Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 3

Learning and generalisation in feed-forward networks — from perceptron learning to backprop

Magnus Tronstad, Teo Jansson Minne and Quintus Roos

February 18, 2021

1 Main objectives and scope of the assignment

The main objective with this assignment is to further extend our knowledge of Hopfield networks (autoassociative networks, attractor dynamics and energy function for Hopfield network, effect of noise, recall patterns and capacity of Hopfield newtorks).

2 Methods

This assignment is written in Python 3. We used the libraries Numpy and Scipy for scientific calculations and Matplotlib for creating plots.

3 Results and discussion

3.1 Convergence and Attractors

In this first section a discrete Hopfield network was tasked with storing bipolar patterns of size 8. Three patterns were stored using the Hebbian learning rule and a synchronous update scheme was used for recall. The stored patterns as well as three slightly distorted versions of them were successfully remembered by the network and even recall of patterns entirely distorted converged to one of the attractors after just a few updates. Since there only exists $2^8 = 256$ unique 8-bit patterns these fix point attractors could be found by simply testing the single update recall stability of all these patterns. As expected the three stored patterns did constitute fix point attractors, as well as their sign inverses. This is a general attribute of the discrete Hopfield network, it is sign blind, meaning that if $\vec{X}$ is an fix point attractor, so is $-\vec{X}$. Three spurious patterns, and their respective sign inverses, were also found and a grand total of 12 attractors were found in the network. These spurious patterns did however disappear if

one removed the diagonal elements of the weight matrix, showcasing why this should usually be done.

3.2 Sequential Update

For this part of the assignment, the provided dataset *pict.dat* was used. The dataset contains nine original and two distorted patterns, each containing 1024 bits, increasing the number of neurons in the network to 1024. Without the little model, the first three patterns p1, p2 and p3 where stable during learning, meaning that none of them showed any changes when updating the network.

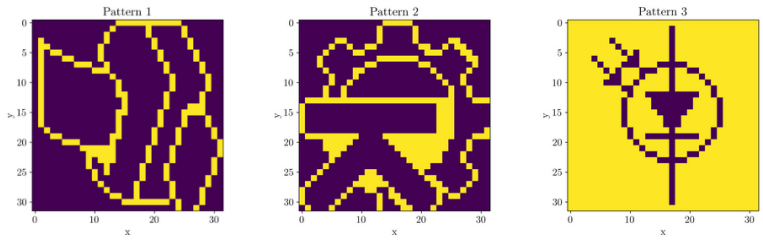


Figure 1: *First three patterns p1, p2 and p3.*

When using p1, p2 and p3 together with the little model, the network was only able to complete the degraded pattern p10 (50% p1 and 50% noise) and not p11 (50% p2 and 50% p3). The resulting patterns after processing both p10 and p11 can be seen in figure 2.

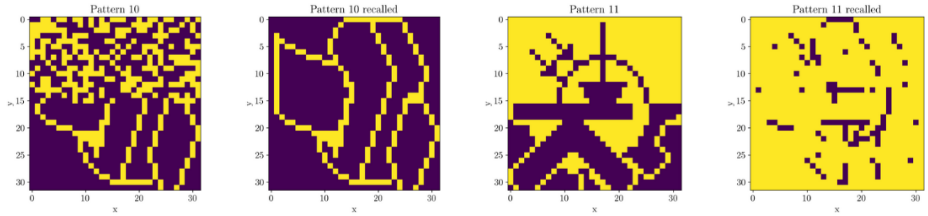


Figure 2: *Pictures of p10 and p11 before and after the network's attempt to complete them.*

When using asynchronous updates, iteratively selecting units to update randomly and without using the little model, p11 entirely converged towards p3. After multiple tries, the convergence was consistently completed between 1900-2000 iterations. Figure three shows the progress of the networks convergence at different iterations.

