند. در این روش، ابتدا درجه عضویت متغیرهای ورودی در مجموعههای فازی مربوطه محاسبه میشود. سپس با استفاده از عملگرهای منطقی فازی، درجه عضویت متغیر خروجی در مجموعههای فازی مربوطه تعیین میشود. در اینجا، از عملگر حداکثر برای عملگر **OR** و از عملگر حداقل برای عملگر **AND** استفاده شده است. در نهایت، با استفاده از روش تراکم، مقدار نهایی متغیر خروجی به دست میآید. در اینجا، از روش مرکز میانگین برای تراکم استفاده شده است.

```
rules = [
    ctrl.Rule(x_pos['Positive'] & sorat_zavieh['Positive'], gashtavar['Negative']),
```

```
    ctrl.Rule(x_pos['Positive'] & sorat_zavieh['Negative'], gashtavar['Positive']),
    ctrl.Rule(x_pos['Negative'] & sorat_zavieh['Negative'], gashtavar['Negative']),
    ctrl.Rule(x_pos['Negative'] & sorat_zavieh['Positive'], gashtavar['Positive']),
]
```

•قسمت پیاده سازی و اجرا کردن ایپاک ها و مشاهده بازی و اجرا کردن موارد جایزه و گشتاور و.

این کد یک شبیه سازی از کنترل فازی آونگ معکوس دوار با استفاده از کتابخانه **gym** و کتابخانه **scikit-fuzzy** را اجرا می کند. این کد از محیط **Pendulum-v1** که یک محیط پیش فرض در کتابخانه **gym** است، استفاده می کند. این محیط یک آونگ معکوس دوار را شبیه سازی می کند که باید با اعمال گشتاور به آن، در حالت عمودی نگه داشته شود. این کد از یک سیستم استنتاج فازی به نام **simulation** که با استفاده از کتابخانه **scikit-fuzzy** تعریف شده است، استفاده می کند. این سیستم دارای دو متغیر ورودی و یک متغیر خروجی است که به ترتیب نشان دهنده موقعیت افقی ارابه، سرعت زاویهای آونگ و گشتاور اعمال شده به آونگ هستند. این سیستم دارای چهار قانون فازی است که بر اساس متغیرهای فازی تعریف شدهاند. این قوانین با استفاده از روش (**Mamdani**) اجرا میشوند.

این کد به این صورت عمل می کند:

• ابتدا یک شیء از کلاس **gym.Env** با نام **env** ایجاد می کند که محیط **Pendulum-v1** را نمایش می دهد. پارامتر **render_mode** با مقدار **"human"** تعیین می کند که محیط به صورت گرافیکی نمایش داده شود.

• سپس یک لیست خالی با نام **rewards** ایجاد می کند که برای ذخیره پاداشهای دریافتی از محیط استفاده می شود.

• سپس محیط را با استفاده از متد **reset**() راه اندازی می کند و مقدار اولیه مشاهده و پاداش را در متغیرهای **observation** و _ ذخیره می کند. مقدار مشاهده شامل سه عنصر است که به ترتیب نشان دهنده موقعیت افقی ارابه، سینوس زاویه آونگ و سرعت زاویهای آونگ هستند.

• سپس یک حلقه **for** با ۵۰۰ تکرار ایجاد می کند که در هر تکرار عملیات زیر را انجام می دهد:

• مقدار موقعیت افقی ارابه را از مقدار مشاهده استخراج می کند و آن را به عنوان ورودی **x_pos** به سیستم استنتاج فازی **simulation** می دهد.

• مقدار سرعت زاویهای آونگ را از مقدار مشاهده استخراج می کند و آن را به عنوان ورودی **sorat_zavieh** به سیستم استنتاج فازی **simulation** می دهد.

• سیستم استنتاج فازی را با استفاده از متد **compute**() اجرا می کند و مقدار خروجی گشتاور را در متغیر **total_value** ذخیره می کند.

• مقدار گشتاور را به عنوان عمل به محیط می دهد و با استفاده از متد **step**() یک گام از شبیه سازی را انجام می دهد. مقدار جدید مشاهده، پاداش، پایان یافتن، قطع شدن و اطلاعات را در متغیرهای **observation**، **reward**، **terminated**، **truncated** و **info** ذخیره می کند.

• محیط را با استفاده از متد **render**() به صورت گرافیکی نمایش می دهد.

• پاداش دریافتی را به لیست **rewards** اضافه می کند.

• اگر مقدار **terminated** برابر با **True** باشد، یعنی شبیه سازی به پایان رسیده است و پیام **" You Won in i iterations!!!"** را چاپ می کند و از حلقه خارج می شود.

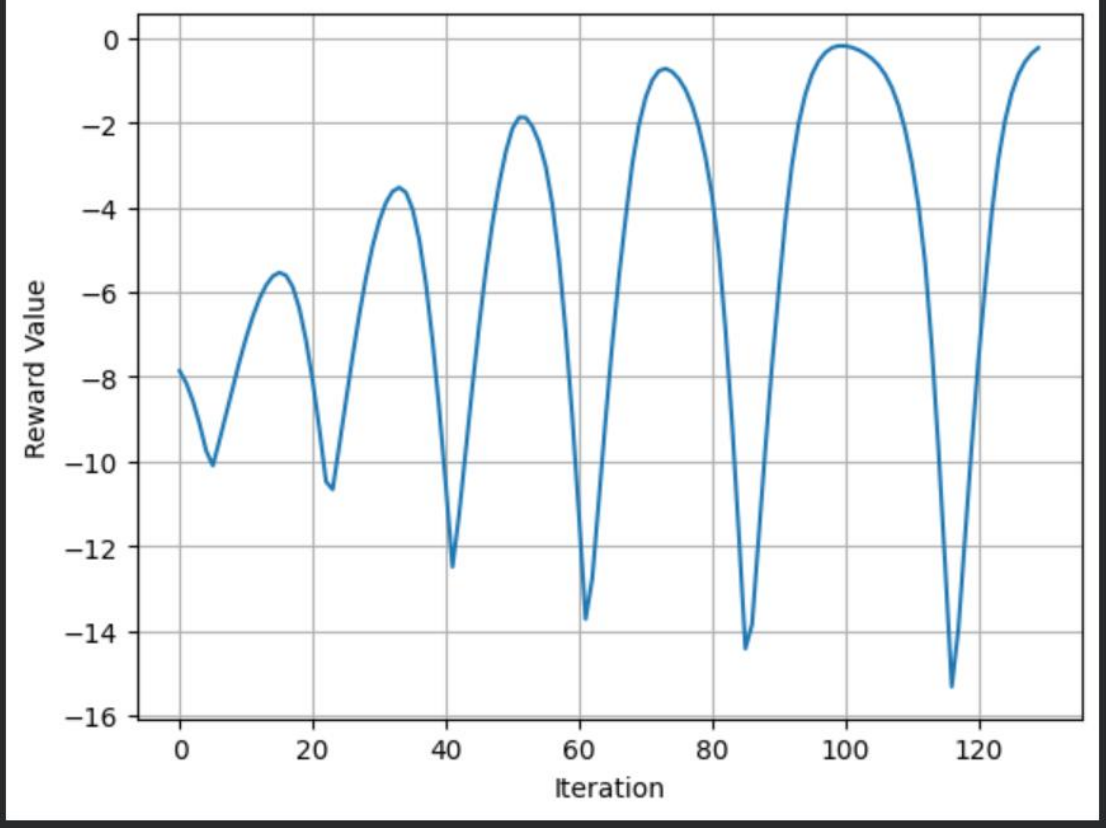
• اگر مقدار موقعیت افقی ارابه بزرگتر از ۰,۹۹ و مقدار مطلق سرعت زاویهای آونگ کوچکتر از ۱,۵ باشد، یعنی شبیه سازی به هدف رسیده است و پیام **"You Won in i iterations!!!"** را چاپ می کند و از حلقه خارج می شود.

• در نهایت، محیط را با استفاده از متد **close**() بسته می شود.

یکی از شرط ها همانگونه که در کلاس حل تمرین گفته شد برای ساده تر شدن می باشد.

•نمودار پاداش های دریافتی

```
plt.show()
✓ 0.3s
```



•تحلیل نمودار پاداش های دریافتی

نمودار پاداش یک روش برای ارزیابی عملکرد یک سیستم کنترلی است که نشان میدهد که چقدر سیستم موفق بوده است تا به هدف خود برسد. در اینجا، هدف کنترل آونگ معکوس دوار این است که آونگ را در حالت عمودی نگه دارد و ارابه را در موقعیت مطلوب قرار دهد. بنابراین، پاداش میتواند بر اساس معیارهایی مانند زاویه آونگ، سرعت زاویهای آونگ، موقعیت ارابه و گشتاور اعمال شده به آونگ تعیین شود. معمولاً پاداش برای هر گام از شبیه سازی محاسبه میشود و در نهایت مجموع پاداشها برای ارزیابی کلی استفاده میشود.

