In the name of God

Assignment 6 Solution

Neural Networks: Fall 1400, Dr. Mozayani

Ghazaleh Mahmoudi

400722156

31 January 2022

Problem 1

- (a)

- تکنیک ANFIS کاربرد محدودی در زمینه پیش‌بینی تقاضای آب شهری دارد. ANFIS در اصل یک شبکه عصبی مصنوعی می‌باشد که با سیستم استنتاج فازی ترکیب شده است. در این الگوریتم از تکنیک back propagation و least squares estimation شبکه‌های عصبی و قوانین اگر-آنگاه و توابع درجه عضویت مربوط به سیستم فازی برای برقراری رابطه بین جفت‌های ورودی و خروجی استفاده می‌شود. به دلیل ترکیب منطق فازی با سازکار شبکه عصبی بازگشتی می‌توان از مزایا هر دو سیستم استفاده کرد و پتانسیل آموزش تابع غیرخطی را دارد.
- از نظر محاسباتی سیستم مرتبه اول Takagi–Sugeno موثر واقع شده است و بنابراین برای ساخت ANFIS از آن استفاده می‌شود.
- تکنیک ANFIS شامل 5 لایه می‌باشد.
  - در لایه اول درجه عضویت (member ship) ورودی‌ها را محاسبه می‌کند.

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

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

  - در لایه دوم درجه عضویت به دست آمده در قسمت قبل، در هم ضرب می‌شوند و وزن W را می‌سازند.

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

$$W_1 = \mu A_1(x)\,\mu B_1(x)$$

$$W_2 = \mu A_2(x)\,\mu B_2(x)$$

- در لایه سوم مقادیر W به دست آمده از لایه دوم به ازای هر نود ورودی نرمالایز می‌شود.

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2},\ i = 1,2$$

- در لایه چهارم برای W های نرمال شده، مقادیر W*f محاسبه می‌شود. تابع f به صورت زیر تعریف می‌شود و پارامترهای p, q, r جز پارامترهای node ها هستند.

$$f_i = \overline{w}_i(p_i x + q_i y + r_i)$$

- در لایه پنجم مجموع حاصل ضرب f*w لایه قبل محاسبه شده و به عنوان خروجی شبکه معرفی می‌شود.

$$O_{5,i} = \sum \overline{w}_i f_i$$

- همچنین اگر به جای Wهای نرمال‌شده از Wهای محاسبه شده در لایه دوم استفاده کنیم، فرمول محاسبه بدین صورت می‌شود.

$$\sum w_i f_i / \sum w_i\ , i = 1,2$$

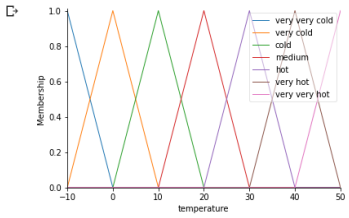
Problem 2

- در ابتدا متغیرهای فازی در بازه‌ای که در صورت سوال ذکر شده تعریف می‌شوند. سپس بسته به تعریف قوانین برای متغیرها در بازه‌های مربوطه، Term های فازی تعریف می‌شود. به عنوان مثال متغیر temperature چون از نوع ورودی است به صورت Antecedent و چون 7 مقداری فازی دارد به این شکل مقداردهی می‌شود.

```
[ ] 1 temperature = ctrl.Antecedent(np.arange(-10, 51, 1), 'temperature')

[ ] 1 temperature.automf(7, names=['very very cold', 'very cold', 'cold', 'medium', 'hot', 'very hot', 'very very hot'])

    1 temperature.view()
```

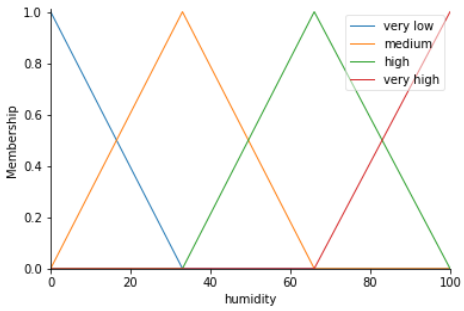


- در متغیر رطوبت چون 4 term فازی داریم مقداردهی به صورت دستی انجام می‌شود.

```
[ ] 1 humidity = ctrl.Antecedent(np.arange(0, 101, 1), 'humidity')

[ ] 1 humidity['very low'] = fuzz.trimf(humidity.universe, [0, 0, 33])
    2 humidity['medium'] = fuzz.trimf(humidity.universe, [0, 33, 66])
    3 humidity['high'] = fuzz.trimf(humidity.universe, [33, 66, 100])
    4 humidity['very high'] = fuzz.trimf(humidity.universe, [66, 100, 100])

[ ] 1 humidity.view()
```

