Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 3

Hopfield networks

Robert Cedergren, Viktor Törnegren and Lingxi Xiong

February 19, 2019

1 Main objectives and scope of the assignment

- To explain the principles underlying the operation and functionality of autoassociative networks
- To train the Hopfield network and explain the attractor dynamics of Hopfield networks the concept of energy function,
- To demonstrate how autoassociative networks can do pattern completion and noise reduction,
- To investigate the question of storage capacity and explain features that help increase it in associative memories.

2 Method

Python 3.6 is used to conduct the tasks and experiments in this lab. The basic Numpy library is used to implement the algorithm and the Matplotlib is used for visualization.

3 Results and discussion

3.1 Convergence and attractors

The Hopfield network is able to store all three patterns. Since the x1d, x2d, and x3d are close to the attractors x1, x2 and x3 (with only one or two bit error), they converge towards to the corresponding stored patterns in 1, 1 and 2 iterations of synchronous mode, respectively. For this 8-node network, we

find 14 attractors including the x1 x2 and x3, by using activation function of $sign(x) = \begin{cases} 1 & x >= 0 \\ -1 & x < 0 \end{cases}$. The attractors are shown in Table 1.

Also, when using more dissimilar starting patterns (more than half of the bits are wrong), they will converge to other attractions that are the closest to them.

Table 1: Attractors

1(x2)	-1	-1	-1	-1	-1	1	-1	-1
2(-13)	-1	-1	-1	-1	1	-1	-1	-1
3(-11)	-1	-1	1	-1	-1	1	-1	1
4(x1)	-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(x3)	-1	1	1	-1	-1	1	-1	1
8	-1	1	1	-1	1	-1	-1	1
9(-x3)	1	-1	-1	1	1	-1	1	-1
10(-x2)	1	1	-1	1	-1	1	1	-1
11(-3)	1	1	-1	1	1	-1	1	-1
12	1	1	-1	1	1	1	1	-1
13(-2)	1	1	1	1	-1	1	1	1
14(-x1)	1	1	1	1	1	-1	1	1

3.2 Sequential Update

Both synchronous approach and asynchronous approach couldn't always complete all kinds of degraded patterns. The recover result is shown in Figure 1. The intermediate images of pattern 10 during updating are shown in Figure 2.

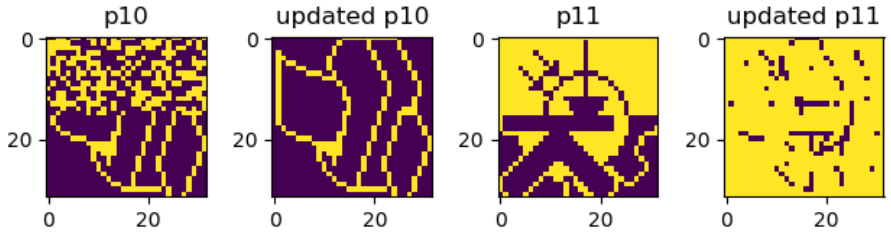


Figure 1: Update result of distorted patterns

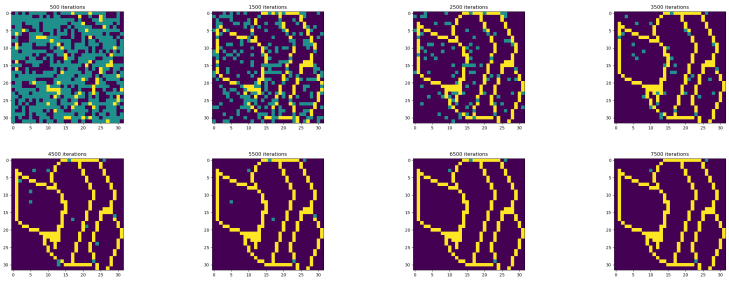


Figure 2: Images of pattern 10 in every 1000 iterations

3.3 Energy

Energy of different attractors and patterns are shown in Table 2.

Table 2: Attractors

attractor/pattern	energy
p1	-1473936.0
p2	-1398416.0
p3	-1497344.0
p10	-425964.0
p11	-177664.0

The energy changes from iteration to iteration to approach an attractor with sequential update rule can be seen in Figure 3.

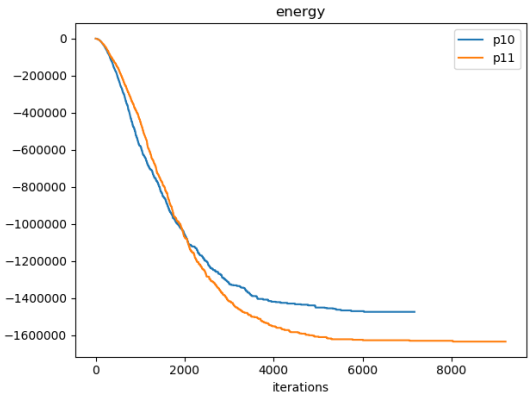


Figure 3: Energy changes by using sequential update rule

When we set the weight matrix to normally distributed random number instead of XX^T that learn the information from input data, both distorted patterns couldn't converge within 10^4 iterations (here one iteration runs through 1024 random nodes). Visualizing the output/final state, we can see they become more and more chaotic.

Another case is to randomly set weight matrix as a symmetric one. Both distorted patterns are able to converge to the stored/original patterns this time with 60 and 47 iterations respectively by using the asynchronous approach. In general, it takes less time and iterations to converge by using symmetric weight matrix.

