دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

گزارش ۹ درس هوش مصنوعی

پیاده‌سازی یک سیستم فازی به منظور کنترل یک مسئله با ورودی‌های زبانی

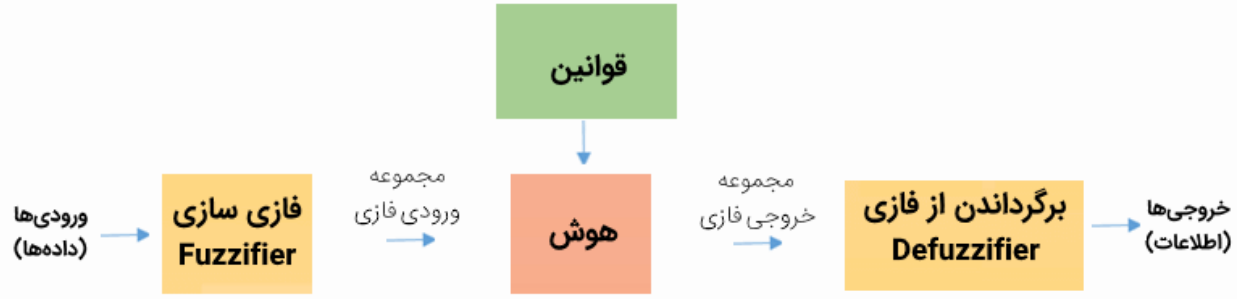
نگارش:
امیررضا رادجو

استاد
دکتر مهدی قطعی

تیر ۱۴۰۰

منطق فازی چیست؟

منطق فازی دارای چهار بخش اصلی است که در ادامه معرفی شده‌اند. همچنین در نمودار زیر نحوه ارتباط این بخش‌ها به خوبی دیده می‌شود.



قوانین پایه: این بخش، شامل همه قاعده‌ها و شرایطی است که به صورت «اگر...آنگاه» توسط یک متخصص مشخص شده‌اند تا قادر به کنترل تصمیمات یک «سیستم تصمیم‌گیری» (Decision-making system) باشند. با توجه به روش‌های جدید در نظریه فازی، امکان تنظیم و کاهش قواعد و قوانین بوجود آمده است به طوری که با کمترین قوانین می‌توان بهترین نتیجه را گرفت.

فازی سازی: در گام فازی سازی، ورودی‌ها به اطلاعات فازی تبدیل می‌شوند. به این معنی که اعداد و ارقام و اطلاعاتی که باید پردازش شوند، به مجموعه‌ها و اعداد فازی تبدیل خواهند شد. داده‌های ورودی که مثلا توسط حسگرها در یک سیستم کنترل، اندازه‌گیری شده‌اند، به این ترتیب تغییر یافته و برای پردازش برمبنای منطق فازی آماده می‌شوند.

موتور استنتاج یا هوش: در این بخش، میزان انطباق ورودی‌های حاصل از فازی سازی با قوانین پایه مشخص می‌شود. به این ترتیب براساس درصد انطباق، تصمیمات مختلفی به عنوان نتایج حاصل از موتور استنتاج فازی تولید می‌شود.

برگرداندن از فازی: در آخرین مرحله نیز نتایج حاصل از استنتاج فازی که به صورت مجموعه‌ها فازی هستند به داده‌ها و اطلاعات کمی و رقمی تبدیل می‌شوند. در این مرحله شما با توجه خروجی‌ها که شامل تصمیمات مختلف به همراه درصدهای انتطباق‌های متفاوتی هستند، دست به انتخاب بهترین تصمیم می‌زنید. معمولا این انتخاب برمبنای بیشترین میزان انطباق خواهد بود.

