

# Short report on lab assignment 3

## Hopfield Networks

Chiachen Ho, Angelos Stais and Ruxue Zeng

October 5, 2021

### **1 Main objectives and scope of the assignment**

Our major goal in the assignment is to study Hopfield networks and associative memory. Our objectives are:

- Understand the principles underlying the operation and functionality of auto-associative networks
- Train the Hopfield network
- Study the attractor dynamics of Hopfield networks the concept of the energy function
- Understand how auto-associative networks can do pattern completion and noise reduction
- Investigate the question of storage capacity and explain features that help increase it in associative memories

### **2 Methods**

We have used Python in this lab with libraries below:

- Numpy, Math, Random, Matplotlib, Spicy

### 3.1 Convergence and attractors

- **Stored patterns**

```
Old pattern was [-1. -1. 1. -1. 1. -1. -1. 1.] updated pattern is [-1. -1. 1. -1. 1. -1. -1. 1.]
They're the same
Old pattern was [-1. -1. -1. -1. -1. 1. -1. -1.] updated pattern is [-1. -1. -1. -1. -1. 1. -1. -1.]
They're the same
Old pattern was [-1. 1. 1. -1. -1. 1. -1. 1.] updated pattern is [-1. 1. 1. -1. -1. 1. -1. 1.]
They're the same
```

The network is able to store all three patterns.

- **Distorted patterns**

```
The new patterns is [-1. -1. 1. -1. 1. -1. -1. 1.] and the correct pattern was [-1. -1. 1. -1. 1. -1. -1. 1.]
They're the same
The new patterns is [-1. 1. -1. -1. -1. 1. -1. -1.] and the correct pattern was [-1. -1. -1. -1. -1. 1. -1. -1.]
The new patterns is [-1. 1. 1. -1. -1. 1. -1. 1.] and the correct pattern was [-1. 1. 1. -1. -1. 1. -1. 1.]
They're the same
```

$x1d$  and  $x3d$  converge towards stored patterns. After 3 iterations,  $x2d$  does not converge toward  $x2$ .

- **Number of attractors in the network**

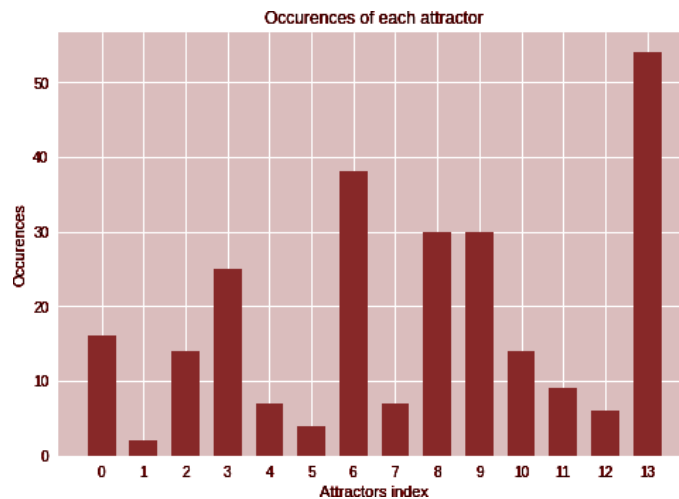


Figure 3.1: Occurrences for each attractor

We found 14 attractors in this network. Moreover, if a pattern is an attractor, the opposite pattern is an attractor too.

- |                              |                              |
|------------------------------|------------------------------|
| (1) [-1 -1 -1 -1 -1 1 -1 -1] | (2) [-1 -1 -1 -1 1 -1 -1 -1] |
| (3) [-1 -1 1 -1 -1 1 -1 1]   | (4) [-1 -1 1 -1 1 -1 -1 1]   |
| (5) [-1 -1 1 -1 1 1 -1 1]    | (6) [-1 1 -1 -1 -1 1 -1 -1]  |
| (7) [-1 1 1 -1 -1 1 1 -1]    | (8) [-1 1 1 -1 1 1 -1 1]     |
| (9) [ 1 -1 -1 1 1 -1 1 -1]   | (10) [ 1 1 -1 1 -1 1 1 -1]   |
| (11) [ 1 1 -1 1 1 -1 1 -1]   | (12) [ 1 1 -1 1 1 1 1 -1]    |
| (13) [ 1 1 1 1 -1 1 1 1]     | (14) [ 1 1 1 1 1 -1 1 1]     |

