Amirkabir University of Technology
(Tehran Polytechnic)

AI Module

Fuzzy logic robot controller

Professor:
Dr. M.Ghatee (Ghatee@aut.ac.ir)
Teaching Assistants:
Eshagh, Amir, Sajad , Mehrad

Student:
Mohammadreza Ardestani **(**ardestani.zr@gmail.com**)**

9513004

11, July, 2021

0) Introduction

هدف این پروژه گرفتن وررودی های زبانی در رنج های مختلف و استفاده از آن برای کنترل سرعت یک ربات است.
به این علت که هدف این تمرین آشنایی پیدا کردن با منطق فازی است، در پیاده سازی کنترلر ربات خود تنها 3 قانون ساده تعریف کرده ام. اضافه کردن قوانین بیشتر، تنها نیاز به کپی کردن خط های قوانین دیگر است اما برای ساده نگه داشتن و قابل فهم شدن پروژه تا جای ممکن سعی در عدم پیچیده کردن شده است.

در اجرای پروژه بهتر است از Google Colab استفاده شود (ممکن است ورژن numpy روی سیستم لوکال شما با ورژن مورد نیاز skfuzzy کانفلیکت داشته باشد. کدی که در انتهای پروژه آمده است را خط به خط در گوگل کولب اجرا کنید. ریکوآیرمنت دیگری نیاز ندارد.

Phase 1) Fuzzy Vs PID controller

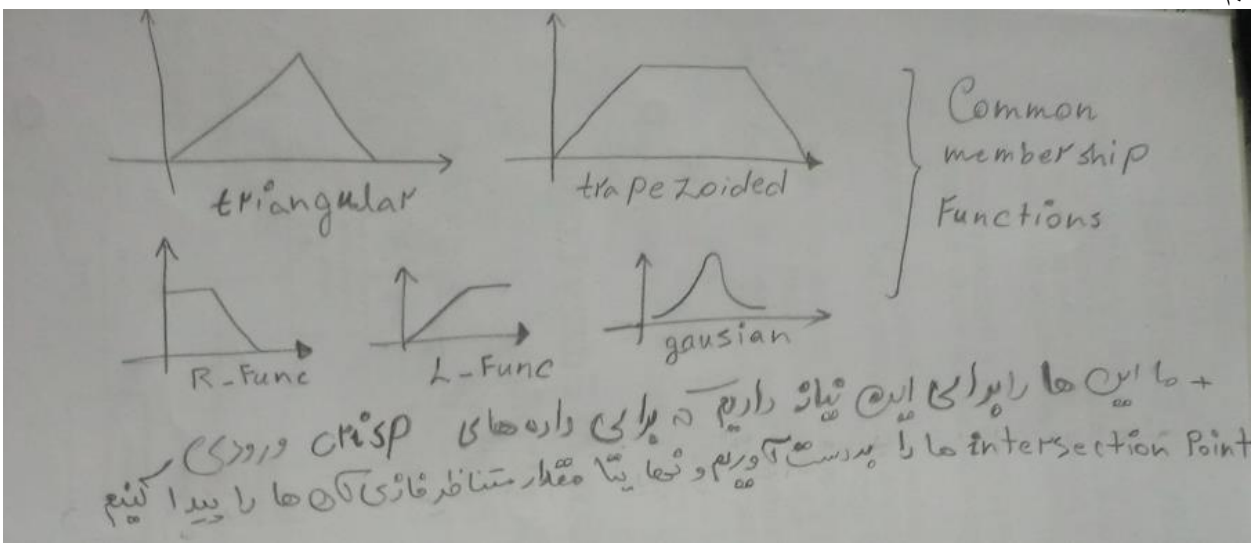
در روش (Proportional Derivative Controller) PID که برای کنترل ماشین های خودران استفاده میشود از منطق بولین استفاده میشود. منطق بولین از نسبت به تغییرات پله ای (Step function) عمل میکند و این تغییرات پله ای موجب فشار آوردن به موتور خودرو و آسیب دیدن ان میشود اما Fuzzy logic که یک modification بر روی آن است soft computing (معادل واژه fuzzy logic در زمینه هایی مثل رباتیک از "محاسبات نرم" استفاده میشود.
پیاده سازی کردن کردن منطق فازی بر روی PID کنتلر در مهندسی برق پیگیری میشود و در این جا فقط به انگیزه و کاربرد های منطق فازی اشاره شد.
در ادامه منطق فازی را بر روی یک ورژن ساده شده Vehicle controller استفاده میکینم.

