به نام خدا

درس مبانی هوش محاسباتی

تمرین سری سوم

مدرس درس:
جناب آقای دکتر مزینی

تهیه شده توسط:
الناز رضایی ۹۸۴۱۱۳۸۷

تاریخ ارسال: ۱۴۰۱/۱۰/۱۵

سوال ۱:

یک شبکه هاپفیلد پیاده‌سازی کنید که بتواند الگوهای زیر را به خاطر بسپارد:

$$X_1 = [-1, -1, 1, -1, 1, -1, -1, 1]$$

$$X_2 = [-1, -1, -1, -1, -1, 1, -1, -1]$$

$$X_3 = [-1, 1, 1, -1, -1, 1, -1, 1]$$

آ) نشان دهید که آیا الگوهای بالا برای شبکه هاپفیلد پایدار هستند یا نه.

ب) حالا می‌خواهیم ببینیم شبکه ما قابلیت به خاطرآوردن الگوهای ورودی نویزی را دارد؟ برای این کار الگوهای زیر که نمونه نویزی الگوهای اصلی هستند را تعریف می‌کنیم:

$$X_{1n} = [1, -1, 1, -1, 1, -1, -1, 1]$$

$$X_{2n} = [1, 1, -1, -1, -1, 1, -1, -1]$$

$$X_{3n} = [1, 1, 1, -1, 1, 1, -1, 1]$$

X_{1n} دارای یک بیت خطا است ولی X_{2n} و X_{3n} دارای دو بیت خطا هستند.

این الگوها را به شبکه بدهید و تا زمانی که به یک نقطه پایدار برسید عملیات به‌روزرسانی را انجام دهید. آیا تمام الگوها به الگوهای متناظر خودشان همگرا شده‌اند؟

ت) غیر از الگوهایی که شبکه را با آنها آموزش دادیم، چه الگوهای پایدار دیگری داریم که شبکه به سمت آنها همگرا خواهد شد؟ این الگوها را به دست آورید.

ث) اگر الگوهایی که به عنوان ورودی به شبکه می‌دهیم دارای نویز بیشتری نسبت به موارد قبل باشند (برای مثال بیش از نصف بیت‌ها دچار خطا شده باشند) چه اتفاقی خواهد افتاد؟ در مورد همگراشدن و تعداد iteration لازم برای آن توضیح دهید.

پاسخ ۱:

به منظور پیاده‌سازی شبکه هاپفیلد، از کد زیر استفاده می‌کنیم. N در اینجا، تعداد نورون، P لیست Patternها و W وزن‌هایمان می‌باشد. ابتدا در تابع learning، با استفاده از Hebb rule به محاسبه ماتریس وزن‌ها پرداختیم. برای این کار، هر Pattern را در transpose آن ضرب کرده و با ماتریس وزن‌ها جمع می‌کنیم. در تابع predict، ورودی را گرفته و سپس به تعداد epochها، خروجی هر نورون را به دست آورده و یک activation function از نوع sign روی آن زده و اختلاف آن‌ها را

محاسبه می‌کنیم. اگر پایداری اتفاق افتاد که الگویی که کمترین انرژی را دارد برگردانده و در خروجی نیز کلمه stable را نمایش داده و از حلقه خارج می‌شود؛ اما در صورتی که پایداری اتفاق نیفتاد، الگوریتم به تعداد epochها تکرار می‌شود. در حل این سوال، از روابط ریاضی زیر استفاده شد.

$$w_{i,j} = \sum_{k=1}^{N} x_i^k x_j^k \quad , \quad w_{i,i} = 0 \quad , \quad w_{i,j} = w_{j,i}$$