- **More dissimilar patterns**

```
The new patterns is [ 1. 1. 1. 1. 1. -1. 1. 1.] and the correct pattern was [-1. -1. 1. -1. 1. -1. -1. 1.] The convergence takes 3 iterations
They are different
The new patterns is [ 1. 1. 1. 1. 1. -1. 1. 1.] and the correct pattern was [-1. -1. -1. -1. -1. 1. -1. -1.] The convergence takes 3 iterations
They are different
The new patterns is [ 1. -1. -1. 1. 1. -1. 1. -1.] and the correct pattern was [-1. 1. 1. -1. -1. 1. -1. 1.] The convergence takes 2 iterations
They are different
```

It seems that if the input patterns are very dissimilar, the memory cannot recall the stored patterns.

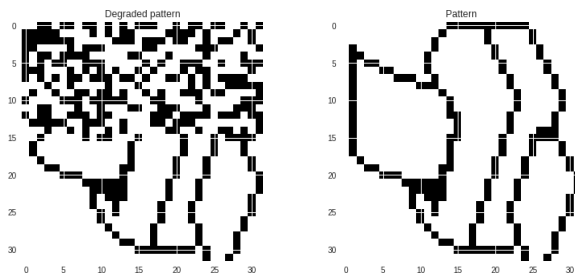
## 3.2 Sequential Update

- Check that the three patterns are stable

Old pattern was `[-1 -1 -1 ... -1 -1 -1]` updated pattern is `[-1. -1. -1. ... -1. -1. -1.]` and the convergence takes 1 iterations  
They're the same  
Old pattern was `[-1 -1 -1 ... -1 -1 -1]` updated pattern is `[-1. -1. -1. ... -1. -1. -1.]` and the convergence takes 1 iterations  
They're the same  
Old pattern was `[1 1 1 ... 1 1 1]` updated pattern is `[1. 1. 1. ... 1. 1. 1.]` and the convergence takes 1 iterations  
They're the same

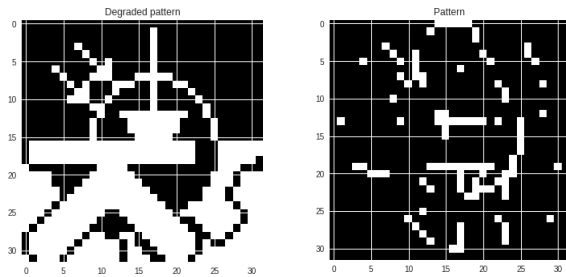
After the update process, the three patterns don't change. They are stable.

- Can the network complete a degraded pattern?



After the figure 3.2, we observe the network transform the p10, the distorted version of P1, to P1. So the network can complete a degraded pattern.

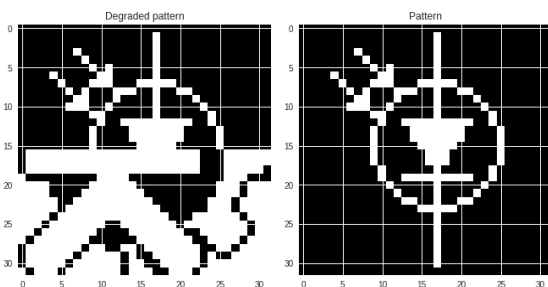
Figure 3.2: Degraded pattern p10 (L) Restored pattern p1(R)



After the figure 3.3, we observe the pattern restored from p11 is different than p2 and p3. So, the network cannot complete a pattern that is a mixture of two learned patterns in batch mode.

