Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 3

Hoppfield networks

Ziyuan Wang, Emil Kerakos, Ming Jiang

February 18, 2021

1 Main objectives and scope of the assignment

In this lab our aim was to get familiar with Hopfield Networks. More specifically, our major goals in the assignment were to:

- understand the principles underlying the operation and functionality of autoassociative networks,
- get experience training Hopfield networks,
- understand the attractor dynamics and the concept of an energy function,
- see how autoassociative networks can do pattern completion and noise reduction,
- investigate the question of network storage capacity and explore features that can help increase it.

In this exploration we have limited ourselves to the discrete Hopfield network and have not considered any extension of the original model. The scope of this particular assignment is well-structured and fulfilled for our capacity. And the limitation for the given project is that some tasks are similar but are separated into different sections

2 Methods

The coding language for our task is built upon Python 3.6 due to the overall universality for dependences. For the entire project, we conduct the code based on NumPy 1.19.2 and Matplotlib 3.3.3 for visulization.

3 Results and discussion

3.1 Convergence and Attractors

We first calculate the weight of Hopfield Network with the given three patterns, and try to recall the patterns based on the three distorted data. And then we run all the patterns with eight bits to search for the attractors in this network. Finally, we flip more bits and check if they can converge or not.

- $x1d$ and $x3d$ can converge towards correctly to the stored patterns, while $x2d$ can not.

- The distortion of $x1d$ and $x3d$ do not change a lot of the data distribution of pattern $x1$ and $x3$, while $x2d$ might have been changed a little bit more, because in pattern $x2$ -1 covers most of the bits, and after the distortion it's not. This is the reason that $x2d$ can no longer converge.
- We find out 14 attractors in total for this 8×8 network. And they are 7 pairs of orthogonal vectors as shown in Table 1.
- When half bits of a pattern are inverse, it never converge.

Table 1: Attractors

Attractors	Orthogonal Attractors	Energy
[-1. -1. -1. -1. -1. 1. -1. -1.]	[1. 1. 1. 1. 1. -1. 1. 1.]	-8.5
[-1. -1. -1. -1. 1. -1. -1. -1.]	[1. 1. 1. 1. -1. 1. 1. 1.]	-4.5
[-1. -1. 1. -1. -1. 1. -1. 1.]	[1. 1. -1. 1. 1. -1. 1. -1.]	-8.5
[-1. -1. 1. -1. 1. -1. -1. 1.]	[1. 1. -1. 1. -1. 1. 1. -1.]	-8.5
[-1. -1. 1. -1. 1. 1. -1. 1.]	[1. 1. -1. 1. -1. -1. 1. -1.]	-7.0
[-1. 1. -1. -1. -1. 1. -1. -1.]	[1. -1. 1. 1. 1. -1. 1. 1.]	-7.0
[-1. 1. 1. -1. -1. 1. -1. 1.]	[1. -1. -1. 1. 1. -1. 1. -1.]	-9.0

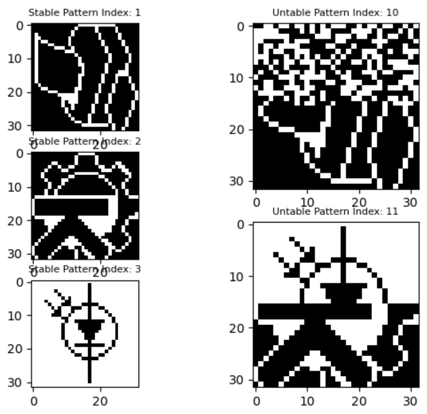


Figure 1: Stable Patterns and Degraded Patterns

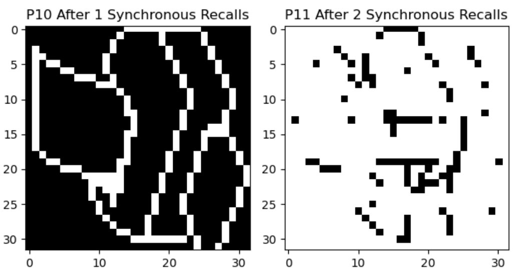


Figure 2: Convergence of Degraded Patterns

3.2 Sequential Update

In this part, we load the pictures data given in *pict.dat* file. We check the patterns and find that the first three pictures are attractors, which can also be called stable. Fig. 1 displays the three stable picture data$p1$, $p2$ and $p3$(on the left side) and the two degraded picture data $p10$ and $p11$(on the right side). We train the Hopfield Network with the three stable ones and check the recall performance for degraded ones. Finally, we check the converge state on sequential update every 256 iterations.

- In Fig. 2 we can see $p10$ can converge to $p1$ while $p11$ fails to converge.
- In Fig. 3 we can see $p10$ gradually converge to $p1$. When the iteration reaches around 1024 the converge is finished, which is exactly the bits number in the pattern.