این نمودار نشان میدهد که پاداش در ابتدای شبیه سازی منفی و کم است، زیرا آونگ از حالت عمودی دور است و ارابه از موقعیت مطلوب فاصله دارد. اما با گذشت زمان، پاداش مثبت و زیاد میشود، زیرا سیستم کنترل فازی موفق میشود که آونگ را به سمت حالت عمودی ببرد و ارابه را به موقعیت مطلوب نزدیک کند. این نشان دهنده عملکرد خوب و مقاوم سیستم کنترل فازی است که با استفاده از قواعد فازی، میتواند به هدف خود برسد.
مودار پاداش آونگ معکوس نشان دهنده میزان پاداشی است که یک عامل هوشمند برای کنترل آونگ معکوس دریافت می کند. این پاداش معمولا بر اساس فاصله زاویهای آونگ از حالت عمودی و سرعت زاویهای آن محاسبه

می شود. هدف از کنترل آونگ معکوس این است که آونگ را در حالت عمودی نگه دارد و ارابه را در موقعیت مطلوب قرار دهد. بنابراین، پاداش بیشتری دریافت می کند که زاویه آونگ کوچکتر و سرعت زاویهای آن کمتر باشد.

نمودار پاداش آونگ معکوس حالت سینوسی دارد، زیرا آونگ معکوس یک سیستم نوسانی است که پاسخ آن به صورت ترکیبی از توابع سینوسی و کسینوسی بدست می آید. این نوسانات می توانند به دو دسته تقسیم شوند:

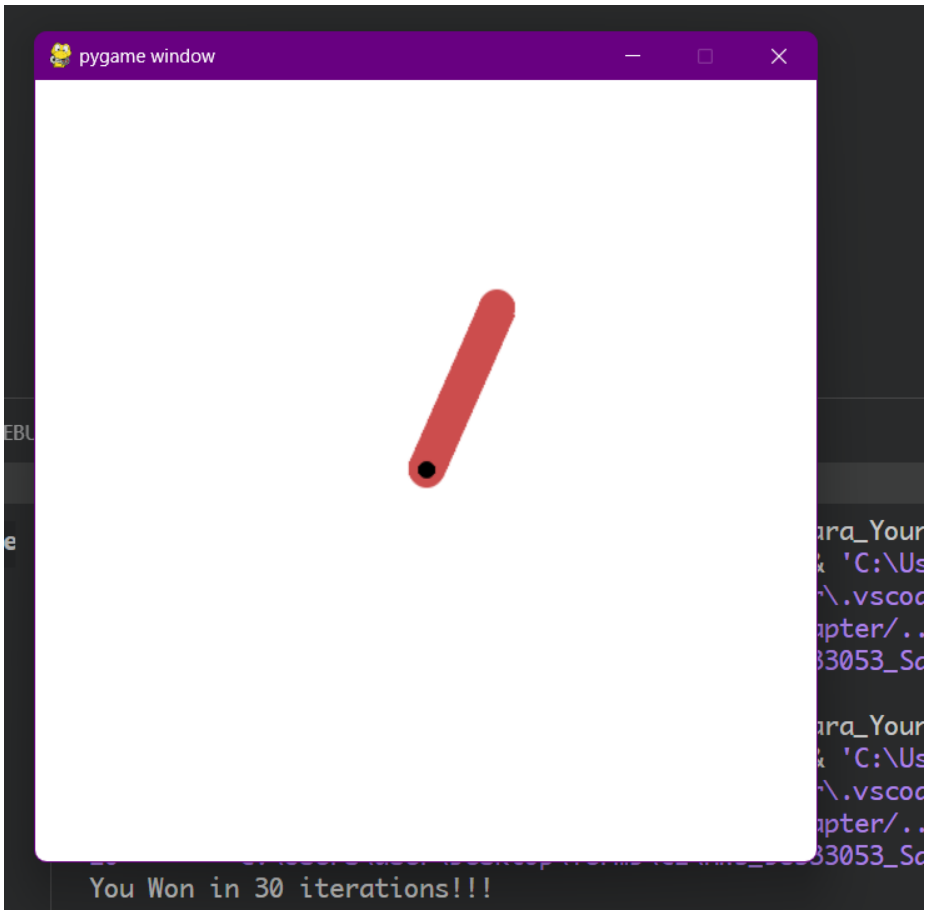
• نوسانات اصلی: این نوسانات نشان دهنده حرکت هارمونیک آونگ معکوس هستند که با فرکانس طبیعی آونگ رخ می دهند. این نوسانات باعث می شوند که زاویه آونگ در بازههای متناوبی از حالت عمودی منحرف شود و سپس به آن برگردد. این نوسانات باعث می شوند که پاداش دریافتی نیز در بازههای متناوبی افزایش و کاهش یابد.

• نوسانات ضربان: این نوسانات نشان دهنده تداخل دو حرکت هارمونیک با فرکانسهای مختلف هستند که باعث می شوند که دامنه نوسانات اصلی افزایش و کاهش یابد. این نوسانات باعث می شوند که پاداش دریافتی نیز دارای نوسانات بزرگتر و کوچکتر باشد.

به طور کلی، نمودار پاداش آونگ معکوس نشان دهنده این است که چقدر عامل هوشمند موفق است که آونگ را در حالت عمودی نگه دارد و ارابه را در موقعیت مطلوب قرار دهد. اگر عامل هوشمند از یک کنترل کننده بهینه استفاده کند، می تواند پاداش بیشتری دریافت کند و نوسانات را کاهش دهد.

این نمودار تناوبی دارد مقدار دامنه ها یکسان است اما مقدار برد به مرور زمان افزایش می یابد تا جایی که به مقدار ماکسیمم خود می رسد و طبق تصویر زیر آونگ به بالاترین نقطه می رسد. اینکه مقدار برد ابتدا کم است و به مرور زیاد می شود این است که با استفاده از قوانین فازی می تواند مقدار پاداش خود را افزایش دهد و هر بار به مقداری پاداش ان افزایش می یابد.و جایی که به ماکسییم خود می رسد برنده می شود و به اتمام می رسد و این حالت تناوبی در هر ایپاک تکرار می شود ابتدا نوسان اونگ کم و بی هدف است و با مرور زمان با توجه به قواعد تعریف شده به هدف خود نزدیک میشود.

برای تحلیل دقیق‌تر و بهبود نمودار پاداش، بهتر است سیستم را مورد بررسی قرار داده و تنظیمات منطق فازی را بهینه سازی کنید. این می‌تواند شامل تنظیم دقیق‌تر توابع عضویت، قوانین، و فاکتورهای ورودی باشد تا سیستم به‌طور مؤثری به تغییرات واکنش نشان دهد و به حالت تعادل پایدارتری برسد.

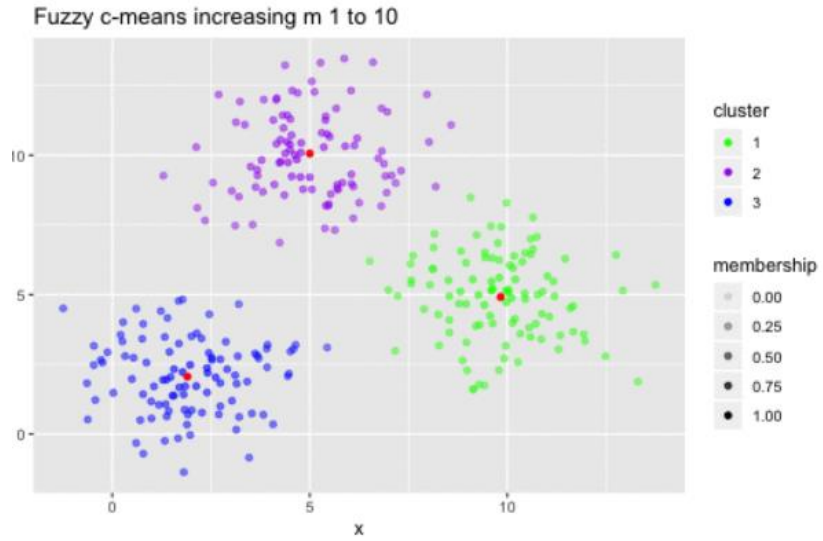


سوال ۲

الف)

عملکرد الگوریتم fuzzy C-Means

از اصول منطق فازی می توان برای خوشه بندی داده های چند بعدی استفاده کرد و به هر نقطه عضویت در هر مرکز خوشه از ۰ تا ۱۰۰ 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 .واضح است که مجموع عضویت هر نقطه داده باید برابر با یک باشد .این یک الگوریتم خوشه‌بندی بدون نظارت است که به ما امکان می‌دهد یک پارتیشن فازی از داده‌ها بسازیم .الگوریتم به پارامتر m بستگی دارد که با درجه مبهم بودن جواب مطابقت دارد .مقادیر بزرگ m کلاس ها را محو می کند و همه عناصر به همه خوشه ها تعلق دارند .راه حل های مسئله بهینه سازی به پارامتر m بستگی دارد .یعنی انتخاب های مختلف m معمولاً منجر به پارتیشن های متفاوتی می شود .در زیر یک gif ارائه شده است که تأثیر انتخاب m به دست آمده از c-means فازی را نشان می دهد.