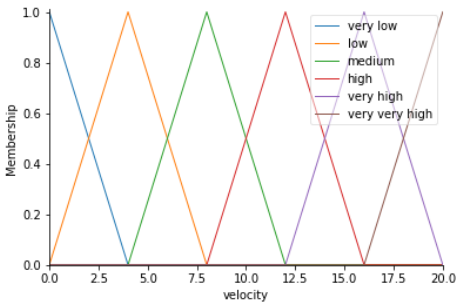


- متغیر سرعت که به عنوان خروجی سیستم فازی می‌باشد نیز به صورت زیر تعریف می‌شود.

```
[ ]  1  velocity = ctrl.Consequent(np.arange(0, 21, 1), 'velocity')
```

```
[ ]  1  velocity['very low'] = fuzz.trimf(velocity.universe, [0, 0, 4])
     2  velocity['low'] = fuzz.trimf(velocity.universe, [0, 4, 8])
     3  velocity['medium'] = fuzz.trimf(velocity.universe, [4, 8, 12])
     4  velocity['high'] = fuzz.trimf(velocity.universe, [8, 12, 16])
     5  velocity['very high'] = fuzz.trimf(velocity.universe, [12, 16, 20])
     6  velocity['very very high'] = fuzz.trimf(velocity.universe, [16, 20, 20])
```

```
     1  velocity.view()
```



- پس از تعریف کلیه متغیرها زمان تعریف قوانین و ساخت سیستم فازی بر اساس قوانین فرا می‌رسد. تعریف متغیرها نیز به صورت زیر انجام می‌شود. لازم به ذکر است در رابطه با قوانینی که فقط برای یکی از ورودی‌ها شرط گذاشته و برای ورودی دیگر شرطی ندارد، باید قوانین به نحوی که به ازای همه مقادیر ورودی دیگر خروجی داشته باشیم بازنویسی شود. به عنوان مثال در قانون اول داریم: اگر درجه حرارت خیلی خیلی سرد بود، آنگاه سرعت پنکه خیلی خیلی کم است. در این حالت باید انواع ورودی رطوبت را هم در نظر بگیریم. این قانون بدین صورت نوشته خواهد شد.

```
rule1 = ctrl.Rule((temperature['very very cold'] & humidity['very low'])|
                  (temperature['very very cold'] & humidity['medium'])|
                  (temperature['very very cold'] & humidity['high'])|
                  (temperature['very very cold'] & humidity['very high'])
                  , velocity['very low'])
```

- سایر قوانین نیز به صورت زیر نوشته می‌شود.

```
rule2 = ctrl.Rule((temperature['very cold'] & humidity['very low'])|
                  (temperature['very cold'] & humidity['medium'])|
                  (temperature['very cold'] & humidity['high'])|
                  (temperature['very cold'] & humidity['very high']),
                  velocity['very low'])

rule3 = ctrl.Rule((temperature['cold'] | temperature['medium']) &
                  humidity['very low'],
                  velocity['low'])

rule4 = ctrl.Rule((temperature['medium'] | temperature['hot'])&
                  (humidity['medium'] | humidity['high']),
                  velocity['medium'])

rule5 = ctrl.Rule(temperature['hot'] & humidity['very high'],
                  velocity['high'])

rule6 = ctrl.Rule((temperature['very hot'] & humidity['very low'])|
                  (temperature['very hot'] & humidity['medium'])|
                  (temperature['very hot'] & humidity['high'])|
                  (temperature['very hot'] & humidity['very high']),
                  velocity['very high'])

rule7 = ctrl.Rule((temperature['very very hot'] & humidity['very low'])|
                  (temperature['very very hot'] & humidity['medium'])|
                  (temperature['very very hot'] & humidity['high'])|
                  (temperature['very very hot'] & humidity['very high']),
                  velocity['very very high'])
```

- در نهایت سیستم فازی با توجه به قوانین تعریف می‌شود.

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

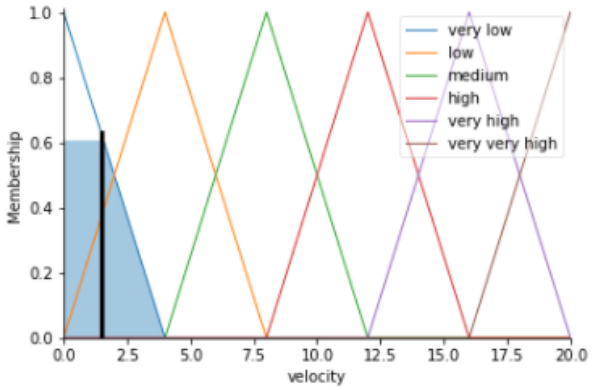
```
[ ]  1  fan_velocity = ctrl.ControlSystemSimulation(fan_ctrl)
```

- برای دو حالت ورودی خروجی به صورت زیر به دست می‌آید.

```
[ ]  1  fan_velocity.input['temperature'] = 0
     2  fan_velocity.input['humidity'] = 20
     3
     4  fan_velocity.compute()
```

```
[ ]  1  print(fan_velocity.output['velocity'])
     2  velocity.view(sim=fan_velocity)
```