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

```python
class Hopfield:
  def __init__(self, N):
    self.N = N
    self.P = 0
    self.W = np.zeros((N, N), dtype=np.int32)

  def learning(self, P):
    for p in P:
      self.W += np.dot(np.vstack(p), np.vstack(p).T)
    np.fill_diagonal(self.W, 0.0)
    return self.W

  def predict(self, y, epochs):
    i = np.array(y)
    yhats_losses = [(i, math.inf)]
    stable = False
    for j in range(epochs):
      z = np.dot(i, self.W)
      y_hat = np.sign(z)
      equality = np.equal(y, y_hat)
      temp = np.count_nonzero(equality)
      loss = self.P - temp
      yhats_losses.append((y_hat, loss))
      stability = np.array_equal(yhats_losses[-1][0], yhats_losses[-2][0])
      if (stability):
        if j == 0:
          stable = True
        return y_hat, stable

    yp = min(result, key = lambda t: t[1])[0]
    return yp, stable
```

آ) برای حل این بخش، ابتدا شبکه خود را با الگوهای داده شده آموزش داده و سپس پایداری آن‌ها را با استفاده از تابع predict به دست می‌آوریم. کد زیر نشان می‌دهد که هر ۳ الگو پایدار هستند.

```
x1 =  np.array([-1, -1, 1, -1, 1, -1, -1, 1])
x2 = np.array([-1, -1, -1, -1, -1, 1, -1, -1])
x3 = np.array([-1, 1, 1, -1, -1, 1, -1, 1])
patterns = np.array([x1, x2, x3])
hopfield = Hopfield(8)
hopfield.learning(patterns)
for pattern in patterns:
  result, stability = hopfield.predict(pattern, epochs=10)
  print("pattern: ", pattern," ", "Result: ", result, " ", "Stabiity: ", stability)
```

```
pattern:  [-1 -1  1 -1  1 -1 -1  1]   Result:  [-1 -1  1 -1  1 -1 -1  1]   Stabiity:  True
pattern:  [-1 -1 -1 -1 -1  1 -1 -1]   Result:  [-1 -1 -1 -1 -1  1 -1 -1]   Stabiity:  True
pattern:  [-1  1  1 -1 -1  1 -1  1]   Result:  [-1  1  1 -1 -1  1 -1  1]   Stabiity:  True
```

ب) با دادن این الگوها به شبکه، الگوی اول مطابق الگوی متناظر خودش همگرا شد و مابقی اختلاف داشتند. این می‌تواند به علت این موضوع باشد که در الگوی اول، اختلاف یک بیت بود و در بقیه الگوها دو بیت. بنابراین نتیجه می‌گیریم شبکه‌مان قابلیت یادگیری داده‌های نویزی را ندارد. البته می‌توان گفت ممکن است شبکه قابلیت یادگیری داده‌های نویزی با حداکثر اختلاف ۱ بیت را داشته باشد، اما این هم قابل اثبات نیست.

```
x1n =  np.array([1, -1, 1, -1, 1, -1, -1, 1])
x2n = np.array( [1, 1, -1, -1, -1, 1, -1, -1])
x3n = np.array([1, 1, 1, -1, 1, 1, -1, 1])
patterns = np.array([x1n, x2n, x3n])
for pattern in patterns:
  result, stability = hopfield.predict(pattern, epochs=10)
  print("pattern: ", pattern," ", "Result: ", result, " ", "Stabiity: ", stability)
```

```
pattern:  [ 1 -1  1 -1  1 -1 -1  1]   Result:  [-1 -1  1 -1  1 -1 -1  1]   Stabiity:  False
pattern:  [ 1  1 -1 -1 -1  1 -1 -1]   Result:  [-1 -1 -1  1 -1  1  1 -1]   Stabiity:  False
pattern:  [ 1  1  1 -1  1  1 -1  1]   Result:  [-1  1  1 -1 -1 -1 -1  1]   Stabiity:  False
```