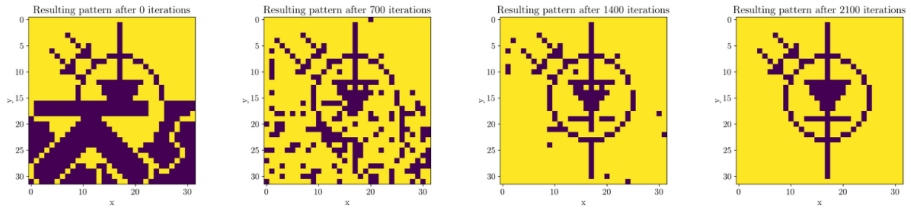


Figure 3: *Pictures of p11 converging to p3 at 0, 700, 1400 and 2100 iterations.*

3.3 Energy

The energy levels for the picture patterns p_1, p_2, p_3 as well as for their distorted counter-parts p_{10} and p_{11} are given in Table 1:

	p_1	p_2	p_3	p_{10}	p_{11}
E	-491312	-466139	-499115	-141988	-59221

Table 1: *Energy levels for different picture patterns.*

Note that the attractors, p_1, p_2, p_3, have *lower* energies than the distorted pictures, which was expected since the fix point attractors correspond to minima of the energy function. The relationship between patterns and their energy levels was studied further by monitoring the convergence of energies when the distorted patterns p_{10} and p_{11} were recalled iteratively, using asynchronous updates. The results are shown in Figure X. We note that the convergence in energies reflects the findings in the earlier part of the lab: p_{10} tends to the energy level of p_1 and p_{11} tends to the energy level of p_3.

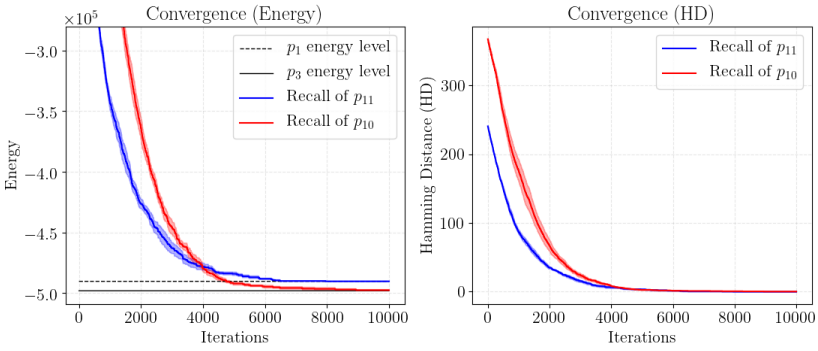


Figure 4: *Convergence in energy (left plot) and Hamming Distance (HD) (right plot), when the network is given the patterns p_{10} (red) and p_{11} (blue) respectively.*

Next, we illustrate the salience of imposing that the weight matrix be *symmetric*. In Figure 5, the energy convergence curves are shown for the case when the elements of the weight matrix W are generated randomly from a normal distribution. Clearly this does not lead to convergence. However, we see that when the weight matrix is symmetric (we set it to $W' = (W^{\mathrm{T}} + W)/2$), the

energy eventually converges to some minimum. This is expected since we are guaranteed convergence when the weight matrix is symmetric (although the proof rests also on the fact that the diagonal elements are set to zero, which is not the case here, it is a necessary condition for convergence).

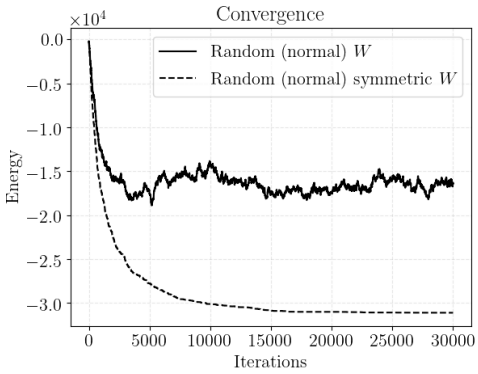


Figure 5: *Convergence in energy from an arbitrary 1024-dim bipolar pattern vector when the weight matrix W is randomly generated with normal distributed elements (solid) line, and when the weight matrix is set to $W' = 0.5(W^{\mathrm{T}} + W)$ (dashed line).*

3.4 Distortion Resistance

To investigate the networks distortion resistance, noise was added to the three patterns and then put into the network again ($\mathrm{p1}_{noisy}$, $\mathrm{p1}_{noisy}$ and $\mathrm{p1}_{noisy}$). The added noise started from 0% to 100% with increments of 2%. The resulting performance is shown in figure 6.

