Computational Intelligence
Dr. Mozayani
Fall 2022
Hoorieh Sabzevari - 98412004
HW3

۱. ابتدا یک کلاس برای شبکه‌ی هاپفیلد خود تعریف می‌کنیم که الگوی ورودی را از طریق ورودی constructor دریافت می‌کند. پارامترهای این کلاس خود الگو، تعداد نورون‌ها، ماتریس وزن‌ها و تعداد ایپاک است.

```python
class Hopfield:
    def __init__(self, patterns):
        self.patterns = patterns
        self.n_patterns = len(patterns[0])
        self.w = None
        self.epoch = None
```

در این کلاس دو تابع تعریف می‌کنیم: تابع آپدیت ماتریس وزن‌ها و تابع پیش‌بینی به ازای الگوی ورودی.

برای تابع اول، با توجه به لیست الگوها و قاعده‌ی هب ماتریس وزن‌ها را محاسبه می‌کنیم. ماتریس وزن‌ها یک ماتریس $n \times n$ است که قطر اصلی آن صفر است و از ضرب داخلی هر الگو در وارونش بدست آمده است.

$$w_{i,j} = \sum_{k=1}^{K} x_i^k \, x_j^k$$

$$w_{i,i} = 0$$

$$w_{i,j} = w_{j,i}$$

```python
def calculate_weight(self):
    temp = np.dot(self.patterns.T, self.patterns)
    np.fill_diagonal(temp, 0.0)
    self.w = temp
    return self.w
```

برای تابع دوم، تخمین شبکه از پایداری یک الگوی ورودی را تعیین می‌کنیم.

$E(i,t)$: Energy of neuron i at time t

$$E(i,t) = -a(i,t)u(i,t) = -a(i,t)(\sum_{j=1}^{N} w_{i,j}\, a(j,t) - \theta_i)$$

$u(i,t+1)$: linear activation of neuron i at time $t+1$

$$u(i,t+1) = \sum_{j=1}^{N} w_{i,j}\, a(j,t) - \theta_i$$

$E(t)$: Total Energy

$$E(t) = \sum_{i=1}^{N} E(i,t) = -\sum_{i=1}^{N} a(i,t)\left(\sum_{j=1}^{N} w_{i,j}\, a(j,t) - \theta_i\right)$$

$$= -\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j} a(i,t) a(j,t) + \sum_{i=1}^{N} a(i,t)\theta_i$$

$a(i,t+1)$: sign activation of neuron i at time $t+1$

$$a(i,t+1) = sign(u(i,t+1))$$

برای بررسی پایداری باید طی چند ایپاک مقادیر a و انرژی را بدست آورده و دو حالت آخر با هم برابر باشند. در این صورت کمترین انرژی را بعنوان خروجی می‌دهیم. اگر در لحظه‌ی اول a با ورودی برابر باشد، ورودی پایدار است. حد آستانه نیز صفر در نظر گرفته شده است.

الف) ابتدا ماتریس وزن‌ها را برای الگوهای ورودی محاسبه می‌کنیم:

```
# given patterns
P = np.array([
    [-1, -1, 1, -1, 1, -1, -1, 1],
    [-1, -1, -1, -1, -1, 1, -1, -1],
    [-1, 1, 1, -1, -1, 1, -1, 1]
])

# create hopfield object
h = Hopfield(P)
w = h.calculate_weight()
print(w)
```

✓ 0.4s

```
[[ 0  1 -1  3  1 -1  3 -1]
 [ 1  0  1  1 -1  1  1  1]
 [-1  1  0 -1  1 -1 -1  3]
 [ 3  1 -1  0  1 -1  3 -1]
 [ 1 -1  1  1  0 -3  1  1]
 [-1  1 -1 -1 -3  0 -1 -1]
 [ 3  1 -1  3  1 -1  0 -1]
 [-1  1  3 -1  1 -1 -1  0]]
```

سپس پایداری الگوها را چک می‌کنیم. همانطور که مشاهده می‌کنیم، هر سه الگو پایدار هستند.

```
    # Check if the given inputs are stable or not
    P = np.array([
        [-1, -1, 1, -1, 1, -1, -1, 1],
        [-1, -1, -1, -1, -1, 1, -1, -1],
        [-1, 1, 1, -1, -1, 1, -1, 1],
    ])

    for i in range(3):
        closest, energy, acc, res = h.calculate_closest(P[i], 5)
        print("\nPattern {}: {}".format(i + 1, res))
```

