Computational Neuroscience
IIT Madras

Assignment 4 : Hopfield network

Ganga Meghanath
EE15B025

November 10, 2018

1 Aim

- Understand and Develop code for Hopfield Network for storing single and multiple patterns (images)
- Retrieve one of the stored patterns from the network corresponding to the input trigger
- Visualise the input triggers, original pattern and retrieved patterns
- Introduce noise into the strorage weights and analyse the effects on pattern retreival

2 Preliminaries

Since the given patterns are black and white (binary with 1 and -1), I have opted to use Discrete Hopfield Network over continuous since it's a simpler and direct implementation in such a scenario.

The patterns are stored in the weights matrix of the fully connected Hopfield Network as :

$$\boldsymbol{W} = \frac{1}{N}\sum_{i}^{N_p} \boldsymbol{s}_i \boldsymbol{s}_i^T$$

where $\boldsymbol{W} \in \{1,-1\}^{N\times N}$, $\boldsymbol{s}_i \in \{1,-1\}^{N\times 1}$ denotes the i^{th} pattern to be stored, N_p denotes the total number of patterns to be stored in the network and N denotes the total number of neurons in the network.

Note that all the patterns have been flattened out before storing it in the network and the all the patterns need to of the same dimension as that of the no. of neurons in the network. The weights from the j^{th} neuron to the i^{th} neuron is given by $\boldsymbol{W}[i][j]$.

The updates have been done in 2 different fashion :

1 **Synchronous** updates : Here all the neurons are updated simultaneously at each iteration. Hence the current step update is made to all the neurons depending on the previous time step values of all the neurons.

$$\boldsymbol{v}(t) = \sigma\big(\boldsymbol{W} \times \boldsymbol{v}(t-1)\big) \tag{1}$$

where $\boldsymbol{v} \in \{1,-1\}^{N\times 1}$, $\boldsymbol{W} \in \{1,-1\}^{N\times N}$, t denotes the time step or iteration in this case and σ denotes the sign function :

$$\sigma(x) = \begin{cases} \frac{x}{|x|}, & \text{if } x \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

2 **Asynchronous** updates : Here one randomly selected neuron is updated at each iteration. Hence the current step update is made to a randomly selected neuron depending on the previous time step values of all the neurons.

$$v_i(t) = \sigma\big(\boldsymbol{W}[\boldsymbol{i},:] \times \boldsymbol{v}(t-1)\big) \tag{2}$$

where $\boldsymbol{v} \in \{1,-1\}^{N\times 1}$, $\boldsymbol{W} \in \{1,-1\}^{N\times N}$, v_i denotes i^{th} neuron, $\boldsymbol{W}[\boldsymbol{i},:] \in \{1,-1\}^{1\times N}$ denotes i^{th} row of $\boldsymbol{W}$, t denotes the time step or iteration in this case and σ denotes the sign function :

$$\sigma(x) = \begin{cases} \frac{x}{|x|}, & \text{if } x \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

For all the below experiments, the hyperparameters used are :

Method	No. of Iterations	No. of Neurons
Synchronous Update	5	9000
Asynchronous Update	100000	9000

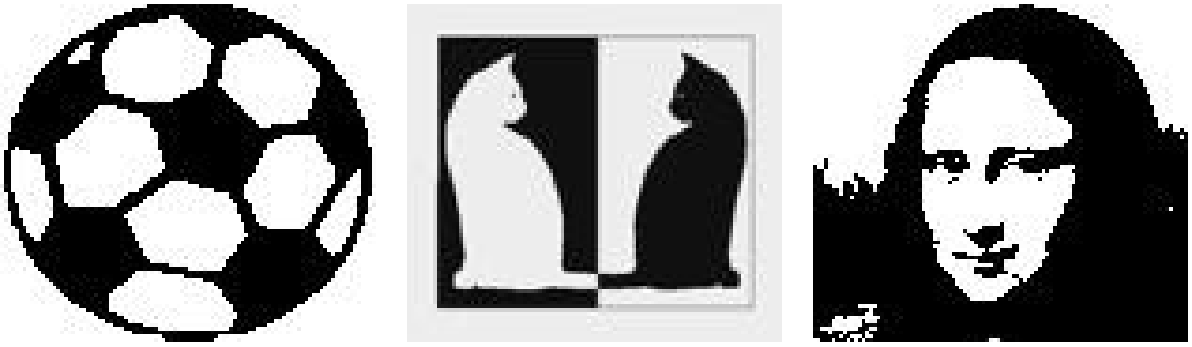
The flattened input triggers are used to initialise $\boldsymbol{v}$.

