Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 3

Hopfield networks

Shuyuan Zhang, Xinxing Guo and Wenqin Chen

September 26 2019

1 Main objectives and scope of the assignment

- To learn the training and iteration process of the Hopfield Network
- To understand and demonstrate the role of auto-associative networks in pattern recognition and noise reduction tasks.
- To investigate features that can increase storage capacity of the Hopfield Network.

2 Methods

We worked with Python as our programming language in this project, and the numpy/matplotlib libraries in Python were adopted.

3 Results and discussion

3.1 Convergence and attractors

First, all three patterns in part 2 are stored in the network.
Table 1 shows that all the distorted patterns converge towards stored patterns after applying the update rule repeatelly.

	Case 1	Case 2	Case 3
Distorted	[1. -1. 1. -1. 1. -1. -1. 1.]	[1. 1. -1. -1. -1. 1. -1. -1.]	[1. 1. 1. -1. 1. 1. -1. 1.]
Test result	[-1. -1. 1. -1. 1. -1. -1. 1.]	[-1. -1. -1. -1. -1. 1. -1. -1.]	[-1. 1. 1. -1. -1. 1. -1. 1.]
Original	[-1. -1. 1. -1. 1. -1. -1. 1.]	[-1. -1. -1. -1. -1. 1. -1. -1.]	[-1. 1. 1. -1. -1. 1. -1. 1.]

Tabell 1: Comparison between the original patterns and distorted patterns after updating

We obtained 10 attractors in this case. The attractors are shown in figure 1.

Table 2 shows the result after making the start pattern even more dissimilar to the stored ones. And it seems that the network cannot recall the stored patterns and went to some spurious attractors.

```
[-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.]
[-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.]
```

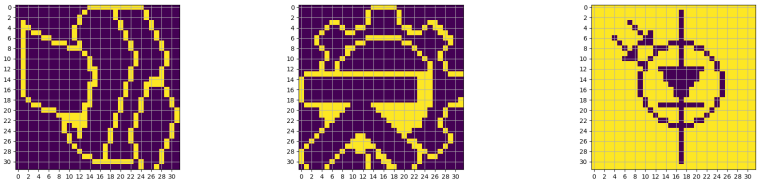
Figur 1: Attractors

	Case 1	Case 2	Case 3
Distorted	[-1. 1. -1. 1. -1. 1. -1. 1.]	[-1. -1. 1. -1. -1. 1. -1. 1.]	[-1. -1. -1. 1. -1. -1. -1. 1.]
Test result	[-1. -1. 1. -1. -1. 1. -1. 1.]	[-1. -1. 1. -1. -1. 1. -1. 1.]	[-1. -1. 1. -1. -1. 1. -1. 1.]
Original	[-1. -1. 1. -1. 1. -1. -1. 1.]	[-1. -1. -1. -1. -1. 1. -1. -1.]	[-1. 1. 1. -1. -1. 1. -1. 1.]

Tabell 2: Comparison between the original patterns and dissimilar patterns after updating

3.2 Sequential update

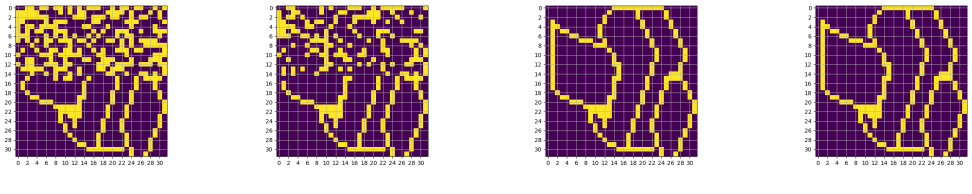
Figure 2 shows the first three patterns and they are stable.



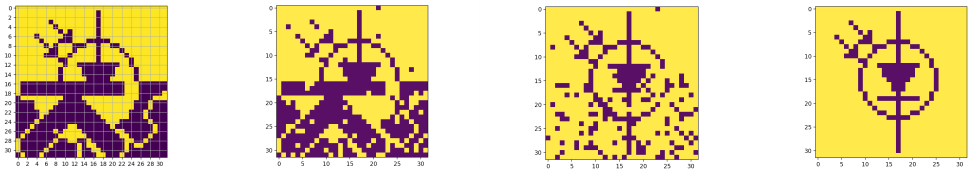
(a) Pattern 1 (b) Pattern 2 (c) Pattern 3

Figur 2: The first three patterns

If we select units randomly, the images are demonstrated in figure 3 with iteration. As figure 3(a) to figure 3(d) demonstrate, the pattern 10 could converge to pattern 1 in a few iterations, but the number of iterations is not constant. While figure 3(e) to figure 3(h) demonstrate that the network can converge pattern 11 that is a mixture of two learnt patterns only to pattern 3 across many trials.



(a) Pattern 10 (b) Pattern 10 (c) Pattern 10 (d) Pattern 10



(e) Pattern 11 (f) Pattern 11 (g) Pattern 11 (h) Pattern 11

Figur 3: Pattern 10 and Pattern 11 with randomly selected units

3.3 Energy

The energy at the different attractors are shown as table 3.

