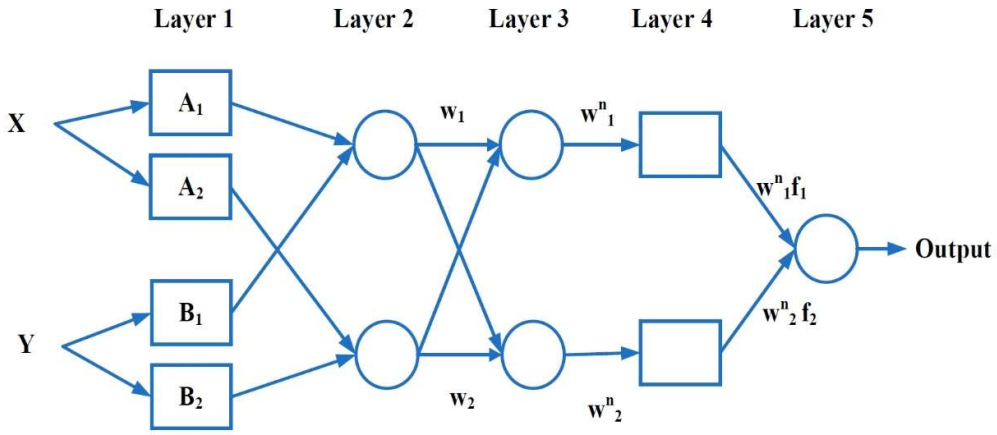
تمرین سری ششم درس شبکه های عصبی مصنوعی

پوریا محمدی نسب

(۴۰۰۷۲۲۱۳۸)

۱- مقاله زیر را با دقت مطالعه کنید و قسمت Adaptive Neuro Fuzzy Inference System (ANFIS) در روش پیشنهادی را به صورت کامل و با ذکر جزئیات توضیح دهید. (۱۵ امتیاز)

در این مقاله با ترکیب DWT(Discrete Wavelet Transform) و ANFIS(Adaptive Neuro-fuzzy Inference System) یک متد جدید برای پیشبینی میزان آب مصرفی ماهانه شهری بر اساس تاریخچه مصرف آب و چندین پارامتر دیگر پیشنهاد شده است. تکنیک ANFIS توانایی این را دارد که time series غیرخطی را مدلسازی کند که در بسیاری از کاربردها با اهمیت است. ANFIS در اصل یک شبکه عصبی مصنوعی است که با یک سیستم inference فازی ترکیب شده است. قسمت ANN از تکنیک Back propagation و least square estimation استفاده میکند و قسمت FIS ، شامل تعدادی قانون if-then است که در همه سیستم های فازی رایج هستند. ANFIS به دلیل ترکیب این دو موضوع قابلیت بهره بردن از هر مزایای هر دو قسمت را داراست و همینطور با انتخاب تابع عضویت های مناسب میتوان عملکرد آن را بهبود داد. ANFIS یک معماری ۵ لایه شامل توابع عضویت، قوانین، نرمالیزیشن دیتا، توابع و خروجی است که در شکل زیر مشاهده میکنید:



در لایه اول هر نود شامل adaptive node ها هست که با دو رابطه ی زیر محاسبه میشود. که در واقع این دو مقدار، مقادیر تابع عضویت برای هر نود هستند.

$$O_{1,i} = \mu A_i(x)$$

$$O_{1,i} = \mu B_i(y)$$

لایه دوم ضرب مقادیر لایه قبل میباشد.

$$O_{2,i} = W_i = \mu A_i(x)\mu B_i(y), \quad i = 1,2$$

در لایه ی سوم، خروجی لایه قبل نرماليز میشود.

$$O_{3,i} = \bar{\omega}_i = \frac{\omega_i}{\omega_1 + \omega_2}, \quad i = 1,2$$

لایه ی چهارم، از یک تابع استفاده میکنیم برای هر نود که ۳ پارامتر دارد و کار لینک کردن نود ها را انجام میدهد.

$$O_{4,i} = \bar{\omega}_i f_i = \bar{\omega}_i(p_i x + q_i y + r_i)$$

و لایه ی آخر مجموع تمام سیگنال های خروجی را محاسبه میکند.

