Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 3

Hopfield Networks

Côme Lassarat, Victor Sanchez, Marie-Ange Stefanos

February 23, 2022

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to manipulate principles underlying the operation and functionality of auto-associative networks by training Hopfiel networks
- to familiarise with attractors dynamics of Hopfield networks the concept of energy function
- to analyse the influence of auto-associative networks on pattern completion and noise reduction
- to investigate the question of storage capacity and explain features that help increase it in associative memories.

2 Methods

In order to reach the intended goals, the different methods and algorithms were coded in Python, a wide-spread programming language. Several libraries were also imported and used:

- **Numpy** to handle multi-dimensional arrays
- **Matplotlib** to construct and plot graphs
- **Random** to handle with randomisation of introduce special distribution of data

3 Results and discussion - Hopfield networks

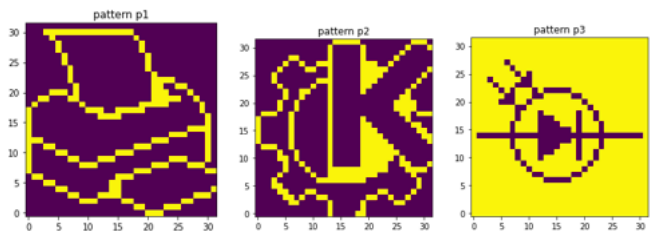
3.1 Convergence and attractors

The goal here is to analyse the convergence and the attractors over three simple memory patterns.
We first implement the update rule and we observe that the fixed point is reached after only two iterations, which is quite efficient.

However, this network may encompass other attractors, that we have found by automating the searching: there are 14 attractors.

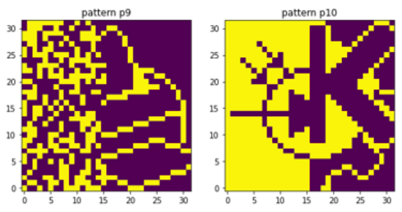
3.2 Sequential Update

We know load *pict.dat* file and switch from a small 8- to a 1024-neuron network that can be plotted as 32x32 images. We learn the first three, that can be seen on figure 1 and check that they are stable.



Figur 1: Patterns contained in the neural network memory

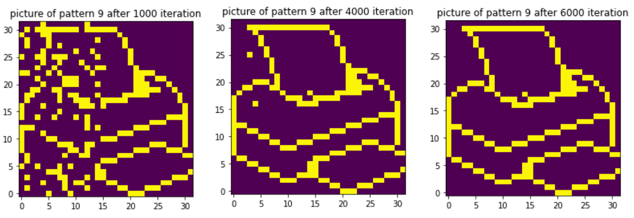
We also want to know if the network can complete a degraded pattern. To do so, we give to the network the two following patterns p9 and p10 (figure 2) that are respectively a degraded version of p1 and a mixture of p2 and p3.



Figur 2: Degraded patterns p9 and p10 given as input to the network

The network can complete both patterns. We can see that after 6000 iterations, the recall converges to fixed points. Whereas p9 unsurprisingly converges to p1, p10 is transformed as p3, not p2. However, convergence is quite slow. A solution is to use random selection of weight, since it makes learning more efficient.

For instance, the evolution of the result of p9 image recall can be seen on the figure 3.



Figur 3: Degraded patterns p9 and p10 given as input to the network

3.3 Energy

For networks with a symmetric connection matrix, it is possible to define a simple energy function as given in the instructions. It has the property of having a minimum at least somewhere so the dynamics must end up in an attractor.

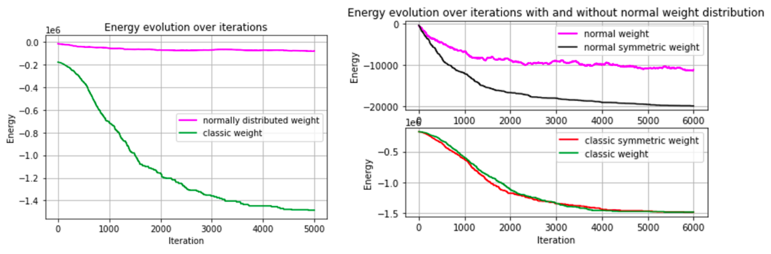
At the different attractors the energy has low values and can reach minima when the recall converges.

Attractors	Energy
p1	-1439
p2	-1365
p3	-1462
p10	-416
p11	-173

Tabell 1: Energy of the different patterns (attractors p1, p2, p3) and distorted patterns (p10, p11)

Then we want to compare the current weight matrix to a weight matrix when setting the weights to normally distributed random numbers and to the weight matrix made symmetrical. The obtained results can be seen on the figure 4.

