تمرین سری پنجم

هوش محاسباتی

بکتاش انصاری

99521082

سوال ۱:

برای این سوال گفته شده بود که اگر پاندول مورد نظر به مقدار X عه بالای 0.99 رسید , در حالی که قدر مطلق سرعت بازو به کمتر از 1.5 برسه مسئله رو حل شده در نظر بگیرید. اما من در نظر گرفتم که پاندول قبل از 500 مرحله به نقطه بالا برسد و ثابت بماند. (چون در حالت اول اگر پاندول با سرعت کم از نقطه مورد نظر رد شود هم انگار مسئله حل شده در صورتی که اینطور نیست)

برای حل این سوال مراحل زیر را به ترتیب اجرا کردم و در نهایت یک ویدیو از اجرای بازی درون فایل زیپ قرار داده‌ام.

مراحل:

1. ابتدا کتابخانه های مورد نظر نظیر gym و scikit-fuzzy و matplotlib را نصب کردم
2. متغیر های زبانی را با توجه به ورودی های سوال طرح تعریف کردم.

این متغیر ها شامل مختصات x و y و سرعت زاویه‌ای سر پاندول و همینطور و action مورد نظر که میزان گشتاور است می‌باشد.

```
y_position = ctrl.Antecedent(np.arange(-1, 1.0 + 0.1, 0.1), 'y_position')

x_position = ctrl.Antecedent(np.arange(-1, 1.0 + 0.1, 0.1), 'x_position')

ang_velocity = ctrl.Antecedent(np.arange(-8, 8 + 0.8, 0.8), 'ang_velocity')

# define the output (action)
action = ctrl.Consequent(np.arange(-2, 2 + 0.2, 0.2), 'action')
```

بازه این متغیرهای زبانی بر اساس فضای حالت مسئله در نظر گرفته شده و گام های xو y را 0.1 و ang_velocity را 0.8 و action را 0.2 در نظر گرفتم.

3. حال برای هر کدام از این متغیرها بازه‌هایی را در نظر گرفتم.

نکته قابل توجه در این مسئله این بود که برعکس معمول متغیر x ما بصورت سینوسی و متغیر y ما بصورت کسینوسی بود و همینطور متغیر y ما در سمت چپ دایره مثلثاتی مقدار مثبت و در سمت راست آن مقداری منفی دارد.

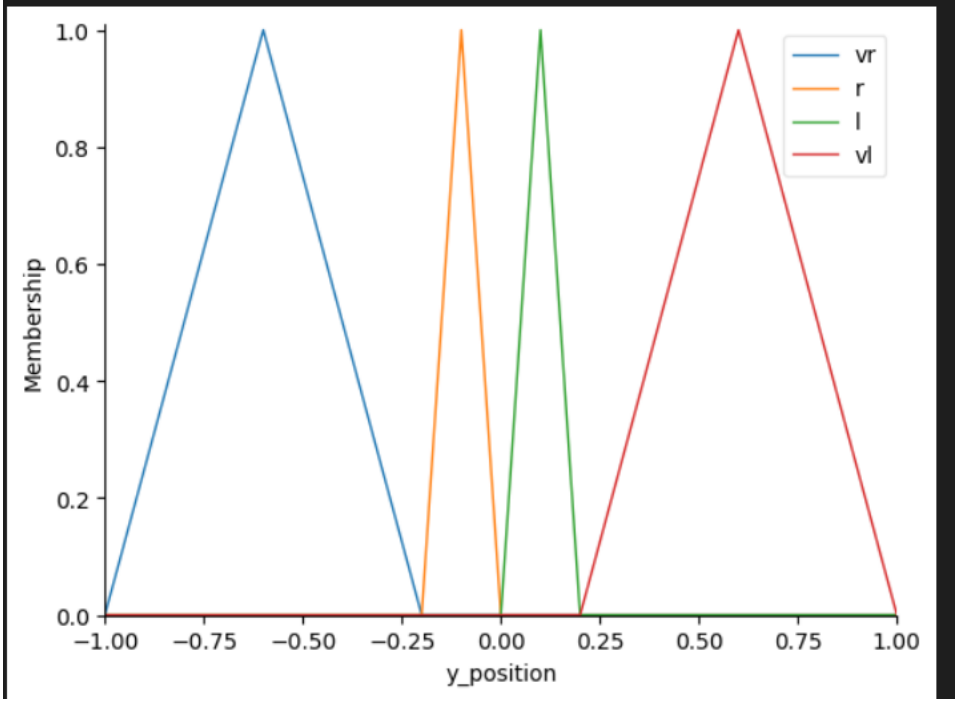
مقادیر سرعت زاویه و گشتاور نیز در مقادیر مثبت بصورت پادساعتگرد و در مقادیر منفی بصورت ساعتگرد می‌باشد.