In order to avoid ambiguity while reading from the image file and ensure all elements are from $\{-1,0,1\}$, corresponding text files have been saved for original images as well as image patches which are read from inorder to store the patterns in the Hopfield Network.

3 Question 1

Image Visualization

The visualised patterns are :



NOTE : The code for visualization can be found in *"visualise.py"*

The code for Hopfield network with N=9000 neurons which are fully connected can be found in :

- *"Discrete/Single Pattern/hopfield_ ball.py"*
- *"Discrete/Multiple Pattern/hopfield_ multi.py"*
- *"Discrete/Multiple Pattern/hopfiled_ weight_ noise.py"*

4 Question 2

4.1 Single pattern stored

The image of the ball is saved in the network.

The code for Hofield Network can be found in *"Discrete/Single Pattern/hopfield_ ball.py"*.

4.2 Input Trigger

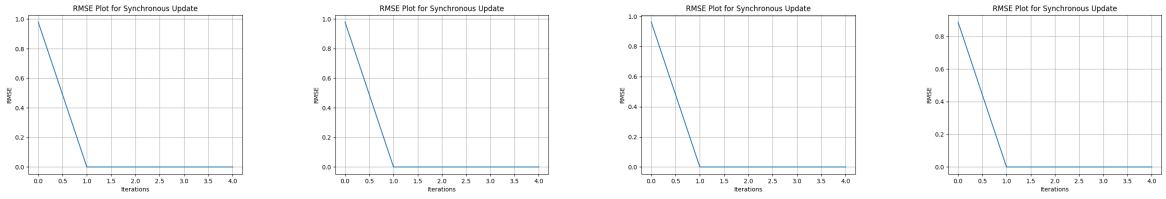
The input triggers are given by :

NOTE : The code forgenerating the input triggers can be found in *"Discrete/Single Pattern/generate_ patch.py.py"*

4.3 RMS Error and Pattern Retrieval

4.3.1 Synchronous Updates

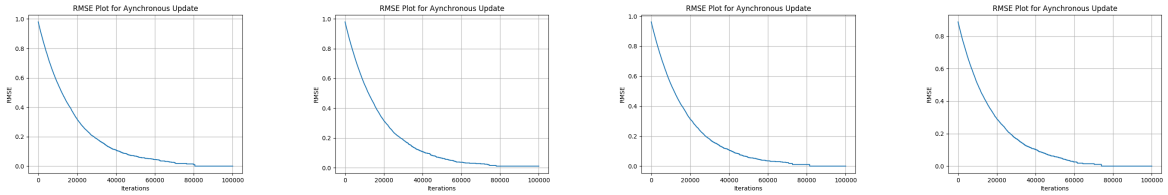
The corresponding RMS errors are given by :



The corresponding reconstructions are given by :

4.3.2 Asynchronous Updates

The corresponding RMS errors are given by :



The corresponding reconstructions are given by :