Phase 2) **Control flow of the program & inference rules &Result**

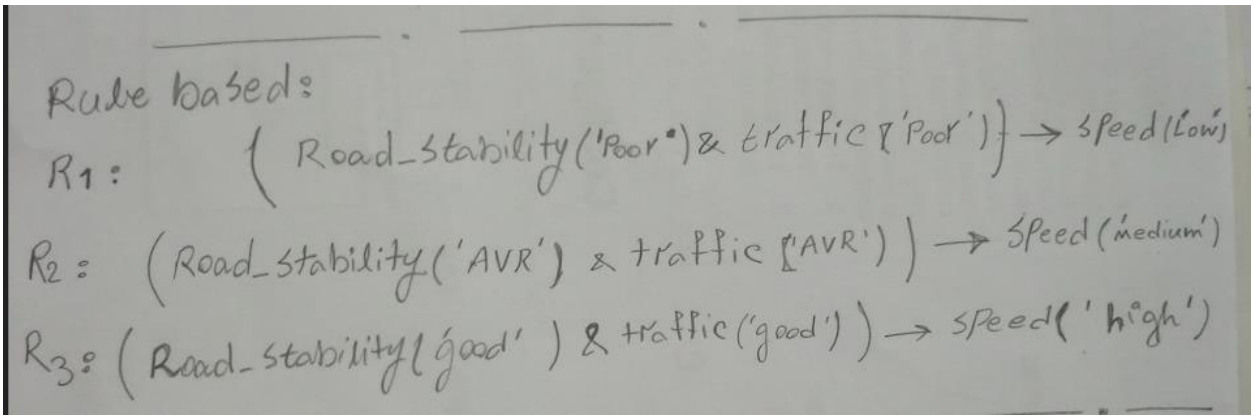
1) در ابتدای کار ما باید state vars and control vars ها را تعیین کنیم.

دراین برنامه دو ورودی " میزان لغزنده بودن جاده" و " میزان ترافیک" برای تنظیم خروجی " میزان سرعت" استفاده میشود. میزان لغزنده بودن جاده در 5 سطح : { خیلی کم ، کم ، متوسط ، زیاد ، خیلی زیاد } به سیستم داده میشود و همین طور برای میزان ترافیک در سه سطح { low , medium , high} میباشد و نهایتا خروجی برنامه کنترلر بعد از دی فازیفیکیشن به صورت عددی crisp بین 0 تا 100 میباشد.

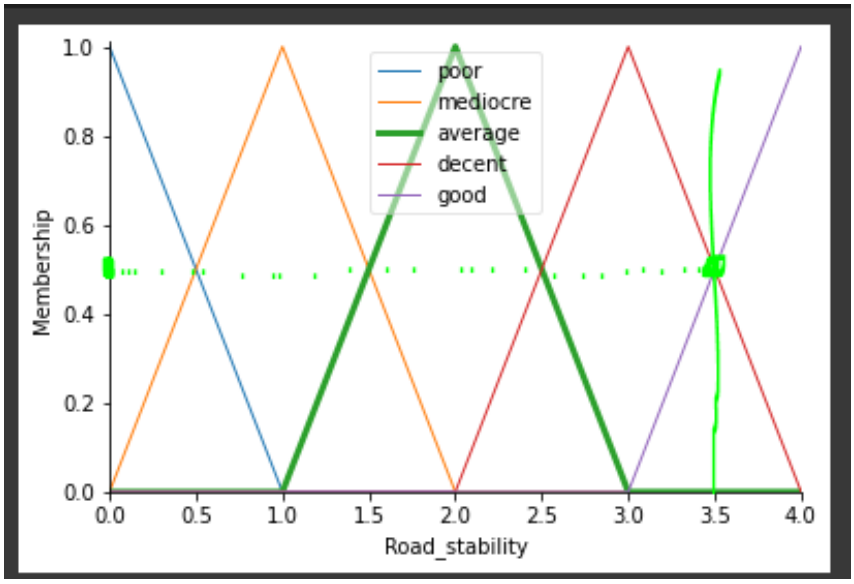
2) حال درباره member ship function ها و انواع آن ها توضیح خواهیم داد. ما از تابع عضویت مثلثی استفاده کرده ایم.