من برای حل این سوال برای هر بازه از توابع عضویت مثلثی fuzz.trimf استفاده کردم. که برای هر متغیر داریم:

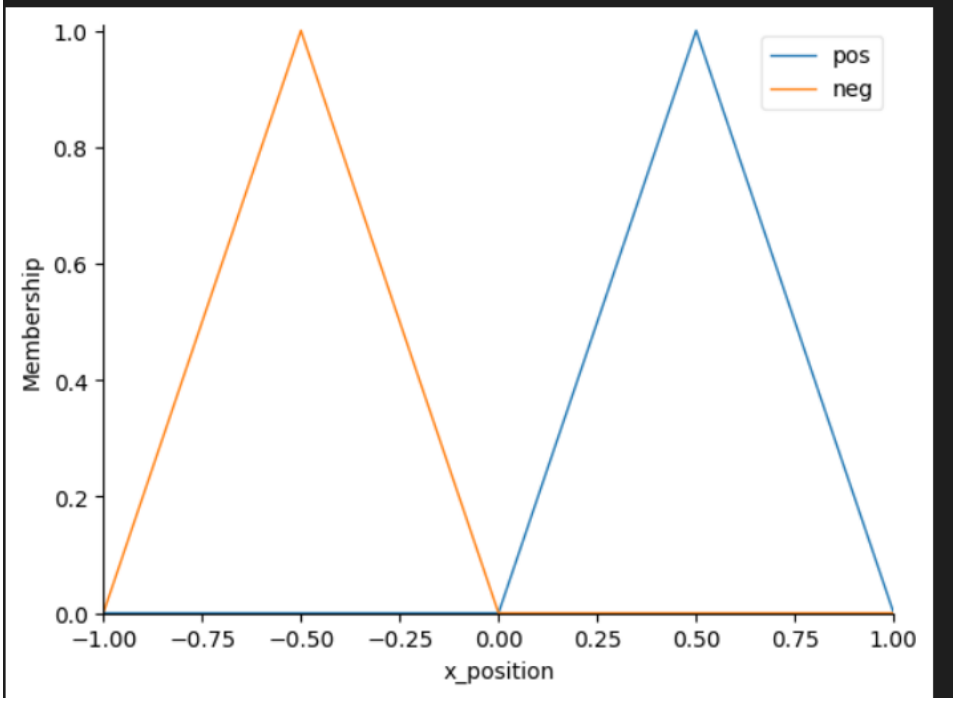
- برای y_position : برای این متغیر مقادیر very right , right , very left, left را تعریف کردم. دلیل آن هم این است که میخواستم وقتی پاندول به مقادیری نزدیک صفر رسید. (یعنی سر پاندول نزدیک محور عمودی باشد) rule های خاصی را تعریف کنم.
- برای x_position : برای این متغیر نیز تنها دو مقدار مثبت و منفی را تعریف کردم تا صرفا مشخص شود سر پاندول در بالای محور افقی است یا پایین آن.
- برای ang_velocity: مقادیر clock-wise_high, clock_wise_medium, clock_wise_low, counter_clock_wise_high , ... را تعریف کردم که ابتدا بفهمم سر پاندول را حال حرکت در چه سمتی است و همینطور متوجه شوم با چه سرعتی به آن سمت حرکت میکند و به همین دلیل میزان سرعت را به سه بخش کم متوسط زیاد تقسیم کردم. برای این متغیر دو مقدار مثبت و منفی نیز اعمال کردم که هر جا حس کردم نیازی به میزان آن ندارم تنها جهت آن را بدانم.
- برای action نیز مانند ang_velocity ابتدا جهت و میزان گشتاور اعمالی را با سه مقدار کم زیاد و متوسط اعمال کردم. همینطور یک action عه nothing نیز با مقادیر بسیار کم تعریف کردم که هرجا نیاز نیست عملی انجام شود از آن استفاده کنم.