$$O_{5,i} = \sum \bar{\omega}_i f_i \quad , i = 1,2$$

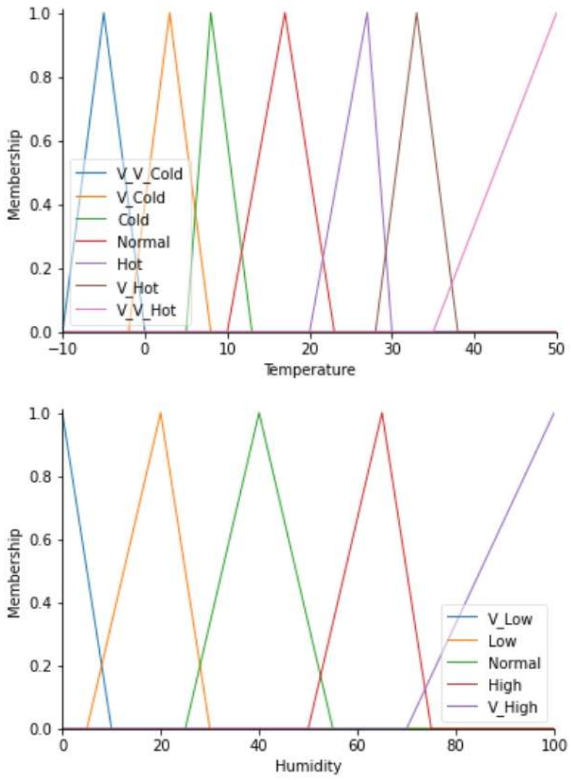
۲- در این سوال هدف پیادهسازی یک سیستم کنترلی فازی ساده با استفاده از پایتون است. برای این سوال از کتابخانه *scikit-fuzzy* استفاده خواهیم کرد. در لینک زیر یک مثال ساده از این کتابخانه برای مسئله انعام دادن آورد ه شده است. لطفا لینک زیر را با دقت بررسی بفرمایید و سپس یک سیستم کنترلی فازی برای مسئله ای که در ادامه آورده شده است، طراحی بفرمایید. لازم است که برای این سوال گزارش طراحی کنید و قسمتهای مختلف کد خود را شرح دهید .

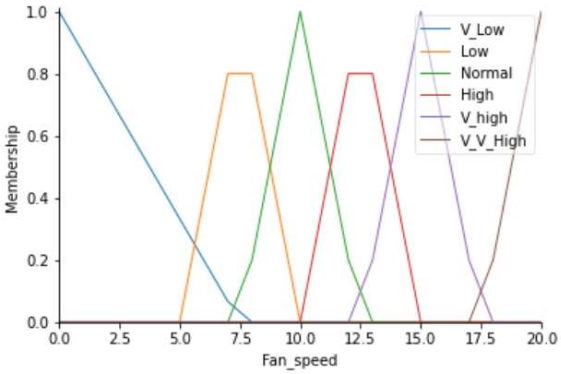
برای حل این سوال ۲ متغیر ورودی و ۱ متغیر خروجی را با دامنه ی هر کدام معرفی میکنیم و سپس برای *descriptor* های هر متغیر از تابع *trimf* که حالت مثلثی دارد استفاده میکنیم. در این تابه ۳ پارامتر ورودی داریم که دو تا از آنها نقاط شروع و پایان بازه هستند و یکی از پارامتر ها مرکز را نشان میدهد.

```
Temperature = ctrl.Antecedent(np.arange(-10, 51, 1), 'Temperature')
Humidity = ctrl.Antecedent(np.arange(0, 101, 1), 'Humidity')
Fan_speed = ctrl.Consequent(np.arange(0, 21, 1), 'Fan_speed')
```

```
# Custom membership functions can be built interactively with a familiar, pythonic API
Temperature['V_V_Cold']   = fuzz.trimf(Temperature.universe, [-10, -5, 0])
Temperature['V_Cold']   = fuzz.trimf(Temperature.universe, [-2, 3, 8])
Temperature['Cold']   = fuzz.trimf(Temperature.universe, [5, 8, 13])
Temperature['Normal'] = fuzz.trimf(Temperature.universe, [10, 17, 23])
Temperature['Hot']   = fuzz.trimf(Temperature.universe, [20, 27, 30])
Temperature['V_Hot']    = fuzz.trimf(Temperature.universe, [28, 33, 38])
Temperature['V_V_Hot']   = fuzz.trimf(Temperature.universe, [35, 50, 50])

Humidity['V_Low']  = fuzz.trimf(Humidity.universe, [0, 0, 10])
Humidity['Low']       = fuzz.trimf(Humidity.universe, [5, 20, 30])
Humidity['Normal']        = fuzz.trimf(Humidity.universe, [25, 40, 55])
Humidity['High']      = fuzz.trimf(Humidity.universe, [50, 65 , 75])
Humidity['V_High']      = fuzz.trimf(Humidity.universe, [70, 100, 100])

Fan_speed['V_Low']        = fuzz.trimf(Fan_speed.universe, [0, 0, 7.5])
Fan_speed['Low']      = fuzz.trimf(Fan_speed.universe, [5, 7.5, 10])
Fan_speed['Normal'] = fuzz.trimf(Fan_speed.universe, [7.5, 10, 12.5])
Fan_speed['High']        = fuzz.trimf(Fan_speed.universe, [10, 12.5, 15])
Fan_speed['V_high']      = fuzz.trimf(Fan_speed.universe, [12.5, 15, 17.5])
Fan_speed['V_V_High'] = fuzz.trimf(Fan_speed.universe, [17.5, 20, 20])
```

