Computational Intelligence *third* assignment

Mohammad ali farahat

97521423

1.1) First point is that when we save an input into Hopfield network, the reverse of that input also saves into network. So, for saving this list: [(1,1,1,1) , (-1,-1,-1,-1) , (1,1,-1,-1) , (-1,-1,1,1)] , we only need to save (1,1,1,1) and (1,1,-1,-1), the others will save automatically and we don't need to add them.

Therefore, answer is **YES**, we can save them all.

Here is calculation of weights matrix:

X	0	0	0
0	X	0	0
0	0	X	0
0	0	0	X

Apply (1,1,1,1) →

X	1	1	1
1	X	1	1
1	1	X	1
1	1	1	X

X	1	1	1
1	X	1	1
1	1	X	1
1	1	1	X

Apply (1,1,-1,-1) →

X	2	0	0
2	X	0	0
0	0	X	2
0	0	2	X

in the next page I'm going to test network and find energy of each input to see if it is stable or not.

$(1,1,1,1) \dashrightarrow 0 + 2 + 0 + 0 = 2 \rightarrow 1$
$2 + 0 + 0 + 0 = 2 \rightarrow 1$
$0 + 0 + 0 + 2 = 2 \rightarrow 1$
$0 + 0 + 2 + 0 = 2 \rightarrow 1$

STABLE

$(-1,-1,-1,-1) \dashrightarrow 0 - 2 + 0 + 0 = -2 \rightarrow -1$
$-2 + 0 + 0 + 0 = -2 \rightarrow -1$
$0 + 0 + 0 - 2 = -2 \rightarrow -1$
$0 + 0 - 2 + 0 = -2 \rightarrow -1$

STABLE

$(1,1,-1,-1) \dashrightarrow 0 + 2 + 0 + 0 = 2 \rightarrow 1$
$2 + 0 + 0 + 0 = 2 \rightarrow 1$
$0 + 0 + 0 - 2 = -2 \rightarrow -1$
$0 + 0 - 2 + 0 = -2 \rightarrow -1$

STABLE

$(-1,-1,1,1) \dashrightarrow 0 + 2 + 0 + 0 = -2 \rightarrow -1$
$-2 + 0 + 0 + 0 = -2 \rightarrow -1$
$0 + 0 + 0 + 2 = 2 \rightarrow 1$
$0 + 0 + 2 + 0 = 2 \rightarrow 1$

STABLE

All of them are stable, so all of them are local minimum and we don’t need calculate energy for them.
But instead of this work we could calculate energy of each possible input (16 different input) and show that these inputs are minimum of them.

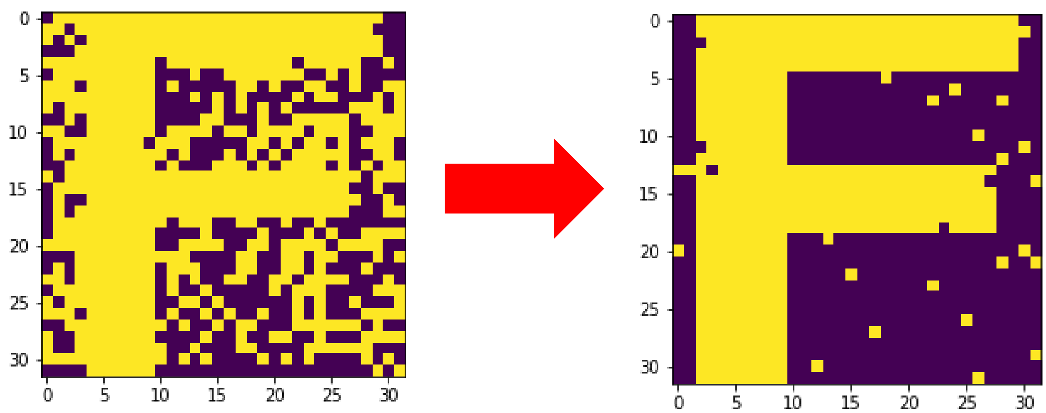
1.2) In this question we are going to implement simple Hopfield network, and save 2 inputs in it. And after that we show that inputs are stable.