بازه های هر کدام از متغیرهای زبانی به شکل زیر است:

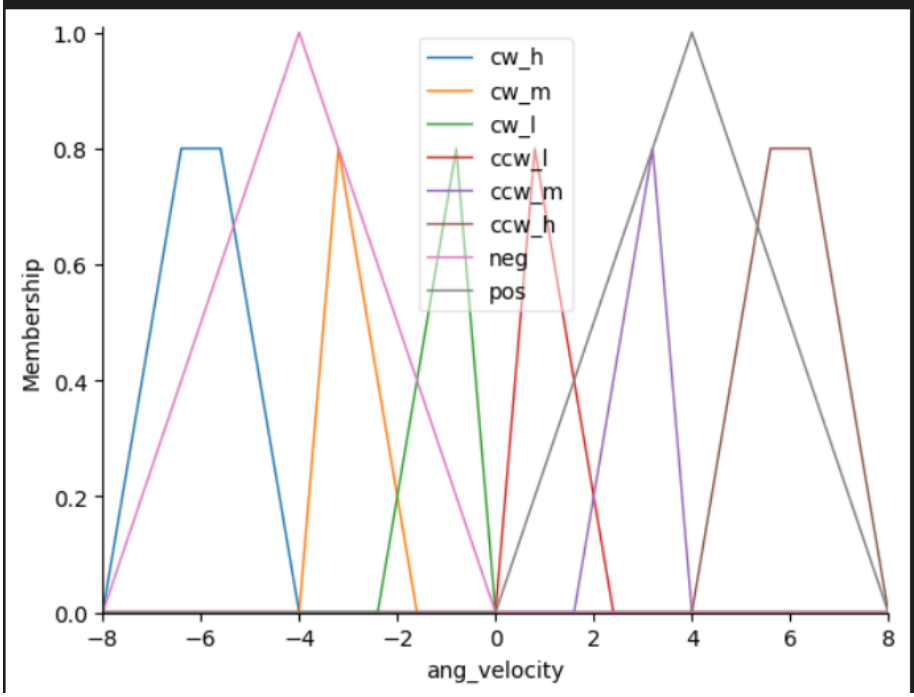
```
y_position.view()
```



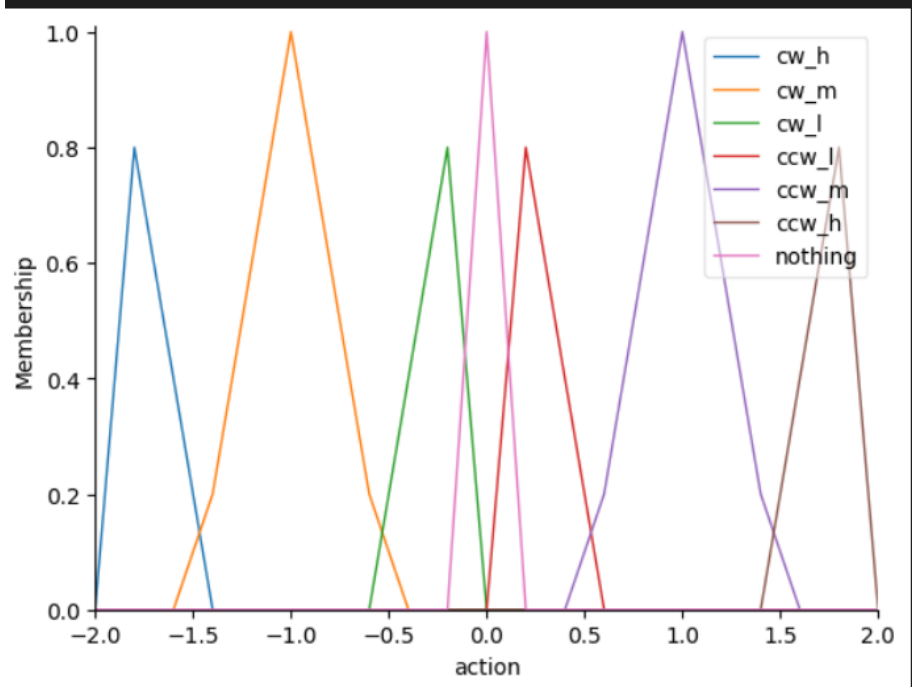

```
x_position.view()
```



```
ang_velocity.view()
```




```
action.view()
```

