WARSAW UNIVERSITY OF TECHNOLOGY

FACULTY OF MATHEMATICS AND
INFORMATION SCIENCE

FIRST SEMESTER 2021/2022

NEURAL NETWORK

PROJECT 1

BY

Swetha Suresh Kumar (324353)

Andra Umoru (324334)

SUBMITTED TO:

Prof. Jacek Mańdziuk

Table of Contents

List of figures

1. Project description

This project is an implementation of program that builds and tests the Hopfield Associative Memory with the Hebb rule.

Note: All parts of the program except for GUI must be written without using external libraries (in particular the NN-related libraries).

1.1. Requirements

- **INPUT** in the form of an external file or provided in an interactive way by the user. both possibilities (external txt file and by means of GUI) need to be programmed.
- Calculation of the Hopfield Associative Memory.
- Iterative testing of any input vector: the user provides a test vector and the system outputs the HAM vector; next the user may input another vector – receive the output, etc.
- The program must allow calculation (testing) the output for ANY input vector of appropriate size not only one of the base/stored vectors
- The program must be able to handle both UNIPOLAR and BIPOLAR vectors.

NOTE that, sometimes you must iterate several times. You may end iterations ONLY in either one of the two following cases:

- In the current iteration the output vector is equal the input one → then you

print this vector as the network's response; OR

- you enter a cycle of length two → in which case you print an information

about entering a cycle and output both alternating outputs.

2. Theoretical description

2.1. Associative Memory

Hopfield networks are a type of recurrent neural network that can be utilized to perform associative memory tasks. The main purpose of associative memory networks is to connect an input with its similar pattern. To put it another way, the goal is to save and retrieve patterns.

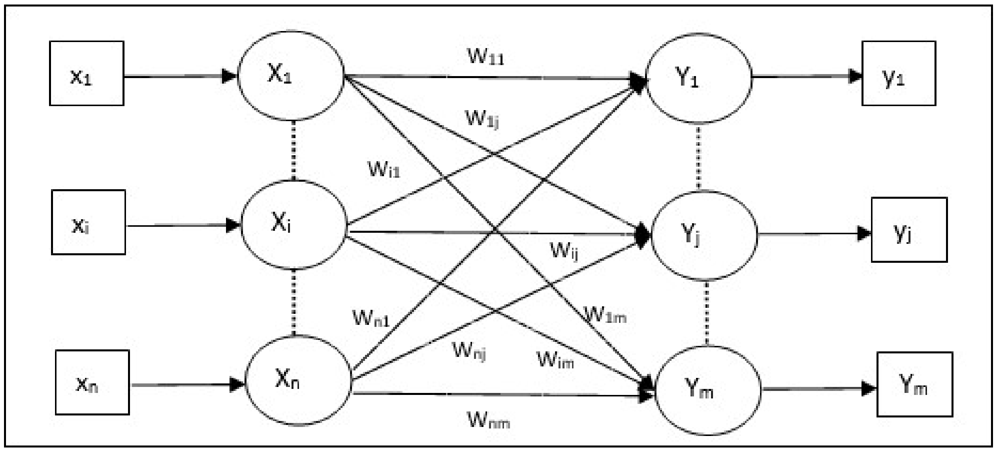


Figure 1: Associative Memory

2.2. Hopfield Network

A Hopfield network is used to implement associative memory. Memory addresses are treated as training vectors for the artificial neural network in the suggested method. The effectiveness of memory search is linked to the solution of the overfitting problem. It is proposed a strategy for breaking the training and input network vectors into pieces that require a fewer number of neurons to process. The results of a series of tests on Hopfield network models with various numbers of neurons trained with various numbers of vectors and operated under various noise circumstances are reported. The simplest associative memory is just a sum of outer products of the N patterns {xi}Ni=1 that we want to store (Hebbian learning rule). In classical Hopfield Networks these patterns are polar (binary), i.e. xi {−1,1}d ∈, where d is the length of the patterns. The corresponding weight matrix W is:

$$\mathbf{W} = \sum_i^N \boldsymbol{x_i}\ \boldsymbol{x_i^T}$$

The weight matrix W stores the patterns, which can be retrieved starting with a state pattern.