- My Hopfield class have two methods: Save() , CheckInput() Save method is used to save input list into network, and CheckInput is used to check if an input is stable or not and if input was unstable tell us nearest saved data.
- We could also find nearest stable data in a loop, but for this example we don't need more depth to find it, one is enough.
- If we input pattern (-1,1,1,-1,-1,-1), our network finds nearest stable pattern to it, and as shown in output, it is : (1,1,1,-1,-1,-1)
- Code is also clear and commented for better understanding.
- Output should look like this:

```
updated weights by input:  [ 1  1  1 -1 -1 -1] is :
 [[ 0  1  1 -1 -1 -1]
 [ 1  0  1 -1 -1 -1]
 [ 1  1  0 -1 -1 -1]
 [-1 -1 -1  0  1  1]
 [-1 -1 -1  1  0  1]
 [-1 -1 -1  1  1  0]]
updated weights by input:  [ 1 -1  1 -1  1 -1] is :
 [[ 0  0  2 -2  0 -2]
 [ 0  0  0  0 -2  0]
 [ 2  0  0 -2  0 -2]
 [-2  0 -2  0  0  2]
 [ 0 -2  0  0  0  0]
 [-2  0 -2  2  0  0]]
[ 1  1  1 -1 -1 -1] is STABLE
[ 1 -1  1 -1  1 -1] is STABLE
[-1  1  1 -1 -1 -1] is UNSTABLE, the nearest saved data is:  [ 1  1  1 -1 -1 -1]
```

1.3) In this question I trained a Hopfield network for English alphabets. After training I added **gaussian** noise to each image and give them to Hopfield network to detect them. The difference between this question and previous question is that I used <u>Async update</u> for checking inputs and it is so much better than before.

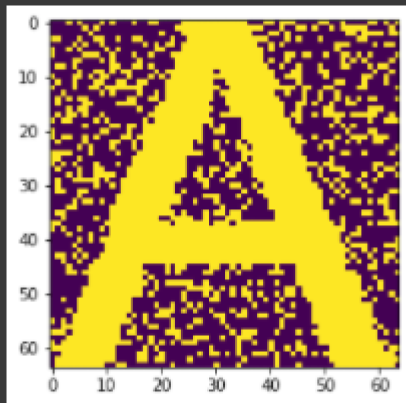
- For better view I used images to show output in each step.
- We can see an input and result as an example (60% of noise):



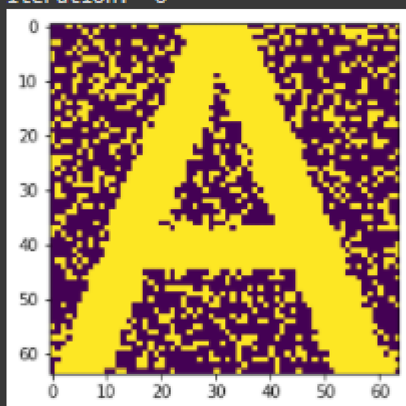
- For measure accuracy I write a method that gives 2 image and return the number of differences in pixels.
- Here is the list of accuracy for "A":

font_size \ Noise	10%	30%	60%
16	56% ×	56% ×	56% ×
32	98.7% ✓	97.9% ✓	97.1% ✓
64	99.3% ✓	97.7% ✓	94.5% ✓

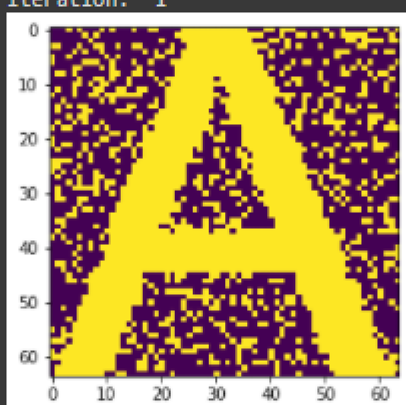
You can run the code to see result like this :