خروجی بازه های تعریف شده در قطعه کد بالا به صورت زیر است:





در طراحی یک کنترلر فازی، مرحله پس از ایجاد توابع عضویت، تعریف قوانین است. قوانین تعریف شده در صورت مسئله به شکل زیر پیاده سازی میشوند و سپس این قوانین سیستم فازی ما رو خواهند ساخت.

```
# rules
rule1 = ctrl.Rule(Temperature['V_V_Cold'] , Fan_speed['V_Low'])
rule2 = ctrl.Rule(Temperature['V_Cold'] , Fan_speed['V_Low'])
rule3 = ctrl.Rule((Temperature['Cold'] | Temperature['Normal']) & Humidity['V_Low'], Fan_speed['Low'])
rule4 = ctrl.Rule((Temperature['Normal'] | Temperature['Hot']) & (Humidity['Normal'] | Humidity['High']), Fan_speed['Normal'])
rule5 = ctrl.Rule(Temperature['Hot'] & Humidity['V_High'], Fan_speed['High'])
rule6 = ctrl.Rule(Temperature['V_Hot'], Fan_speed['V_high'])
rule7 = ctrl.Rule(Temperature['V_V_Hot'] , Fan_speed['V_V_High'])
```

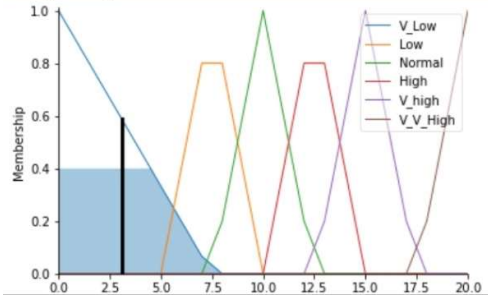
```
Fan_Ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7])
```

```
Fan = ctrl.ControlSystemSimulation(Fan_Ctrl)
```