2.3. Hebb Rule

For a given input vector, the input is said to be stable when signum function of the product of the weight matrix and the input vector is same as the input vector.

That is V is stable if sign (T x V) = V.

Another possibility is a cycle of length 2:

Sign (T x V) = V' and

Sign (T X V') = V

Where T is the weight matrix and V is the input Vector.

3. Algorithm

3.1. Variables

In this section, we describe all variables used in this implementation.

- V_i: represents a single neuron. Is binary, with 1 representing firing and -1 not firing.
- V = $[v_1, v_2.., v_n]^T$: represents the entire neural network, in vector form.
- X: Number of input vectors V_i .
- w_{ij} : represents the strength or weight of a connection between neurons V_i and V_j .This is determined with Hebbian Learning.
- W: the weights put together in matrix form. Note that w_{ii} = 0 and this matrix is symmetric.
- U_i : The fixed threshold determining neuron state. Unless otherwise stated, U_i = 0.
- N_i = $[v_i, \dots, v_n]^T$: a new training input pattern where vi is the state of neuron i of the new pattern and vi is either 1 or -1.

3.2. Hopfield Network Training Algorithm

- **Storage (learning)**

  In the learning step of Hopfield network, we need to find the weight matrix for X of patterns (fundamental memories: $[v_1, v_2.., v_x]$ stored in the synaptic weights of the network according to the equation:

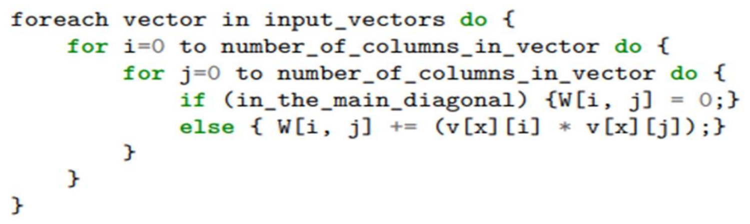
$$W_{ij} = \begin{cases} \sum_{x=1}^{X} v_{x,i} v_{x,j} & i \neq j \\ 0 & \text{i=j} \end{cases}$$

$v_{x,i}$ and $v_{x,i}$, i_{th} and j_{th} elements of the fundamental memories V_X in matrix form:

$$W = \begin{vmatrix} 0 & w_{12} & \dots & w_{1i} & \dots & w_{1n} \\ w_{21} & 0 & \dots & w_{2i} & \dots & w_{2n} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ w_{i1} & w_{i2} & \dots & 0 & \dots & w_{in} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ w_{n1} & w_{n2} & \dots & w_{ni} & \dots & 0 \end{vmatrix}$$

Where $w_{ij} = w_{ji}$, The weight matrix W is remains fixed.

To implement the equation 1, we adopted the following algorithm:



```
foreach vector in input_vectors do {
    for i=0 to number_of_columns_in_vector do {
        for j=0 to number_of_columns_in_vector do {
            if (in_the_main_diagonal) {W[i, j] = 0;}
            else { W[i, j] += (v[x][i] * v[x][j]);}
        }
    }
}
```

In the above figure, to create the initial wight matrix w of size i * j for each vector V we have used i * j number of iteration for each vector iteration. When the last vector is reached, the matrix W with all, final and correct w_{ij} is finalized.