5 Question 3

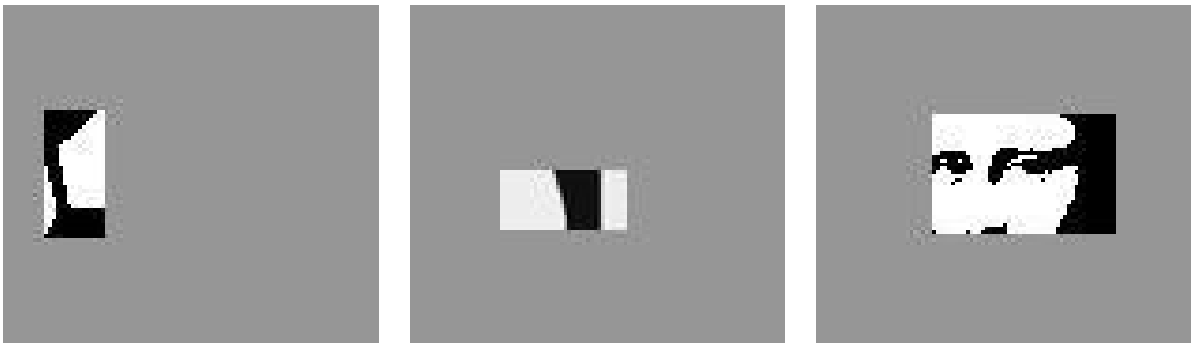
5.1 Multiple patterns stored

The image of the ball, cat and monalisa are saved in the network.

The code for Hofield Network can be found in *"Discrete/Multiple Pattern/hopfield_multi.py"*.

5.2 Input Trigger

The input triggers for *ball*, *cat* and *Monalisa* are given by :

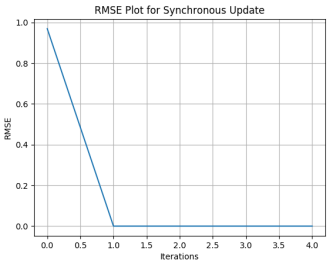


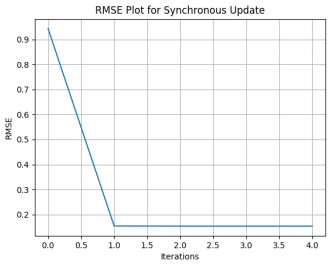
NOTE : The code for generating the input triggers can be found in *"Discrete/Multiple Pattern/generate_patch.py"*.

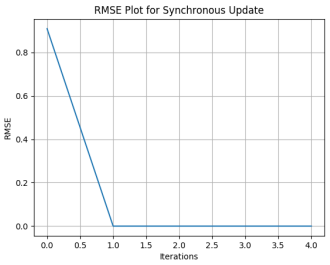
5.3 RMS Error and Pattern Retrieval

5.3.1 Synchronous Updates

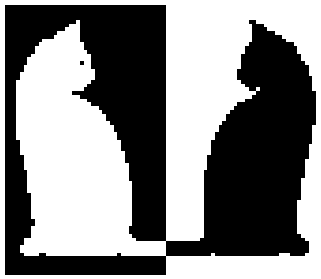
The corresponding RMS errors are given by :





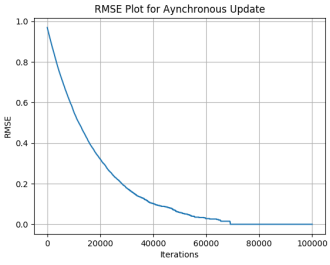


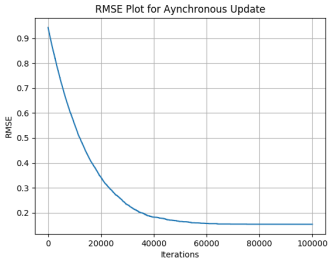
The corresponding reconstructions are given by :

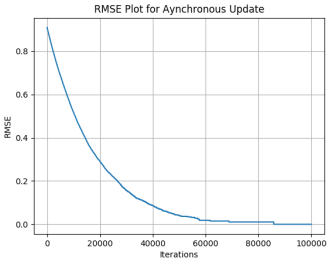




5.3.2 Asynchronous Updates

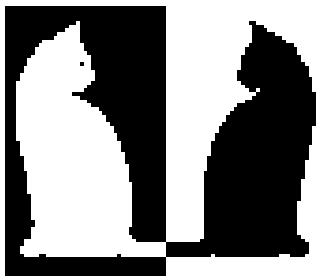
The corresponding RMS errors are given by :







The corresponding reconstructions are given by :


5.4 Corrupting the weights that store the patterns

X% of weights are made to be zero (randomly).