در نهایت پس از ایجاد سیستم فازی و اعمال *rules* به آن میتوان ورودی های *crisp* به سیستم داد و خروجی *crisp* نیز دریافت کرد. در مثال زیر مشاهده میکنیم که با دادن مقادیر *0* و *20* به عنوان دما و رطوبت دور موتور حدود *3.1* میشود و خط آخر کد نیز شبیه سازی را روی شکل نمایش میدهد که کدام قسمت از بازه توان *Fan* به کار افتاده است.

```
# run inputs:
Fan.input['Temperature'] = 0
Fan.input['Humidity'] = 200

Fan.compute()

print('Fan Speed: ',Fan.output['Fan_speed'])

Fan_speed.view(sim=Fan)
```

```
Fan Speed:  3.0931034482758624
C:\ProgramData\Anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



۳- در این سوال قصد داریم که با پیادهسازی یک سیستم کنترلی پیچیدهتر آشنا شویم. برای این مسئله از محیط *MountainCarContinuous-v0* استفاده خواهی مکرد. برای آشنایی بیشتر با این محیط به لینک زیر مراجعه کنید:

https://gym.openai.com/envs/MountainCarContinuous-v0

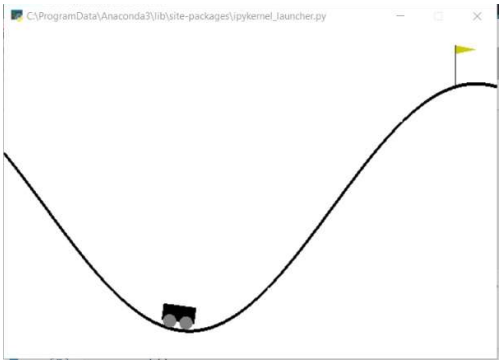
برای حل این سوال ابتدا باید با محیط و *State* ها و *Action* های محیط آشنا شد. این مسئله ۲ ورودی به عنوان موقعیت واگن و یک ورودی به عنوان سرعت دارد و اکشنی که انجام میدهد اعمال نیرویی به واگن است.

```
env = gym.make('MountainCarContinuous-v0')

env.seed(101)
np.random.seed(101)

print('observation space:', env.observation_space)
print('action space:', env.action_space)
print('  - low:', env.action_space.low)
print('  - high:', env.action_space.high)
```

```
observation space: Box([-1.2  -0.07], [0.6  0.07], (2,), float32)
action space: Box([-1.], [1.], (1,), float32)
  - low: [-1.]
  - high: [1.]
```

```
env.reset()
for _ in range(200):
    env.render()
    a = env.action_space.sample()
    o,r,d,p = env.step([a]) # take a random action
env.close()
```

دو قطعه کد بالا نمایشی از ویژگی های محیط و اجرای گرافیکی محیط به ما میدهد.

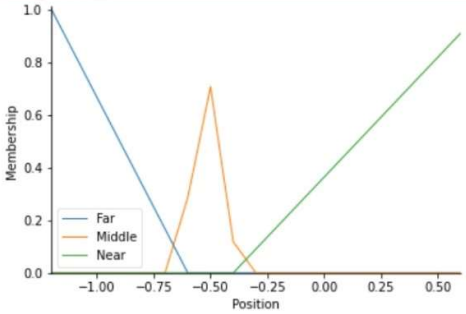


در ابتدا شروع به تعریف توابع عضویت میکنیم:

```
Position = ctrl.Antecedent(np.arange(-1.2, 0.7,0.1), 'Position')
Speed = ctrl.Antecedent(np.arange(-0.07, 0.07, 0.01), 'Speed')
Power = ctrl.Consequent(np.arange(-1, 1.1, 0.1), 'Power')
```

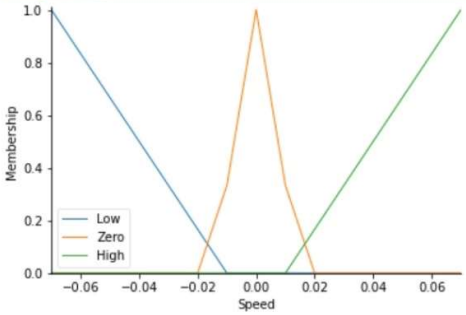
```
# Custom membership functions can be built interactively with a familiar, pythonic API
Position['Far']   = fuzz.trimf(Position.universe, [-1.3, -1.2, -0.6])
Position['Middle']   = fuzz.trimf(Position.universe, [-0.62, -0.55, -0.38])
Position['Near']   = fuzz.trimf(Position.universe, [-0.4, 0.7, 0.7])

Position.view()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



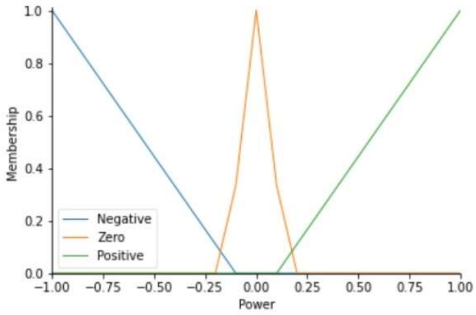
```
Speed['Low']   = fuzz.trimf(Speed.universe, [-0.07, -0.07, -0.01])
Speed['Zero']   = fuzz.trimf(Speed.universe, [-0.015, 0.0, 0.015])
Speed['High']   = fuzz.trimf(Speed.universe, [0.01, 0.07, 0.07])

Speed.view()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```