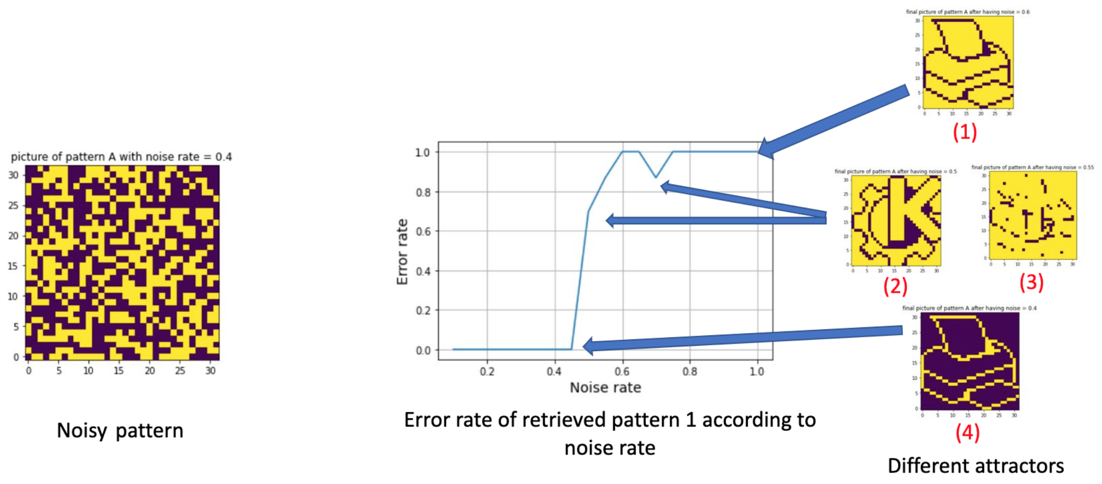
The left-hand pannel of figure 4 shows that the values of the energy for normally distributed weights are way higher than for the classic weight matrix, which is an inconvenient since attractors are characterized by minima. However, the right-hand pannel shows that if the weight matrix is made symmetrical, the results are different. Only making it symmetrical does not change much the energy, but making symmetrical the normalized distributed matrix not only lowers the energy but also make the converge faster.



Figur 4: Evolution of the energy through iterations for classical weight matrix W, randomly computed and symmetrized

3.4 Distortion Resistance

In this section, the patterns are noised by flipping a pourcentage of its pixels values. Given a *flipping*, the convergence of the networks towards the right attractor (the original pattern without noise) is checked. Thus, the error rate (percentage of pixels with the wrong value for the output image, compared to the original noise-less image) is plotted according to the noise rate (Figure 5)



Figur 5: Distortion resistance of the network

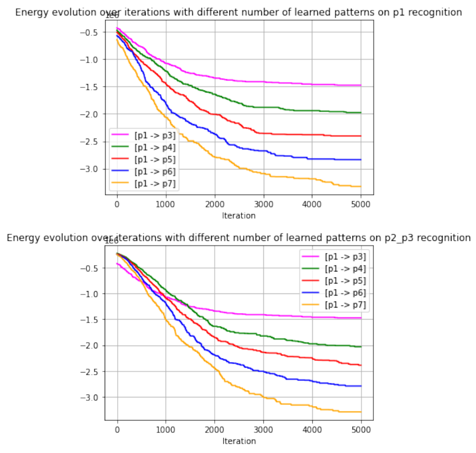
- Retrieving easily the right pattern until 40% of noise (Figure 5 image (4), error = 0)
- For a lot of noise ($\geq$ 60%), the retrieved pattern is most of the time the initial one with its pixels flipped Figure 5 image (1), error = 1)
- Existence of other attractors: another pattern (Figure 5 image (2)) or a mixture of different patterns (Figure 5 image (3))

3.5 Capacity

We want here to analyze the capacity of memorisation of the Hopfield Network.

3.5.1 Memorisation of image pattern

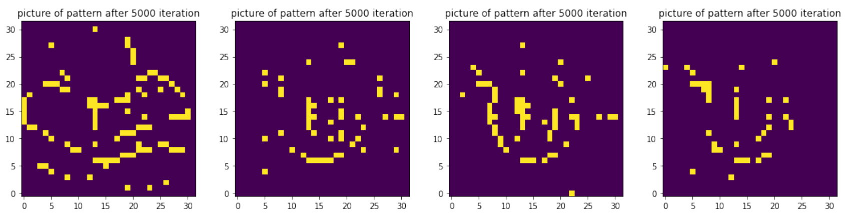
We start here by storing more than the three first patterns in our memory and we try as before to recall what was learned. We plot the energy of the recall and we observe that the more we have pattern in our memory, the more the energy is low, which is quite nice in terms of efficiency.