Figure 3.3: Degraded pattern p11 (L) Restored pattern (R)

- Sequential update with random units



With a sequential update, the pattern converges most of the time to pattern p3 in a few iterations. The number of iterations needed for convergence is not constant.

Figure 3.4: Degraded pattern p11 (L) Restored pattern p3 (R)

### 3.3 Energy

In order to study the convergence of our network, since we have a symmetric connection matrix, we define an energy function, a finite-valued function of the state that always decreases as the state changes. Since it has to have a minimum at least somewhere the dynamics must end up in an attractor. We observe that the energy at the different attractors is much lower than at the points of their distorted patterns.

	Energy at the Attractors	Energy at the point of the Distorted Patterns
P1	-1439.39	-415.98 (P10)
P2 / P3	-1365.64 -1462.25	-173.5 (P11)

Figure 3.3 shows the energy evolution of P10 with three different weight matrices using the sequential update rule. From iteration to iteration, when we use the weight matrix initialized depending on the patterns, the energy keeps decreasing until it attains its minimum, *i.e.* in an attractor. But if the weight matrix is initialized with normally distributed random numbers, then the energy doesn't converge, it keeps cycling between different states forever. If we make this random matrix symmetric, *i.e.* by setting  $w = 0.5 * (w + w')$ , then the energy keeps decreasing and ends up in an attractor with much more iterations than the first model, and this attractor is not the one we expected. By setting the matrix symmetric, we obtain a symmetric connection matrix, but the numbers composed of this matrix are independent of the patterns, so they are in the memory cue situated in the basin of a different attractor, which drives it to these different attractors.

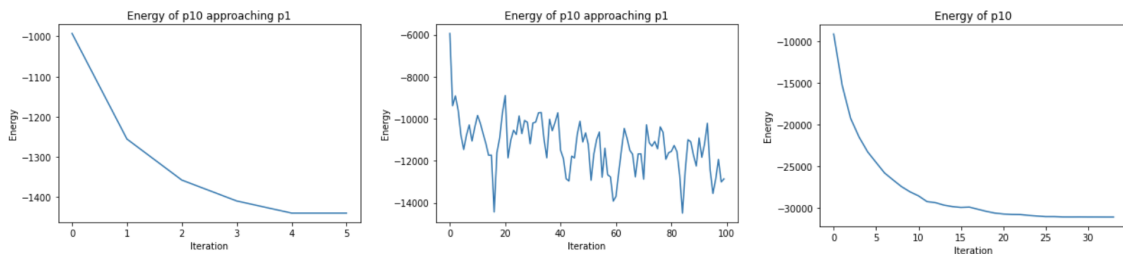


Figure 1 - Energy evolution of P10 approaching P1 with three different weight matrix :  
Created depending on patterns (L) Normally distributed random numbers (C) Matrix symmetric of normally distributed random number (R)

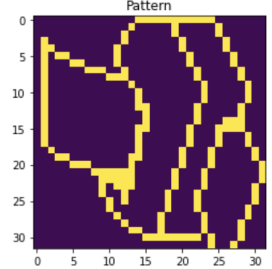
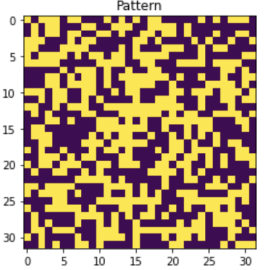
Weight Matrix	$W = XX'$	$WN =$ Normally distributed random	$WS = 0.5 * (WN + WN')$
Attractors		No exist	
Number of Iterations	6	Inf	34

Table 3.3 - Attractors of P10 with three different weight matrix

### 3.4 Distortion Resistance

Now, we want to study the resistance of the Hopfield network to noise and distortion. By randomly flipping a selected number of units, we generate noisy test stimuli. In particular, we train the network with p1, p2, and p3.