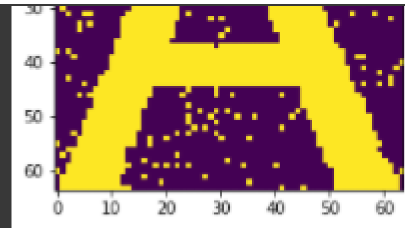
```
iteration:  0
```



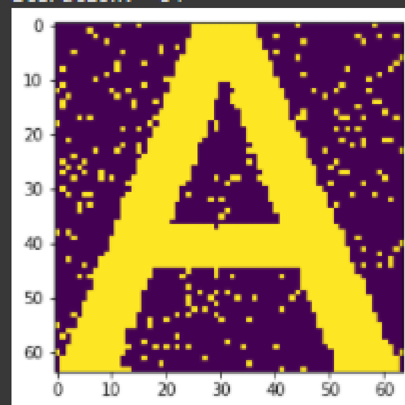
```
iteration:  1
```



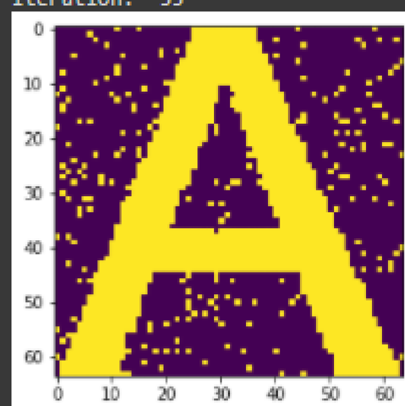
```
iteration:  2
```



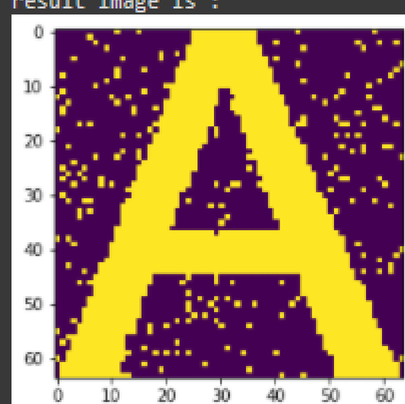
```
iteration:  54
```



```
iteration:  55
```



```
Network find an answer
result image is :
```



```
finished in: 55
ACC is:  0.94580078125
```

FUZZY

2.1)

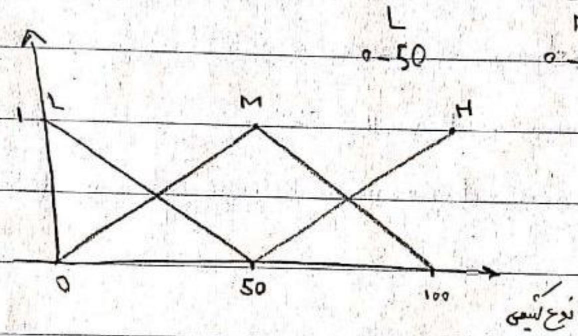
۱) طراحی کنترلگر ماشین لباس شویی

قواعد :

← نوع کثیفی : برای اینکه ورودی‌ها فازی باشند ، من نوع کثیفی را از 0 تا 100 در نظر می‌گیرم و

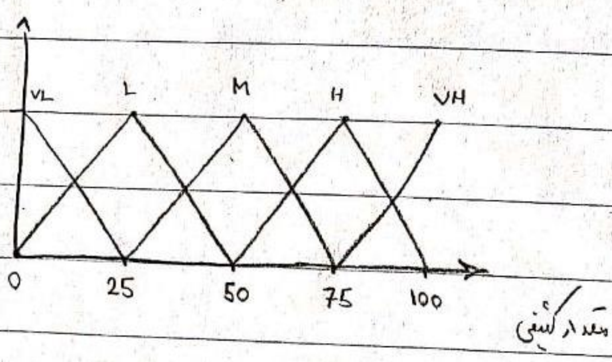
به سه قسمت تقسیم می‌کنم که شبیه‌ترها کنار هم باشند :