```
Power['Negative']   = fuzz.trimf(Power.universe, [-1, -1, -0.1])
Power['Zero']   = fuzz.trimf(Power.universe, [-0.15, 0.0, 0.15])
Power['Positive']   = fuzz.trimf(Power.universe, [0.1, 1, 1])

Power.view()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



پس از تعریف و رسم توابع *membership* نوبت به تعریف قوانین میرسد که با توجه به سادگی مسئله با ۵ قانون مسئله حل میشود:

۱) اگر واگن در قعر دره است و سرعتش کم است با نیروی منفی به سمت چپ هل داده شود.

۲) اگر واگن در قعر دره است و سرعت صفر است با نیروی منفی به سمت چپ هل داده شود.

۳) اگر واگن در سمت چپ داره قرار دارد با نیروی مثبت به سمت راست هل داده شود.

۴) اگر واگن در قعر دره است و سرعتش زیاد است با نیروی مثبت به سمت راست هل داده شود.

۵) اگر واگن در سمت راست دره است با نیروی مثبت به سمت راست هل داده شود.

```
rule1 = ctrl.Rule(Position['Middle'] & Speed['Low'] , Power['Negative'])
rule2 = ctrl.Rule(Position['Middle'] & Speed['Zero'], Power['Negative'])
rule3 = ctrl.Rule(Position['Far'] , Power['Positive'])
rule4 = ctrl.Rule(Position['Middle'] & Speed['High'], Power['Positive'])
rule5 = ctrl.Rule(Position['Near'], Power['Positive'])
```

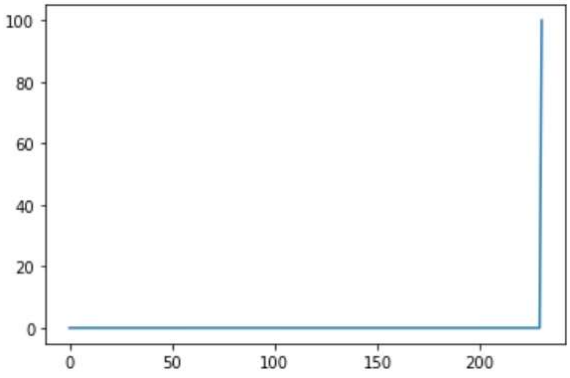
```
Car_Ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5])
```

```
Car = ctrl.ControlSystemSimulation(Car_Ctrl)
```

قطعه کد بالا پیاده سازی قوانین و ایجاد سیستم کنترل فازی را نمایش میدهد. پس از این مرحله نوبت به تست سیستم میرسد که در حلقه ای که انیمیشن را *render* میکنیم به جای انتخاب یک عمل تصادفی با توجه به *State* واگن مقدار نیروی مورد نظر را محاسبه میکنیم.

```
env.reset()

Car.input['Position'] = -0.2
Car.input['Speed'] = -0.02
Car.compute()
a = Car.output['Power']
Rewards = []
for _ in range(250):
    env.render()

    #print(a)

    State,Reward,Done,tmp = env.step([a]) # take a random action
    print(State[0] , '-' , State[1])
    Car.input['Position'] = State[0]
    Car.input['Speed'] = State[1]
    Car.compute()
    a = Car.output['Power']

    Rewards.append(Reward)

    if Done == True:
        break

env.close()
plt.plot(Rewards)
```

پس از رسم نمودار *reward* ها به شکل زیر میرسیم که نشان میدهد در طول مسیر واگن *reward* ای دریافت نکرده است ولی در انتها که به بالای تپه ی سمت راست و به پرچم رسیده است یک *reward* زیاد دریافت کرده است.



