Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 3

Hopfield Networks

Mustafa Al-Janabi, Einar Lennelöv - Group 15

February 17, 2020

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- To understand, study and explain the principles behind the functionality of an auto associative network.
- To train a Hopfield network and study its attractor dynamics, energies, noise reduction and storage capacities.

We have focused on qualitative findings that help in the understanding of the algorithms, as opposed to more rigorous numerical results.

2 Methods

All of the code was written in Python. The data sets used were those provided in the assignment.

3 Results and discussion

3.1 Convergence and attractors

In the first part of this assignment the focus is on ensuring that the Hebbian learning in the Hopfield network is implemented properly. Three patterns of 8 bits each where used as training data. Proper is assumed to be one where the network is able to recall the patterns it was presented with once it is presented with them again. Furthermore, the network's capability of recalling patterns based on distorted patterns as inputs.

The following shows the distorted patterns that were used as inputs and their convergence:

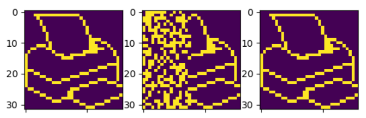


Figure 1: Example of memory (left), input (center), and output (right).

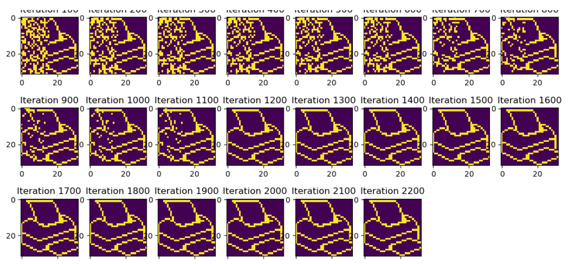


Figure 2: Sequence of outputs with sequential updates.

x1d=[1 -1 1 -1 1 -1 -1 1] $\implies$ [-1 -1 1 -1 1 -1 -1 1]
x2d=[1 1 -1 -1 -1 1 -1 -1] $\implies$ [-1 1 -1 -1 -1 1 -1 -1]
x3d=[1 1 1 -1 1 1 -1 1] $\implies$ [-1 1 1 -1 -1 1 -1 1]
Out of those, the middle one doesn't belong to the original three patterns that were used for training. That suggests there are other attractors and indeed we found **11 attractors** in total when trying every possible combination of the 8-bit input pattern. Among those 11 is the original three that the network was able to store.

3.2 Sequential Update

We now turn to dealing with a more realistic data-set, consisting of 1024 dimensional binary vectors. These can conveniently be represented as 32x32 images. We begin by examining the basic behaviour. Figure 2 shows the memory (left), input (center) and output (right). The Hopfield network is clearly able to recover the correct memory, despite the chunk of noise.

It is also possible to perform image recovery using sequential steps. This is illustrated by Figure **??**. It shows snapshots of the output every 100 steps. Convergence is obviously much slower, but only one update per pixel is necessary.

3.3 Energy

When the weight matrix is symmetric it is possible to derive a Lyapunov function that describes the decrease in the energy of the system for every time-step taken. Indeed, that was found for the Hopfield network in this section of the assignment.

When on the first three patterns P1,P2 and P3, the energy of the attractors was found to be

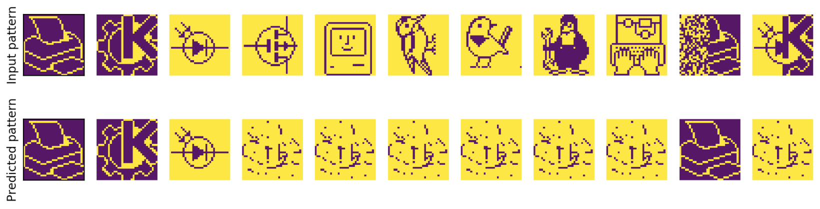


Figure 3: The input on the first row and the prediction on the second

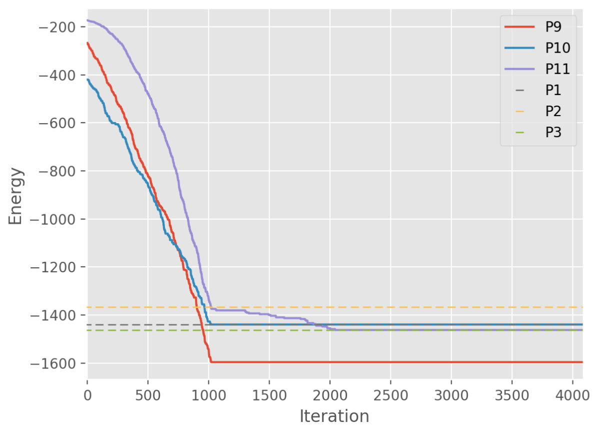


Figure 4: The energies of the last three patterns with energies of the attractors corresponding to the first three patterns.