ت) در اینجا ۲۵۶ حالت را باید بررسی کنیم تا ببینیم شبکه به سمت آن‌ها همگرا می‌شود یا خیر؛ چرا که هر الگو دارای ۸ بیت است و هر بیت یا ۱ است یا ۱-. بنابراین در حلقه for اول، تمامی الگوهای ممکن را در می‌آوریم و در یک آرایه ریخته تا در مرحله بعد، بررسی کنیم که شبکه به سمت کدام یک‌ها همگرا می‌شود. در حلقه for دوم، شبکه هاپفیلد را بر روی هر یک از این الگوها بررسی کرده و الگوهای پایدار را چاپ می‌کنیم. همچنین تعداد الگوهای پایدار را محاسبه کرده و آن هم به همراه خروجی نمایش می‌دهیم. کدی که برای این بخش زده شد، در تصویر زیر قابل مشاهده است.

```
patterns = []
for i in range(256):
  x = bin(i)[2:].zfill(8)
  pattern = [2*int(x[i])-1 for i in range(8)]
  patterns.append(pattern)

count = 0
print("Stable patterns are: ")
for pattern in patterns:
  result, stability = hopfield.predict(pattern, epochs=10)
  if(stability):
    print(result)
    count += 1
print("The number of stable patterns is: ", count)
```

```
Stable patterns are:
[-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]
The number of stable patterns is:  6
```

ث) اگر pattern دارای نویز نیز پایدار باشد، مشکلی نداریم و شبکه همگرا می‌شود؛ اما اگر پایدار نباشد، می‌توانیم از شبکه هاپفیلد asynchronization یا رویکرد stochastic استفاده کنیم، قطعا همگرا می‌شود اما اگر از synchronization هاپفیلد بهره ببریم، ممکن است همگرا نشود. در مورد تعداد iterationهای لازم نظر قطعی نمی‌توان داد. در اینجا چون تعداد نورون‌ها کم است، احتمالا بعد از ۱۰ iteration همگرا شویم و ممکن هم هست به‌طور کامل همگرا نشویم.

لینک استفاده شده در حل این سوال:

https://ttu-ir.tdl.org/bitstream/handle/2346/10832/31295005976443.pdf;sequence=1

سوال ۲:

می‌خواهیم با استفاده از منطق فازی یک ماشین ظرفشویی طراحی کنیم و اطلاعات زیر موجود است: دمای آب بین ۲۰ درجه تا ۷۰ می‌تواند باشد، وزن ظروف بین صفر تا ۵ کیلو، کثیفی ظروف با یک سنسور چربی بین صفر (کمی کثیف) تا ۵۰ (خیلی کثیف) سنجیده می شود. سرعت چرخش موتور بین صفر تا ۶۰ دور در دقیقه و زمان شستشو بین ۱۰ دقیقه تا ۱۰۰ دقیقه است.

برخی از قواعد تجربی که توسط یک متخصص ارایه شده است به قرار زیر است:

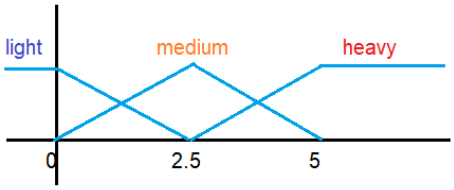
- اگر ظروف کمی کثیف و وزن سبک و آب سرد باشد باید سرعت موتور زیاد و زمان شستشو کوتاه باشد.
- اگر ظروف کمی کثیف و وزن متوسط و آب سرد باشد باید سرعت موتور خیلی زیاد و زمان شستشو متوسط باشد.
- اگر ظروف کمی کثیف و وزن زیاد و آب ولرم باشد باید سرعت موتور متوسط و زمان شستشو طولانی باشد.
- اگر ظروف کثیف و وزن سبک و آب سرد باشد باید سرعت موتور زیاد و زمان شستشو خیلی طولانی باشد.
- اگر ظروف خیلی کثیف و وزن زیاد و آب ولرم باشد باید سرعت موتور خیلی کم و زمان شستشو خیلی طولانی باشد.
- اگر ظروف کثیف و وزن زیاد و آب گرم باشد باید سرعت موتور کمی زیاد و زمان شستشو نسبتا طولانی باشد.
- اگر ظروف خیلی کثیف و وزن زیاد و آب سرد باشد باید سرعت موتور خیلی زیاد و زمان شستشو خیلی طولانی باشد.