✓ 0.6s

```
Pattern 1: Stable!

Pattern 2: Stable!

Pattern 3: Stable!
```

ب) سپس الگوهای نویزی داده‌شده را تست می‌کنیم.

```
# Check noisy patterns
P = np.array([
    [1, -1, 1, -1, 1, -1, -1, 1],
    [1, 1, -1, -1, -1, 1, -1, -1],
    [1, 1, 1, -1, 1, 1, -1, 1]
])

for i in range(3):
    closest, energy, acc, res = h.calculate_closest(P[i], 100)
    print("\nPattern {}: {}".format(i + 1, res))
    print("Closest node is {} with accuracy {}%".format(closest, acc))
    print("Energy is {}".format(energy))
```

```
Pattern 1: Not Stable!
Closest node is [-1 -1  1 -1  1 -1 -1  1] with accuracy 87.5%
Energy is -44

Pattern 2: Not Stable!
Closest node is [ 1  1 -1 -1 -1  1 -1 -1] with accuracy 100.0%
Energy is -12

Pattern 3: Not Stable!
Closest node is [-1 -1  1 -1  1  1 -1  1] with accuracy 75.0%
Energy is -32
```

هر سه الگو ناپایدار هستند و فقط الگوی اول به الگوی متناظر خود همگرا شده است.

ت) برای این سوال ابتدا تمام الگوهای ممکن را تولید کرده و سپس پایداری هرکدام را توسط شبکه‌ی خود تست می‌کنیم.

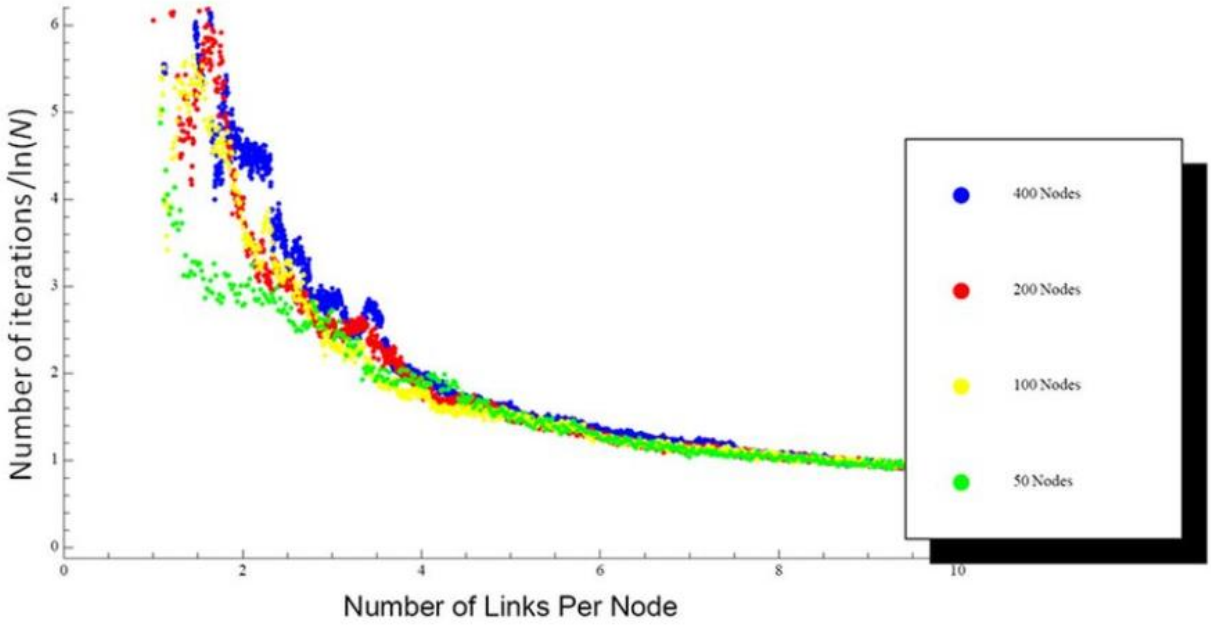
```python
l = [-1, 1]
all_patterns = list(product(l, repeat=8))
stable_patterns = []

for i in range(256):
    closest, energy, acc, res = h.calculate_closest(all_patterns[i], 5)
    if(res=='Stable!'):
        stable_patterns.append(all_patterns[i])

print("{} stable patterns are found!".format(len(stable_patterns)))
for p in stable_patterns:
    print(p)
```