عرق	روغن	گل
L	M	H
0_50	0_100	50_100



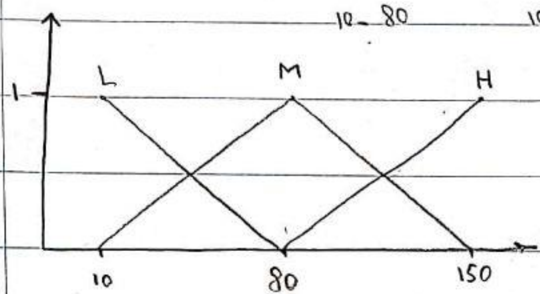
← مقدار کثیفی : پنج فاز برای مقدار کثیفی در نظر می‌گیریم.

خیلی کم	کم	متوسط	زیاد	خیلی زیاد
VL	L	M	H	VH
0_25	0_50	25_75	50_100	75_100



برای خروجی هم زمان بستری را از 10 تا 150 دقیقه و در سه فاز قرار دادیم

زیاد	متوسط	کم
H	M	L
80-150	10-150	10-80



حالا فرمول هر قسمت را پیدا می کنیم:

← نوع کشتی : X

نوع کشتی ←

$\mu_L(X) = (50-X)/150 \;,\; 0 < X < 50$

$\mu_m(X) = \{X/50 \;,\; 0 < X < 50\}$ and $\{(100-X)/50 \;,\; 50 \leq X < 100\}$

$\mu_H(X) = \{(X-50)/50 \;,\; 50 < X < 100\}$

← مقدار کشتی : Y

$\mu_{VL}(Y) = \frac{(25-Y)}{25} \;,\; 0 < Y < 25$

$\mu_L(Y) = \{Y/25 \;,\; 0 \leq Y < 25\}$ and $\{(50-Y)/25 \;,\; 25 < Y < 50\}$

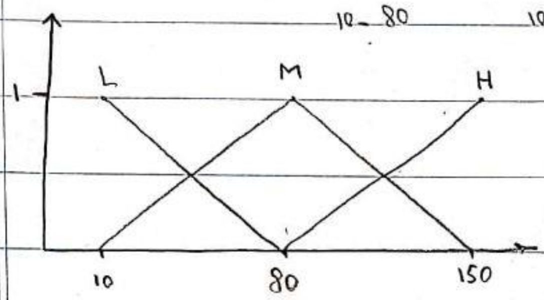
$\mu_M(Y) = \{(Y-25)/25 \;,\; 25 \leq Y < 50\}$ and $\{(75-Y)/25 \;,\; 50 < Y < 75\}$

$\mu_H(Y) = \{(Y-50)/25 \;,\; 50 \leq Y < 75\}$ and $\{(100-Y)/25 \;,\; 75 \leq Y < 100\}$

$\mu_{VH}(Y) = \{(Y-75)/25 \;,\; 75 \leq Y < 100\}$

برای خروجی هم، زمان بستری را از ۱۰ تا 150 دقیقه و در سه فاز قرار دادیم

کم	متوسط	زیاد
L	M	H
10 - 80	10 - 150	80 - 150



حالا فرمول هر قسمت را پیدا می کنیم:

← نوع کشتی : X

← نوع کشتی

$\mu_L(X) = (50-X)/150 \ , \ 0 \leq X < 50$

$\mu_m(X) = \{X/50 \ , \ 0 \leq X < 50\}$ and $\{(100-X)/50 \ , \ 50 \leq X < 100\}$

$\mu_H(X) = \{(X-50)/50 \ , \ 50 \leq X < 100\}$

← مقدار کشتی : Y

$\mu_{VL}(Y) = \frac{(25-Y)}{25} \ , \ 0 \leq Y < 25$

$\mu_L(Y) = \{Y/25 \ , \ 0 \leq Y < 25\}$ and $\{(50-Y)/25 \ , \ 25 \leq Y < 50\}$