3.4 Distortion Resistance

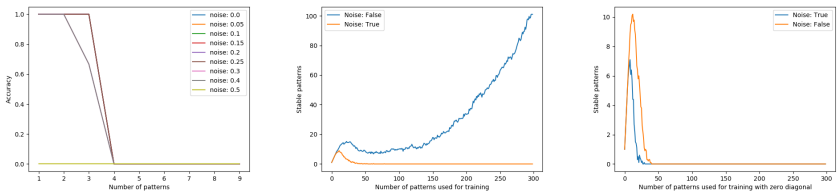
The accuracy, i.e. the rate of recovering the original patterns correctly at 100 runs of adding different random normal noise with zero mean and 2 std is shown in the table 2. We can see that in pattern 1, when the added noise becomes greater than 30%, the network start to fail in successfully recovering. Also, there are difference between attractors with regard to noise tolerance. For example pattern 2 handles well with much more noise, while pattern 3 generate worse result. And if we combine these phenomenon with the energy of those attractors, it seems that larger energy has higher tolerance towards to noise.

Table 3: Correct recovery rate (accuracy) among different patterns with different percentage of noise

	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
p1	1.	1.	1.	0.96	0.94	0.9	0.9	0.86	0.84	0.78
p2	1.	1.	1.	1.	1.	1.	0.99	0.99	0.95	0.98
p3	0.99	0.98	0.98	0.85	0.85	0.86	0.91	0.84	0.82	0.78

3.5 Capacity

We add more and more memories to the network and see that for noise up to 30 % and up to three pictures used for training, three pictures (patterns) can safely be stored. When the number of patterns used for training are more than three the drop in performance is abrupt and the network can not store any pictures (patterns), independently of the level of noise. This can be seen in Figure 4 (a).



(a)

Figure 4: (a) Plot of number of patterns stored given number of patterns used for training and noise level. All lines for noise less than 0.4 (40 %) are equal. (b) Plot of. (c) Plot of .

The experiment is repeated with learning a few random patterns instead of the pictures. The result shows that with noise up to 40 % all the patterns can be stored. For the noise level of 50 % no patterns can be stored.

In the case of independent (orthogonal) patterns the the capacity of a Hopfield network is 0.138N, where N is the dimension of the patterns used in training, i.e. the number of nodes. Correlation between memory patterns limits the capacity. This is demonstrated with our pictures and our random patterns. In the case of the pictures recollection of the wrong pattern occurs due to correlations between the pictures whereas random patterns are more likely to be uncorrelated and thus the network can memorize more patterns.

We create 300 random patterns and train a 100-unit network with them. After each new pattern has been added to the weight matrix, the number of earlier patterns that remain stable (a single iteration does not cause them to change) are calculated. As seen in Figure 4 (b) the stable patterns increases with the number of patterns used for training. This is due to the fact that we have non-zero diagonals which gets larger and larger in relation to all the other elements (the cross terms) as the number of patterns used for training increases. I.e. the network does not learn, it simply gives back the same pattern just like a multiplication with the identity matrix would. Note that the scaling of the W matrix does not matter since we only use 1s and -1s. When recalling noisy patterns (10 % noise level) we basically get back the noisy patterns when the number of patterns increases, also due to the large numbers on the diagonal in relation to the other elements. Hence, the number of stable pattern goes to zero as the number of pattern increases. We remove the self-connections and in Figure 4 (c) it can be seen that the difference between the curves with pure and noisy patterns for large number of patterns goes away. The maximal number of retrievable pattern for this network is 11 and 7 with pure and noisy patterns, respectively.

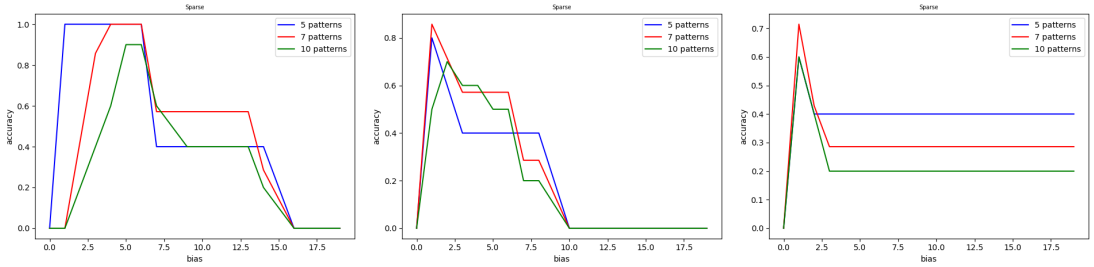
By adding bias to the training patterns the number of retrieved patterns decreases, and they are doing so for the same reason that less patterns could be retrieved when using pictures rather than random patterns. Adding bias of 0.5 means that each element in each pattern are more likely to be set to 1 instead

of -1. Hence, the patterns with bias will be more correlated and will thus limit the capacity of the network.

3.6 Sparse Patterns

When we create a sparse data set with 10% activity, then only 10% of the elements in each pattern will have the value one. Thus, we need to add a bias term to favour the value zero for each element. In Figure 3a it can be seen that we can store at most 7 patterns for a data set with 10% activity and we need the bias term θ to lie in the interval $[3, 6]$. If we use a higher bias than 6, then we will favour the value one, and since only 10% of the elements in each pattern have the value one, we will get a quite bad accuracy.

Moreover, when we create a data set with with 5% respectively 1% activity, all of the patterns will be almost identically. Hence, it will be almost impossible to separate one pattern from another. This can be clearly seen in Figure 3b and Figure 3c.



(a) 10% activity (b) 5% activity (c) 1% activity

Figure 5: Accuracy for randomly created sparse pattern with bias, θ, in the range [0, 20].