Tabell 3: Energy at different attractors

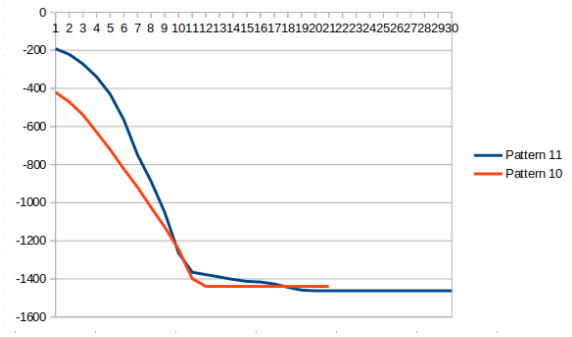
Layer	Energy Value
Pattern 1	-1439.390625
Pattern 2	-1365.640625
Pattern 3	-1462.25

The energy at the points of the distorted patterns are demonstrated in table 4.

Tabell 4: Energy of the distorted patterns

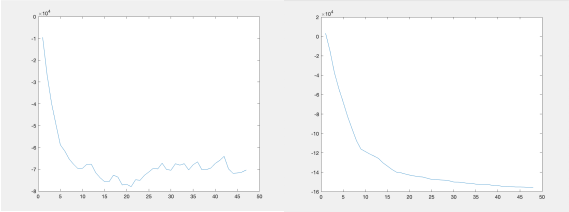
Layer	Energy Value
Pattern 10	-415.98046875
Pattern 11	-173.5

Figure 4 show how the energy changes from iteration to iteration when we use the sequential update rule to approach an attractor. It is clear that pattern 10 starts to converge at the 12-hundredth iteration, and pattern 11 begins to converge at 22-hundredth iteration.



Figur 4: Energy changes from iteration to iteration

When generating a matrix by setting the weights to normally distributed random numbers, and iterating an arbitrary starting state, we found that the energy oscillates instead of decreasing monotonically since the weight matrix is not symmetric.

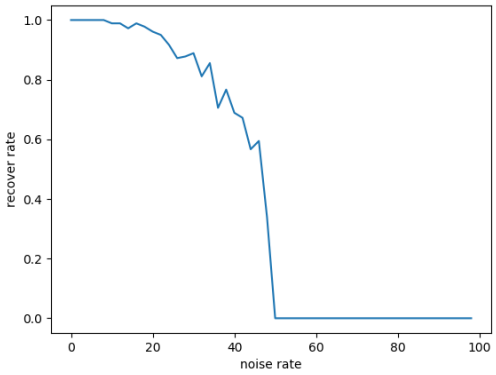


(a) Random (b) Symmetric

Figur 5: Random vs. Symmetric W matrices

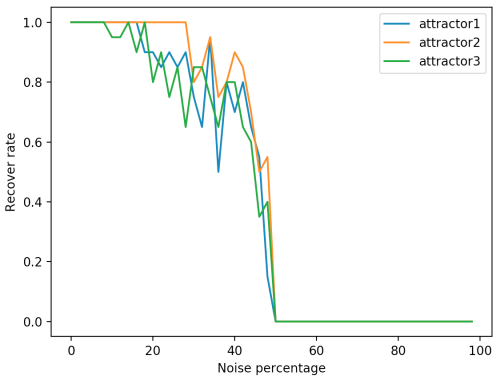
When making the weight matrix symmetric, we found that the energy would decrease monotonically. So the symmetricity of the weight matrix is a necessary condition for the convergence of energy function.

3.4 Distortion resistance



Figur 6: Noise resistance

We ran the test 30 times to calculate the recover rate. Figure 6 shows that the network is able to recover pictures with about 45% noise. When the noise increases, we approach the same pattern but with inverted colours. Thus, the network can no longer restore the original picture, but converge to a spurious attractor.



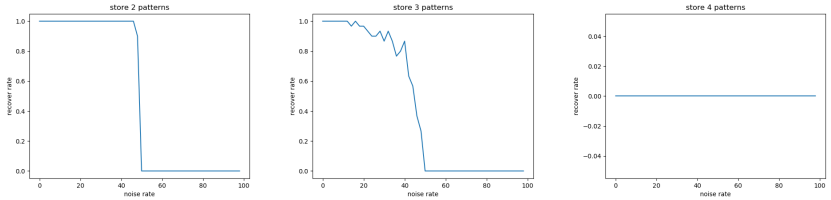
Figur 7: Noise tolerance of 3 attractors

It seems that attractor 2 has the best noise tolerance.

The network would converge to the right attractor if the noise rate is not large. The extra iterations would not help since the process has already converged. There are indeed some other spurious attractors.