این کنترلر فازی را دقیقا طراحی کنید و بگویید برای وضعیت ۴ کیلو ظروف با درجه چربی ۴۵ و دمای آب ۲۰ درجه، خروجی چه باید باشد؟

پاسخ ۲:

Weight of the container: [0 - 5]

light - medium - heavy

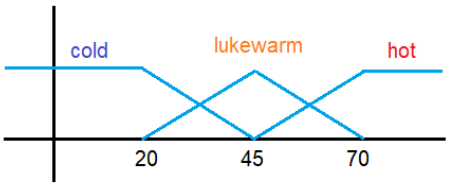


$$\mu_l(w): \begin{cases} 1 & w < 0 \\ -0.4w + 1 & 0 < w < 2.5 \end{cases} \qquad \mu_m(w): \begin{cases} 0.4w & 0 < w < 2.5 \\ -0.4w + 2 & 2.5 < w < 5 \end{cases}$$

$$\mu_h(w): \begin{cases} 0.4w - 1 & 2.5 < w < 5 \\ 1 & w > 5 \end{cases}$$

Water temperature: [20 - 70]

cold - lukewarm - hot

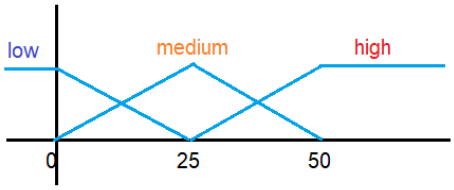


$$\mu_c(t): \begin{cases} 1 & t < 20 \\ -0.04t + 1.8 & 20 < t < 45 \end{cases} \qquad \mu_l(t): \begin{cases} 0.04t - 0.8 & 20 < t < 45 \\ -0.04t + 2.8 & 45 < t < 75 \end{cases}$$

$$\mu_h(t): \begin{cases} 0.04t - 1.8 & 45 < t < 70 \\ 1 & t > 70 \end{cases}$$

Dirt of dishes: [0 - 50]

low - medium - high



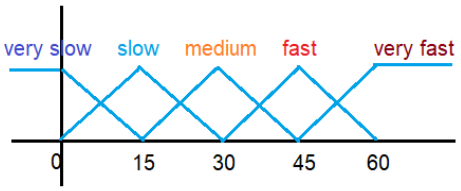
$$\mu_l(d): \begin{cases} 1 & d < 0 \\ -0.04d + 1 & 0 < d < 25 \end{cases} \qquad \mu_m(d): \begin{cases} 0.04d & 0 < d < 25 \\ -0.04d + 2 & 25 < d < 50 \end{cases}$$

$$\mu_h(d): \begin{cases} 0.04d - 1 & 25 < d < 50 \\ 1 & d > 50 \end{cases}$$

Engine rotation speed: [0 - 60]

very slow - slow - medium -

fast - very fast



$$\mu_{vs}(s): \begin{cases} 1 & s < 0 \\ -\frac{s}{15} + 1 & 0 < s < 15 \end{cases} \quad \mu_s(s): \begin{cases} \frac{s}{15} & 0 < s < 15 \\ -\frac{s}{15} + 2 & 15 < s < 30 \end{cases}$$

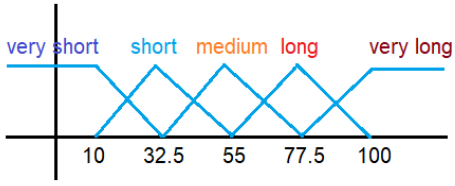
$$\mu_m(s): \begin{cases} \frac{s}{15} - 1 & 15 < s < 30 \\ -\frac{s}{15} + 3 & 30 < s < 45 \end{cases} \quad \mu_f(s): \begin{cases} \frac{s}{15} - 2 & 30 < s < 45 \\ -\frac{s}{15} + 4 & 45 < s < 60 \end{cases}$$

$$\mu_{vf}(s): \begin{cases} \frac{s}{15} - 3 & 45 < s < 60 \\ 1 & s > 60 \end{cases}$$