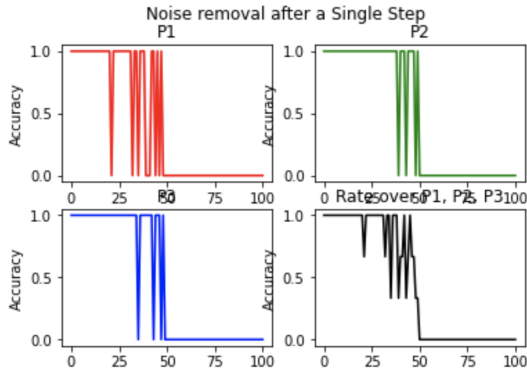


Figure 3.4.1 - Noise removal after a Single-Step

We observe that the network doesn't always converge to the right attractor. The extra iterations do help to improve the convergence, we observe that there are fewer oscillations. Furthermore, we notice another attractor which is the opposite of the right attractor. It has the same energy as the right attractor, so with the Little model, we converge also to this opposite attractor.

According to Figure 3.4, using the Little model with one single-step, we observe that from 50% noise, the accuracy to recover the original clean patterns becomes zero, so only a maximum of 50% of noise can be removed. With regard to noise tolerance, it varies depending on attractors. We observe that p2 has the shortest oscillation bearing, so it has the biggest noise tolerance. On the other hand, p1 has the longest oscillation bearing, so it has the smallest noise tolerance.