3) حال قوانین خود را معرفی کنیم که knowledge based آن ها را تعیین کرده است و در آن ذخیره شده است. (در حافظه سیستم هم قرار داده شده است.)

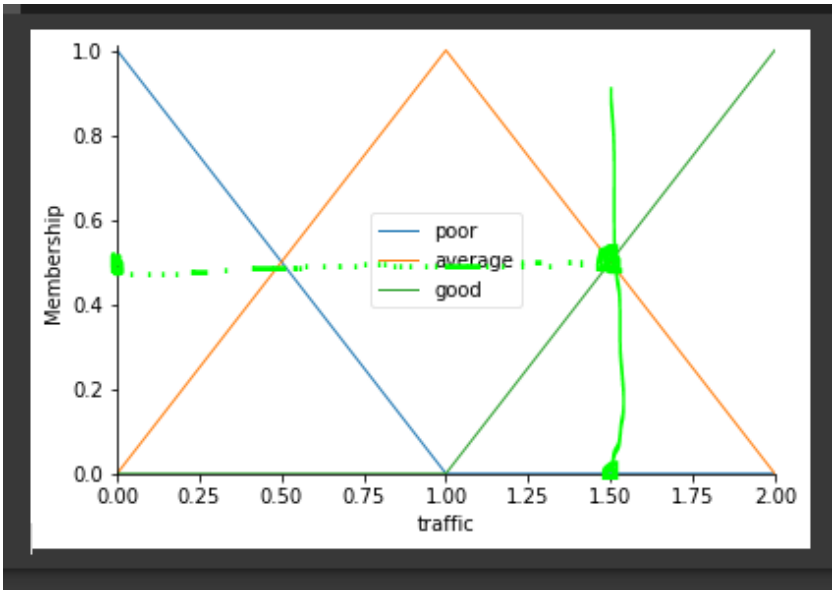


4) حال نوبت به گرفتن داده های ورودی است. به طور مثال 3.5 و 1.5 را به ترتیب برای "وضعیت لغزندگی جاده" و " میزان ترافیک" به سیستم میدهیم.



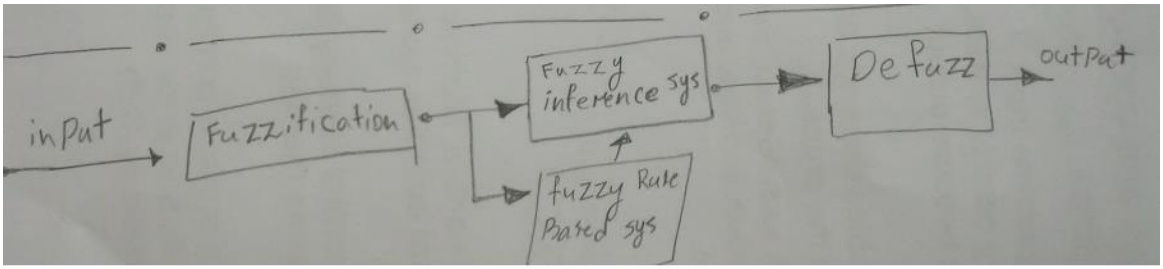
این داده ها crisp هستند و روش فازی کردن آن ها این است که از روی member ship function ما intersectionها را پیدا کنیم و مقدار متناظر آن روی مجور yها درجه عضویت فازی آن خواهد بود.