منطق فازی و احتمال

همانطور که گفته شد، در منطقی فازی درجه قطعیت یا میزان درستی یک گزاره توسط یک عدد در فاصله ۰ تا ۱ بیان می‌شود. در نظریه احتمال نیز برای وقوع یک پیشامد از عددی بین ۰ تا ۱ استفاده می‌کنیم. به این ترتیب به نظر می‌رسد که انطباقی بین این دو مفهوم و البته با کاربرد متفاوت وجود دارد. بنابراین اگر با دید نظریه احتمال بگوییم: «با احتمال ۹۰٪، یک فرد، عینکی هستند.»، می‌توان آن را در منطق فازی به صورت: «درجه عضویت فردی به گروه افراد عینکی برابر با ۰,۹ است.» نشان داد. به این ترتیب می‌توان گفت که احتمال، یک مدل ریاضی برای پدیده‌های نامشخص و تصادفی است و از طرفی منطق فازی نیز مدلی برمبنای ریاضیات است که برای تعیین حقیقت برای هر پدیده، از مقداری به عنوان «میزان درستی» (Truth Degree) استفاده می‌کند.

منطق فازی و منطق بولی

در این بخش نیز به مقایسه دو منطق فازی و منطق بولی که گاهی به منطق دیجیتال نیز معروف است، می‌پردازیم. در منطق بولی درستی هر گزاره براساس دو مقدار «درست» (True) و «نادرست» (False) تعیین می‌شود. در حالیکه در منطق فازی درستی هر گزاره منطقی براساس «درجه عضویت» (Degree of Membership) مشخص خواهد شد. البته گاهی با تفکیک گزاره‌ها به جملات ساده‌تر می‌توان ارزش درستی را به صورت منطق فازی تغییر دارد ولی نمی‌توان ارزش گزاره‌ها را در منطق فازی به صورت فقط دو وضعیت درست یا نادرست تبدیل کرد.

در منطق بولی، درستی یک گزاره و نقیض آن، همزمان برقرار نیست، درحالیکه در منطق فازی می‌توان براساس درجه عضویت، درستی هر دو گزاره و نقیض آن گزاره را مشخص کرد. به بیان دیگر به کمک ترکیب گزاره‌ها در منطق بولی گزاره $P \vee \sim P \equiv T$ است. به این معنی که ترکیب فصلی هر گزاره با نقیض خودش، یک گزاره همیشه درست یا تاتولوژی ایجاد خواهد کرد. در حالیکه در منطق فازی، ترکیب فصلی هر گزاره با نقیضش دارای درجه قطعیت است که لزوما یک گزاره همیشه درست نخواهد بود.

نظریه فازی و کلاسیک مجموعه‌ها

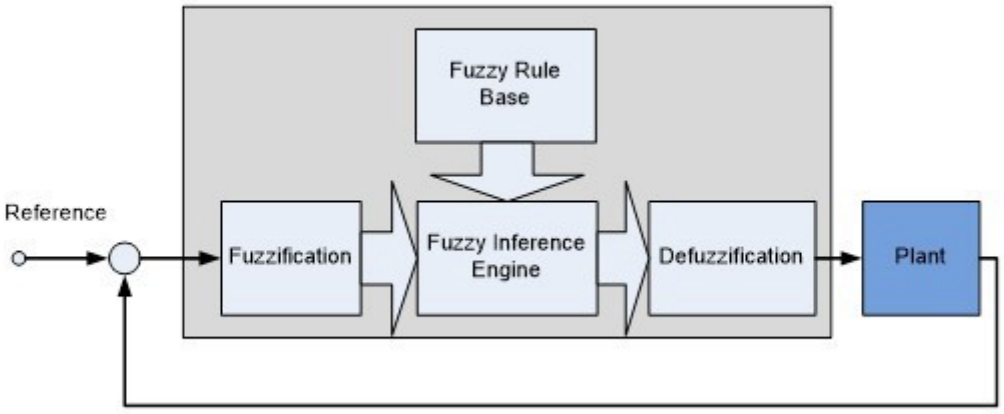
در نظریه کلاسیک مجموعه‌ها، براساس قانون عضویت مشخص می‌شود که نشان می‌دهد هر شئ به یک مجموعه تعلق دارد یا خیر. در نتیجه محدوده یک مجموعه و اعضای آن کاملا شفاف و قابل تعیین است. در حالیکه در نظریه مجموعه‌ها با رویکرد فازی، تعلق هر شئ به یک مجموعه با درجه عضویت که مقداری بین ۰ تا ۱ است تعیین می‌شود. برای مثال، اگر a یک شئ باشد، می‌تواند با درجه عضویت ۰,۳ به مجموعه A تعلق داشته باشد و با درجه عضویت ۰,۷ به مجموعه B متعلق باشد. بنابراین کران‌های یک مجموعه فازی به روشن، قابل تعیین نیست و نمی‌توان به طور قطع در مورد حدود یا اعضای یک مجموعه قضاوت کرد.