3.3 Energy

In the first two questions, we took the data in Section 3.1 due to it has less computational complexities, as for the remaining questions, we take the data given by the *pict.dat* file. We could see, if we normalized

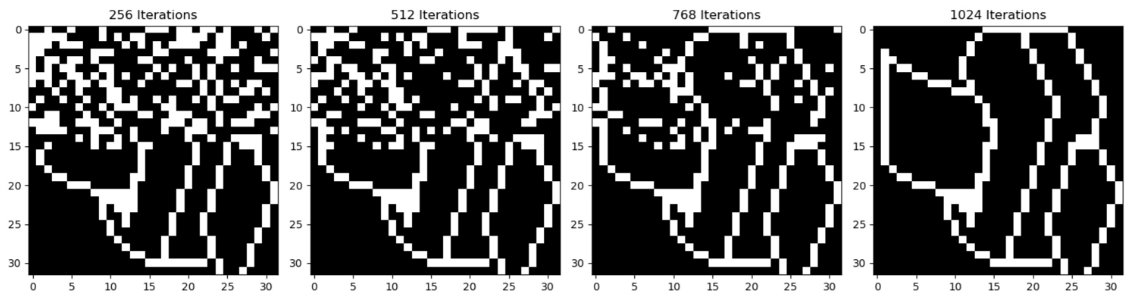


Figure 3: Convergence State of $p10$ with Sequential Update

the weight, we can get the energy from -9.0 to -4.5 in Table 1, and for the distorted pattern, the energy is ranging from -5.0 to -4.5. This stands for the attractors have the distorted version of given input. And we could see the energy direction from iteration to iteration in Fig 5a, which is sequential rule. We

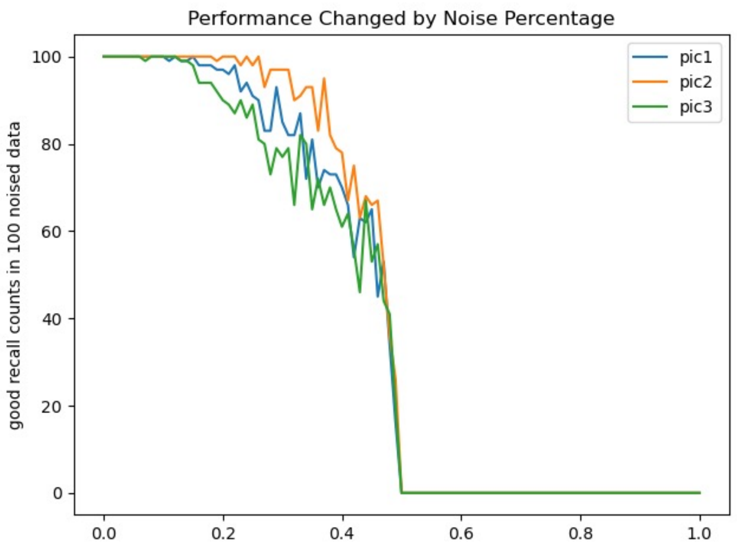


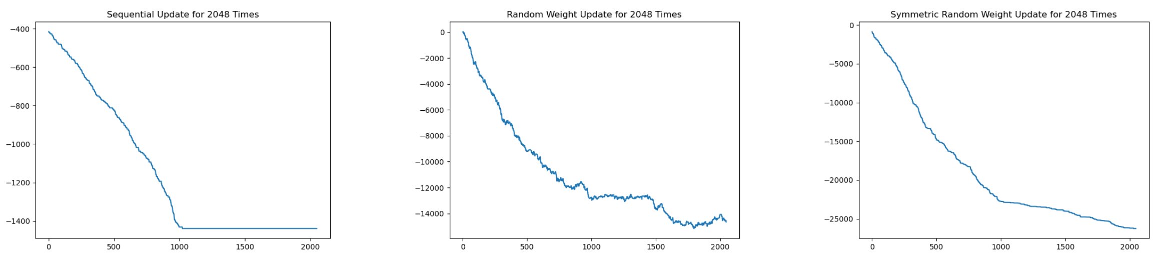
Figure 4: Good Recall Numbers Changed by Different Noise Percentages

can observe the pattern is reach minimal in 1024 times which is the dimension of input space. As for the Random Weight Sequential Update with Random Weight in Fig 5b, we could see that due to randomized weight, the network can not find the global minimal, but if we set the random weight into symmetric, we could see that in the iteration of 1024 times, we could see that it have a abrupt drop in the speed of reach minimal, and after 1024 iterations, the speed is much slower. This is due to the fact that even though the value is randomized, it share some similarity with the trained weight.