دو عکس به رو به ترتیب برای road stability and traffic میباشند.



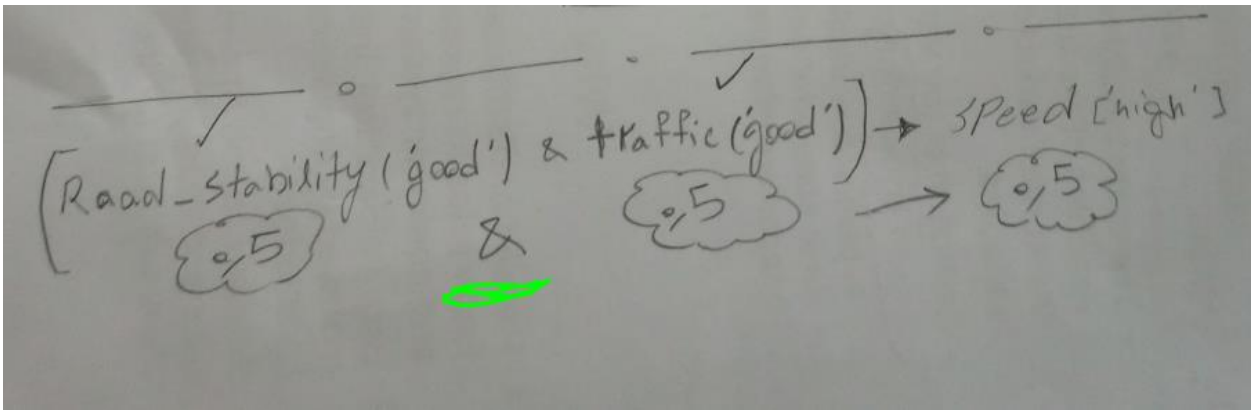
اگر نقاط برخورد خط سبز رنگ x = 1.5 , x = 3.5 را با توابع عضویت برخورد دهیم و مقدار تصویر شده آن روی مجور y را به دست آوریم، همان مقدار فازی شده خواهد بود.

5) حال بگذارید که یک بار flowکل برنامه را داشته باشیم:



همان طور که در تصویر بخش 5 مشخص است، ما تا به حال "فازی فیکیشن " را انجام داده ایم حال به سراغ مرحله های بعدی میرویم.

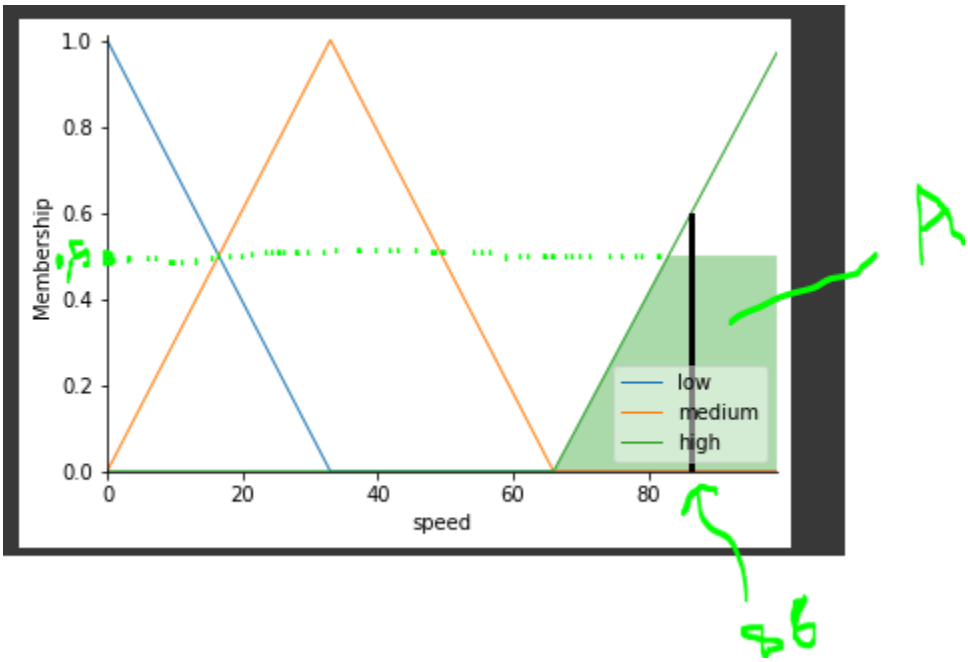
6) مرحله " پیدا کردن قوانین فعال شده " :
در این مرحله ما آن قوانینی که فعال شده اند را استخراج میکنیم:



تنها قانون شماره 3 فعال شده است. توجه شود که بین مقادیر and هست پس باید بین مقادیر سمت چپ مینیمم را انتخاب کنیم که البته این جا همه مقادیر سمت چپ مساوی اند -> پس نهایتا مقدار سرعت 0.5 میشود.

7) مرحله آخر که " دیفازیفکشن " نام دارد ما با داشتن مقدار y یا همان مقدار فازی مقدار xیا همان دیفازی شده را به دست میآوریم.

توجه شود که برای دیقازیفیکشن روش های متفاوتی وجود دارد که ما روش " میانگین ناحیه فازی" را دنبال میکنیم. (A = fuzzy region , 86 = defuzzified value = out put)



Phase 5) **References**

در این پروژه از ویدئوهای زیر به عنوان رفرنس استفاده شده است:

https://www.youtube.com/watch?v=vG6aZEgbAVU

https://www.youtube.com/watch?v=6i5bHQZ-c5U

https://www.youtube.com/watch?v=wVu1pjhTfCc

https://www.youtube.com/watch?v=xD1c8jTFF78

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

Phase 6) **Appendix (code)**

```
pip install -U scikit-fuzzy
```

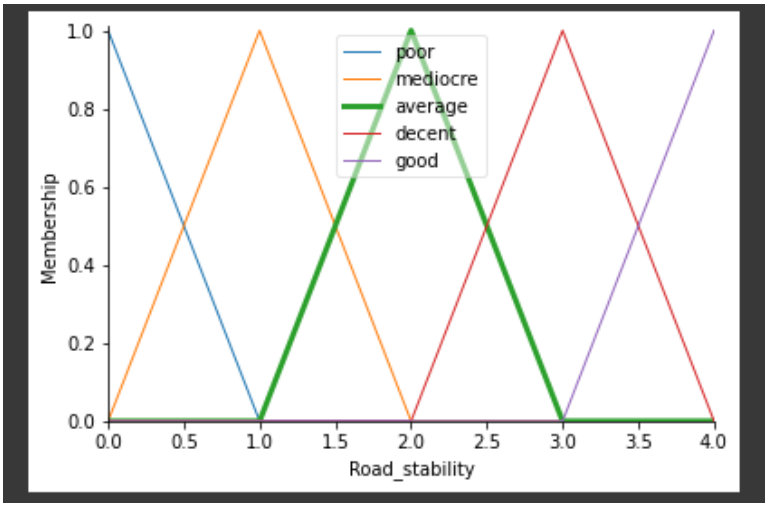
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
```