✓ 0.8s

```
6 stable patterns are found!
(-1, -1, -1, -1, -1, 1, -1, -1)
(-1, -1, 1, -1, 1, -1, -1, 1)
(-1, 1, 1, -1, -1, 1, -1, 1)
(1, -1, -1, 1, 1, -1, 1, -1)
(1, 1, -1, 1, -1, 1, 1, -1)
(1, 1, 1, 1, 1, -1, 1, 1)
```

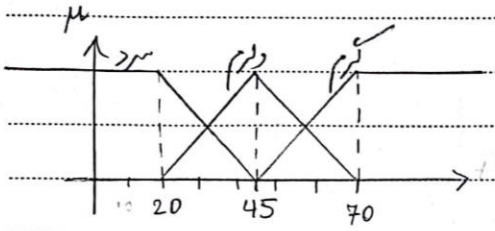
۶ الگوی پایدار تشخیص داده شد.

ث) شبکه فقط در صورتی همگرا نمی‌شود که در یک لوپ گیر کند. این مورد زمانی پیش می‌آید که شبکه‌ی ما سنکرون نباشد و هر لحظه آپدیت نشود (مانند مثال کوییز کلاسی). برای حل این مشکل شبکه باید از ابتدا سنکرون باشد. اگر الگو پایدار باشد که همگرا می‌شود و اگر هم پایدار نباشد باز به یکی از الگوهای نزدیک شده و همگرا می‌شود. همچنین در شبکه‌ی هاپفیلد اگر تمام بیت‌ها را برعکس کنیم باز هم الگو متناظر با یک نقطه مینیمم انرژی در شبکه است. لذا اگر بیشتر از نصف بیت‌ها را معکوس کنیم ممکن است به معکوس یکی از الگوهایی که آموزش داده شده است برسد و یا نزدیک شود که باز هم همگرا خواهد شد. برای تعداد iteration های رسیدن به همگرایی نیز نمودار زیر موجود است که نشان می‌دهد بستگی به تعداد نورون‌ها و وزن‌ها دارد. (لینک کمکی)

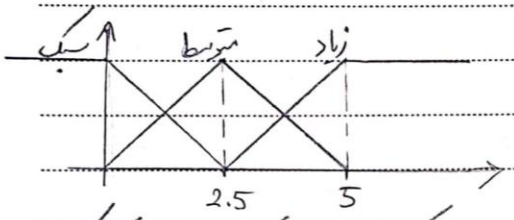


(2)

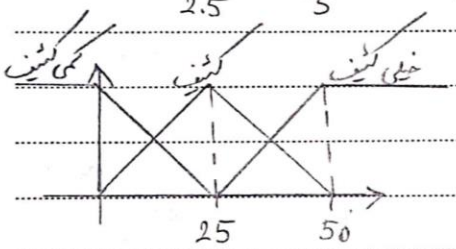
دمای آب [20-70]
- گرم
- ولرم
- سرد



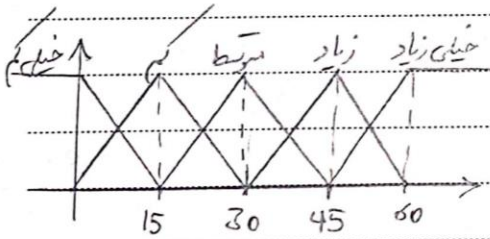
وزن ظرف [0-5]
- زیاد
- متوسط
- سبک



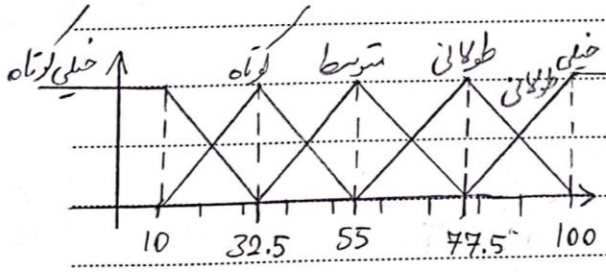
کثیفی [0-50]
- خیلی کثیف
- کثیف
- کمی کثیف