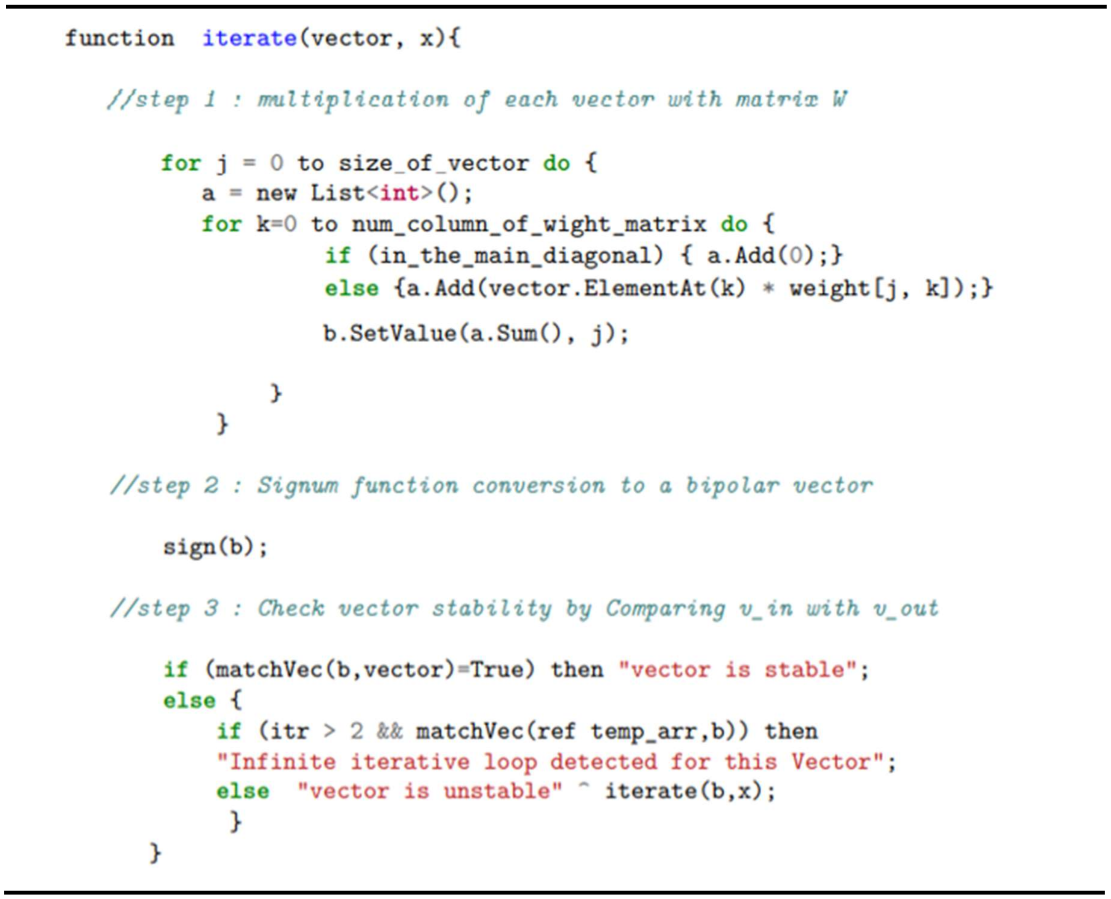
3.3. Testing Vector's Stability

Given an input memory or pattern, called V_{in} , Then, using the equation below to iterate through all vectors input for updating.

$$\boldsymbol{V_i^n} = \begin{cases} \boldsymbol{-1}, & \boldsymbol{if \sum_{i \neq 1} W_{ij}\, V_j > U_i} \\ \boldsymbol{1}, & \boldsymbol{otherwise} \end{cases}$$

where U_i = 0 unless otherwise stated. This process continues, with synchronous update of neurons, until Vin reaches a learned memory/pattern, which will be a stable point. At this point, it can be considered the output memory which the Hopfield Network has identified and recovered.

To implement the above equation, we utilized the following algorithm:



```
function  iterate(vector, x){

  //step 1 : multiplication of each vector with matrix W

      for j = 0 to size_of_vector do {
         a = new List<int>();
         for k=0 to num_column_of_wight_matrix do {
                  if (in_the_main_diagonal) { a.Add(0);}
                  else {a.Add(vector.ElementAt(k) * weight[j, k]);}

                  b.SetValue(a.Sum(), j);

              }
          }

  //step 2 : Signum function conversion to a bipolar vector

      sign(b);

  //step 3 : Check vector stability by Comparing v_in with v_out

      if (matchVec(b,vector)=True) then "vector is stable";
      else {
          if (itr > 2 && matchVec(ref temp_arr,b)) then
          "Infinite iterative loop detected for this Vector";
          else  "vector is unstable" ^ iterate(b,x);
           }
      }
```

Figure 2: Algorithm for testing the stability of the vectors

3.4. Convergence

Present an unknown n-dimensional vector, N, (corrupted or incomplete version of a pattern from fundamental memories) to the network and retrieve a stored association (stable state). To get the convergence vector for specific test vector, we utilized the same algorithm used in subsection 3.3.