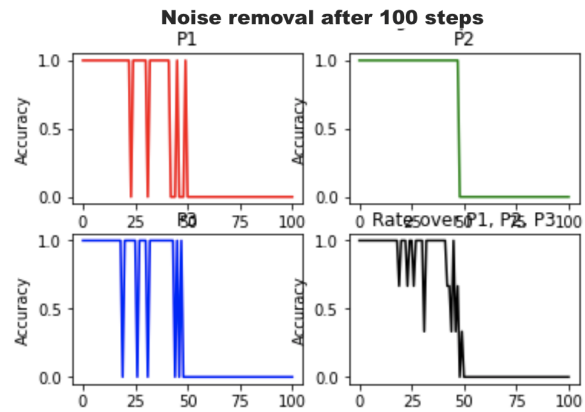


Figure 3.4.2 - Noise removal after 100 Steps

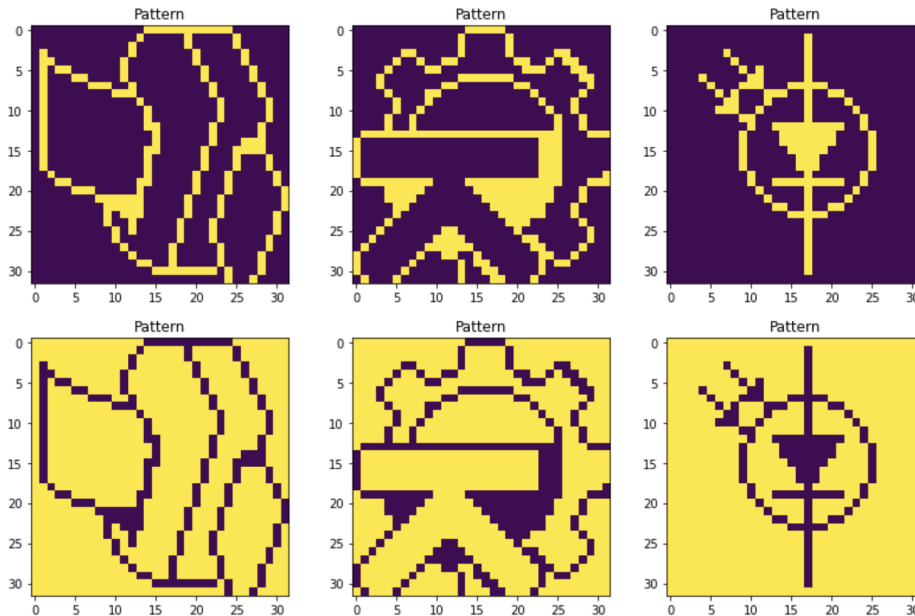
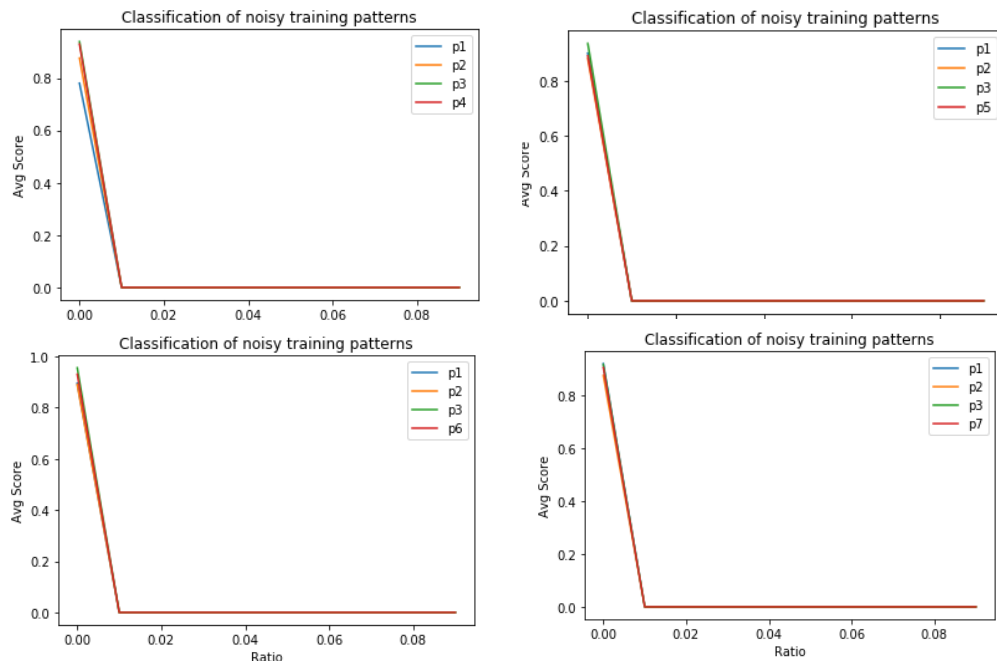


Figure 3.4.3 - Attractors and opposite attractors

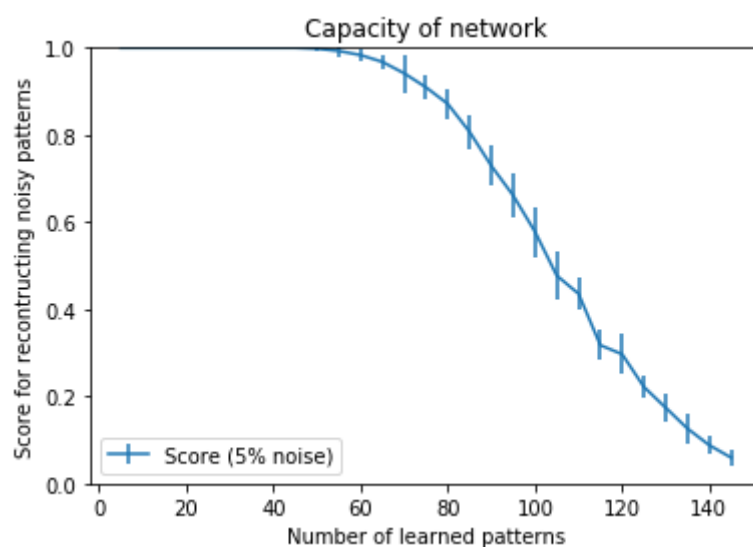
### 3.5 Capacity



The network doesn't seem to be able to recall these 4 patterns well when the distortion is more than 1%. The value on the y-axis is the average score obtained from 10 runs. If the network is able to store  $p$ , then we set the score to 1. We see an abrupt drop in performance when the noise rate is at 1%. We tested out different combinations of 4 patterns and found that [p2,p5,p8,p9] is the best. The explanation for this might be that among the combinations shown in the above plots, there are some similarities shared between pictures.