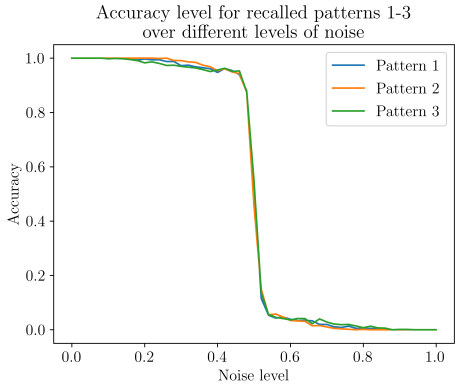


Figure 6: *Accuracy for patterns 1, 2 and 3 with noise from 0-100%.*

From the resulting graph, all three patterns seemed to have reach a recall-limit of around 50% before drastically loosing accuracy. There also seems to be small differences between the patterns when it comes to distortion resistance. In addition, the network seems to always converge to the right attractor when the noise is below around 45-50%, while it converges to the other attractor when

the noise is above 50-55%. Therefore the conclusion is that low level of noise leads to high accuracy, and high level of noise leads to low accuracy (i.e. all elements where flipped). In order to investigate which attractors the network converges to between noise levels 40-60%, all results were plotted in this range. When inspecting the plots, it turns out that there is no consistency in the convergence. For instance, sometimes p1_{noisy} would converge to p2 and other times to p3, both being strong attractors. It is therefore difficult to draw any conclusion from that specific range. Adding extra iterations beyond a single-step recall together with the little model does not seem to change anything in particular.

3.5 Capacity

To illustrate factors influencing the storage capacity of a Hopfield network, Figure 7 plots the performance in terms of the fraction of correctly recalled elements as a function of the fraction of distorted elements compared with the original pattern (p_1 in this case).

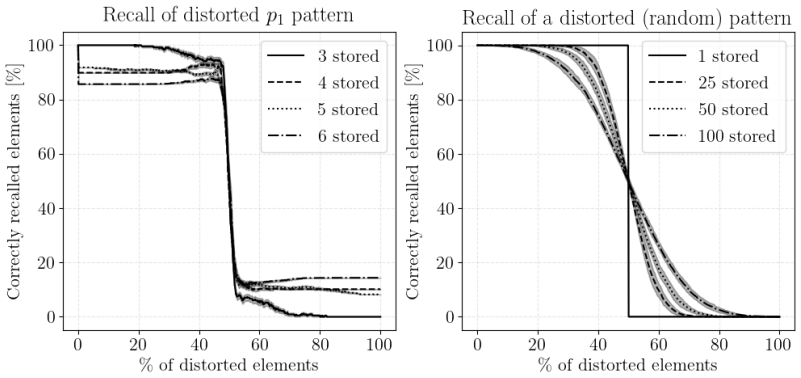


Figure 7: *Fraction of correctly recalled elements after a one-shot recall with the little model for picture patterns (left) and random patterns (right). Different degrees of noise is added to the patterns before recall (horizontal axis). Since the noise is added randomly, mean values are computed over 30 different runs. The shaded areas correspond to 5 standard deviations.*

First, looking at the left hand plot, we note that when the three first patterns are stored, the network is able to correctly recall patterns that have up to 20% of their elements disturbed (chosen at random) from the original p_1 pattern. Storing more patterns worsened performance. Not only is the network not able to *safely* store more than 3 patterns, as more patterns are added, the performance after 1 recall update decreases with the number of new patterns being stored.

The performance is very different when random (1024 dimensional) patterns are recalled. The right hand plot in Figure 6 shows the same experiment run for *random* patterns. While storage capacity decreases as the number of stored patterns increases, the network is able to store many more patterns than when the pictures were used. This increase in storage capacity can be explained by the fact that random vectors likely "more orthogonal", and thus, the so-called

cross-talk term for any given recall should be minimal. That these random vectors indeed show a higher degree of orthogonality than the picture patterns is easily verified by checking the matrix $X^T X$: we found that the off-diagonal elements— corresponding to the scalar products of different vectors—were much bigger for the picture patterns compared with the random patterns.

Figure 7 shows the results from training a 100-unit network with increasing number of (random) patterns. Interestingly, we see that without removing the diagonal elements of the weight matrix, the storage capacity appears to improve when a very large number of patterns (more than 100) are stored. However, when the recall is made starting from a noisy pattern, this is not the case. We hypothesized this had to do with an increase of spurious solutions as more and more patterns are stored. When the diagonal elements are kept, these will begin to dominate such that the weight matrix will start tending towards the identity matrix (since it is normalized over the number of stored patterns). Hence, eventually, without removing the diagonal terms, as the number of patterns tend to infinity, we expect that any 100-unit pattern will be a fix point of the network. Indeed, when the diagonal elements $w_{ii} = 0$, this behavior goes away.