4. Application Manual

This application was developed using C#. It can be executed by opening the file in Visual studio 2019.

The following steps should be followed for the expected result.

⇨ Click start button to execute the program

⇨ The application will pop up GUI of Hopfield Association Memory Simulation

- ⇨ The user can enter the values of vector in the input phase of GUI. This can be done in two ways as mentioned below,
    - ▪ Click Add vector button to add the values of the vectors
    - ▪ Click Read from a file button to read a value of vectors in a text file
- ⇨ To reset the values in input phase, click Reset button in the input phase
- ⇨ After adding the values of the vectors, click Train pattern button to get the weight matrix
- ⇨ Then Click Test stability button to check the stability of each vector
- ⇨ In the test convergence phase, Enter the test vector but using the -1,0,1 button
- ⇨ After entering the test vectors, click Test convergence button to test the stability of unknown vector
- ⇨ If you want to clear all the input and output, click Reset all button in the program control phase to reset everything
- ⇨ To close the GUI, click Exit button

5. Result

5.1. GUI

Figure 1 show the user interface of this implementation.

- Input phase

  We have two ways of getting the value of vectors as input

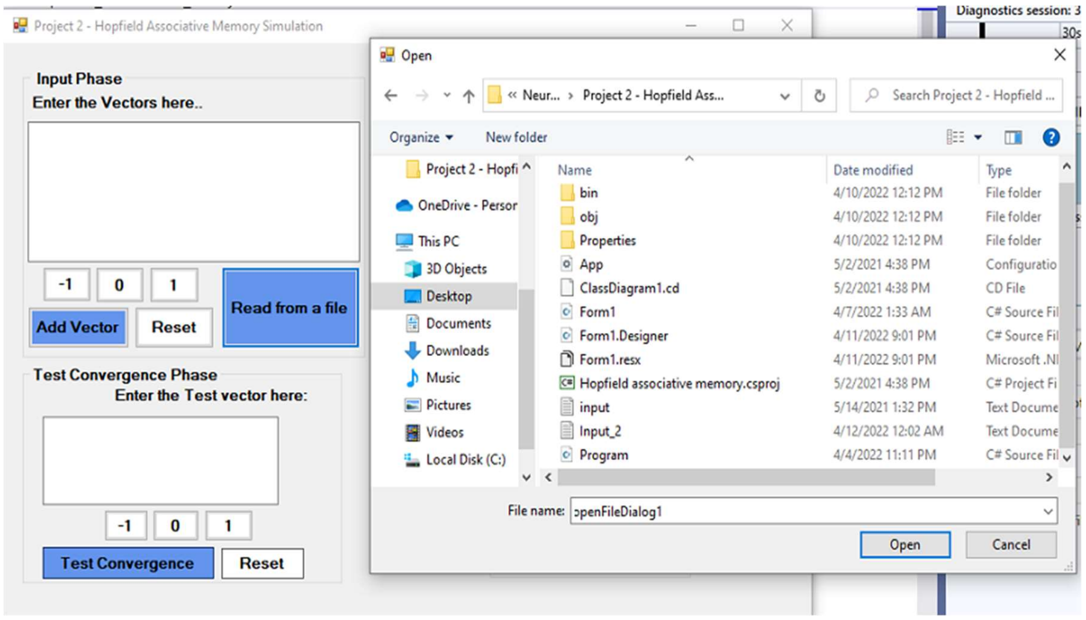
    - ▪ **Manually entering the value of the vector by using the buttons given**
    - ▪ **Reading a file which consists of the vectors from the user**

- Training phase

  The Train_patterns button is used to generate the weight matrix using the input vectors from the input phase. After generating the weight matrix, the Test_stability button is used to check the stability of each vector and identify the stable state of specified vector in each iteration.