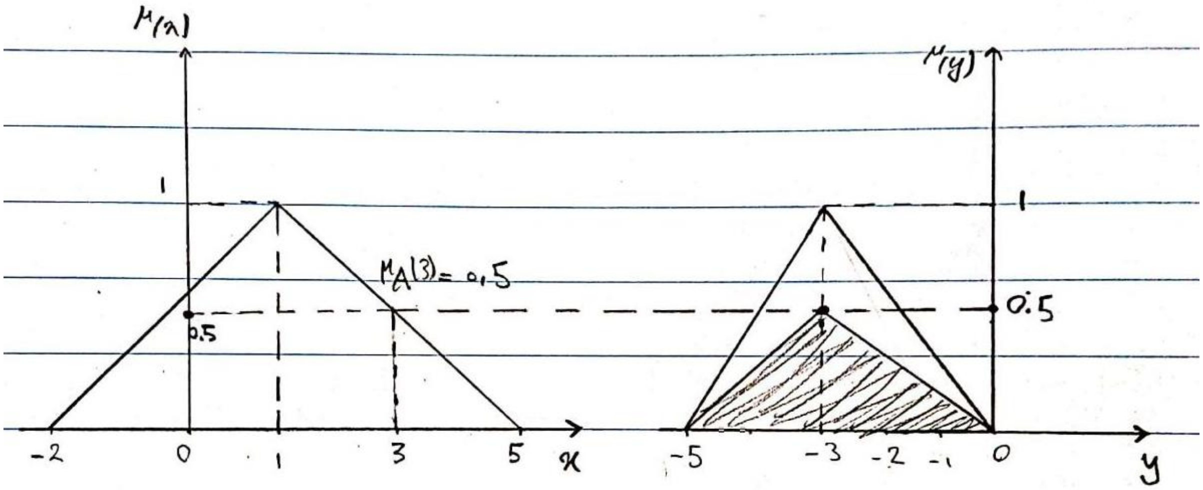
$\mu_M(Y) = \{(Y-25)/25 \ , \ 25 \leq Y < 50\}$ and $\{(75-Y)/25 \ , \ 50 \leq Y < 75\}$

$\mu_H(Y) = \{(Y-50)/25 \ , \ 50 \leq Y < 75\}$ and $\{(100-Y)/25 \ , \ 75 \leq Y < 100\}$

$\mu_{VH}(Y) = \{(Y-75)/25 \ , \ 75 \leq Y < 100\}$

2.2)

(2



$\mu_B' = \mu_{A(3)}\,\mu_B = \frac{1}{2}\mu_B$

$= \{(-4, \frac{1}{4}), (-3, \frac{1}{2}), (-2, \frac{1}{3}), (-1, \frac{1}{6})\}$

2.3) for this part we have to set some fuzzy rules to balance our Pendulum, inverted. My rules are:

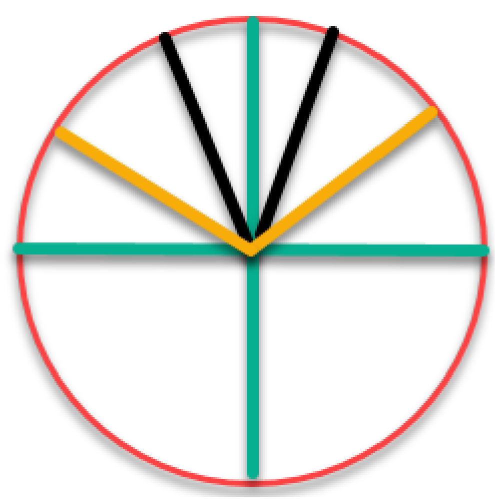
- when Pendulum is down (-90 , 90), if speed of Pendulum was not too fast I push it in same direction . by this work Pendulum will go high and higher each time.

- when Pendulum reached top of circle, it's time to balance it there. So, I added 4 rules to it, first and second one is when Pendulum is going to fall in left or right direction. If it happens, I push some low force in opposite direction to don't let it fall. And this is for only 3 degrees from top. But next two rules are like this two, but the difference is they are stronger and don't let Pendulum fall at all (for our example it is impossible, but in higher speeds of Pendulum it maybe happens)

** these steps make Pendulum balance and in most cases the speed of wain is very low. but if we want to stop it completely, we have to do this:

In balance situation we push the wain in same direction of its speed! but after that when Pendulum lean to opposite direction, gravity takes Pendulum down and we can push the wain in opposite direction to stop it. Because of lack time I couldn't implement this , but it will work.