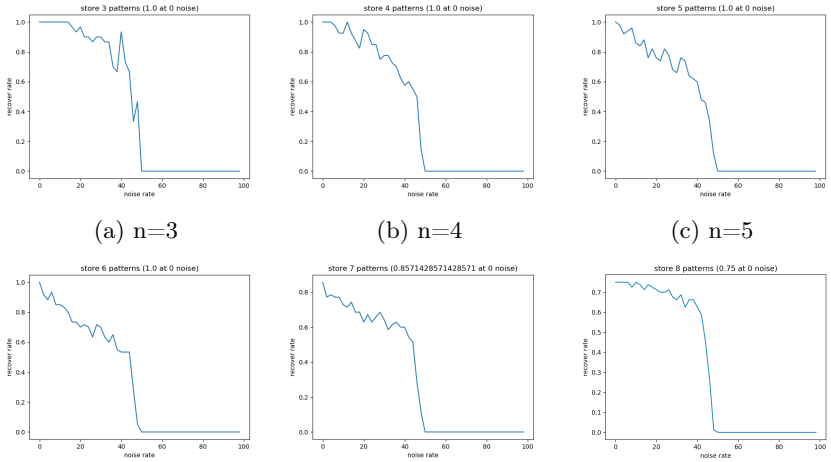
3.5 Capacity

According to figure 8, we found 3 patterns can be safely stored. But if we add the fourth attractor(or more attractors), the network lose its capability to restore pictures. The drop in performance is abrupt.
According to figure 9, If we store random generated patterns instead of picture patterns, the store capacity become better, now it can safely store 6-7 random patterns with 100% recovery rate of inputs(little noise).



(a) n=2 (b) n=3 (c) n=4

Figur 8: Recover rate of different number of stored picture patterns



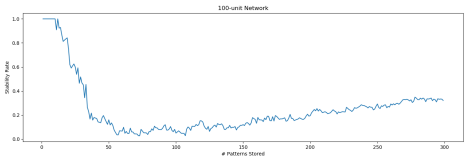
(a) n=3 (b) n=4 (c) n=5

(d) n=6 (e) n=7 (f) n=8

Figur 9: Recover rate of different number of stored random patterns

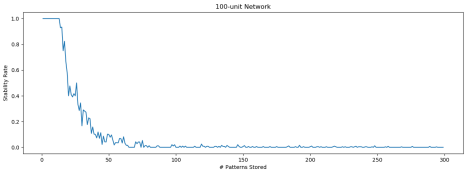
The reason why the network has a better performance on random patterns is that the stored patterns are less related to each other. The picture patterns are somehow more related to each other(patterns are placed in the center of a picture) and this may generate many spurious attractors.

Next, we used the 100-unit network.



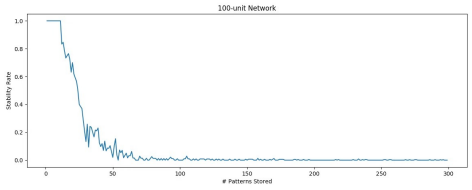
Figur 10: Recovery rate of different number of patterns

It can perfectly store 10-20 patterns before stability rate started to drop. However, with the number of stored patterns keep increasing, the recover rate of the orignal stored version went up again.



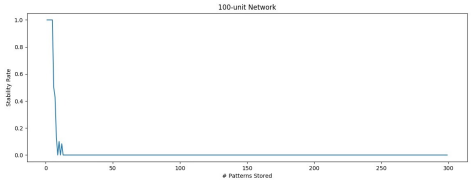
Figur 11: Recovery rate of different number patterns when input a noisy version of stored patterns

When we consider the network's capability of restoring noisy version of the stored patterns, the performance before 10 patterns was almost same with the former one, but it didn't improve when the number of patterns increasing. So it loses its restoring capability when the number of stored patterns increases. This can be explained by the increase of number of stored patterns also introduces even more spurious attractors and noisy patterns may converge to them.



Figur 12: Stability rate of different number of patterns

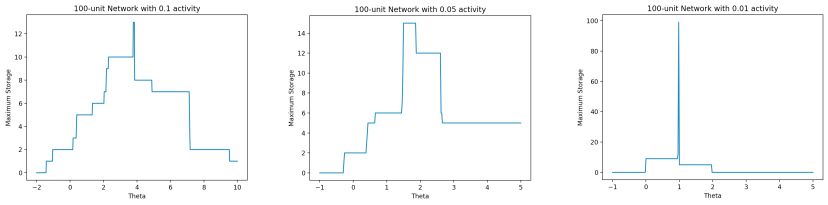
The maximum number of retrievable patterns for this network is 11. Also, after removed self-connections, the recovery rate won't rise if we add more stored patterns.



Figur 13: Stability rate of different number of patterns biased

The capacity became smaller if we biased the patterns. Because biased patterns are more related with each other.

3.6 Sparse patterns



(a) $\rho = 0.1$ (b) $\rho = 0.05$ (c) $\rho = 0.01$

Figur 14: Sparse data capacity

When $\rho = 0.1$, the network can store 13 patterns when $\theta = 4$. When the data became sparser, the best θ also decreases.

4 Final remarks

In lab3, we tried to experiment on auto-associative memory using Hopfield network, we mainly focus on its capabilities(noise resistance, restore capability), capacity(how many states can be stored) and limitations.