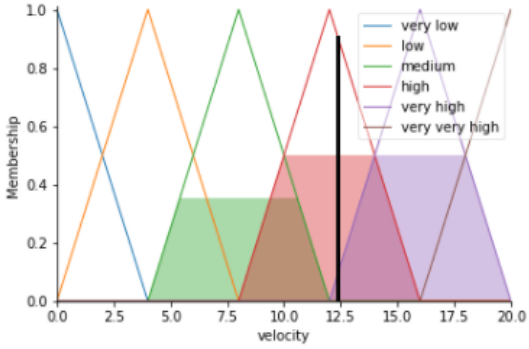
```
1.4817742643829603
```




```
[ ]  1  fan_velocity.input['temperature'] = 35
     2  fan_velocity.input['humidity'] = 88
     3
     4  fan_velocity.compute()
```

```
[ ]  1  print(fan_velocity.output['velocity'])
     2  velocity.view(sim=fan_velocity)
```

```
12.380216273430282
```



Problem 3

- در این سوال ابتدا فضا پیوسته صورت سوال را به فضای گسسته تبدیل می‌کنیم. به این منظور دو متغیر پیوسته فضای حالت ورودی را به 40 مقدار (مقادیر 0 تا 39) تقسیم می‌کنیم. این کار در تابع obs_to_state صورت می‌گیرد. همچنین متغیر فضای خروجی به اعداد 10- تا 10 تبدیل می‌شود.
- در مرحله بعد برای دو ورودی و یک خروجی سیستم را برای سیستم فازی تعریف می‌کنیم.
- در ادامه قوانین مورد نیاز برای راه‌اندازی سیستم را نوشته و کنترلر فازی را تعریف می‌کنید.
- در نهایت به ازای هر state ورودی خروجی را از کنترلر به عنوان "نیروی وارد به ماشین" تعیین می‌شود. فیلم نحوه عملکرد در پوشه MountainCarContinuous-v0-results قابل مشاهده است.

Problem 4 (Bonus)

- **Error**

- **به علت اروری که عکس آن در ادامه آورده شده است، از pendulum-v1 استفاده شده است.**

```
Box([-1. -1. -8.], [1. 1. 8.], (3,), float32)
```

```
KeyError: 'Pendulum-v0'

During handling of the above exception, another exception occurred:

DeprecatedEnv                             Traceback (most recent call last)
<ipython-input-6-2342bdc40ca7> in <module>
      1 if __name__ == '__main__':
----> 2     env = gym.make('Pendulum-v0')
      3     env.seed(0)
      4     array.random.seed(0)
      5     print ('----- Start Learning -----')

~\anaconda3\lib\site-packages\gym\envs\registration.py in make(id, **kwargs)
    233
    234 def make(id, **kwargs):
--> 235     return registry.make(id, **kwargs)
    236
    237

~\anaconda3\lib\site-packages\gym\envs\registration.py in make(self, path, **kwargs)
    126         else:
    127             logger.info("Making new env: %s", path)
--> 128         spec = self.spec(path)
    129         env = spec.make(**kwargs)
    130         return env

~\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'])
```

- ابتدا فضا پیوسته ورودی را به stateهایی در فضای گسسته map می‌کنیم. بدین صورت مقدار cos , sin زاویه یکی از اعداد مجموعه زیر خواهد بود.

```
[-0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 -0.   0.1  0.2  0.3  0.4
  0.5  0.6  0.7  0.8  0.9]
```

- همچنین مقدار ورودی theta dot به صورت زیر می‌شود.

```
[-7. -6. -5. -4. -3. -2. -1.  0.  1.  2.  3.  4.  5.  6.  7.]
```

- در ادامه با توجه به متغیرهای ورودی q-table ساخته شده تا به کمک آن برای انجام اکشن در هر استیت تصمیم گرفته شود.

- به تعداد iteration مشخص الگوریتم آموزش را تکرار می‌کنیم. به کمک پارامتر eps تعیین می‌کنیم که چه زمانی به صورت رندوم action را انتخاب کند و چه زمانی بر اساس action موجود در q-table عمل کند.

- در هر مرحله و با توجه به شرایط مطرح شده در قسمت قبل یک action انتخاب شده و با توجه به reward اکشن انتخاب شده در محیط q-table آپدیت می‌شود.

$$Q(a,i) \leftarrow Q(a,i) + \alpha(R(i) + \gamma \max_{a'} Q(a',j) - Q(a,i))$$

```
q_state_table[current_state][action_index] = q_state_table[current_state][action_index] \
    + alpha * (reward + np.max(q_state_table[new_state])- q_state_table[current_state][action_index])
```

- ضریب یادگیری آلفا با پیشروی الگوریتم و تا مقدار 0.03 کاهش می‌یابد.

- بدین ترتیب و پس از طی کردن تعداد مشخصی iteration و تکمیل q-table حال به ازای هر استیت ورودی اکشن به صورتی انتخاب می‌شود که پاندول به صورت معکوس بایستد.

- نتیجه اجرا در پوشه Pendulum-v1_result قابل مشاهده است.