یک مثال مقایسه‌ای برای منطق فازی و منطق بولی

منطق بولی یا دیجیتال، در بسیاری از سیستم‌های باینری به کار می‌رود. البته منطق فازی نیز امروزه در سیستم‌های کنترل بسیار کارا و موثر ظاهر شده است. در ادامه به بررسی و مقایسه منطق فازی و منطق بولی با استفاده از یک مثال می‌پردازیم. فرض کنید یک سیستم فازی طراحی شده است که توسط آن می‌توان میزان امانت‌داری افراد را مشخص کرد. در مقابل براساس یک سیستم دیجیتال یا منطق بولی نیز امانت‌داری افراد اندازه‌گیری و تعیین می‌شود. به نمودار زیر و تفاوت نتایج حاصل از این دو سیستم توجه کنید.
مشخص است که نمی‌توان در مورد امانت‌داری افشین در منطق فازی به طور قطع نظر داد در حالیکه در منطق دیجیتال این کار به راحتی امکان‌پذیر است.

در این گزارش میخواهیم با استفاده از منطق فازی برای یک ربات که میخواهد در یک پیاده رو حرکت کند سرعت حرکت کردن را تعیین میکنیم.
مدلی که تعریف میکنیم به این شکل است که بنا به منطق فازی متغیرهای عددی و کیفی را دریافت کرده و بر اساس قوانین فازی که برایش تعریف شده است برای نوع حرکت این ربات و همچنین سرعت حرکت کردن آن تصمیم‌گیری می‌کند.

در منطق فازی روند کلی به شکل زیر خواهد بود:



مساله به این شکل می‌باشد که سه ورودی زیر توسط مدل دریافت شده و توسط این پارامترها میزان مناسب سرعت برای رباتی که در حال حرکت در خیابان است انتخاب میشود.

ورودی‌ها:

۱. تعداد افرادی که در خیابان حضور دارند و شلوغی خیابان:

میزان شلوغی خیابان‌ها به صورت کیفی تعیین میشود. این میزان به این شکل است که تقسیم‌بندی شلوغی خیابان‌ها بر اساس افرادی که در خیابان‌ها حرکت می‌کنند به سه بخش خلوت-معمولی-شلوغ تقسیم می‌شود و با توجه به چیزی که توسط سنسورها دریافت می‌شود در مدل ذخیره می‌شود.

۲.سرعت حرکت کردن افراد:

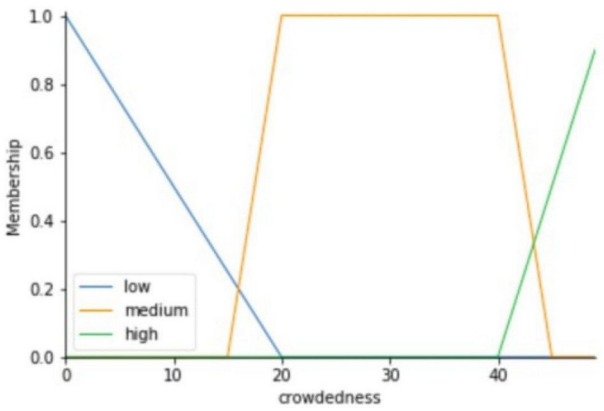
سرعت حرکت کردن افراد نیز توسط سنسور های ربات تعیین خواهد شد و به این شکل است که با ارزیابی حرکت انسان‌ها میزان سرعت آن‌ها به عنوان ورودی به مدل داده شود. میزان شلوغی خیابان‌ها میتواند در حالت‌های مختلف تفاوت داشته باشد به طور مثال در صورتی که این ربات در حال حرکت کردن در مترو باشد احتمالاً سرعت افراد بسیار زیاد است و اگر در یک کافی شاپ یا رستوران در حال حرکت باشد حرکت افراد بسیار آرام‌تر خواهد بود. این بخش را نیز به ۳ دسته طبقه بندی خواهیم کرد.