الگوریتم فازی C-Means (FCM) رویکردی برای خوشه بندی است که به یک قطعه داده اجازه می دهد به دو یا چند خوشه تعلق داشته باشد. این یک روش خوشه بندی است که به یک قطعه داده اجازه می دهد تا به دو یا چند خوشه تعلق داشته باشد. این روش (در سال ۱۹۷۳ توسط دان توسعه یافت و در سال ۱۹۸۱ توسط بزدک بهبود یافت) اغلب در تشخیص الگو استفاده می شود. این مبتنی بر کمینه سازی تابع هدف زیر است:

[J_m = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^mx_i - c_j\2]

جایی که:

(n) تعداد نقاط داده است.

(ج) تعداد خوشه ها است.

(x_i) نقطه داده (i^{th}) است.

(c_j) مرکز خوشه (j^{th}) است.

(u_{ij}) درجه عضویت (x_i) در خوشه (j^{th}) است.

(m) یک عدد واقعی بزرگتر از ۱ (اغلب ۲) است که بر درجه همپوشانی فازی تأثیر می گذارد. هر چه (m) بیشتر باشد، مرزهای خوشه مبهم‌تر می‌شود.

(x_i - c_j\2) مجذور فاصله اقلیدسی بین (x_i) و (c_j) است.

الگوریتم FCM شامل مراحل زیر است:

مراکز خوشه را به صورت تصادفی راه اندازی کنید.

هر قطعه داده به صورت "فازی" به هر خوشه اختصاص داده می شود. این بدان معناست که به جای اینکه فقط به یک خوشه تعلق داشته باشد، هر نقطه داده دارای درجه ای از ارتباط با همه خوشه ها است. اینها توسط عضویت (u_{ij}) برای هر نقطه داده (x_i) به هر مرکز خوشه (c_j) داده می‌شود، که با استفاده از اندازه‌گیری فاصله (معمولاً فاصله اقلیدسی) محاسبه می‌شود.

مراکز خوشه را به عنوان میانگین وزنی تمام نقاط داده محاسبه کنید، جایی که اوزان درجات عضویت هستند.
عضویت (u_{ij}) را برای هر نقطه داده و خوشه با توجه به مراکز خوشه جدید به روز کنید.
مراحل ۲ تا ۴ را تکرار کنید تا زمانی که تغییر ضرایب بین دو تکرار دیگر قابل توجه نباشد (زیر یک آستانه مشخص).
مزیت اصلی FCM توانایی آن در مدیریت ابهام در داده ها با معرفی مفهوم مقادیر عضویت است که امکان تخصیص خوشه های انعطاف پذیرتر را فراهم می کند، که به ویژه در بسیاری از موقعیت های دنیای واقعی که نقاط داده لزوماً به طور انحصاری به آن تعلق ندارند مفید است. یک خوشه و می تواند بخشی از چندین خوشه به درجات مختلف باشد. این مشخصه FCM را به یک الگوریتم خوشه بندی بسیار انعطاف پذیر و با کاربرد گسترده تبدیل می کند، به ویژه در زمینه هایی که با ساختارهای داده پیچیده و مبهم سروکار دارند.

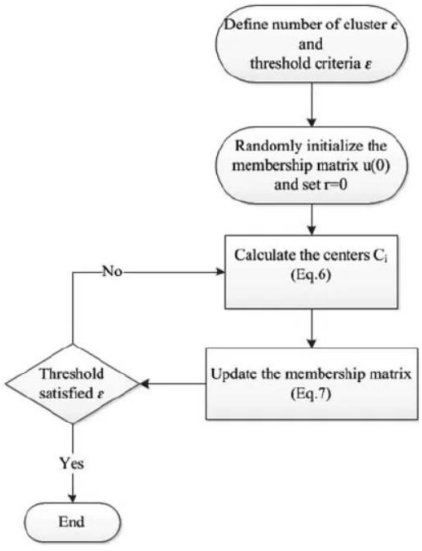


Image Credits: https://www.researchgate.net

جریان فرآیند c-means فازی در زیر برشمرده شده است:

تعداد ثابتی از خوشه ها را k فرض کنید.
مقداردهی اولیه: به طور تصادفی k-means μk مرتبط با خوشه ها را مقداردهی کنید و احتمال اینکه هر نقطه داده xi عضوی از یک خوشه معین k، P است را محاسبه کنید (نقطه xi دارای برچسب k|xi، k است).
تکرار: مرکز خوشه را مجدداً به عنوان مرکز وزن دار با توجه به احتمال عضویت همه نقاط داده xi محاسبه کنید:

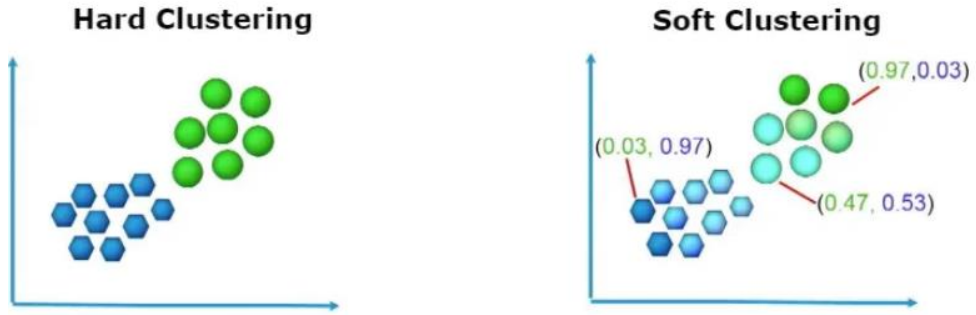
$$\mu_k(n+1) = \frac{\sum_{x_i \epsilon k} x_i * P(\mu_k|x_i)^b}{\sum_{x_i \epsilon k} P(\mu_k|x_i)^b}$$

۴. خاتمه: تکرار تا زمان همگرایی یا تا رسیدن به تعداد تکرار مشخص شده توسط کاربر (تکرار ممکن است در حداکثر یا حداقل محلی به دام افتاده باشد).

پیاده سازی در پایتون را می توانید در اینجا پیدا کنید. یک توضیح ریاضی را می توان در اینجا یافت.

تفاوت آن با نسخه کلاسیک آن یعنی K-Means

- فازی C-Means به یک پارامتر فازی نیاز دارد که درجه فازی بودن خوشه بندی را کنترل می کند، در حالی که k-means به چنین پارامتری نیاز ندارد .
- فازی C-Means از نظر محاسباتی گرانتر از k-means است، زیرا شامل عملیات و تکرارهای بیشتری برای به روز رسانی مقادیر عضویت و مراکز خوشه است.
- Fuzzy C-Meansنسبت به k-means در برابر نویز و نقاط پرت مقاوم تر است، زیرا می تواند نقاط داده مبهم یا دارای همپوشانی را بهتر مدیریت کند
- فازی C-Means می تواند خوشه های فشرده تر و به خوبی از هم جدا شده نسبت به k-means تولید کند، زیرا می تواند با شکل و اندازه خوشه ها سازگاری بهتری داشته باشد.



- انتساب به یک خوشه: در خوشه بندی فازی، هر نقطه احتمال تعلق به هر خوشه را دارد، نه اینکه به طور کامل فقط به یک خوشه تعلق داشته باشد، همانطور که در k-means سنتی وجود دارد. در خوشه‌بندی Fuzzy-C Means، هر نقطه دارای وزنی است که با یک خوشه خاص مرتبط است، بنابراین یک نقطه به اندازه ارتباط ضعیف یا قوی با خوشه که با فاصله معکوس تعیین می‌شود، «در یک خوشه» قرار نمی‌گیرد. به مرکز خوشه.
- سرعت: میانگین C فازی نسبت به میانگین K کندتر عمل می کند، زیرا در واقع کار بیشتری انجام می دهد. هر نقطه با هر خوشه ارزیابی می شود و عملیات بیشتری در هر ارزیابی درگیر می شود. K-Means فقط باید یک محاسبه فاصله انجام دهد، در حالی که میانگین c فازی باید یک وزن دهی کامل با فاصله معکوس انجام دهد.

- نظر شخصی: FCM/Soft-K-Means نسبت به Hard-K-Means «کمتر احمقانه» است که به خوشه های کشیده می رسد (زمانی که نقاطی که در ابعاد دیگر ثابت هستند تمایل دارند در طول یک یا دو بعد خاص پراکنده شوند).

ما باید متوجه باشیم که c-means فازی یک مورد خاص از K-means است وقتی که تابع احتمال مورد استفاده به سادگی ۱ است اگر نقطه داده نزدیکترین نقطه به مرکز باشد و ۰ در غیر این صورت.

مزایا

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.

منفی

مشخصات پیشینی تعداد خوشه ها.

با مقدار کمتر β، 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.