You can see my results in the next page



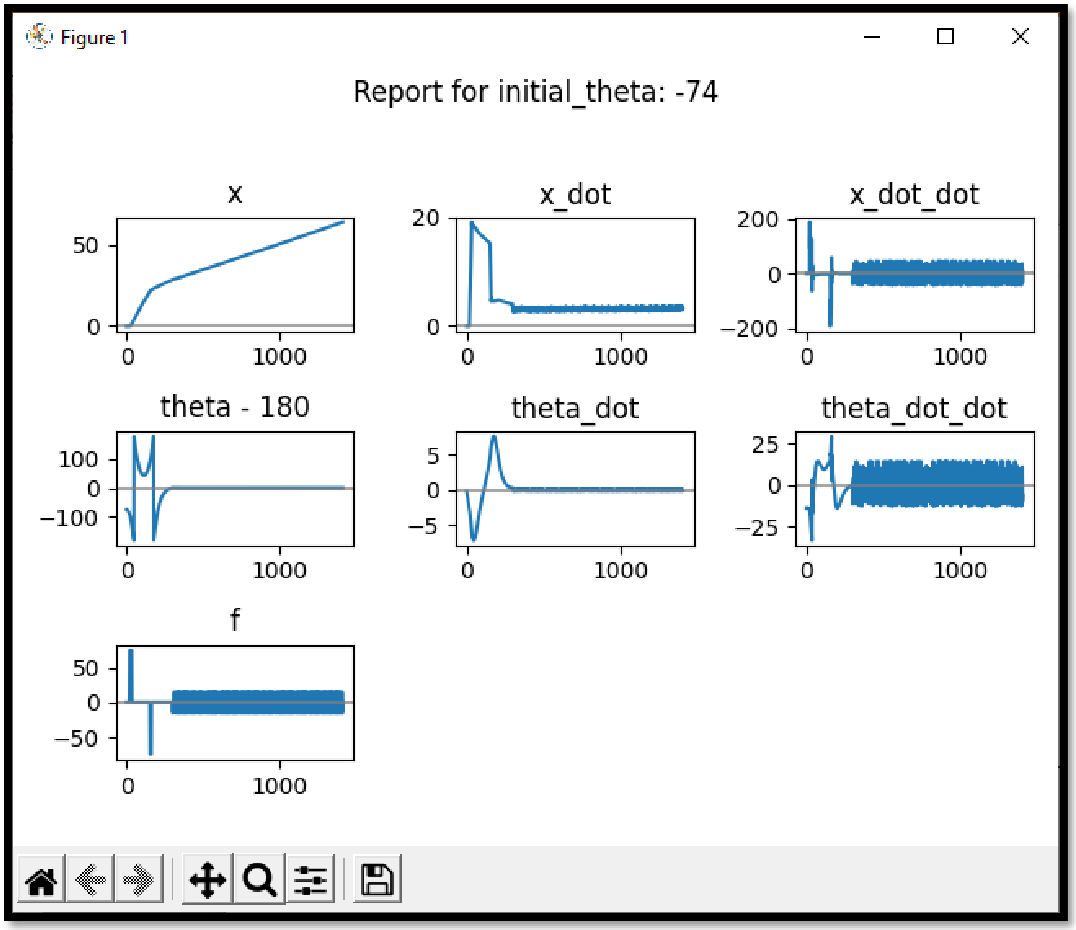


Figure 1
Report for initial_theta: -74
x
50
0
0
1000
x_dot
20
0
0
1000
x_dot_dot
200
0
−200
0
1000
theta - 180
100
0
−100
0
1000
theta_dot
5
0
−5
0
1000
theta_dot_dot
25
0
−25
0
1000
f
50
0
−50
0
1000