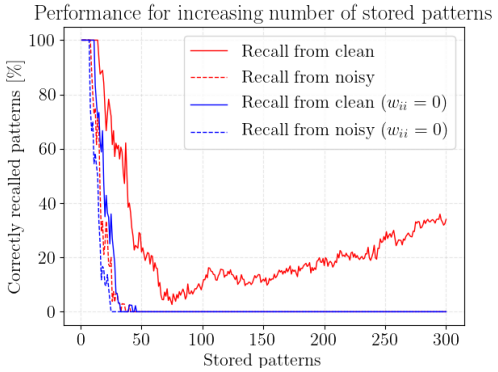


Figure 8: *Recall performance of random 100-dimensional patterns. The dashed lines correspond to recall from a noisy pattern (10% of the attractors patterns were changed).*

Adding similar levels of bias to that of the pictures (+1s to -1s ratio of around 0.43) to random patterns lead to worse performance compared to when there was no bias in the data. Thus we expect this to be one factor in explaining why the pictures were so hard to store, since the corresponding "biases" for the picture patterns were 0.21, 0.32, 7.0 for p_1, p_2, p_3 respectively.

3.6 Sparse Patterns

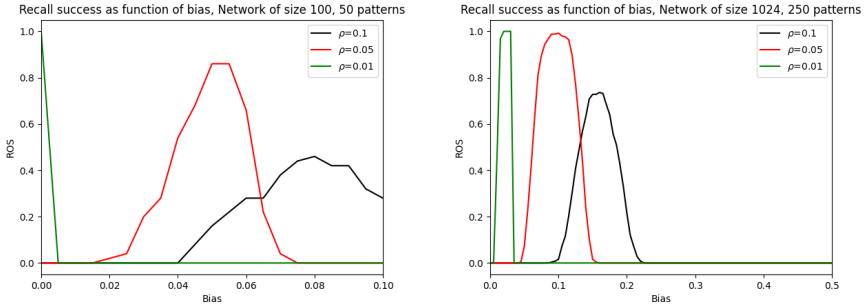
Real world data, as well as biological input, is in general biased. As shown in the previous section this poses a problem for the Hopfield network. One can, however, change the learning rule slightly by compensating for the average activity in the stored patterns and add a bias term to the updating schema to try and resolve this issue.

Binary patterns were randomly generated with a given activity and the storage capacity of the Hopfield network with these new rules was tested for different activities. Both a 100-bit and a 1024-bit sized network was tested. The results of plotting the recall ratio of sparse patterns versus the bias from one simulation can be seen in figure 9.a and 9.b for the 100-bit and 1024-bit sized network respectively. Some key takeaways here is that for very sparse patterns the coding space shrinks drastically, for example in the case of the 100-bit network and activity of 0.01 only 100 unique patterns exist, and this facilitates storage.

Finding a good value of the bias is crucial for good capacity and particularly for the large network the optimal bias varied by a fair amount between pattern sets, suggesting use of a preliminary search for optimal bias.

50 simulations with different randomly generated sparse patterns were run for both networks, noting the recall ratio when using the best performing bias in each simulation. The results can be seen in table 2 below. It is evident that this modified algorithm performs better on sparse and biased data although it requires some more fine-tuning than the more general basic Hopfield network. A final note here is that the behaviour was a lot less predictable when the self-connecting weights were kept.

Figure 9: *Bias dependency of the recall ratio for sparse Hopfield network.*



a) Network size 100, patterns stored = 50.

b) Network size 1024, patterns stored = 250.

Activity of patterns	Recall ratio, 100-bit network 100 stored patterns	Recall ratio, 1024-bit network 500 stored patterns
$\rho = 0.1$	0.04 ± 0.02	0.0056 ± 0.0008
$\rho = 0.05$	0.25 ± 0.06	0.70 ± 0.02
$\rho = 0.01$	0.74 ± 0.02	0.988 ± 0.005

Table 2: *Recall ratio, sparse Hopfield network. N sims = 50. (mean ± std)*