The input triggers for *ball*, *cat* and *Monalisa*, used for the purpose are given by :

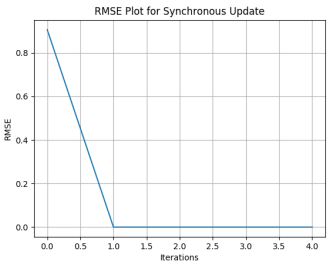
NOTE : The code for generating the input triggers can be found in *"Discrete/Multiple Pattern/generate_patch.py"*.

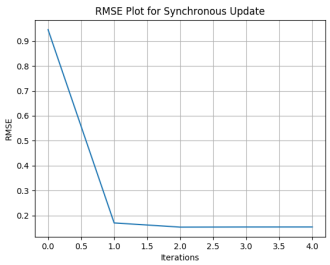
For generation of random X% of noise, the following snippet has been utilised :

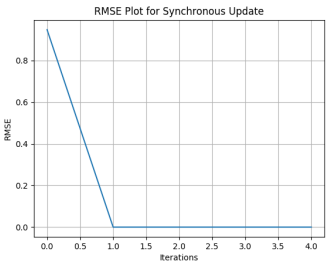
- Consider $\boldsymbol{W} = \begin{bmatrix} 1 & 1 & -1 \\ -1 & -1 & -1 \\ -1 & 1 & 1 \end{bmatrix}$
- X_val = int(X*N*N) gives no. of random 0s to be generated. In this case $N = 3$ and $X = 25\%$ gives X_val = 2.
- mask = np.hstack((np.ones((1, int(N*N-X_val))) , np.zeros((1,X_val)))) gives : $mask = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$
- mask = np.reshape(mask[0][np.random.permutation(N*N)], [N,N]) gives : $mask = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- W = W*mask (element-wise product) gives : $\boldsymbol{W} = \begin{bmatrix} 1 & 0 & -1 \\ -1 & -1 & 0 \\ -1 & 1 & 1 \end{bmatrix}$

5.4.1 X = 25%

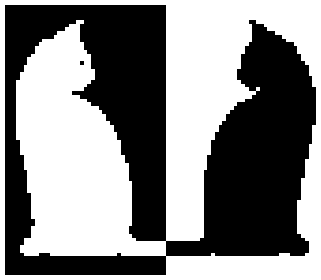
Synchronous Updates
The corresponding RMS errors are given by :





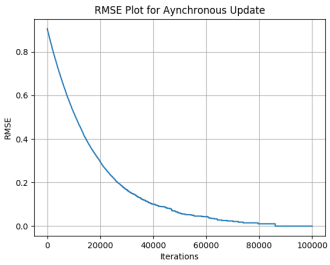


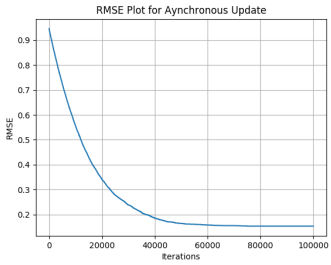
The corresponding reconstructions are given by :

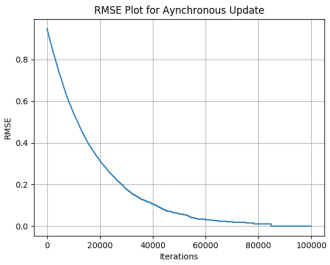




5.4.2 Asynchronous Updates

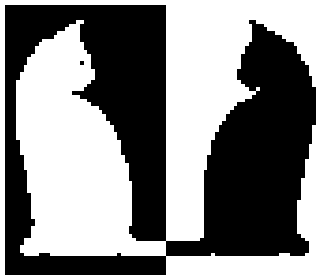
The corresponding RMS errors are given by :