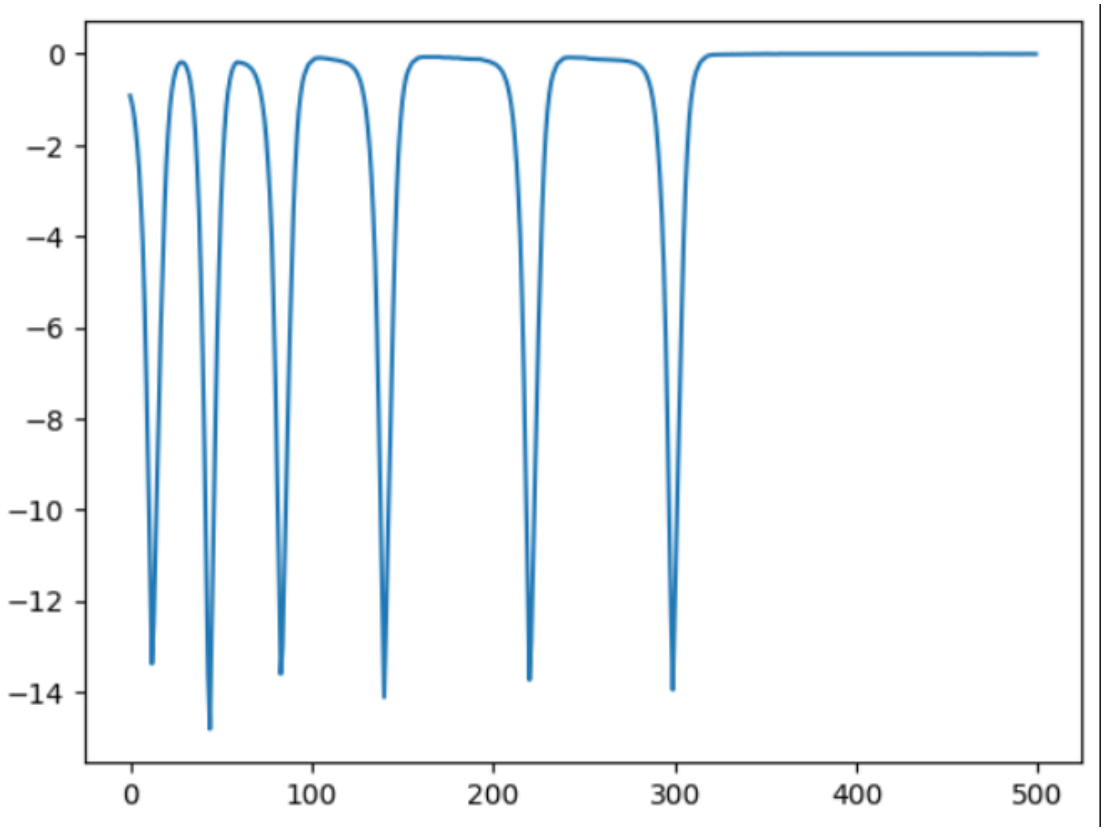


- برای تعریف قوانین: ابتدا دایره مثلثاتی را به 4 بخش تقسیم کردم و برای هر کدام در برخی حالت های پاندول قوانینی مطرح کردم. و برای بقیه حالت ها نیز از action عه nothing استفاده کردم.

  در کل 20 قانون ایجاد کردم و قوانین را با آزمون و خطا اصلاح کردم.

  تمامی 20 قانون با کامنت درون کد مشخص است. با توجه به توابع عضویت تعریف شده فهم قوانین بسیار آسان است.

- در نهایت نیز مسئله را با حداکثر 500 گام اجرا کردم(در عمده اجراها در 200 الی 300 گام پاندول در موقعیت قرار میگیرد و آن موقعیت را حفظ میکند.)
- نمودار پاداش‌های یکی از اجراها به شکل زیر است: (ویدیو همین اجرا درون فایل زیپ موجود است.)