۳.میزان عجله ربات برای رسیدن به مقصد:

میزان عجله‌ای که ربات برای رسیدن به مقصد دارد یا درواقع میزان ریسکی که این ربات میخواهد به خرج بدهد تا در زمان زودتری به مقصد برسد می‌باشد که این بخش نیز به سه دسته بندی کم متوسط زیاد تقسیم‌بندی میشود.

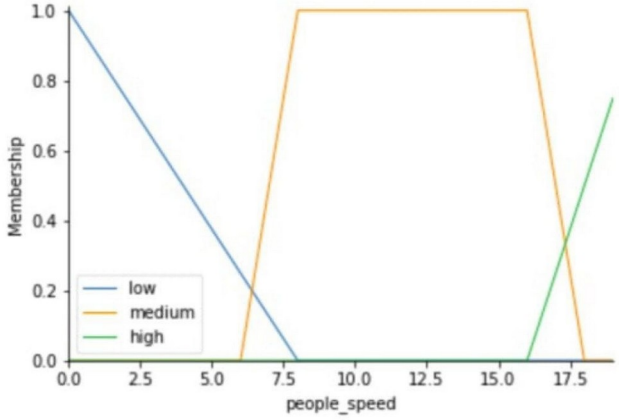
شلوغی محل در حال عبور:

همانطور که گفته شد برای این ویژگی یک متغیر در نظر گرفته و سه میزان کیفی مختلف را به آن نسبت میدهیم.
با توجه به تعداد افرادی که توسط سنسورهای ربات که در یک فاصله خاص از ربات قرار دارند تشخیص داده می‌شوند سه کمیت کیفی مختلف تخصیص داده میشود.
تعداد افرادی که مد نظر میباشد به طور استاندارد چیزی بین ۰ تا ۵۰ نفر خواهد بود.
تعداد افراد بین ۰ تا ۲۰ نفر در طبقه خلوت طبق یک نمودار مثلثی
افراد بین ۱۵ تا ۴۵ در طبقه متوسط طبق یک نمودار ذوزنقه‌ای
و افراد بین ۳۰ تا ۵۰ در طبقه شلوغ قرار خواهند داشت که نمودار مربوط به آن را در شکل زیر میبینید.



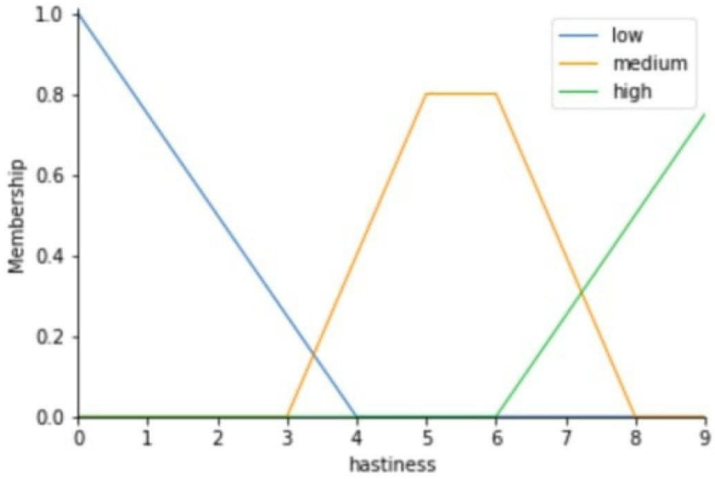
سرعت حرکت کردن افراد:

سرعت حرکت کردن افراد نیز یک ویژگی دیگر است که به سه دسته بندی آرام معمولی و سریع طبقه بندی شده است. این اطلاعات نیز توسط سنسور سرعت سنج که در ربات کار گذاشته شده است قرار دارد به دست می‌آید و به این طریق اطلاعات عددی توسط سنسور به دست خواهد آمد. فرم کلی و منطقی این نمودار بسیار شبیه به نمودار حالت قبل میباشد. نمودار های اول و سوم به شکل مثلثی و طبقه سرعت معمولی به شکل ذوزنقه شکل خواهد گرفت. که میتوانید نمودار مربوط به آن‌ها را نیز در شکل زیر مشاهده کنید.



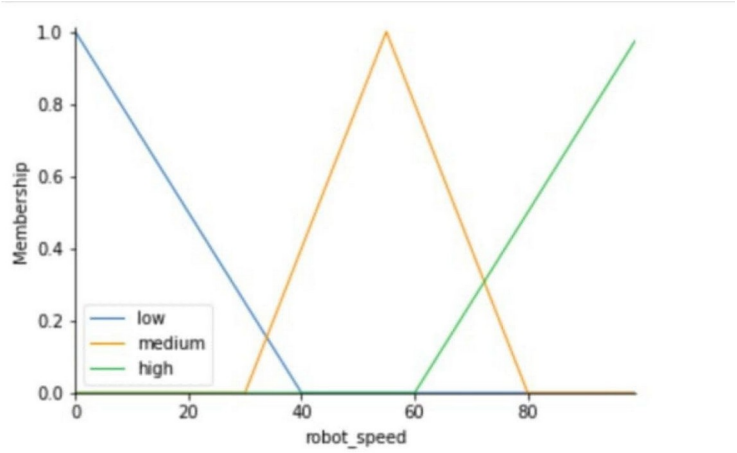
میزان عجله ربات:

این ویژگی درواقع طبق برنامه‌ریزی های موجود در ربات ما خواهد بود و سنسور خارجی برای آن موجود نیست. فرض کنید که ربات باید به مقصد خاصی برسد. توسط برنامه‌ای که در ربات وجود دارد و زمان مورد انتظار رسیدن ربات مقدار عجله آن برای رسیدن به مقصد تعیین خواهد شد.
این ویژگی سه حالت کم متوسط و زیاد را دارد که بر خلاف دو نمودار قبل در اینجا هر سه نمودار به فرم مثلثی خواهند بود که نمودار مربوط به آن‌ها را نیز در شکل زیر میوانید مشاهده نمایید.



سرعت حرکت ربات:

با توجه به ورودی های مد نظر سرعت حرکت ربات انتخاب خواهد شد.
این مقدار درواقع مقدار خروجی مساله ما خواهد بود که به این شکل است که ربات مورد نظر در کدام یک از سرعت های مورد نظر حرکت کند. این مقدار نیز با سه حالت مختلف آرام معمولی و سریع بیان خواهد شد که با توجه به مثال‌های مشابه موجود این حالت دارای نموداری به شکل ۳ نمودار گاوسی خواهد بود.



قوانین این مدل:

با توجه به مقدار کیفی که از پارامتر هایی که در مدل وجود دارد داریم میتوانیم حالت‌های مختلف را در نظر داشته باشیم و با کمک این قوانین که در پیاده‌سازی با جزییات بیشتری آن‌ها را میبینیم میتوانیم برای سرعت ایده‌آل ربات خود تصمیم گیری نماییم.

پیاده سازی:

در مرحله اول نیاز است که کتابخانه skfuzzy را ایمپورت کنیم چون اکثریت کار ما در این پروژه به کمک این کتابخانه خواهد بود.