3.4 Distortion Resistance

We generate the noised data based on $p1$, $p2$, and $p3$ for the noise percentage range from 1% to 100% with step size 1%. For each noise percentage, we randomly generate 100 noised data for one picture, and test the average recall performance. Fig. 4 shows the average good recall counts for each noise percentage of one pattern. And in Fig. 6 we visualize the convergence state of noised $p1$ with the noise percentage in [0.2, 0.5, 0.8].

- With noise less than 10%, nearly all noised patterns can be recovered. The good recall numbers



(a) Sequential Update (b) Random Weight (c) Symmetric Random Weight

Figure 5: Sequential Updating for 2048 Times with various Weight

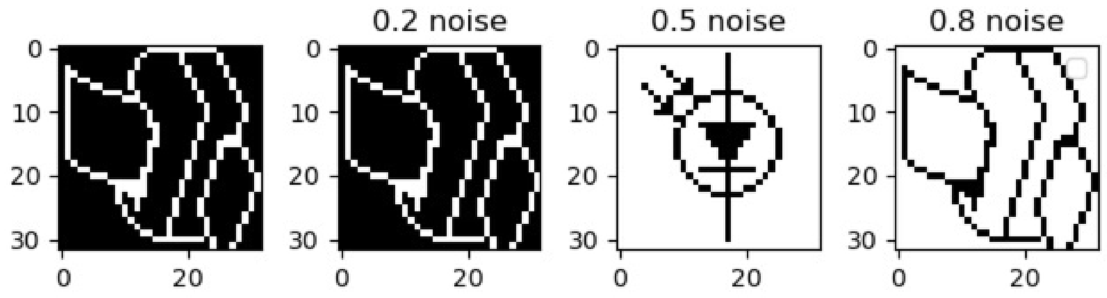


Figure 6: $p1$ Restoring Performance with Different Noise Percentage

begin to reduce but still have the possibility to converge when the noise percentage is larger than 10% and less than 50%. However, when the noise beyond 50%, none of the patterns can be recovered successfully.

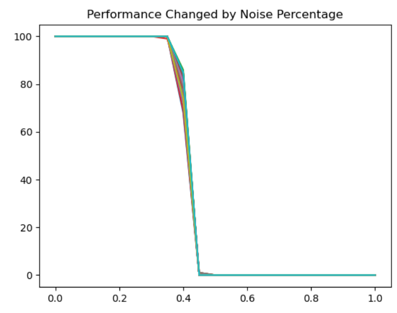
- Pattern2, i.e., $p2$, seems to have a relatively higher noise tolerance.
- When the noise beyond 50% the network never converge to the right attractors and always converge the other attractors. The extra iterations can not help. When the noise percentage is really large, like 80%, the noised pattern will converge to orthogonal attractor.

3.5 Capacity

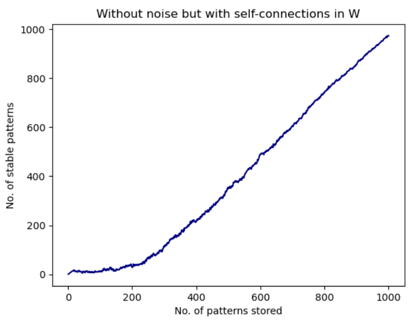
One question that is obviously important in evaluating a memory-system such as a Hopfield Network is storage capacity. To investigate this property we first looked at how many of the images already encountered could be stored in a Hopfield network with the same size as the images. The results: storage capacity depends on the specific set of images chosen for storage. For instance, if the network was trained on images 1-3 it could recall all three but if image 4 was added, no image could be recalled. If we instead trained the network on images 3-5 it would only remember one image. The maximum number of images we were able to store in our experiments where four: when the network was trained on the last four images.

Such a low storage capacity is surprising as it's theoretically proven that the memory capacity is $0.138d$ and one would thus expect the network with 1024 nodes to be able to store 140 images. This might be because the images have pixels that are heavily correlated. To test this further we initialized random patterns of the same sizes as the images with each bit being drawn independently from a Bernoulli(0.5) distribution. In fig 7(a) we trained the Hopfield Network on 50 such patterns (each a line in the plot). The results are that all 50 patterns are stored and stable, also we can see it handles noise very well.

The network's autoassociative memory ability can be viewed as it's ability to give an output equivalent to it's input if that input is a pattern the network has previously been trained on. Thus, disregard-

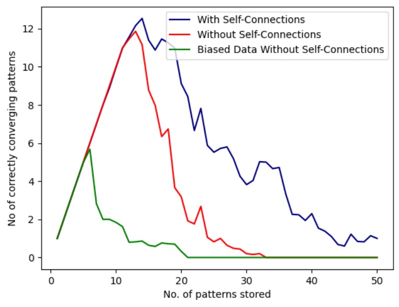


(a) Percentage of randomly distorted patterns converging to the correct attractor (y-axis) as noise-level (x-axis) change. Each line corresponds to one of 50 images.