ب)

برای فایل data1.csv

۱. خط اول کد (dataset_1 = pd.read_csv('data1.csv') یک فایل csv را با نام data1.csv میخواند و آن را به صورت یک جدول به نام dataset_1 ذخیره میکند .csv مخفف comma-separated values است و یک فرمت رایج برای ذخیره و انتقال دادههای جدولی است.

```
print("******** INFORMATION FROM DATASET 1 ********")
dataset_1 = pd.read_csv('data1.csv')
dataset_1
```

```
******** INFORMATION FROM DATASET 1 ********
```

	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 × 3 columns

۲. خط اول scaler_standard = MinMaxScaler() یک شیء از کلاس MinMaxScaler را با بازهی پیشفرض بین صفر و یک ایجاد میکند.

خط دوم scaler_standard.fit(dataset_1[['X','Y']]) شیء را با دادههای ستونهای X و Y از جدول dataset_1 آموزش میدهد. این خط باعث میشود که شیء مقادیر کمینه و بیشینهی هر ستون را بدست آورد و برای تبدیل بعدی ذخیره کند.

خط سوم data_normal = scaler_standard.fit_transform(dataset_1[['X', 'Y']]) شیء را با دادههای ستونهای X و Y از جدول dataset_1 آموزش میدهد و سپس آنها را تبدیل میکند. این خط باعث میشود که دادههای ستونهای X و Y به بازهی بین صفر و یک تغییر یابند و در یک جدول جدید به نام data_normal ذخیره شوند.

خط چهارم dataset_1[['X', 'Y']]= scaler_standard.transform(dataset_1[['X', 'Y']]) شیء را با دادههای ستونهای X و Y از جدول dataset_1 تبدیل میکند. این خط باعث میشود که دادههای ستونهای X و Y در جدول اصلی dataset_1 به بازهی بین صفر و یک تغییر یابند.

******** INFORMATION FROM DATASET 1 AFTER NORMALIZE AND FITTING********

	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
...	...	...	...
207	0.653707	0.751728	0
208	0.576983	0.605908	0
209	0.753758	0.642363	0
210	0.763608	0.671904	0
211	0.884914	0.832810	0

همانگونه که میبینیم مقادیر داده ها بین صفر و یک نرمالایز شدند بر خلاف جدول اول.

۳. • قسمت اول C = range(۲, ۱۱) یک بازهی عددی از ۲ تا ۱۰ را به عنوان تعداد گروههای مورد نظر ایجاد میکند.

• قسمت دوم graph, plot = plt.subplots(5, 2, figsize=(20, 30)) یک شیء نمودار و یک آرایهی دوبعدی از شیء زیرنمودار را با ابعاد ۵ در ۲ و اندازهی ۲۰ در ۳۰ ایجاد میکند. این آرایهی زیرنمودارها برای نمایش نتایج خوشهبندی برای هر تعداد گروه استفاده میشود.

• قسمت سوم plot = plot.ravel() آرایهی دوبعدی زیرنمودارها را به یک آرایهی یکبعدی تبدیل میکند تا دسترسی به آنها راحتتر شود.

• قسمت چهارم Set_color = ['red', 'orange', 'yellow', 'green','purple', 'orange','blue', 'indigo', 'violet', 'gray'] یک آرایهی رنگها را برای استفاده در نمودارهای پراکندگی ایجاد میکند. این رنگها برای تفکیک گروهها و مراکز آنها در نمودارها استفاده میشوند.

• قسمت پنجم for g, number in enumerate(C): یک حلقهی for را شروع میکند که در آن متغیر g شمارهی زیرنمودار و متغیر number تعداد گروهها را نشان میدهد. این حلقه برای هر تعداد گروه از ۲ تا ۱۰ اجرا میشود.

• تابع fuzz.cluster.cmeans را از کتابخانه scikit-fuzzy می نامد که الگوریتم خوشه بندی c-means فازی را بر روی داده های نرمال شده ذخیره شده در data_normal انجام می دهد. انتقال داده ها، شماره خوشه، پارامتر فازی m، تحمل خطا، حداکثر تعداد تکرارها و مراکز اولیه خوشه را به عنوان آرگومان ارسال می کند. چندین خروجی را برمی گرداند، اما تنها دو مورد مرتبط هستند: tmp، که یک آرایه دو بعدی از مختصات مراکز خوشه است، و n1، که یک آرایه دو بعدی از مقادیر عضویت هر نقطه داده برای هر خوشه است.

• تابع np.argmax را از کتابخانه numpy فراخوانی می کند، که شاخص حداکثر مقدار را در امتداد یک محور معین از یک آرایه پیدا می کند. آرایه n1 و محور ۰ را به عنوان آرگومان ارسال می کند. آرایه‌ای تک بعدی از شاخص‌های خوشه‌هایی که بالاترین مقدار عضویت را برای هر نقطه داده دارند، برمی‌گرداند. این آرایه را در متغیر mui ذخیره می کند.

• بر روی شاخص های خوشه از ۰ تا عدد-۱ حلقه می زند و شاخص حلقه را به j اختصاص می دهد.

• دو متغیر k و h را تعریف می کند و به ترتیب مقادیر ۰ و ۱ را به آنها اختصاص می دهد. این متغیرها برای دسترسی به مختصات x و y مراکز خوشه از آرایه tmp استفاده می شوند.

• متد plot[g].scatter را از کتابخانه matplotlib فراخوانی می کند، که نمودار پراکنده ای از نقاط داده متعلق به خوشه j را ترسیم می کند. مختصات x و y نقاط داده را از جدول data_1 که با شرط mui == j، رنگ آرایه Set_color و برچسب به عنوان آرگومان فیلتر شده است، ارسال می کند. برچسب 'Cluster {j+1}' را به نمودار اضافه می کند، جایی که {j+1} شماره خوشه است.

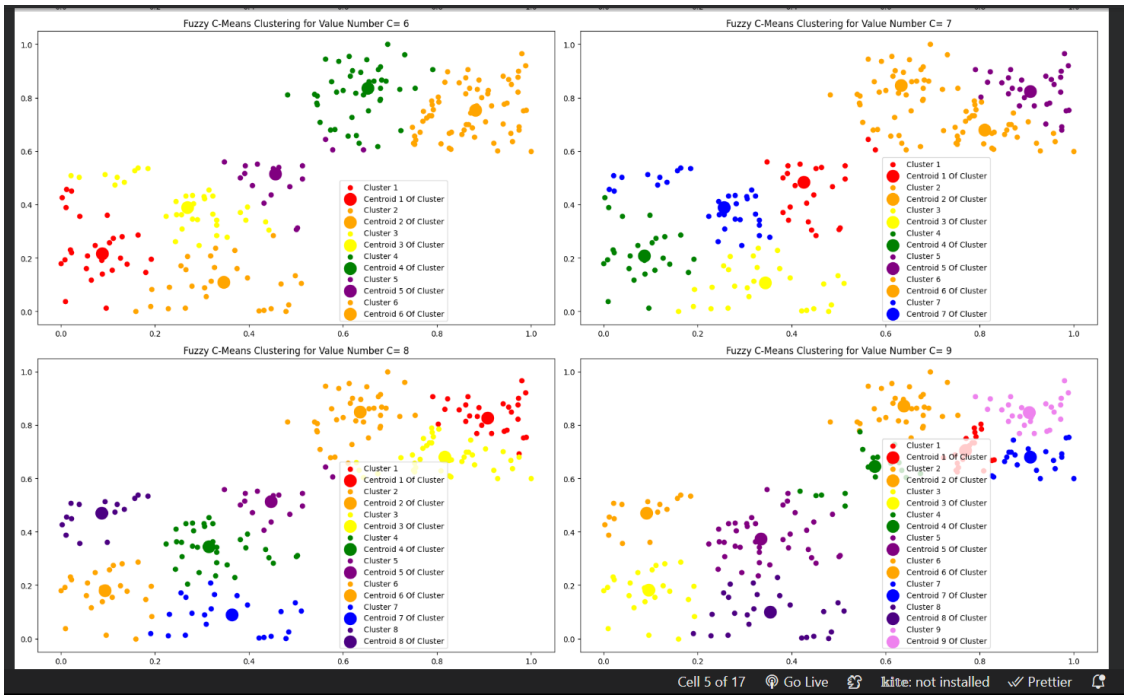
• دوباره متد plot[g].scatter را فراخوانی می کند، اما این بار مرکز خوشه j را رسم می کند. مختصات x و y مرکز خوشه را از آرایه tmp، رنگ را از آرایه Set_color، شکل نشانگر را به عنوان 'o'، اندازه نشانگر را به عنوان ۲۵۰، و برچسب را به عنوان آرگومان ارسال می کند. برچسب "Centroid {j+1} Of Cluster" را به نمودار اضافه می کند، جایی که {j+1} شماره خوشه است.

• متد plot[g].set_title را از کتابخانه matplotlib فراخوانی می کند، که عنوان طرح فرعی g را تعیین می کند. عنوان را به عنوان 'Fuzzy C-Means Clustering for Value Number C= {number}' منتقل می کند، جایی که {number} شماره خوشه است.