۴- در این سوال میخواهیم هم هی مراحل اجرا شده در سوال قبل را برای یک محیط دیگر اجرا کنیم. در این سوال هدف اجرا و کنترل محیط *Pendulum-v0* است. برای آشنایی بیشتر با این محیط به لینک زیر مراجعه فرمایید:

https://gym.openai.com/envs/Pendulum-v0/

این محیط یک محیط پیچیده تر با ویژگیهای بیشتر نسبت به سوال قبل است. برای این سوال دقیقا باید مراحل سوال قبل طی شود. یعنی شما باید محیط مورد نظر را اجرا کنید و با استفاده از یک سیستم کنترل فازی تلاش کنید به هدف و خواسته تعریف شده در محیط برسید. برای این سوال باید مراحل سوال قبل طی شود و گزارش هم نوشته شود. در صورت پیاده سازی این محیط با استفاده از سیستم کنترل فازی ۵۰ امتیاز مثبت دریافت خواهید کرد. همچنین می توانی د از یادگیری تقویتی برای حل این سوال استفاده کنید که در صورت استفاده از یادگیری تقویتی ۳۰ امتیاز مثبت دریافت خواهید کرد. اگر از یادگیری تقویتی برای حل این سوال استفاده کردید باید در گزارش خود به صورت کامل الگوریتم پیاده سازی شده و مراحل مختلف آن را شرح دهید .

برای حل این سوال ما از یادگیری تقویتی استفاده میکنیم. در صورت سوال ذکر شده است که از *Pendulum-v0* استفاده شود اما در نسخه ی جدید کتابخانه *gym* تنها ورژن *1* این *environment* وجود دارد و با فراخوانی ورژن *0* به ارور زیر برخورد میکنیم:

```
C:\ProgramData\Anaconda3\lib\site-packages\gym\envs\registration.py in spec(self, path)
    183             ]
    184             if matching_envs:
--> 185                 raise error.DeprecatedEnv(
    186                     "Env {} not found (valid versions include {})".format(
    187                         id, matching_envs

DeprecatedEnv: Env Pendulum-v0 not found (valid versions include ['Pendulum-v1'])
```

پس برای ساخت محیط شبیه سازی و تنظیم پارامترها از کد زیر استفاده میکنیم:

[1]:
```
import gym
import random
import numpy as np
import time
import matplotlib.pyplot as plt
```

[2]:
```
# paramters and environments

env = gym.make("Pendulum-v1")

discrete_os_size = (13, 11, 12)
discrete_os_win_size = (env.observation_space.high - env.observation_space.low)/discrete_os_size
discrete_os_nsize = (14, 12, 13)
discrete_action_size = (2)
discrete_action_win_size = (env.action_space.high - env.action_space.low)/discrete_action_size
q_table = np.random.random(discrete_os_nsize + (discrete_action_size,))
```

با توجه به پیوستگی فضای محیط نیاز است که تابعی تعریف کنیم تا *state* های پیوسته را به یک *state* گسسته تبدیل کند. و در سپس میتوانیم شروع به تعریف کلاسی برای *agent* کنیم.