```
quality = ctrl.Antecedent(np.arange(0,11,1), "quality")
service = ctrl.Antecedent(np.arange(0,11,1), "service")
tip = ctrl.Consequent(np.arange(0,26,1), "tip")
```

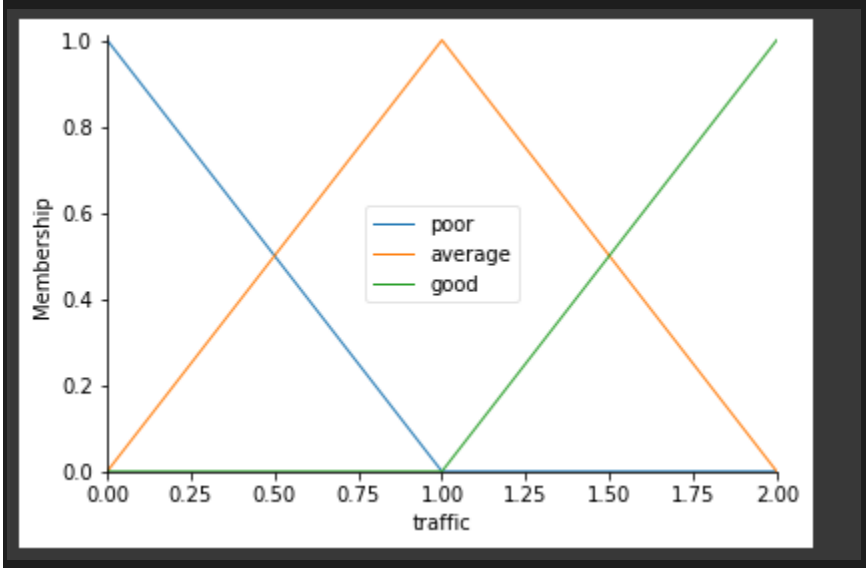
```
Road_stability.automf(5)
traffic.automf(3)
```

```
speed['low'] = fuzz.trimf(speed.universe, [0,0,33] )
speed['medium'] = fuzz.trimf(speed.universe, [0,33,66] )
speed['high'] = fuzz.trimf(speed.universe, [66,100,100] )
```

```
Road_stability['average'].view()
```




```
traffic.view()
```




```
rule_1 = ctrl.Rule(Road_stability['poor'] & traffic['poor'], speed['low'])
rule_1.view()
rule_1.view()
```

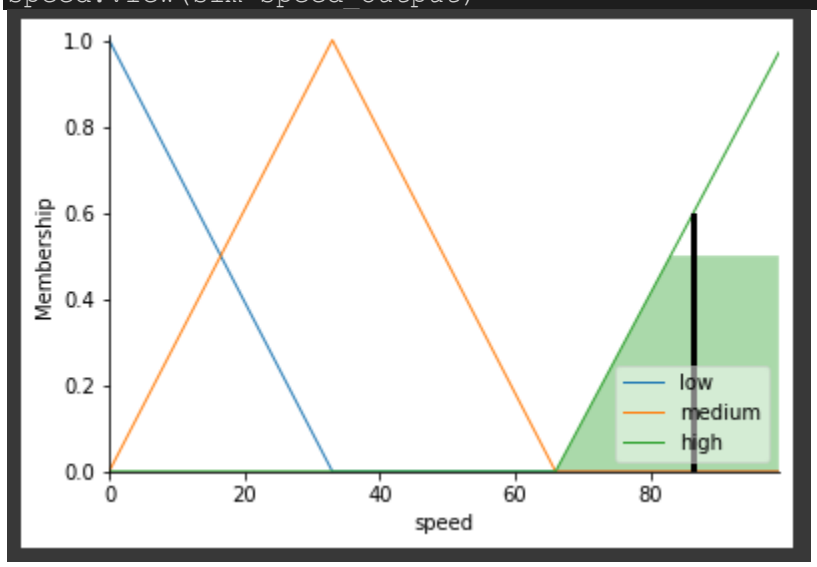
```
rule_2 = ctrl.Rule(Road_stability['average'] & traffic['average'], speed['
medium'])
rule_3 = ctrl.Rule(Road_stability['good'] & traffic['good'], speed['high']
)
```

```
speed_ctrl = ctrl.ControlSystem( [rule_1,rule_2,rule_3])
speed_output = ctrl.ControlSystemSimulation(speed_ctrl)
```

```
speed_output.input["Road_stability"] = 3.5
speed_output.input["traffic"] = 1.5
speed_output.compute()
```

```
print(speed_output.output['speed'])
```
```
86.25850340136056
```

```
speed.view(sim=speed_output)
```



Thanks for your attention.