• متد plot[g].legend را از کتابخانه matplotlib فراخوانی می‌کند، که یک افسانه به نمودار فرعی g اضافه می‌کند. مکان را به عنوان "بهترین" و کادر محدود را به عنوان (۰، ۰، ۰,۸، ۱) به عنوان آرگومان ارسال می کند. افسانه برچسب ها و رنگ های خوشه ها و مراکز آنها را نشان می دهد.

• پس از پایان حلقه، تابع plt.tight_layout را از کتابخانه matplotlib فراخوانی می کند، که فاصله بین نمودارهای فرعی را برای تناسب با شکل تنظیم می کند. مستطیل را به صورت (۰، ۰، ۱، ۱) به عنوان آرگومان ارسال می کند که به معنای کل منطقه شکل است.

• تابع plt.show را از کتابخانه matplotlib فراخوانی می کند، که شکل را با نمودارهای فرعی نمایش می دهد. در شکل زیر خوشه های تولید شده با رنگ های مختلف به همراه مراکز ان ها با دایره بزرگتر با تعداد خوشه های مختلف را نشان دادیم :



۴. در نهایت، باید با استفاده از معیار FPC بهترین تعداد خوشه را مشخص کنید. معیار FPC یا Partition Coefficient یک معیار ارزیابی برای خوشه بندی فازی است که نشان میدهد چقدر خوشه ها از هم جدا شدهاند. مقدار FPC بین ۰ و ۱ است و هر چه بیشتر باشد، نشان میدهد که خوشه بندی بهتر است. معیار FPC را میتوان با فرمول زیر محاسبه کرد.

"ضریب پارتیشن فازی" در محدوده بین ۰ و ۱ است که در آن ۱ بهترین است. این معیار به کاربر می گوید که داده ها توسط یک مدل چقدر شفاف است. سپس مجموعه داده چندین بار با تعداد خوشه های بین ۲ تا ۱۰ خوشه بندی می شود .سپس نتایج خوشه بندی نمایش داده می شود و نمودار ضریب تقسیم فازی رسم می شود. هنگامی که مقدار FPC به حداکثر می رسد، داده ها به بهترین شکل توصیف می شوند.

همانطور که می بینید، تعداد ایده آل مراکز دو است. اما به طور کلی استفاده از FPC زمانی بسیار مفید است که ساختار داده شفاف نباشد. توجه به این نکته ضروری است که کار از دو مرکز خوشه شروع می شود. خوشه بندی یک مجموعه داده با تنها یک مرکز خوشه یک راه حل واضح است و طبق تعریف FPC == 1 را برمی گرداند (در واقع فقط یک خوشه وجود دارد که همه داده ها به آن تعلق دارند).

بهترین تعداد خوشه و مشخص کردن FPC برای دسته های مختلف و لیبل زدن به داده ها :

بهترین خوشه c=2 با مقدار FCP = ۰,۸ می باشد. داده های ما با تعداد خوشه ۲ بهترین توصیف از مدل را دارد.۵۳

```
Best for Value Number C Fuzzy C-Means Clustering  : 2
Best FCP Fuzzy C-Means Clustering  : 0.8685894251413883
```

```
FPC FOR 2 CLUSTER:  0.86758070269301
FPC FOR 3 CLUSTER:  0.7226017962230996
FPC FOR 4 CLUSTER:  0.6595697590659074
FPC FOR 5 CLUSTER:  0.6283134374230792
FPC FOR 6 CLUSTER:  0.5821329799020413
FPC FOR 7 CLUSTER:  0.5738297689600216
FPC FOR 8 CLUSTER:  0.5322548066183401
FPC FOR 9 CLUSTER:  0.5429729251772679
FPC FOR 10 CLUSTER:  0.5277550794118445
```

	X	Y	Class	Cluster Class
0	5.50	7.00	1	3
1	9.40	13.00	1	1
2	6.00	6.80	1	3
3	12.50	13.00	0	1
4	5.50	5.60	1	3
...	...	...	...	...
207	12.72	12.05	0	7
208	11.24	9.73	0	7
209	14.65	10.31	0	8
210	14.84	10.78	0	8
211	17.18	13.34	0	6

212 rows × 4 columns

توضیح کد های این قسمت :

یک لیست خالی fval برای ذخیره مقادیر تابع هدف برای اعداد خوشه های مختلف تعریف می کند.

• روی اعداد خوشه ای در C که محدوده ای از ۲ تا ۱۱ است حلقه می زند و متغیر حلقه را به عدد اختصاص می دهد.

• تابع fuzz.cluster.cmeans را از کتابخانه scikit-fuzzy می نامد که الگوریتم خوشه بندی c-means فازی را بر روی داده های نرمال شده ذخیره شده در data_normal انجام می دهد. انتقال داده ها، شماره خوشه، پارامتر فازی m، تحمل خطا، حداکثر تعداد تکرارها و مراکز اولیه خوشه را به عنوان آرگومان ارسال می کند. چندین خروجی را برمی گرداند، اما تنها یکی مربوط است: n6، که مقدار تابع هدف نهایی است.

• مقدار تابع هدف n6 را به لیست fval اضافه می کند.

• پس از پایان حلقه، تابع np.argmax را از کتابخانه numpy فراخوانی می کند که شاخص حداکثر مقدار را در یک آرایه پیدا می کند. آرایه fval را به عنوان آرگومان ارسال می کند. ایندکس شماره خوشه ای را که دارای کمترین مقدار تابع هدف است، برمی گرداند، که به معنای بهترین نتیجه خوشه بندی است. برای بدست آوردن شماره خوشه واقعی، ۲ را به ایندکس اضافه می کند و آن را در متغیر C_Best ذخیره می کند.

• همچنین حداکثر مقدار را در آرایه fval که کمترین مقدار تابع هدف است پیدا می کند و آن را در متغیر FPC که مخفف Fuzzy Partition Coefficient است ذخیره می کند. FPC معیاری است که نشان می‌دهد الگوریتم خوشه‌بندی فازی چقدر مقادیر عضویت را به هر نقطه داده برای هر خوشه اختصاص می‌دهد.

• بهترین عدد خوشه و بهترین مقدار FPC را در کنسول چاپ می کند.

• تابع fuzz.cluster.cmeans را دوباره فراخوانی می کند، اما این بار با بهترین شماره خوشه ذخیره شده در C_Best. همان خروجی‌های قبلی را برمی‌گرداند، اما فقط دو مورد مرتبط هستند: tmp، که آرایه‌ای دو بعدی از مختصات مراکز خوشه است، و n1، که آرایه‌ای دو بعدی از مقادیر عضویت هر نقطه داده است. هر خوشه
• دوباره تابع np.argmax را فراخوانی می کند، اما این بار با آرایه n1 و محور ۰ به عنوان آرگومان. آرایه‌ای تک بعدی از شاخص‌های خوشه‌هایی که بالاترین مقدار عضویت را برای هر نقطه داده دارند، برمی‌گرداند. این آرایه را در متغیر mui ذخیره می‌کند، که نشان‌دهنده برچسب‌های خوشه برای هر نقطه داده است.
• تابع pd.read_csv را از کتابخانه pandas فراخوانی می کند، که یک فایل مقادیر جدا شده با کاما (csv) را می خواند و یک قاب داده را برمی گرداند. نام فایل 'data2.csv' را به عنوان آرگومان ارسال می کند. این قاب داده را در مجموعه داده متغیر ذخیره می‌کند /
• یک ستون جدید به مجموعه داده قاب داده_۲ با نام 'کلاس کلاستر' اضافه می کند و مقادیر آرایه mui را به آن اختصاص می دهد. این ستون برچسب خوشه ای را برای هر نقطه داده در قاب داده نشان می دهد.

• مجموعه داده قاب داده_۲ را در کنسول چاپ می کند.