در ادامه به ازای هر ویژگی که داریم یک متغیر ساخته و نوع آن را مشخص میکنیم. متغیر های ورودی ما Antecedent و متغیر های خروجی ما Consequent خواهند بود.
با استفاده از کد زیر این متغیر ها را تعریف میکنیم و برای آن‌ها بازه مشخص میکنیم.

```
crowdedness = ctrl.Antecedent(np.arange(0, 50, 1), 'crowdedness')
people_speed = ctrl.Antecedent(np.arange(0, 20, 1), 'people_speed')
hastiness = ctrl.Antecedent(np.arange(0, 10, 1), 'hastiness')

robot_speed = ctrl.Consequent(np.arange(0, 100, 1), 'robot_speed')
```

حال در مرحله بعدی در هر متغیر برای بازه موجود در آن طبقه بندی های مختلف را انجام میدهیم و هر بازه از آن را به یک دسته خاص اختصاص میدهیم و همچنین نوع نمودار آن را بسته به مثلثی و یا ذوزنقه ای بودن انتخاب میکنیم(در اینجا نوع دیگری مورد نیاز نبود)

```
crowdedness['low'] = fuzz.trimf(crowdedness.universe, [0, 0, 20])
crowdedness['medium'] = fuzz.trapmf(crowdedness.universe, [15, 20, 40, 45])
crowdedness['high'] = fuzz.trimf(crowdedness.universe, [40, 50, 50])
```

این عمل را برای قسمت‌های مختلف کد انجام میدهیم. نمودار مربوط به هر کدام را نیز میتوان با تابع view مشاهده کرد.

در ادامه پس از تعیین فازهای مختلف در هر یک از ویژگی‌های موجود نوبت به آن می‌رسد که قانون گذاری های مربوطه را انجام دهیم. در اینجا قانون‌گذاری ها را به شکل زیر انجام دادیم و در مدل تعیین کردیم که تحت هر کدام از فازها چه رفتاری از خود نشان دهد.

```
rule1a1 = ctrl.Rule(crowdedness['low'] | people_speed['low'] | hastiness['low'],
                    robot_speed['medium'])
rule1a2 = ctrl.Rule(crowdedness['low'] | people_speed['medium'] | hastiness['low'],
                    robot_speed['medium'])
rule1a3 = ctrl.Rule(crowdedness['low'] | people_speed['medium'] | hastiness['medium'],
                    robot_speed['medium'])
rule1a4 = ctrl.Rule(crowdedness['low'] | people_speed['low'] | hastiness['medium'],
                    robot_speed['high'])
rule1a5 = ctrl.Rule(crowdedness['low'] | people_speed['high'] | hastiness['high'],
                    robot_speed['high'])
rule1a6 = ctrl.Rule(crowdedness['low'] | people_speed['low'] | hastiness['high'], robot_speed['high'])
rule1a7 = ctrl.Rule(crowdedness['low'] | people_speed['high'] | hastiness['low'], robot_speed['low'])
rule1a8 = ctrl.Rule(crowdedness['low'] | people_speed['high'] | hastiness['medium'],
                    robot_speed['medium'])
rule1a9 = ctrl.Rule(crowdedness['low'] | people_speed['medium'] | hastiness['high'],
                    robot_speed['high'])

rule1b1 = ctrl.Rule(crowdedness['medium'] | people_speed['low'] | hastiness['low'],
                    robot_speed['medium'])
rule1b2 = ctrl.Rule(crowdedness['medium'] | people_speed['medium'] | hastiness['low'],
                    robot_speed['low'])
rule1b3 = ctrl.Rule(crowdedness['medium'] | people_speed['medium'] | hastiness['medium'],
                    robot_speed['medium'])
rule1b4 = ctrl.Rule(crowdedness['medium'] | people_speed['low'] | hastiness['medium'],
                    robot_speed['high'])
rule1b5 = ctrl.Rule(crowdedness['medium'] | people_speed['high'] | hastiness['high'],
                    robot_speed['medium'])
rule1b6 = ctrl.Rule(crowdedness['medium'] | people_speed['low'] | hastiness['high'],
                    robot_speed['high'])
```