$E_{P1} = -1439.4$, $E_{P2} = -1365.6$, $E_{P3} = -1462.3$
Based on that, the prediction of all the patterns in the data-set is show in figure 3. The results suggest that there are at least four attractors. When studying the energy level for the last three patterns, the result show in figure 4 confirms that result. P9 converges to the fourth attractor while the distorted version of P1 and P3 each converges to their corresponding attractors. Note however that P11 does not always converge to P3. Sometimes it converges to the fourth attractor as is shown in figure 3. This shows that depending on the order of the update in the sequential method, the network can exhibit a forgetting behaviour.

Further analysis of the energy function reveals that when using only randomly generated weights, only a symmetric matrix exhibits the qualities of a Lyapunov function. The energy function using non-symmetric weight matrix is not strictly decreasing, as seen in figure 5. Furthermore the energy is much lower than that of the attractor, but that is expected since no Hebbian learning takes place.

3.4 Distortion Resistance

One of the main applications for Hopfield networks is noise reduction. A key property to investigate is thus how much noise the network can handle. We

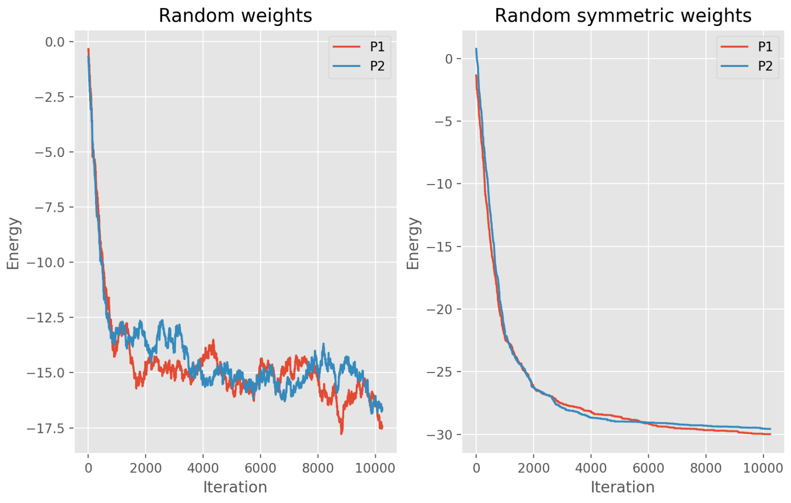
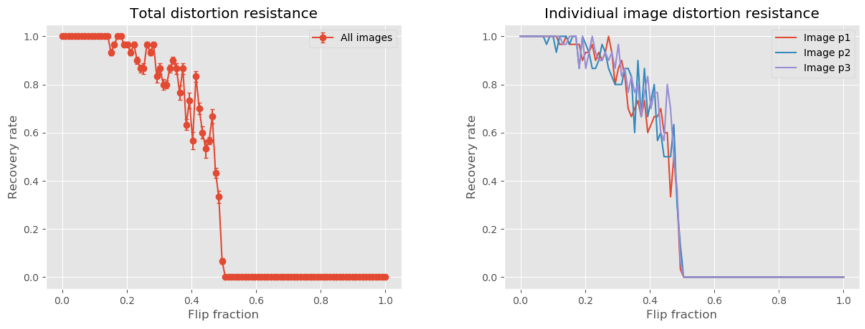


Figure 5: The energy function



(a) All 3 images.

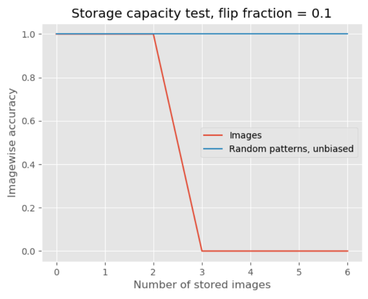
(b) Individual images.

Figure 6: Recovery rate as function of noise fraction.

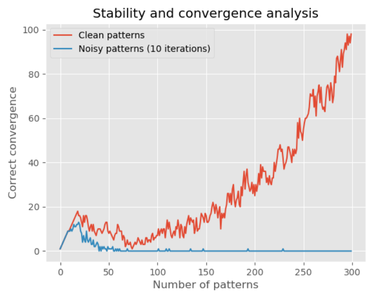
conduct and experiment where we train the network on the images $p1, p2, p3$, apply noise to each image, and then attempt to recover the original. Figure 6a shows the average rate of full recovery (over a 100 trials) as a function of the fraction of flipped pixels. A clear avalanche behaviour can be observed. When more than half of the pixels are flipped, the network is unable to recover the image. Figure 6b shows the same measure for each individual image, indicating that the behaviour is the same for each.