برای فایل data2.csv

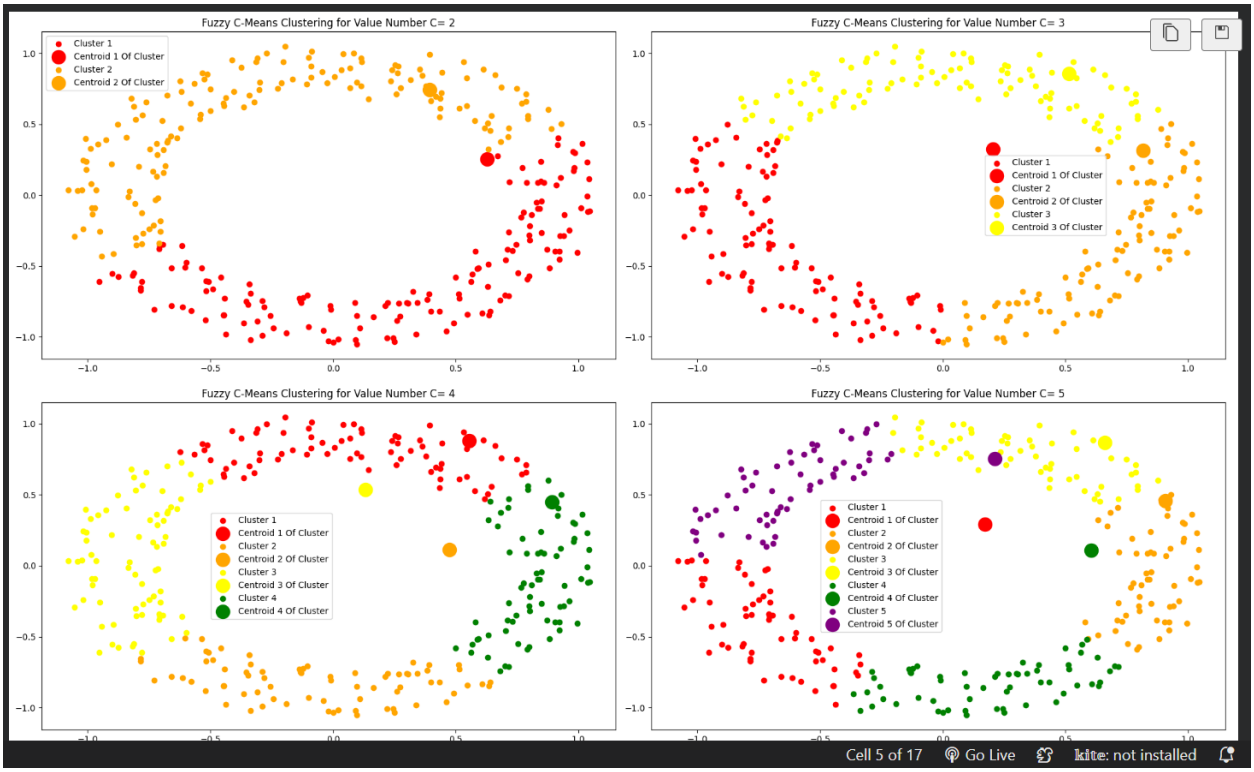
۱.صرفا برای فایل دوممان همان روش ها و رویکرد هارا تکرار می کنیم با این تفاوت که فایل دوم را می خوانیم از دایرکتوری.نمایی از خوشه بندی برای دیتاست دوم و همانطور که میبینیم بعد از نرمال کردن اعداد منفی پاک شدند :

	X	Y	Class
0	0.111887	0.695514	0
1	0.553537	0.094573	1
2	0.054103	0.517320	0
3	0.430661	0.890623	1
4	0.637230	0.861541	1
...	...	...	...
295	0.260532	0.889525	0
296	0.900176	0.542289	1
297	0.084345	0.236098	0
298	0.256775	0.256050	1
299	0.061144	0.377528	0

300 rows × 3 columns

	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 × 3 columns



بهترین تعداد خوشه و مشخص کردن FPC برای دسته های مختلف و لیبل زدن به داده ها :

بهترین خوشه c=2 با مقدار FCP = ۰,۷ می باشد.

```
Best for Value Number C Fuzzy C-Means Clustering  : 2
Best FCP Fuzzy C-Means Clustering  : 0.7065604018728259
```

```
FPC FOR 2 CLUSTER:  0.7071130620590377
FPC FOR 3 CLUSTER:  0.6774143475403843
FPC FOR 4 CLUSTER:  0.6603052579621941
FPC FOR 5 CLUSTER:  0.6500623929680728
FPC FOR 6 CLUSTER:  0.6321969610071321
FPC FOR 7 CLUSTER:  0.6229931142841069
FPC FOR 8 CLUSTER:  0.6117786264813322
FPC FOR 9 CLUSTER:  0.5954085949381366
FPC FOR 10 CLUSTER:  0.5746305906103093
```

	X	Y	Class	Cluster Class
0	-0.842046	0.408155	0	6
1	0.096394	-0.852114	1	4
2	-0.964828	0.034454	0	6
3	-0.164699	0.817332	1	2
4	0.274231	0.756343	1	0
...	...	...	...	...
295	-0.526197	0.815028	0	5
296	0.832952	0.086819	1	9
297	-0.900569	-0.555313	0	1
298	-0.534179	-0.513472	1	1
299	-0.949867	-0.258711	0	1

300 rows × 4 columns

نتیجه گیری و تحلیل نهایی

خوشه‌بندی فازی c-means برای توزیع داده‌های دایره‌ای متحدالمرکز خیلی خوب نیست، زیرا تمایل به تولید خوشه‌های کروی با واریانس برابر دارد. حلقه، که خوشه بندی c-means فازی را برای گرفتن ساختار واقعی داده ها دشوار می کند.

خوشه‌بندی فازی c-means بسته به شکل و واریانس خوشه‌ها می‌تواند برای توزیع عادی داده‌ها خوب باشد. اگر خوشه ها کروی باشند و دارای واریانس مساوی باشند، خوشه بندی c-means فازی می تواند نتایج رضایت بخشی ایجاد کند. با این حال، اگر خوشه ها غیر کروی باشند یا دارای واریانس های متفاوتی باشند، خوشه بندی c-means فازی ممکن است نتواند ساختار واقعی داده ها را به تصویر بکشد.

یک بهبود احتمالی برای خوشه بندی c-means فازی استفاده از متریک هسته است که می تواند داده ها را به فضایی با ابعاد بالاتر که در آن خوشه ها قابل تفکیک تر هستند، ترسیم کند. یکی دیگر از پیشرفت ها استفاده از یک عامل فازی مشترک است که می تواند مقادیر عضویت را با توجه به اطلاعات محلی داده ها تنظیم کند.

ک تکنیک خوشه‌بندی بهتر برای توزیع داده‌های دایره‌ای متحدالمرکز، خوشه‌بندی مبتنی بر چگالی است، مانند DBSCAN یا OPTICS، که می‌تواند خوشه‌هایی با شکل و اندازه دلخواه را بر اساس چگالی محلی نقاط داده شناسایی کند.

برخی از معایب خوشه بندی c-means فازی عبارتند از:

• نیاز به مشخصات قبلی تعداد خوشه ها دارد که ممکن است از قبل مشخص نباشد یا بسته به داده ها متفاوت باشد.

• به انتخاب مراکز اولیه خوشه و پارامتر فازی حساس است که می تواند بر همگرایی و کیفیت نتایج خوشه بندی تاثیر بگذارد.

• از نظر محاسباتی گران است، زیرا به تکرارهای متعدد و محاسبات فاصله برای هر نقطه داده و هر خوشه نیاز دارد.

• فرض می کند که خوشه ها کروی هستند و دارای واریانس مساوی هستند، که ممکن است برای برخی از توزیع های داده درست نباشد.

• ممکن است نتواند ویژگی های پر سر و صدا، نامربوط یا اضافی را کنترل کند، که می تواند اندازه گیری فاصله و مقادیر عضویت را مخدوش کند.

لگوریتم خوشه‌بندی فازی C-Means (FCM) بر اساس به حداقل رساندن واریانس درون خوشه‌ها است، که به طور موثر فرض می‌کند که خوشه‌ها تقریباً کروی یا فوق کروی در فضای ویژگی هستند. با توجه به این فرض، FCM زمانی به خوبی کار می کند که خوشه ها همسانگرد و فشرده باشند، به این معنی که نقاط داده هر خوشه در اطراف یک نقطه میانگین مرکزی جمع شده اند.

اکنون، بیایید توزیع داده های دایره ای متحدالمرکز را در نظر بگیریم، که در آن شما خوشه های دایره ای در اطراف یکدیگر دارید. در چنین سناریویی، فرض همسانگرد برقرار نیست زیرا ساختار داده ذاتاً غیر کروی است و

به صورت دایره ای گسترش می یابد. هر خوشه شکل حلقه مانندی دارد که «مرکز» خود توخالی است یا حاوی نقاطی از یک خوشه دیگر است.

هنگام اعمال FCM برای داده های دایره متحدالمرکز:

۱. **از دست دادن روابط فضایی**: FCM روابط فضایی بین نقاط داده را که در شناسایی دایره های متحدالمرکز ضروری هستند در نظر نمی گیرد. FCM احتمالات عضویت را بر اساس فاصله تا مراکز خوشه بدون توجه به ساختار کلی اختصاص می دهد.

۲. **سوگیری به سمت خوشه های کروی**: از آنجایی که هدف FCM به حداقل رساندن فاصله بین نقاط داده و نزدیکترین مرکز خوشه است، برای یافتن مرکز مناسب برای شکل حلقه مشکل خواهد داشت. در نتیجه، مراکز خوشه فازی تمایل دارند به سمت میانگین هندسی ساختارها سوگیری کنند، که با یک "مرکز" معنادار برای یک خوشه حلقه‌شکل مطابقت ندارد.