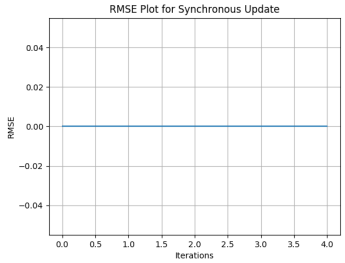
The corresponding reconstructions are given by :

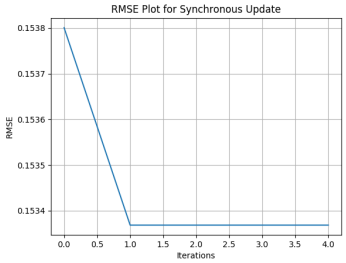



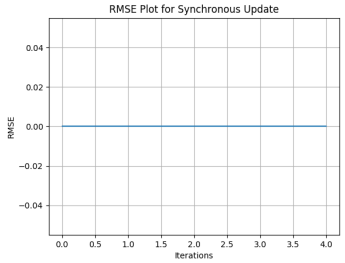

5.4.3 X = 50%

Synchronous Updates

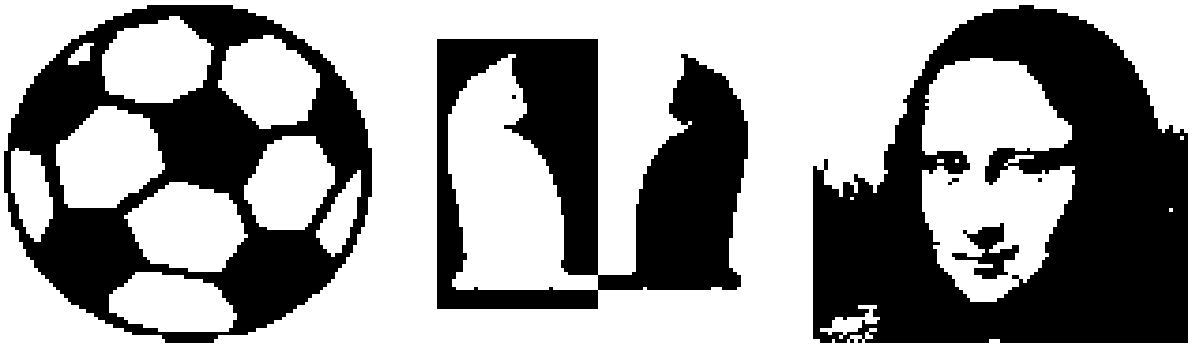
The corresponding RMS errors are given by :





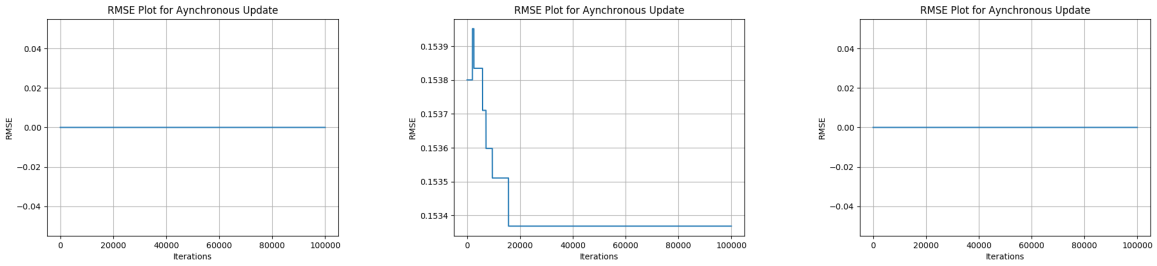


The corresponding reconstructions are given by :

