Artificial Neural Networks

Prof. Dr. Sen Cheng

Jan 20, 2020

Problem Set 13: Hopfield Network

Tutors: Mohammadreza Mohagheghi Nejad (m.mohagheghi@ini.rub.de), Eloy Parra Barrero (eloy.parrabarrero@rub.de)

1. **Asynchrounous updates in Hopfield network.** We stored one input pattern in a Hopfield network, which yielded the following weight matrix:

$$w = \begin{bmatrix} 0 & -1 & -1 & +1 \\ -1 & 0 & +1 & -1 \\ -1 & +1 & 0 & -1 \\ +1 & -1 & -1 & 0 \end{bmatrix}$$

   Based on this, answer the questions below.

   (a) How many elements did the pattern have?

   (b) Starting with any valid pattern you like, use algorithm 13.1 from the lecture notes using pen and paper to find the pattern the network converges to. Is the pattern you have found necessarily the one the network was trained on?

   (c) Now write a Python script that automatizes this process and compare your results.

   (d) Modify your code so as to calculate and store the energy of the network for each step in (c). How does it vary across multiple repetitions of your script?

2. **Pattern storage and retrieval in Hopfield network.** There are seven images stored in `images.npy`. Follow the tasks below to complete this exercise.

   (a) Load the data. You will observe that the shape of the data is 1150×7. The first dimension corresponds to the number of pixels in each image (46×25 pixels stacked column-wise into vectors). The second dimension corresponds to the number of images. The content of each pixel is either -1 or 1, where -1 corresponds to black and 1 to white.

   (b) Reshape the image vectors according to the information provided in 2a and visualize the images.

   (c) Store these images in a Hopfield network using Eq. 2 from the lecture notes.

   (d) Add noise to the images by randomly choosing a subset of pixels and flipping their values from -1/1 to 1/-1. Generate noisy images by setting the subset size to 200 and provide them as input to the network. Run the algorithm for 100 iterations to retrieve the stored images. Visualize this process by showing the original, noisy and retrieved images side by side.

   (e) For each image compute the correlation coefficient (`numpy.corrcoef`) of the original against the retrieved images as a measure of retrieval quality. The correlation coefficient (cc) outputs a number between -1 and 1, $cc \in [-1, 1]$. Interpret what these values mean and which value(s) correspond(s) to lossless retrieval.

   (f) During retrieval, even with 0 noise, one image is confused with another one. Compute the correlation coefficient across all pairs of images. Do you find any relationship between the computed correlation coefficients and the incorrectly retrieved image?

(g) Vary systematically the number of flipped pixels from 0 to 1150 with a step size of 25 and record the correlations between original and retrieved images. Plot these correlations for all images and noise levels, e.g., using `pyplot.matshow`. Describe the effect of noise in the retrieval process.

3. **Capacity of Hopfield network.** To study the memory capacity of the Hopfield network, generate and store random patterns in a network of 100 units. Then, initialize the network with each of the stored patterns and let it evolve for 100 iterations. After this, count the number of pixels which have deviated from the original value (`numpy.count_nonzero` might be of help here). Repeat this process for different numbers of patterns and plot the error rate as a function of the network load (= number of stored patterns / network size). Focus on network loads between 0 and 0.5. At which point does performance start to degrade steeply?