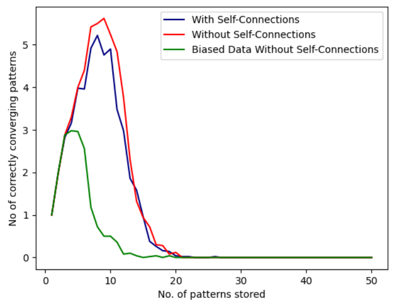


(b) Number of stable patterns as the number of patterns stored in the network was increased. Note that the network includes self-connections

Figure 7



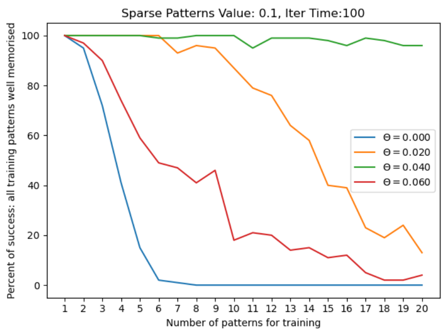
(a) 5-bits flipped for patterns tested for convergence



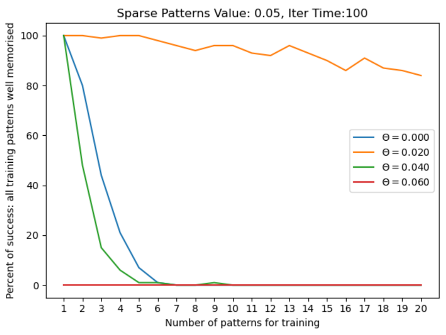
(b) 35-bits flipped for patterns tested for convergence

Figure 8: How storage capacity (the average number of converging patterns) changes as more patterns are added to the network. For every trained network (equivalenly, for each value on the x-axis) 50 flipped patterns were generated for each trained pattern (also x-azis), and the average amount of correctly convering patterns were computed.

ing corrupted or incomplete inputs one might wish to investigate how many such retrievable patterns (equivalently called stable patterns) could be stored in a given network. To this end we generated 300 patterns randomized as before but with $d = 100$ and fed them to the network one-by-one to see how the amount of stable memories changed as more memories where added to the network. The results can be seen in 7(b). We see that the number of stable patterns is less than the number of stored patterns but that the relationship approaches a linear 1:1 ratio when the number of patterns stores is large. This may seem like a high storage capacity but with a closer look (figure 8)(a) we see that it's due to the self-connections and that this seemingly remarkable memory ability disappears when noise is introduced. There we see how many original patterns converged when the network is fed training patterns with a few bits flipped. We thus see that the linear relationship we saw before disappears when we consider these bit-flipped patterns instead (blue curve). Usually one wishes to work without self-connections due to fewer spurious patterns and noise-removal capabilities and in the figure we also see that the capacity then is very similar(red curve. Lastly, we find insights into the observed sub-par storage capacity for the image data before, as we see that the maximum storage capacity with slightly biased data is much less than without (green curve). A bit unexpected is that in figure 8(a) the network with self-connections is more robust to the low noise level (only 5-bits are flipped). If we increase it to 35 bits the network without is only slightly better. It thus seems that storage capacity is related to the distribution of the input patterns, if the input patterns are considered as sparse to each other, the network would consider the sparse input pattern as the same distribution space, which could lead to worse performance, i.e. the storage capacity for converge patterns are exceedingly low.



(a) Sparse Level: 0.1



(b) Sparse Level: 0.05

Figure 9: Sparse Data Training, Bias: $\theta = \{0.00, 0.02, 0.04, 0.06\}$

We also see that our results are in line with the theoretical memory capacity of $0.138d$ and in all cases we see that beyond a certain point, loading the network with more patterns causes it to rapidly "forget" all patterns, including those previously learnt.

3.6 Sparse Patterns

We took the sparse level $\rho = \{0.1, 0.05\}$, and the bias value $\theta = \{0.00, 0.02, 0.04, 0.06\}$. We could see that the overall best bias value is 0.02, with could be considered as the value given in the instruction. We could see that the overall decent performance in Fig 9a as well as Fig 9b. And the trend in Fig 9b is more steep which stands for the extremity in input data space.

4 Final remarks

In this lab session, we have learnt the Hopfield Network from a methodology perspective as well as a practical perspective. The overall experience is significantly beneficial for understanding the foundation of RNN. And we have a better understanding of how can it tolerate the noise and what effect its storage capacity.