Figur 6: Energy on the recall process of pattern p9 and p10

We also notice the convergence to fixed point is slower when we have more and more patterns.

But we now want to see if the recall is efficient by looking at the image obtained with several size of memory. Here we have the proof that even if the energy is

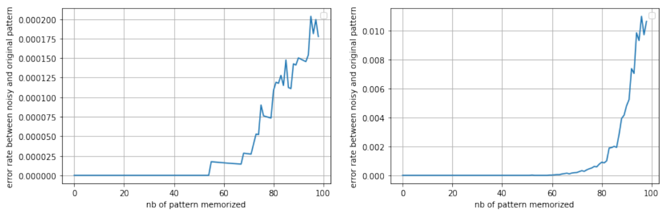


Figur 7: Recalled image with different memory size

lower when we have more pattern memorize, it does not mean that the recall is well made.
This effect is mainly due to the fact that the data are overlapping.

3.5.2 Memorisation of random pattern with 1024 nodes

We want now to see the limits of the memory with random images. To do so we compute patterns of same size as before : 1024.
To map the memorization we decided to store successively patterns in a memory buffer and after each storage we observe the error rate between the original pattern and updated ones.
We also do the same process but with noised data (with the same process as before).

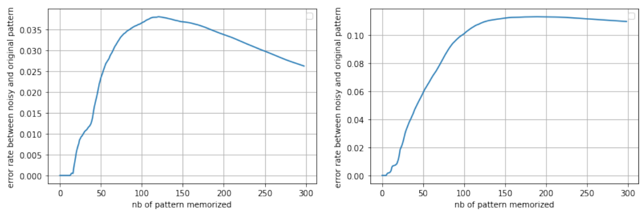


Figur 8: Error rate with memory size growing recalling original (left) and 10% flipped (right) pattern

We can observe that with 1024 units no matter the number of patterns, the network can recall quite efficiently the pattern.
When we flip the data by 30% we observe that we have only an error of 1% after storing 100 patterns.

3.5.3 Memorisation of random pattern with 100 nodes

Now we want to observe the same phenomenon but with patterns that has a different size. We introduce also random patterns but with only 100 units and we repeat the storage of 300 patterns.
We can thus observe the following results. For the recall of original patterns,



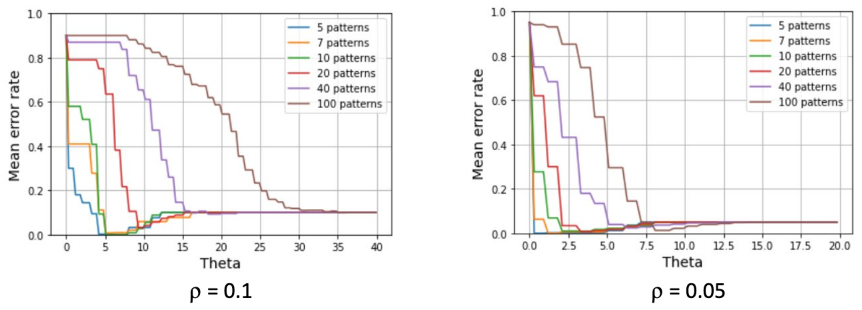
Figur 9: Error rate with memory size growing recalling original (left) and 10% flipped (right) pattern

there is a 100% recall for a certain number of nodes but after a certain threshold, the network get confused. Nevertheless, the error is still under 3% which is acceptable.
For flipped pattern, we can also see at first that the error rate is null. But the

more patterns we learn, the high is the error rate. By flipping 10% of the pattern we obtain a,n error rate of approximately 11%. Which is still acceptable efficient after 300 patterns memorized.

3.6 Sparse Patterns

Thus, the unbalanced characteristic of the data causes a capacity drop. In order to improve this issue, a new updated rule is tested taking into account the activity ρ of a pattern (number of active neurons) and introducing the bias θ (Figure 10).



Figur 10: Influence of the value of the bias θ over the error rate of the retrieved images (and thus the capacity), when varying the number of stored patterns

- Given an activity ρ, there is an upper bounds of number of patterns where the network can retrieved all the patterns (error = 0 for some values of theta)
- When ρ decreases, we are able to store more patterns (error is lower and bigger sets of patterns reach a null error for some values of θ)
- When ρ decreases, the best corresponding θ (with the lowest error rate) decreases

4 Final remarks

This assignment enables us to study and build on our own a Hopfield Network. We have studied the attractor dynamics of Hopfield networks and its links with the energy function. We have also showed that distorted patterns or mixture of patterns can be completed thanks to autoassociative networks, that can do pattern completion and noise reduction. We have also tackled the issue of storage capacity, that can be increased by different means such as using random patterns.