۳. **همپوشانی بین خوشه ها**: FCM سعی می کند با اختصاص عضویت های فازی، انتقال های صاف بین خوشه ها ایجاد کند. برای دایره های متحدالمرکز، این بدان معناست که نقاط روی لبه داخلی دایره بیرونی تقریباً به همان اندازه که به مرکز خوشه واقعی خود نزدیک هستند به مرکز دایره داخلی نزدیک هستند که منجر به طبقه بندی های اشتباه قابل توجهی می شود.

۴. **پیچیدگی الگوریتم**: از آنجایی که FCM از هر نقطه در همه خوشه ها برای محاسبه مرکز استفاده می کند، می تواند از نظر محاسباتی برای اشکال پیچیده بسیار فشرده باشد. در دایره‌های متحدالمرکز، نقاط روی دو طرف یک دایره به طور مساوی به محاسبه مرکز کمک می‌کنند، که توزیع فضایی واقعی و موقعیت نسبی آنها را منعکس نمی‌کند.

این محدودیت ها به این معنی است که در حالی که FCM هنوز می تواند برای داده های دایره متحدالمرکز اعمال شود، نتایج اغلب ضعیف هستند. تکنیک‌های خوشه‌بندی مناسب‌تر برای چنین توزیع‌هایی، آن‌هایی هستند که مفروضات کروی ایجاد نمی‌کنند و می‌توانند شکل و چگالی خوشه‌ها را در نظر بگیرند، مانند خوشه‌بندی فضایی مبتنی بر چگالی برنامه‌ها با نویز (DBSCAN) یا خوشه‌بندی طیفی، که هر دوی آنها می‌توانند کار کنند. اشکال دلخواه مانند دایره های متحدالمرکز.

در خوشه‌بندی دایره‌های متحدالمرکز، مراکز خوشه‌ها در اطراف یکدیگر قرار دارند، زیرا الگوریتم C-means فازی سعی می‌کند تابع هدف را که مجموع فاصله‌های مجذور بین هر نقطه داده و مرکز خوشه متناظر آن است، به حداقل برساند. توسط مقادیر عضویت

با این حال، این تابع هدف، شکل یا چگالی خوشه‌ها را که برای هر دایره متفاوت است، در نظر نمی‌گیرد. بنابراین، الگوریتم نمی تواند بین دایره ها تمایز قائل شود و مراکز خوشه را به طور تصادفی در محدوده داده ها اختصاص می دهد.

یک راه حل ممکن برای این مشکل استفاده از اندازه گیری فاصله متفاوت است، مانند فاصله Mahalanobis، که می تواند واریانس و همبستگی داده ها را توضیح دهد. راه حل دیگر استفاده از یک تکنیک خوشه بندی متفاوت است، مانند خوشه بندی مبتنی بر چگالی، که می تواند خوشه هایی با شکل و اندازه دلخواه را بر اساس تراکم محلی نقاط داده شناسایی کند.

برای داده‌های معمولی توزیع شده (توزیع شده گاوسی)، خوشه‌بندی فازی C-Means (FCM) می‌تواند با این فرض که خوشه‌های داده کروی یا بیضی شکل هستند و هر خوشه تقریباً واریانس یکسانی دارد، کاملاً مؤثر عمل کند. در یک توزیع تصادفی معمولی، نقاط داده تمایل دارند به صورت متقارن حول یک میانگین مرکزی جمع شوند و شکل کلاسیک منحنی زنگی را در یک بعد ایجاد کنند که به اشکال "کروی" در ابعاد بالاتر تعمیم می یابد.

چرا FCM می تواند برای داده های توزیع شده معمولی مناسب باشد

خوشه های همسانگرد: در توزیع های نرمال، خوشه ها تمایل دارند همسانگرد باشند - به این معنی که همه جهات واریانس یکسانی را نشان می دهند که منجر به شکل کروی یا بیضوی می شود - که با فرضیات FCM همسو می شود.

محاسبات مرکز خوشه: FCM مراکز خوشه را با وزن کردن هر نقطه داده با درجه عضویت آن در هر خوشه محاسبه می کند. این به خوبی با داده های توزیع عادی که در آن مراکز خوشه در میانگین نقاط داده قرار دارند، همسو می شود.

Soft Assignments: ماهیت توزیع های نرمال به گونه ای است که داده ها در نزدیکی میانگین متراکم تر و با دور شدن شما پراکنده تر می شوند. تخصیص عضویت نرم به کار گرفته شده توسط FCM می‌تواند درجاتی را که نقاط به خوشه‌ها تعلق دارند، نشان دهد، که با کاهش تدریجی چگالی به دور از میانگین در یک توزیع نرمال هماهنگ است.

مقاوم به همپوشانی: توزیع‌های معمولی اغلب دارای مناطق همپوشانی هستند که واریانس‌ها مشابه هستند و میانگین‌ها خیلی از هم دور نیستند. منطق فازی FCM با ارائه احتمالات (عضویت) که نشان می‌دهد تا چه اندازه نقاط به خوشه‌های همپوشانی تعلق دارند، چنین همپوشانی‌هایی را به خوبی مدیریت می‌کند.

در حالی که FCM هنگام اعمال بر داده‌های توزیع شده معمولی به خوبی کار می‌کند - جایی که هر خوشه در اطراف یک نقطه مرکزی در فضای ویژگی ایجاد می‌شود - ممکن است همچنان با چالش‌های خاصی روبرو شود:

حساسیت به نویز: FCM به نویز و نویز حساس است. از آنجایی که توزیع های نرمال می تواند شامل نقاط پرت باشد (بسته به انحراف استاندارد)، این می تواند با تخصیص خوشه های دقیق تداخل داشته باشد.

فرض واریانس برابر: اگر خوشه‌ها واریانس‌های متفاوتی داشته باشند، FCM ممکن است آنها را به‌طور دقیق خوشه‌بندی نکند، زیرا واریانس مشابهی را در بین خوشه‌ها پیش‌فرض می‌گیرد.

داده های با ابعاد بالا: مانند بسیاری از تکنیک های خوشه بندی، FCM ممکن است به دلیل "نفرین ابعاد" در فضاهای با ابعاد بالا عملکرد خوبی نداشته باشد.

به طور خلاصه، FCM برای توزیع داده‌های تصادفی معمولی مناسب است، زیرا ویژگی‌های اساسی این توزیع با مفروضاتی که خوشه‌بندی FCM بر اساس آن‌ها بنا شده است، همسو است. با این حال، درک ویژگی‌های داده‌ها و شکل‌های خوشه ضروری است، زیرا ممکن است نیاز به تنظیمات یا اتخاذ سایر تکنیک‌های خوشه‌بندی برای مجموعه داده‌های خاص داشته باشد.

الگوریتم خوشه‌بندی فازی C-Means (FCM) عموماً برای داده‌های توزیع‌شده گاوسی به دلیل ویژگی‌های ذاتی چنین داده‌هایی و نحوه عملکرد FCM مؤثر است. خوشه های داده گاوسی یا به طور معمول توزیع شده دارای چندین ویژگی هستند که به خوبی با رویکرد FCM برای خوشه بندی مطابقت دارد:

۱. **شکل کروی/بیضی**: توزیع های گاوسی در فضای چند بعدی تمایل به تشکیل خوشه هایی دارند که شکل کروی یا بیضی دارند. FCM فرض می‌کند که خوشه‌ها همسانگرد هستند، به این معنی که همه ابعاد واریانس یکسانی دارند، بنابراین به طور طبیعی با خوشه‌هایی که تقریباً کروی هستند به خوبی کار می‌کند.

۲. **گرایش مرکزی**: توزیع های گاوسی حول یک مقدار متوسط متمرکز می شوند و FCM برای یافتن مرکز خوشه ها با به حداقل رساندن مجموع وزنی تفاوت های مجذور بین نقاط داده و مرکز طراحی شده است. این با گرایش مرکزی توزیع‌های گاوسی مطابقت دارد و به FCM اجازه می‌دهد تا مرکز یک خوشه معمولی توزیع شده را به طور موثر تعیین کند.

۳. **عضویت نرم**: توزیع های گاوسی دارای نقاطی هستند که به احتمال زیاد نزدیک به میانگین و احتمال کمتری دارند زیرا دورتر هستند. FCM احتمال عضویت را به هر نقطه داده برای هر خوشه بر اساس فاصله آن تا مرکز تعیین می کند، که به خوبی با احتمال یک نقطه داده متعلق به یک توزیع گاوسی خاص مطابقت دارد.

۴. **استحکام نسبت به همپوشانی**: FCM در مدیریت خوشه های همپوشانی مهارت دارد، که اغلب در مورد توزیع های گاوسی صدق می کند، به خصوص زمانی که میانگین ها به یکدیگر نزدیک هستند و واریانس ها خیلی زیاد نیستند. عضویت های فازی در FCM به یک نقطه داده اجازه می دهد تا به خوشه های متعدد با درجات مختلف عضویت تعلق داشته باشد که منعکس کننده همپوشانی ها در توزیع داده ها است.