همانطور که در نمودار و ویدیو قابل مشاهده است. تا گام ها نزدیک به 300 پاندول تلاش به سکون در موقعیت موردنظر میکند و از گام 300 به بعد موقعیت خود را حفظ کرده و همانطور باقی میماند. و مقدار reward به بیشترین حالت خود میرسد.

سوال ۲:

الف)

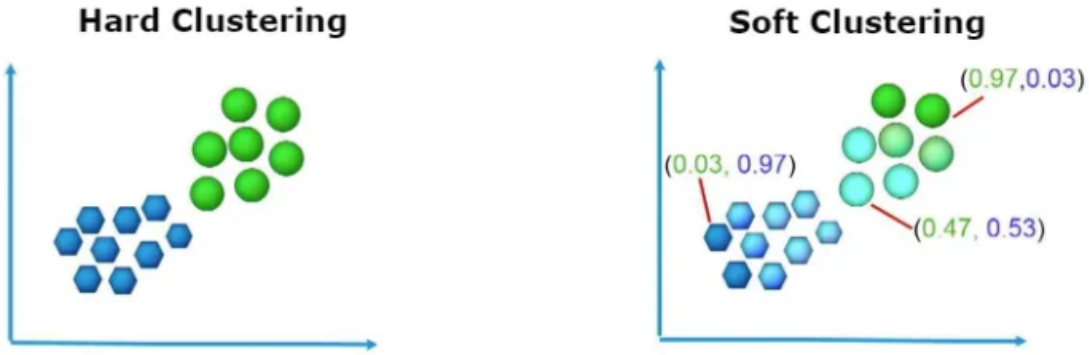
در حالت کلی ما میتوانیم مسئله clustring را با استفاده از سیستم فازی نیز حل کنیم.
در اصل ما می‌توانیم با استفاده از قوانین فازی و تابع عضویت هر نقطه را به یک cluster نسبت دهیم. به این شکل که برای هر نقطه یک مقدار عضویت برای هر cluster وجود دارد و هر چه این مقدار بیشتر باشد نشان دهنده این است که آن نقطه بیشتر عضو آن خوشه می‌باشد.
یعنی هرچه یک نقطه از داده به یک خوشه نزدیک‌تر باشد مقدار تابع عضویت آن بیشتر است.
در این نوع مسئله یک پارامتر m وجود دارد که درجه فازی بودن مسئله را مشخص میکند. می‌توان اینطور درنظر گرفت که مقادیر متفاوت m باعث ایجاد خوش های متفاوت میشود.

تفاوت K-means و Fuzzy C-means :

در حالت کلی ما می‌توانیم بگوییم که k-means یک hard-clustering میباشد. چون یک نقطه برای ر خوشه دو حالت بیشتر ندارد. یا عضو آن است یا عضو آن نیست. به بیان دیگر مقادیر آن crisp هستند.

اما در fuzzy c-means میزان عضویت مطرح است و یک نقطه میتوان کم عضو یک خوشه یا زیاد عضو یک خوشه باشد.

به این شکل توجه کنید:



در اصل هر نقطه یک وزن عضویت برای یک خوشه دارد.

دومین تفاوت **سرعت** اجرای دو الگوریتم است. الگوریتم فازی کند تر از الگوریتم k-means است. چرا؟ چون کار بیشتری انجام میدهد. در k-means ما تنها یک محاسبه فاصله انجام میدهیم و مشخص میشود که نقطه مربوط به کدام خوشه است. اما در فازی باید مقدار عضویت تمام نقاط محاسبه شوند.

مزایای fuzzy c-means :

+ در مواقعی که overlap در نقاط زیاد است می‌تواند بهتر از k-means عمل کند.
+ عضویت هر نقطه 0 و 1 نیست.

معایب آن:

+ هزینه محاسباتی بالا
+ پرفورمنس آن می‌تواند به مقادیر اولیه اعضا و خوشه ها وابسته باشد.

ب)

با استفاده از الگوریتم FCM خوشه بندی را برای مقادیر c بین 2 تا 10 انجام دادیم و نتایج و گراف‌ها درون نوت‌بوک قابل مشاهده است.