- **Repeat this with random patterns**

We set the number of patterns from 0 to  $0.138 \times 1024$  with an interval of 5 and plot out the score. The score here is the average score obtained from 10 runs for each number of patterns. The noise we added is 5%.

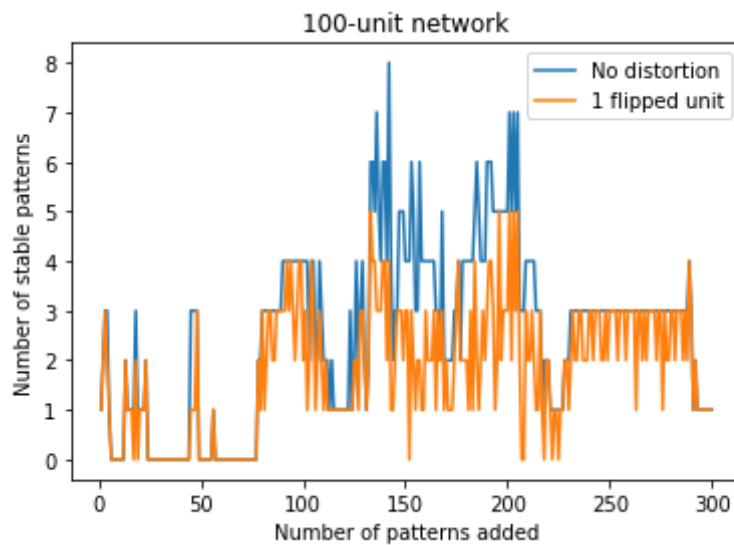


As shown in the plot, the network performs better on random patterns than on pictures. And after the number of patterns exceeds 60, the performance starts to drop.

The reason why such difference exists is probably due to the similarity between pictures whereas for random patterns which are generated via uniform distribution, such similarity doesn't exist.