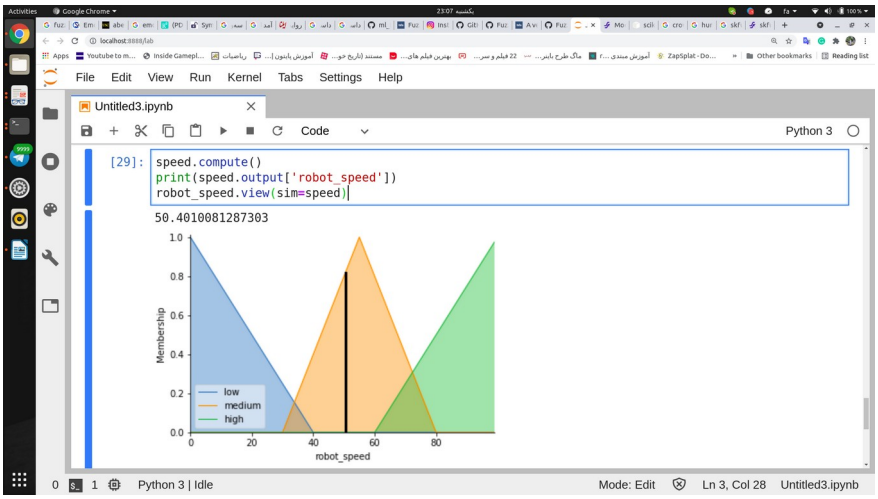
```
rule1b7 = ctrl.Rule(crowdedness['medium'] | people_speed['high'] | hastiness['low'],
                    robot_speed['low'])
rule1b8 = ctrl.Rule(crowdedness['medium'] | people_speed['high'] | hastiness['medium'],
                    robot_speed['medium'])
rule1b9 = ctrl.Rule(crowdedness['medium'] | people_speed['medium'] | hastiness['high'],
                    robot_speed['high'])

rule1c1 = ctrl.Rule(crowdedness['high'] | people_speed['low'] | hastiness['low'],
                    robot_speed['medium'])
rule1c2 = ctrl.Rule(crowdedness['high'] | people_speed['medium'] | hastiness['low'],
                    robot_speed['low'])
rule1c3 = ctrl.Rule(crowdedness['high'] | people_speed['medium'] | hastiness['medium'],
                    robot_speed['medium'])
rule1c4 = ctrl.Rule(crowdedness['high'] | people_speed['low'] | hastiness['medium'],
                    robot_speed['medium'])
rule1c5 = ctrl.Rule(crowdedness['high'] | people_speed['high'] | hastiness['high'], robot_speed['low'])
rule1c6 = ctrl.Rule(crowdedness['high'] | people_speed['low'] | hastiness['high'],
                    robot_speed['medium'])
rule1c7 = ctrl.Rule(crowdedness['high'] | people_speed['high'] | hastiness['low'], robot_speed['low'])
rule1c8 = ctrl.Rule(crowdedness['high'] | people_speed['high'] | hastiness['medium'],
                    robot_speed['low'])
rule1c9 = ctrl.Rule(crowdedness['high'] | people_speed['medium'] | hastiness['high'],
                    robot_speed['medium'])
```

حال بعد از قانون گذاری ها میتوانیم مدل خود را کامل شده در نظر بگیریم. در این مرحله ورودی‌های مختلف را دریافت کرده و طبق همین منطق فازی آن‌ها را تحلیل کرده و خروجی‌های مناسب برای آن‌ها در نظر بگیریم. باقی کار ها با کد زیر برای گرفتن خروجی انجام میشود.

```
speed.compute()
print(speed.output['robot_speed'])
robot_speed.view(sim=speed)
```

تصویری از خروجی برای ورودی های ۱۰و۱۰و۱۰:



منابع:

https://towardsdatascience.com/a-very-brief-introduction-to-fuzzy-logic-and-fuzzy-systems-d68d14b3a3b8

https://towardsdatascience.com/fuzzy-inference-system-implementation-in-python-8af88d1f0a6e

https://github.com/FreakyHarsh/Fuzzy-controller/blob/master/fuzzy.py

https://pythonhosted.org/scikit-fuzzy/auto_examples/index.html

/https://blog.faradars.org/fuzzy-logic