۵. **همگنی واریانس**: اگر خوشه های مجموعه داده دارای واریانس های همگن باشند، برای FCM مفید است زیرا الگوریتم تفاوت های واریانس بین خوشه ها را در نظر نمی گیرد. این ویژگی با بسیاری از مجموعه داده‌های توزیع شده گاوسی در دنیای واقعی که واریانس درون خوشه‌ها تقریباً یکسان است، همسو می‌شود.

از آنجایی که توزیع‌های گاوسی در داده‌های دنیای واقعی کاملاً رایج هستند، مکانیسم FCM به خوبی تنظیم شده است تا ساختار آنها را به تصویر بکشد، و آن را به انتخابی مناسب برای این موقعیت‌ها تبدیل می‌کند. با این حال، مهم است که توجه داشته باشید که در حالی که FCM می تواند

لگوریتم خوشه‌بندی فازی C-Means (FCM) عموماً برای داده‌های توزیع‌شده گاوسی به دلیل ویژگی‌های ذاتی چنین داده‌هایی و نحوه عملکرد FCM مؤثر است. خوشه های داده گاوسی یا به طور معمول توزیع شده دارای چندین ویژگی هستند که به خوبی با رویکرد FCM برای خوشه بندی مطابقت دارد:

۱. **شکل کروی/بیضی**: توزیع های گاوسی در فضای چند بعدی تمایل به تشکیل خوشه هایی دارند که شکل کروی یا بیضی دارند. FCM فرض می‌کند که خوشه‌ها همسانگرد هستند، به این معنی که همه ابعاد واریانس یکسانی دارند، بنابراین به طور طبیعی با خوشه‌هایی که تقریباً کروی هستند به خوبی کار می‌کند.
۲. **گرایش مرکزی**: توزیع های گاوسی حول یک مقدار متوسط متمرکز می شوند و FCM برای یافتن مرکز خوشه ها با به حداقل رساندن مجموع وزنی تفاوت های مجذور بین نقاط داده و مرکز طراحی شده است. این با گرایش مرکزی توزیع‌های گاوسی مطابقت دارد و به FCM اجازه می‌دهد تا مرکز یک خوشه معمولی توزیع شده را به طور موثر تعیین کند.
۳. **عضویت نرم**: توزیع های گاوسی دارای نقاطی هستند که به احتمال زیاد نزدیک به میانگین و احتمال کمتری دارند زیرا دورتر هستند. FCM احتمال عضویت را به هر نقطه داده برای هر خوشه بر اساس فاصله آن تا مرکز تعیین می کند، که به خوبی با احتمال یک نقطه داده متعلق به یک توزیع گاوسی خاص مطابقت دارد.

از آنجایی که توزیع‌های گاوسی در داده‌های دنیای واقعی کاملاً رایج هستند، مکانیسم FCM به خوبی تنظیم شده است تا ساختار آنها را به تصویر بکشد، و آن را به انتخابی مناسب برای این موقعیت‌ها تبدیل می‌کند. با این حال، توجه به این نکته مهم است که اگرچه FCM می‌تواند برای توزیع‌های گاوس مانند مؤثر باشد، اما برای خوشه‌هایی با شکل غیرکروی، اندازه‌های مختلف یا چگالی‌های متفاوت کمتر مناسب است. همچنین ممکن است با داده های با ابعاد بالا به دلیل نفرین ابعاد، عملکرد خوبی نداشته باشد و عملکرد آن می تواند در حضور نویز و نقاط دورافتاده کاهش یابد. در نتیجه، پیش پردازش دقیق و در نظر گرفتن ویژگی های مجموعه داده قبل از انتخاب FCM یا هر الگوریتم خوشه بندی بسیار مهم است.

بنابراین اینکه این الگوریتم چه مزایا و چه معایب دارد و اینکه برای چه دیتاست هایی مناسب تر است و نخوه ی رویکردمان با ان هارا مورد بررسی قرار داده و اینکه مراککز در حالت توزیع گوسین توزیع بهتری دارند تا حالت دایره ی و دلایل ان را بررسی کردیم همچنین یک متریک برای ارزیابی تعداد خوشه انتخابی معرفی کردیم که مخصوصا زمانی که اطلاعاتی از ساختار دیتا ست ها مانند این مسئله نداریم می توانی استفاده کنیم تا بتوانیم بهترین تعاد خوشه مورد نظر را پیدا کنیم و داده را روی ان فیت کنیم.
اینکه مرکز ها به صورت دایره ای کنار هم قرار می گیرند د دیتاست دوم و اختلاف زیادی از خود خوشه ها دارند اما در دیتاست اول در بین خوشه قرار می گیرند و با فاصله از یکدیگرند
برای حل دو مسئله اول از چت جی پی تی و بینگ کمک گرفتم.

سوال ۳

سؤال ۳) ۲ نفر با وزن‌های ۵۵kg و ۹۵kg و سن ۳۵,۴۵ :

$$\mu_{thin} = \begin{cases} 1 & \mu \in [0,25] \\ \left(1+\left(\frac{u-25}{100}\right)^2\right) & \mu \in [25,150] \end{cases} \qquad \mu_{fat} = \begin{cases} 0 & \mu \in [0,50] \\ 1-\left(\frac{u-150}{100}\right)^2 & \mu \in [50,150] \end{cases}$$

الف) نفر دوم نسبتاً جوان‌تر از نفر اول است.

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

بررسی درستی این دو جمله :

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

در این جا دو متغیر زبانی داریم وزن = $\bar{A}$ سن = $\bar{B}$:

ابتدا μ_{fat} ، μ_{thin} و μ_{young} را برای هر دو فرد مورد نظر حساب می‌کنیم. سپس برای موارد نسبتاً deletion و موارد خیلی Concentration استفاده می‌کنیم. یعنی به ترتیب استفاده از $\sqrt{\ }$ و $(\ )^2$. پس برای مورد "تر" از این فرمول استفاده می‌کنیم.

$$\mu'_{(\alpha,\beta)} = \frac{\alpha - \beta + 1}{2} \qquad \begin{array}{l} \alpha=1 , \beta=0 \rightarrow \mu'(\alpha,\beta)=1 \\ \alpha=0 , \beta=1 \rightarrow \mu'(\alpha,\beta)=0 \end{array}$$

برای پیاده‌سازی AND از min . برای پیاده‌سازی OR از Max طبق اصول عمومی استفاده می‌کنیم در اینجا تابع عضویت. یک مرحله اعداد را گرد می‌کنیم.

$$\mu_{thin}(\text{نفر اول}) = \left(1+\left(\frac{55-25}{5}\right)^2\right)^{-1} \xrightarrow{\simeq} 0.02702 \simeq 0.027$$

$$\mu_{thin}(\text{نفر دوم}) = \left(1+\left(\frac{95-25}{5}\right)^2\right)^{-1} \xrightarrow{\simeq} 0.005057 \simeq 0.005$$

$$\mu_{fat}(\text{نفر اول}) = 1-\left(\frac{55-150}{100}\right)^2 \xrightarrow{\simeq} 0.0975$$

$$\mu_{fat}(\text{نفر دوم}) = 1-\left(\frac{95-150}{100}\right)^2 \xrightarrow{\simeq} 0.6975$$

$$\mu_{young}(\text{نفر اول}) = \left(1+\left(\frac{45-25}{5}\right)^2\right)^{-1} \xrightarrow{\simeq} 0.0588 \simeq 0.059$$

$$\mu_{young}(\text{نفر دوم}) = \left(1+\left(\frac{35-25}{5}\right)^2\right)^{-1} \xrightarrow{\simeq} 0.2$$

پاسخ: مقدار بستنی و چای زرد جوان را به ترتیب

۱) هر ۲ بستنی چای را از سر اول

۲) هر ۲ چای را از سر اول

$$\mu_{\text{fairly hot}}^{\text{چای شیرین}}(\text{زیاد}) = \sqrt{\mu_{hot}(\text{زیاد})} = \sqrt{0.0975} \simeq 0.3122$$

$$\mu_{\text{fairly hot}}(\text{کم}) = \sqrt{\mu_{hot}(\text{کم})} = \sqrt{0.6975} \simeq 0.8351$$

$$1)\ \mu'_{\alpha,\beta\ hot} = \frac{0.8351 - 0.3122 + 1}{2} \simeq 0.7615 = 0.76$$

$$2)\ \mu'_{\alpha,\beta\ young} = \frac{0.02 - 0.059 + 1}{2} \simeq 0.4805 = 0.48$$

$$\sigma = \text{چای زرد جوان} = \min(0.76, 0.48) = 0.48$$

مقدار درستی جمله اول

$$\mu_{\text{خیلی داغ}}(\text{زیاد}) = (0.027)^2 = 0.000729 \simeq 0.00073$$

$$\mu_{\text{بستنی}}(\text{سرد}) = \sqrt{0.02} = 0.141421 \simeq 0.1414$$

$$\mu_{\text{خیلی داغ است و بستنی سرد}} = \min(0.00073, 0.1414) = 0.00073$$

مقدار درستی جمله دوم