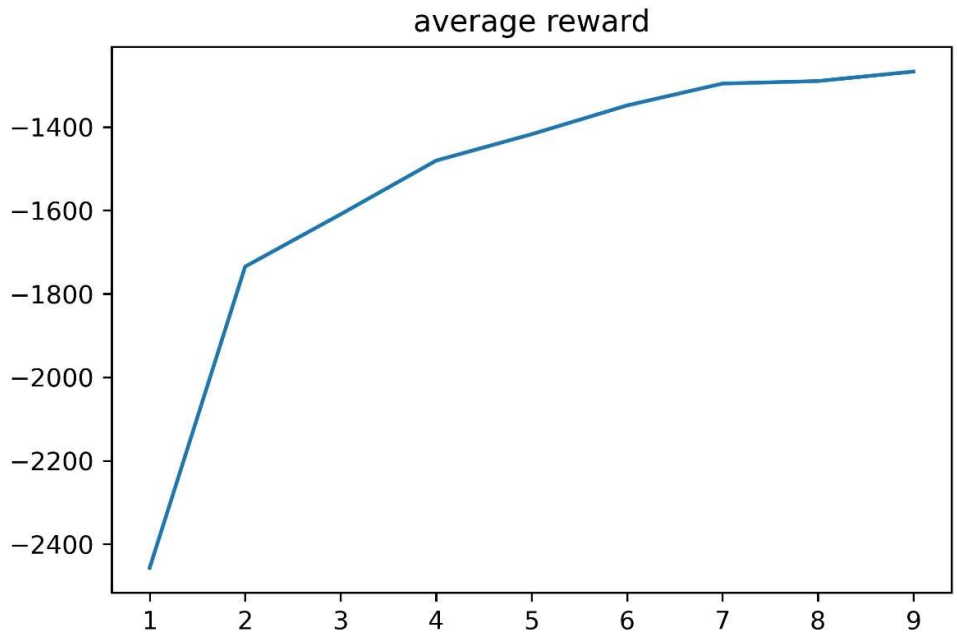
[3]:
```
def get_discrete_state(state):
    discrete_state = (state - env.observation_space.low)/discrete_os_win_size
    return tuple(discrete_state.astype(int))

class QAgent():
    def __init__(self, env):
        self.action_low = env.action_space.low
        self.action_high = env.action_space.high
        self.eps = 0.01
        self.discount = 0.91
        self.lr = 0.06
    def get_action(self, state, env):
        d_state = get_discrete_state(state)
        action = [-2.0 + (np.argmax(q_table[d_state]))*4.0]
        if(random.random() < self.eps):
            return env.action_space.sample()
        else:
            return action
    def train(self, state, action, next_state, reward, done):
        d_state = get_discrete_state(state)
        next_d_state = get_discrete_state(next_state)
        if(done==True):
            future_q = np.zeros(discrete_action_size)
        else:
```

```
        else:
            future_q = q_table[next_d_state]
        current_q = q_table[d_state + (np.argmax(q_table[d_state]),)]
        target_q = reward + self.discount*np.max(future_q)
        update = target_q - current_q
        q_table[d_state + (np.argmax(q_table[d_state]),)] += self.lr*update
        if(done==True):
            self.eps = self.eps*0.98
```

در کلاس *agent* تابع *__init__* با گرفتن مشخصات محیط پارامترهای کمینه و بیشینه *action* را مشخص میکند. همینطور پارامترهای مهم در الگوریتم *Q-learning* شامل مقداز *epsilon* و *discount* و *learning rate* مشخص میشوند. در تابع *get_action* میتوان با دادن حالت محیط و محیط *action* ای که *agent* انجام میدهد را مشخص کرد. و در نهایت مهم ترین تابع این کلاس، تابع *train* است که در واقع عمل ارتباط و اپدیت *q-table* ، تنظیم پارامتر *lr* و *epsilon* با این تابع است.

در آخرین قدم کافیت *object* عامل در محیط را بسازیم و عملکرد آن در محیط را مشاهده کنیم. برای بررسی دقیق نحوه عملکرد عامل ما ۱۰ *episode* مختلف را در نظر میگیریم و *reward* میانگین عامل در هر *episode* را ذخیره و رسم میکنیم تا مشخص شود آیا *agent* حرکتی بهبودی دارد یا خیر.

[4]:
```
agent = QAgent(env)
t_reward = 0
info = 0
aggr_rewards = {'avg': []}

ep_len = 0
avg_reward = 0
state = env.reset()
done = False
for ep in range(10):
    for i in range(200):
        ep_len += 1
        action = agent.get_action(state, env)
        next_state, reward, done, info = env.step(action)
        agent.train(state, action, next_state, reward, done)
        state = next_state
        t_reward += reward
        avg_reward += reward
        env.render()
    aggr_rewards['avg'].append(avg_reward/ep)
env.close()

plt.title('average reward')
plt.plot(aggr_rewards['avg'])
```



۵- در این پیاده سازی از سیستم *ANFIS* برای حل مسائل *Regression* و *Classification* استفاده شده است. در لینک داده شده از چهار مجموعه داده برای حل مسئله استفاده شده است. در این پیاده سازی نیاز است که شما سیستم مورد نظر را بر روی یکی از مجموعه دادهها پیاده سازی کنی د و نتایج را به دست آورید و گزارش کنید. در صورت حل این سوال یک ارائه از شما گرفته خواهد شد که باید کد پیاده سازی شده را توضیح دهید. حل این سوال ۴۰ امتیاز مثبت خواهد داشت.