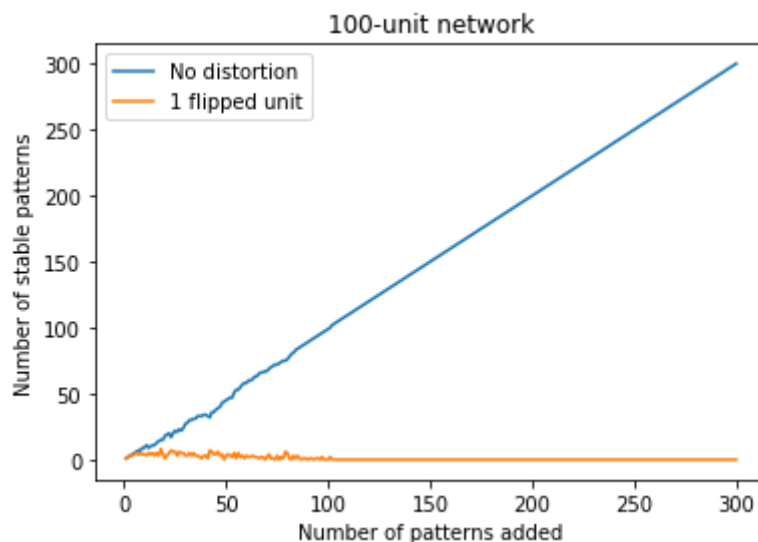
- **A hundred-unit network**

We firstly plot out the results from a 100-unit network without self-connections



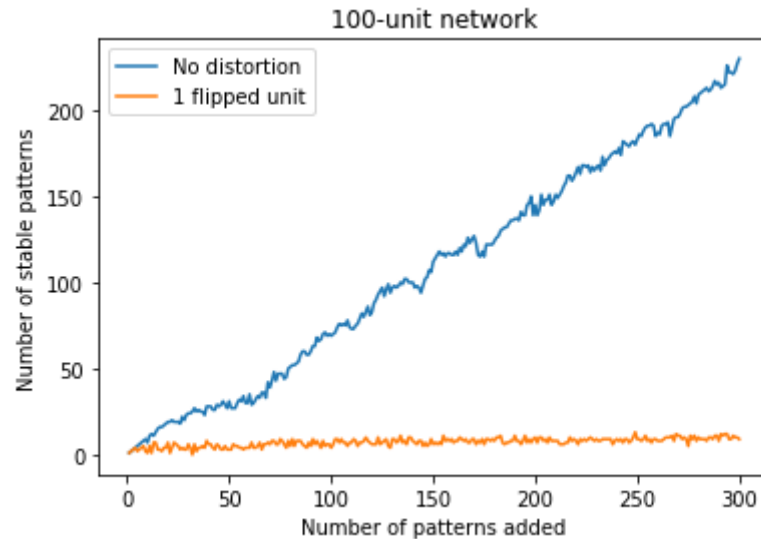
As shown in the plot, no distortion refers to the original patterns. 1-flipped unit refers to the noisy patterns. We can see that even without noise, the network still has a hard time reconstructing the original patterns. The maximum number of stable patterns which can be reconstructed is 8.

Now we add self-connections and plot the results.



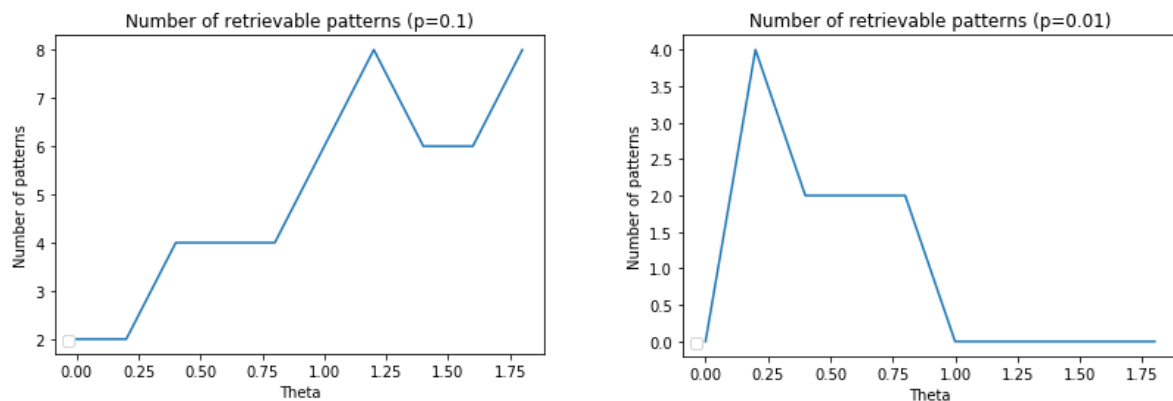
As expected, when introducing self-connections, the difference between pure and noisy patterns appears. The maximum number of retrievable patterns for pure patterns is 300. However, this network has a poorer capability of noise removal.

When we add bias to the patterns, the performance is slightly worse than non-bias patterns.



This phenomenon is similar to picture patterns in that pictures usually are dominated by 1s or 0s.

### 3.6 Sparse Pattern



We set  $\theta$  from 0 to 1.8 with 0.2 intervals and set the number of patterns from 2 to 12. For each number of patterns, we repeat 5 times. The value on the y-axis is the highest number of patterns the network can store among 2 to 12 patterns with 5 runs for each.

From the above two plots, we can see that bias has a great influence on the capacity of the network. With 10% activity, the network can store up to 8 patterns when  $\theta = 1.25$ . Overall, the sparser pattern ( $p=0.01$ ) gives worse results and it needs a smaller value of  $\theta$  to achieve better performance.



## **5. Final remarks**

In this lab, we get more familiar with how the Hopfield network works and its usage. For example, pattern completion and noise reduction. In addition, we were able to apply our knowledge of attractor dynamics of Hopfield networks, energy function, and the capacity of the network to analyze the results with different experiment settings.