سرعت چرخش موتور [0-60]
- خیلی زیاد
- زیاد
- متوسط
- کم
- خیلی کم



زمان شست و شو [10-100]
- خیلی طولانی
- طولانی
- متوسط
- کوتاه
- خیلی کوتاه



MOM

Max-Min

وزن 4kg : $\frac{x-2.5}{2.5} = \frac{4-2.5}{2.5} = \frac{1.5}{2.5} = 0.6$ زیاد

چربی 45 : $\frac{45-25}{25} = \frac{20}{25} = 0.8$ خیلی کثیف

آب $20^{\circ}c$: $\frac{45-20}{25} = 1$ سرد

اگر ظرف خیلی کثیف و وزن زیاد و آب سرد باشد باید سرعت موتور خیلی زیاد و زمان شست وشو خیلی طولانی باشد.

$\min(0.6, 0.8) = 0.6$

$\min(0.6, 1) = 0.6$ $\Longrightarrow$ $\max(0.6, 0.6, 0.8) = 0.8$

$\min(0.8, 1) = 0.8$

سرعت موتور : $\frac{0.8}{1} = \frac{12}{15} \longrightarrow 45 + 12 = \boxed{57}$ دور در دقیقه

زمان شست وشو : $\frac{0.8}{1} = \frac{18}{22.5} \longrightarrow 77.5 + 18 = \boxed{95.5}$ دقیقه

۳.

ابتدا یک کلاس کنترلر فازی تعریف کرده و در تابع define_terms() ترم‌های زبانی خود را تعریف می‌کنیم.

```
def define_terms(self):
    temperture = AutoTriangle(5, terms=['VeryCold', 'Cold', 'Medium', 'Hot', 'VeryHot'], universe_of_discourse=[0, 40])
    humidity = AutoTriangle(5, terms=['VeryLow', 'Low', 'Medium', 'High', 'VeryHigh'], universe_of_discourse=[0, 100])
    rain = AutoTriangle(3, terms=['Low', 'Medium', 'High'], universe_of_discourse=[0, 100])
    height = AutoTriangle(3, terms=['Low', 'Medium', 'High'], universe_of_discourse=[0, 1000])
    self.fuzzy_system.add_linguistic_variable("PastTemperture", temperture)
    self.fuzzy_system.add_linguistic_variable("Humidity", humidity)
    self.fuzzy_system.add_linguistic_variable("Rain", rain)
    self.fuzzy_system.add_linguistic_variable("Height", height)
    self.fuzzy_system.add_linguistic_variable("Temperture", temperture)
```

سپس قواعد خود را تعریف کرده و آن‌ها را نیز به کنترلر خود اضافه می‌کنیم.

```
rules = [
        "IF (PastTemperture IS Medium) AND (Humidity IS High) AND (Rain IS Low) AND (Height IS Medium) THEN (Temperture IS Hot)",
        "IF (PastTemperture IS Hot) AND (Humidity IS Medium) AND (Rain IS Low) AND (Height IS Low) THEN (Temperture IS VeryHot)",
        "IF (PastTemperture IS Hot) AND (Humidity IS High) AND (Rain IS Medium) AND (Height IS High) THEN (Temperture IS Hot)",
        "IF (PastTemperture IS VeryHot) AND (Humidity IS Medium) AND (Rain IS Medium) AND (Height IS Low) THEN (Temperture IS VeryHot)",
        "IF (PastTemperture IS Medium) AND (Humidity IS Medium) AND (Rain IS Medium) AND (Height IS Medium) THEN (Temperture IS Medium)",
        "IF (PastTemperture IS Medium) AND (Humidity IS Low) AND (Rain IS High) AND (Height IS High) THEN (Temperture IS Cold)",
        "IF (PastTemperture IS Cold) AND (Humidity IS High) AND (Rain IS High) AND (Height IS Medium) THEN (Temperture IS Cold)",
        "IF (PastTemperture IS VeryCold) AND (Humidity IS High) AND (Rain IS High) AND (Height IS High) THEN (Temperture IS VeryCold)"
        ]
```

در انتها مقادیر پارامترها را ست می‌کنیم تا دمای امروز را پیش‌بینی کنیم.

```
FC = FuzzyController()
FC.define_terms()
FC.add_rules(rules)
FC.add_variables(values)
temp = FC.inference()
print(temp)
```

```
{'Temperture': 19.999999999999847}
```