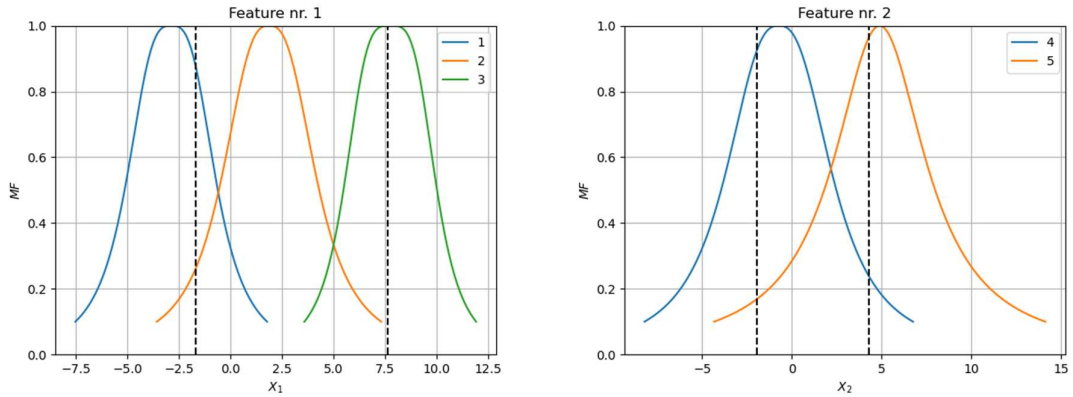
ما برای ایم سوال حل دیتاست معروف *wine-EW* را انجام دادیم که یک *task* کلاسبندی *multi class* در آن نهفته است. با مطالعه *Documentation* کد متوجه شدیم که با دادن آرگومان نام دیتا ست بعد از نام فایل *test.py* مثال اجرا میشود. نکته ی مهم این است که کد طوری نوشته شده است که برای هر مثال انتخابی بهترین پارامتر ها را از لحاظ عملکردی در کد تنظیم میکند تا نتیجه بهتری بدست آید.

پس از اجرای اسکریپت زیر الگوریتم اجرا شد:

python.exe .\test.py wine

در ابتدا مشخصات دیتاست را بررسی میکنیم. این دیتاست *1599* عدد *example* دارد که هر *example* از دو ورودی و یه شماره کلاس خروجی تشکیل شده است اما در اجرای این کد متوجه میشویم که با تبدیل شماره کلاس ها با روش *one-hot* مسلئه را به چند مسئله کلاسبندی *binary* تبدیل کرده است. برای حل این مثال *dataset* به دو بخش *training data* و *test data* تقسیم میشود. قسمت *training* شامل *1119* مثال و مابقی *example* ها (شامل *480* مثال) برای *test* کردن هستند.

بعد از قسمت کار با دیتا الگوریتم به سراغ پارامترهای تنظیم شده ی خود رفته و حل را آغاز میکند. در واقع همانطور که در سوال ۱ نیز توضیح داده شد این الگوریتم با یافتن *membership function* مناسب، مسئله را به بهترین نحو حل میکند. پس از یافتن توابع عضویت آن ها را برای درک بهتر برای کاربر چاپ میکند که در ادامه شکل خروجی آن ها را مشاهده میکنید.



با این توابع عضویتی که بدست آمده، کار *classification* انجام میشود و دقتی که روی داده ی *training* و *test* بدست آمده به ترتیب *58.18* و *59.80* میباشد. این نتایج نشان میدهد با تغییر بعضی پارامتر ها ممکن است عملکرد بهتری داشته باشیم اما این مدل هرگز *overfit* نشده است زیرا خطای آموزش و خطای تعمیم در یک حد هستند.

References

1) www.youtube.com/watch?v=1XRahNzA5bE

2) codecrucks.com/designing-fuzzy-controller-step-by-step-guide/

3) towardsdatascience.com/real-world-applications-of-markov-decision-process-mdp-a39685546026

4) Arzate, Christian & Igarashi, Takeo. (2021). MarioMix: Creating Aligned Playstyles for Bots with Interactive Reinforcement Learning.

5) www.comp.nus.edu.sg/~rishav1/blog/2016/mario-bros-RL/

6) https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c

7) https://people.revoledu.com/kardi/tutorial/ReinforcementLearning/Q-Learning-Example.htm

8) https://blog.floydhub.com/an-introduction-to-q-learning-reinforcement-learning/

9) https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_tipping_problem_newapi.html

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