Washing duration: [10 - 100]

very short - short - medium -

long - very long



$$\mu_{vsh}(d): \begin{cases} 1 & d < 10 \\ -\frac{2}{45}d + \frac{65}{45} & 10 < d < 32.5 \end{cases} \quad \mu_{sh}(d): \begin{cases} \frac{2}{45}d - \frac{20}{45} & 10 < d < 32.5 \\ -\frac{2}{45}d + \frac{110}{45} & 32.5 < d < 55 \end{cases}$$

$$\mu_m(d): \begin{cases} \frac{2}{45}d - \frac{65}{45} & 32.5 < d < 55 \\ -\frac{2}{45}d + \frac{155}{45} & 55 < d < 77.5 \end{cases} \quad \mu_l(d): \begin{cases} \frac{2}{45}d - \frac{110}{45} & 55 < d < 77.5 \\ -\frac{2}{45}d + \frac{200}{45} & 77.7 < d < 100 \end{cases}$$

$$\mu_{vl}(d): \begin{cases} \frac{2}{45}d - \frac{155}{45} & 77.7 < d < 100 \\ 1 & d > 100 \end{cases}$$

Mamdani(max-min):

$$weight = 4kg : \begin{cases} \mu_m(4) = -0.04 * (4) + 2 = 0.4 \\ \mu_h(4) = 0.4 * (4) - 1 = 0.6 \end{cases}$$

$$dirt = 45^\circ : \begin{cases} \mu_m(45) = -0.04 * (45) + 2 = 0.2 \\ \mu_h(45) = 0.04 * (4) - 1 = 0.8 \end{cases}$$

$$temperature = 20^\circ : \begin{cases} \mu_c(20) = 1 \\ \mu_m(20) = 0 \end{cases}$$

Dishes are heavy and very dirty, and the water is cold, so engine speed must be very fast and washing duration should be very long (the last rule).

$min(0.6, 0, 8) = 0.6 \quad , \quad min(0.8, 1) = 0.8 \quad , \quad min(0.6, 1) = 0.6$

$max(0.6, 0.6, 0.8) = 0.8$

$\mu_{vf}(s) = 0.8 \implies \frac{s}{15} - 3 = 0.8 \implies s = 57\ [min]$

$\mu_{vl}(d) = 0.8 \implies \frac{2}{45}d - \frac{155}{45} = 0.8 \implies d = 95.5\ [min]$

سوال ۳:

از شما می‌خواهیم که به کمک سیستم فازی، یک کنترلر بنویسید که با توجه به داده‌های روزهای پیشین و دما در آن روز، نتیجه‌گیری داشته باشید که بتواند با استفاده از داده‌های جدید دمای مورد نظر را پیش‌بینی کنید. فاکتورهایی که می‌توانند در دمای آن روز تاثیر بگذارند:

- دمای هوا در سه روز گذشته
- رطوبت هوا
- میزان بارش باران به میلی‌متر
- ارتفاع آن شهر یا مکان

و از شما با توجه به پارامترهای بالا مقدار دمای هوای امروز را می‌خواهیم. توجه داشته باشید که برای هر پارامتر حداقل ۳ ترم تعریف کنید. مثال برای رطوبت هوا (خیلی کم، کم، متوسط، زیاد، خیلی

زیاد) را می‌توان تعریف کرد که ۵ ترم می‌باشد. و برای هر کدام از آنها بازه مقادیری که می‌توانند بگیرند را نیز باید مشخص کرد که برای رطوبت هوا بین ۰ تا ۱۰۰ می‌تواند باشد. (میزان اعداد مشخص شده به دلخواه می‌باشد). با توجه متغیرهای مشخص شده، باید قوانین را مشخص کنید. مثال "اگر دمای دیروز سرد باشد و میزان بارش باران زیاد باشد، دمای امروز خیلی سرد میباشد." (توجه داشته باشید که قوانین مشخص شده نیز دلخواه هستند که با ترکیب متغیرها می‌توانید قوانین خود را مشخص کنید) — حداقل ۸ قانون را مشخص کنید .در آخر نیز با مقدار دادن به تمام پارامترهای بالا، دمای هوای امروز را پیش‌بینی کنید. برای پیاده‌سازی این سوال از کتابخانه simpful استفاده کنید.