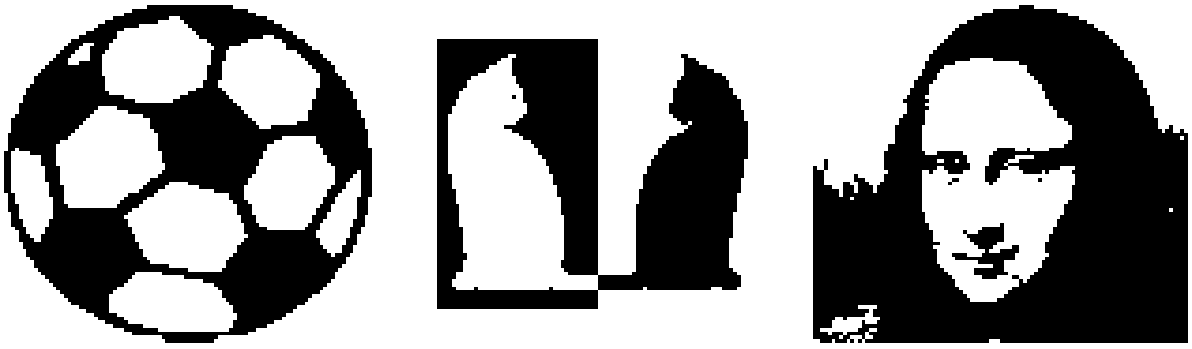


5.4.4 Asynchronous Updates

The corresponding RMS errors are given by :

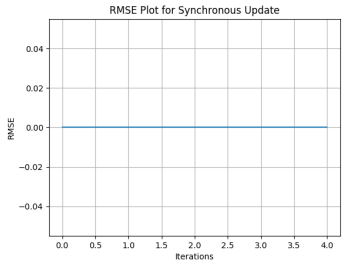


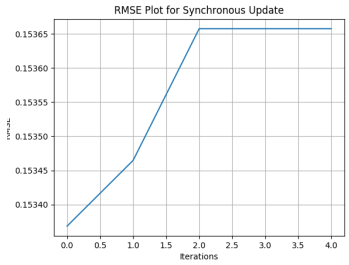
The corresponding reconstructions are given by :

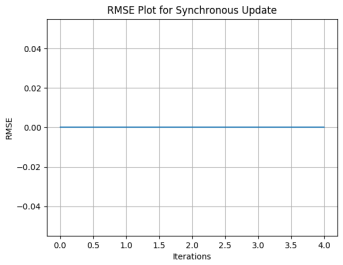


5.4.5 X = 80%

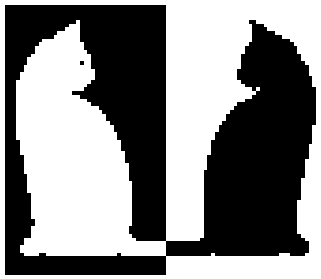
Synchronous Updates
The corresponding RMS errors are given by :





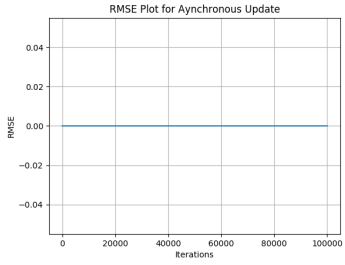


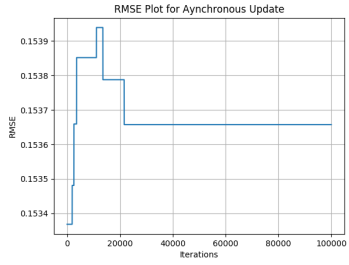
The corresponding reconstructions are given by :

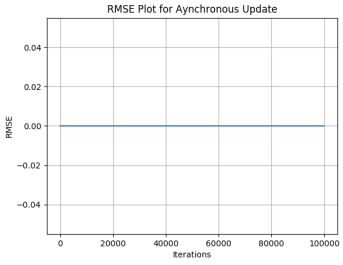




5.4.6 Asynchronous Updates

The corresponding RMS errors are given by :







The corresponding reconstructions are given by :