- Convergence phase

  The Test_Convergence button is used to test the stability of unknown vector and retrieve the vector a stored association (stable state).



Name	Date modified	Type
bin	4/10/2022 12:12 PM	File folder
obj	4/10/2022 12:12 PM	File folder
Properties	4/10/2022 12:12 PM	File folder
App	5/2/2021 4:38 PM	Configuratio
ClassDiagram1.cd	5/2/2021 4:38 PM	CD File
Form1	4/7/2022 1:33 AM	C# Source Fil
Form1.Designer	4/11/2022 9:01 PM	C# Source Fil
Form1.resx	4/11/2022 9:01 PM	Microsoft .NI
Hopfield associative memory.csproj	5/2/2021 4:38 PM	C# Project Fi
input	5/14/2021 1:32 PM	Text Docume
Input_2	4/12/2022 12:02 AM	Text Docume
Program	4/4/2022 11:11 PM	C# Source Fil

File name: openFileDialog1

Figure 3: Input from external file

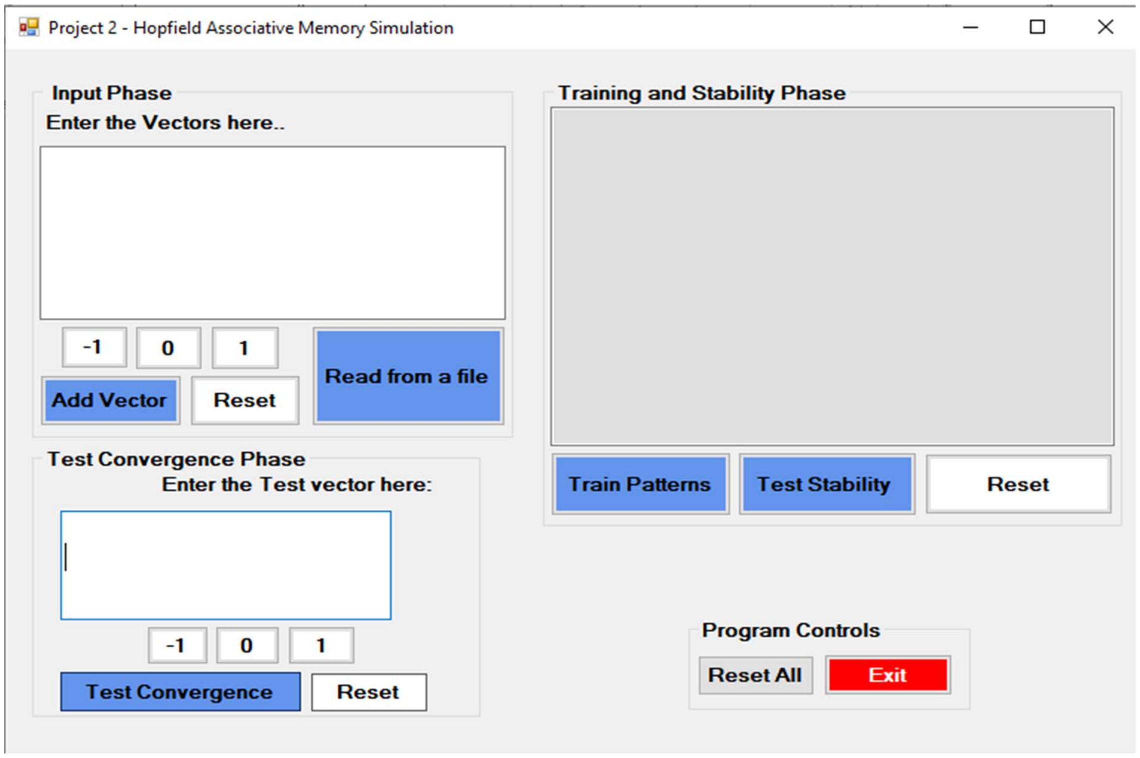
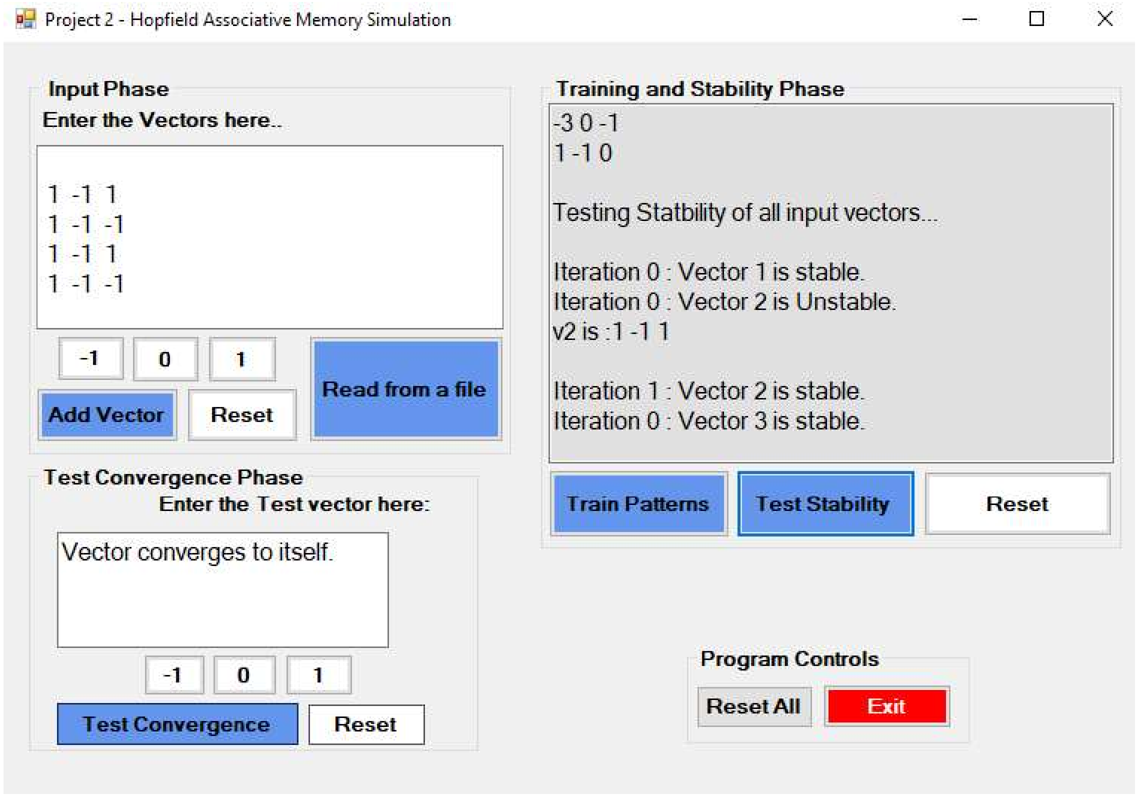


Figure 4: GUI of Hopfield Association Memory Simulation

5.2. Experimental Results

In this subsection, we introduced some output of this implementation using different size of input vectors.



```
1 -1 1
1 -1 -1
1 -1 1
1 -1 -1
```

```
-3 0 -1
1 -1 0

Testing Statbility of all input vectors...

Iteration 0 : Vector 1 is stable.
Iteration 0 : Vector 2 is Unstable.
v2 is :1 -1 1

Iteration 1 : Vector 2 is stable.
Iteration 0 : Vector 3 is stable.
```

```
Vector converges to itself.
```

Figure 5: Inputting, Training and testing of Vectors

6. Reference

- https://en.wikipedia.org/wiki/Hopfield_network
- https://ml-jku.github.io/hopfield-layers/
- https://dl.acm.org/doi/abs/10.1134/S0361768820050023
- https://page.mi.fu-berlin.de/rojas/neural/chapter/K13.pdf
- https://www.emis.de/journals/GM/vol13nr2/popovici/popovici.pdf