پاسخ ۳:

ابتدا سیستم فازی خود را با ترم‌هایش تعریف می‌کنیم.

```
[5] temperature = AutoTriangle(3, terms=['cool', 'norminal', 'warm'], universe_of_discourse=[0, 50])
    humidity = AutoTriangle(5, terms=['very_low', 'low', 'normal', 'high', 'very_high'], universe_of_discourse=[0, 100])
    rain_amount = AutoTriangle(5, terms=['very_low', 'low', 'normal', 'high', 'very_high'], universe_of_discourse=[0, 1000])
    height = AutoTriangle(3, terms=['low', 'normal', 'high'], universe_of_discourse=[0, 1000])
```

سپس متغیرهای خود را که مربوط به آب و هوا هستند اضافه می کنیم تا سیستم فازی ما آن‌ها را بشناسد.

```
FS = FuzzySystem()
FS.add_linguistic_variable("today_tmp", temperature)
FS.add_linguistic_variable("one_prev_day", temperature)
FS.add_linguistic_variable("two_prev_day", temperature)
FS.add_linguistic_variable("three_prev_day", temperature)
FS.add_linguistic_variable("humidity", humidity)
FS.add_linguistic_variable("rain_amount", rain_amount)
FS.add_linguistic_variable("height", height)
```

```
 _____ __  __ ____  _____ _____ _   _ _     v2.9.0
/ ___)(  )(  \/ )(  _ \(  _  )( \( ) ( \(  )
\___ \ )(  )    (  )___/ )(_)(  )  (   )  (
(____/(__)(__\_)(__)  (_____)(_)\_) (_/\_)

Created by Marco S. Nobile (m.s.nobile@tue.nl)
and Simone Spolaor (simone.spolaor@unimib.it)
```

در ادامه، قانون‌های دلخواه را تعریف می‌کنیم.

```
Rules = ["IF (one_prev_day IS cool) AND (rain_amount IS high) THEN (today_tmp IS cool)",
         "IF (one_prev_day IS norminal) AND (rain_amount IS low) THEN (today_tmp IS norminal)",
         "IF (height IS high) AND (rain_amount IS high) THEN (today_tmp IS cool)",
         "IF (one_prev_day IS norminal) AND (two_prev_day IS cool) THEN (today_tmp IS norminal)",
         "IF (height IS low) AND (rain_amount IS very_low) THEN (today_tmp IS warm)",
         "IF (one_prev_day IS cool) OR (rain_amount IS high) THEN (today_tmp IS warm)",
         "IF (three_prev_day IS cool) OR (two_prev_day IS cool) THEN (today_tmp IS cool)",
         "IF (height IS normal)  OR (rain_amount IS very_low) THEN (today_tmp IS norminal)",
         "IF (three_prev_day IS cool) AND (two_prev_day IS warm) AND (one_prev_day IS warm) THEN (today_tmp IS warm)",
         "IF (rain_amount IS high) AND (humidity IS high) THEN (today_tmp IS cool)",
         "IF (humidity IS low) THEN (today_tmp IS warm)"]
FS.add_rules(Rules, verbose=True)
```

حال متغیرهای اعلام شده قبلی خود را به یک عدد تنظیم می کنیم.

```
[8]  FS.set_variable("one_prev_day", 15)
     FS.set_variable("two_prev_day", 10)
     FS.set_variable("three_prev_day", 13)
     FS.set_variable("humidity", 50)
     FS.set_variable("rain_amount", 600)
     FS.set_variable("height", 400)
```

نتیجه حاصل شده به شرح زیر است.

```
[9]  result = FS.inference()
     print(result)
```