3.5 Capacity

Another key property of the Hopfield networks is the storage capacity. Figure 7a shows that for a 100-node network, only the first three images can be succesfully

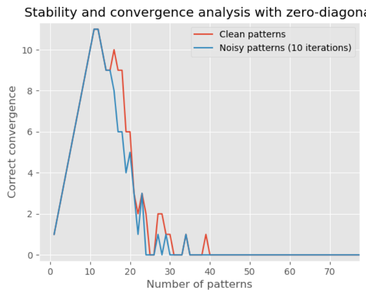


(a) Capacity on real images.

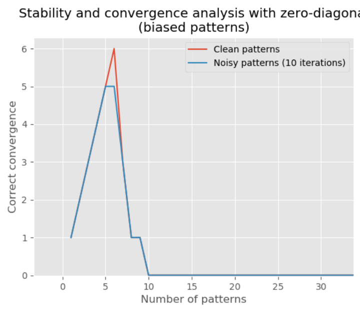


(b) Capacity on random patterns.

Figure 7: Number of successfully recovered images as function of number of stored images.



(a) Unbiased samples.



(b) Biased samples (note y-axis scale).

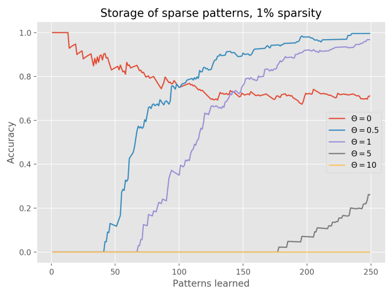
Figure 8: Number of successfully recovered images as function of number of stored images, with diagonal of weight matrix forced to 0.

stored and recovered. If a fourth image is added, none of the images can be recovered. The figure also shows that this does not happen when the images are random sequences of pixel, with equal number of positive and negative pixels (that is, they are unbiased). This warrants further investigation.

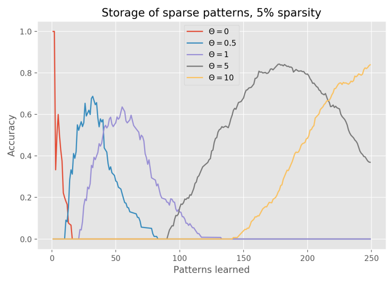
Figure 7b shows the number of unbiased random images that can be recovered, both when the inputs are clean images and when slight noise is added. It is clear that about 100 of the images are fix points, but that they are not stable attractors, since adding some noise makes recovery impossible.

This behaviour is not very useful, since we should expect at least some noise on the inputs. We can avoid it by forcing the diagonal of the weight matrix to 0. Figure 8a shows the corresponding plot when this is done. The two lines are now similar.

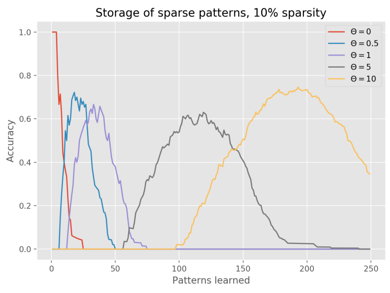
So far we have examined unbiased images, but real images are likely to be biased. We therefore repeat the experiment with random images with 30% negative samples, and 70% positive samples. Figure 8b shows the results. The network is now only able to store and recover about half as many images, illustrating a major issue with the model.



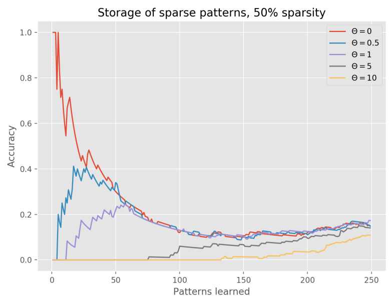
(a) 1% sparsity



(b) 5% sparsity



(c) 10% sparsity



(d) 50% sparsity

Figure 9: The accuracy of the network as a function of the patterns learned for different biases and sparsity levels.

3.6 Sparse Patterns

In Hopfield networks, introduction of bias decreases the storage capacity significantly. In this section we study the extent of that by using sparse patterns with different biases in the data. The additional bias also leads to the necessity of an altered update rule. In figure 9 we find that the higher the bias the more patterns are required to achieve higher accuracy. Thus, performance is degraded. Furthermore, the lower the sparsity the less the performance is degraded. Which is an expected result since low sparsity means the majority of the patterns consist of zeros and, thus, the stored inputs are more likely to be similar.

4 Final remarks

We enjoyed this assignment, primarily because it was quite limited in scope and focused on aspects of Hopfield networks that seem particularly important for practical use, while also illustrating some quite surprising concepts (such as the effect of the biased learning).