معیار FPC:

معیار FPC یک معیار ارزیابی است که در متدهای خوشه‌بندی مبتنی بر منطق فازی (Fuzzy Clustering)، به ویژه در الگوریتم خوشه‌بندیFCM، استفاده می‌شود. این معیار به ارزیابی کیفیت تقسیم فازی داده‌ها به خوشه‌ها می‌پردازد. در الگوریتم FCM، هر داده به هر خوشه با درجه عضویت فازی بیشتر اختصاص می‌یابد. مقدار FPC نشان‌دهنده اندازه واقعی تقسیم فازی است و از فرمول زیر محاسبه می‌شود:

$$FPC = \frac{\sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^{m}}{n^2}$$

در این فرمول:

- مقدار n برابر است با تعداد داده‌ها
- مقدار c تعداد خوشه ها
- مقدار uij درجه عضویت داده i در خوشه j است.

- مقدار m پارامتر فازی است که مقداری بین 1 و 2 انتخاب می‌شود.

هرچه مقدار FPC بیشتر باشد، تقسیم فازی بهتری انجام شده است. بازه مقدار این معیار بین 0 تا 1 است که هرچه این مقدار به 1 نزدیک‌تر باشد تقسیم فازی بهتری انجام شده است. و هر چه به صفر نزدیک تر باشد به معنای عدم تقسیم فازی است.

سوال۳:

سوال ۳ : الف)

$\mu_{thin} \rightsquigarrow \{(55, 0{,}027), (95, 0{,}005)\} = thin$

$\mu_{fat} \rightsquigarrow \{(55, 0{,}0975), (95, 0{,}4975)\} = fat$

$\mu_{young} \rightsquigarrow \{(45, 0{,}0585), (40, 0{,}02)\} = young$

$$a > b \Rightarrow \frac{a-b+1}{2}$$

برای ۲ درجه از این روش استفاده کردم :

ابتدا تم دوم نسبتا" جوان تر از تم اول را محاسبه کنیم :

برای این کار ابتدا نسبتا" جوان رو محاسبه کنیم :

~~$\mu_{\text{نسبتا جوان تر}} = \mu_{\text{جوان تر}}^{1/2}$~~

$\mu_{relatively\ fat} = (\mu_{fat})^{1/2} = \{(55, 0{,}3121), (95, 0{,}8381)\}$

حال نسبتا" جوان تر را محاسبه کنیم (برای هر کدام که شود تم * نسبتا" جوان تر * از تم اول است)

$\mu_{relatively\ fatter} = \{(55, 0{,}2345), (95, 0{,}71855)\}$

حال برای تم دوم حداکثر آن برابر با 0,71855 می باشد .

حال سراغ جوان تر می رویم :

$\mu_{younger} = \{(45, 0{,}819), (40, +0{,}481)\}$

و برای تم دوم داریم : 0,481

و حال باید بین این ۲ حداکثر "min" بگیریم :

$\mu_{younger\ and\ relatively\ fatter} = min(0{,}71855, 0{,}481) = 0{,}481$

ب) برای محاسبه میزان درستی «اگر نفر اول خیلی لاغر باشد آنگاه نفر دوم نسبتاً جوان است» داریم:

ابتدا خیلی لاغر بودن را محاسبه می‌کنیم:

$$\mu_{very\ thin} = (\mu_{thin})^2 = \{(55, 0{,}000729), (95, 0{,}000025)\}$$

$$\mu_{very\ thin}(P_1) = 0{,}000729$$

حال میزان نسبتاً جوان را محاسبه می‌کنیم:

$$\mu_{relatively\ young} = (\mu_{young})^{1/2} = \{(45, 0{,}24), (40, 0{,}14)\}$$

$$\mu_{relatively\ young}(P_2) = 0{,}14$$

حال با توجه به فرمول مقابل مقدار Implication را محاسبه می‌کنیم:

$$a \rightarrow b = \min(1, 1 + b - a)$$

$$\mu_{very\ thin}(P_1) \rightarrow \mu_{relatively\ young}(P_2) = \min(1, 1 + 0{,}14 - 0{,}000729)$$

$$= 1$$