Bidirectional Associative Memory (BAM)

COSC 4P80 Assignment 1

Krish Patel
6949671 kp19hs@brocku.ca

***Abstract*—This report investigates the behaviour of a Bidirectional Associative Memory (BAM) network. A BAM is a type of recurrent neural network that stores associations between two sets of vectors and can recall one set from the other in both directions. The experiments performed in this study included: constructing a weight matrix using Hebbian learning from three vector pairs, testing recalls from both sets, measuring crosstalk and global crosstalk across the stored vectors, analyzing saturation effects when too many associations are stored, and testing the network under noise through random mutations with a 20% mutation rate. Results confirm that the BAM successfully recalls stored associations in most cases, but generates errors due to crosstalk and saturation when capacity is exceeded.**

I. Introduction

Bidirectional Associative Memory (BAM) is a heteroassociative neural network. Unlike autoassociative memories such as Hopfield networks, BAM stores pairs of patterns (A, B) such that recalling one pattern retrieves its partner. The BAM is implemented by constructing a weight matrix

$$W = \sum_i A_i B_i^T$$

which encodes associations between input space A and output space B. The goals of this project were:

1) To implement a BAM using three given pairs of binary patterns encoded as $\{-1, 1\}$.
2) To test recall from set A to B, and from set B to A.
3) To compute crosstalk for each pattern and analyze global crosstalk.
4) To investigate the effect of saturation by adding more vectors than the network can reliably store.
5) To study the ability of BAM to correct errors when inputs are randomly mutated with 20% noise and measure performance in terms of Hamming distance.

II. Experimental Results

A. Weight Matrix and Recall

The weight matrix was computed using the three given vector pairs. The following table shows the recall results in both directions. The second association was recalled incorrectly due to crosstalk effects. This happens because there are overlaps from other associations due to the vectors not being completely orthogonal.

Association	Expected Recall	Actual Recall
$A_1 \to B_1$	$[1, 1, -1, 1]$	$[1, 1, -1, 1]$
$A_2 \to B_2$	$[1, -1, -1, -1]$	$[-1, -1, 1, -1]$
$A_3 \to B_3$	$[-1, -1, 1, 1]$	$[-1, -1, 1, 1]$
$B_1 \to A_1$	$[-1, 1, 1, 1, -1]$	$[-1, 1, 1, 1, -1]$
$B_2 \to A_2$	$[-1, -1, -1, -1, 1]$	$[-1, -1, -1, -1, 1]$
$B_3 \to A_3$	$[-1, -1, -1, 1, 1]$	$[1, -1, -1, 1, 1]$

TABLE I
RECALL RESULTS FOR PART A.

B. Crosstalk

Crosstalk was measured by multiplying each vector A_i with the weight matrix W and comparing the result against its intended output. The global crosstalk, derived from adding up the magnitude of all values in the vectors, was equal to 40.

Vector in A	Crosstalk Value
A_1	[-2, 4, 2, 2]
A_2	[-6, -6, 6, 0]
A_3	[2, -4, -2, -4]

TABLE II
CROSSTALK FOR EACH VECTOR IN A.

C. Saturation

Additional vectors were added beyond the original three. When the number of stored associations increased, the recall performance degraded.

Number of Vectors Stored	Global Crosstalk
3 (original)	40
4	54
5	64
6	88
7	96

TABLE III
CROSSTALK VALUES AS ADDITIONAL VECTORS WERE ADDED (SATURATION TEST).

When analyzing the growth of crosstalk as new vectors were added, I observed a steady increase in crosstalk. With the original three vectors (Part A), global crosstalk was 54, and recall performance was stable aside from one minor mismatch. Adding the fourth vector raised crosstalk to 64, and although most recalls were correct, incorrect associations started to appear. After introducing the fifth vector, crosstalk increased sharply to 88, and recall performance degraded significantly. Finally, with the sixth and seventh vectors included, crosstalk rose to 96, and almost every recall attempt failed. This shows that saturation happened when the fifth vector was added.

Run	Original (O)	Mutated (M)	Hdist(O,M)	Corrected (C)	Hdist(O,C)
1	[-1, 1, 1, 1, -1]	[-1, 1, 1, 1, -1]	0	[-1, 1, 1, 1, -1]	0
2	[-1, 1, 1, 1, -1]	[1, 1, 1, 1, 1]	2	[1, 1, 1, 1, -1]	1
3	[-1, -1, -1, 1, 1]	[-1, -1, 1, 1, 1]	1	[1, -1, -1, 1, 1]	1
4	[-1, -1, -1, 1, 1]	[1, -1, -1, -1, 1]	2	[1, -1, -1, -1, 1]	2
5	[-1, 1, 1, 1, -1]	[1, 1, -1, 1, -1]	2	[1, 1, 1, 1, -1]	1
6	[-1, -1, -1, 1, 1]	[-1, -1, -1, 1, 1]	0	[1, -1, -1, 1, 1]	1
7	[-1, -1, -1, -1, 1]	[1, -1, -1, -1, 1]	1	[1, -1, -1, -1, 1]	1
8	[-1, 1, 1, 1, -1]	[-1, 1, 1, 1, 1]	1	[-1, 1, 1, 1, -1]	0
9	[-1, -1, -1, -1, 1]	[-1, 1, -1, 1, 1]	2	[1, -1, -1, 1, 1]	2
10	[-1, 1, 1, 1, -1]	[-1, 1, 1, 1, -1]	0	[-1, 1, 1, 1, -1]	0
11	[-1, 1, 1, 1, -1]	[1, 1, 1, 1, -1]	1	[-1, 1, 1, 1, -1]	0
12	[-1, 1, 1, 1, -1]	[-1, 1, 1, 1, -1]	0	[-1, 1, 1, 1, -1]	0
13	[-1, -1, -1, -1, 1]	[-1, -1, -1, -1, 1]	0	[1, -1, -1, -1, 1]	1
14	[-1, -1, -1, 1, 1]	[1, -1, 1, 1, 1]	2	[1, -1, -1, 1, 1]	1
15	[-1, -1, -1, -1, 1]	[-1, -1, -1, -1, 1]	0	[1, -1, -1, -1, 1]	1
16	[-1, 1, 1, 1, -1]	[-1, 1, 1, -1, 1]	2	[1, 1, 1, -1, -1]	2
17	[-1, -1, -1, -1, 1]	[-1, -1, -1, -1, -1]	1	[-1, -1, -1, -1, 1]	0
18	[-1, -1, -1, -1, 1]	[-1, 1, 1, -1, -1]	3	[-1, 1, 1, 1, -1]	4
19	[-1, 1, 1, 1, -1]	[1, 1, 1, -1, -1]	2	[1, 1, 1, -1, -1]	2
20	[-1, -1, -1, -1, 1]	[-1, 1, 1, 1, -1]	4	[-1, 1, 1, 1, -1]	4

TABLE IV
20% MUTATION RATE.

D. *Noise and Error Correction*

Across 20 runs, most mutations were corrected perfectly ($H(O,C) = 0$) or partially corrected ($H(O,C) < H(O,M)$). In some cases, the correction worsened the input ($H(O,C) > H(O,M)$), caused by the mutation pushing the vector into the pattern of a different vector. These results show the uncertain nature of BAM: error correction is not guaranteed, but the network often succeeds when mutations are small.

CONCLUSION

From these experiments, we learned that:

1) BAM can reliably recall stored associations in both directions when the number of patterns is within capacity.
2) Crosstalk naturally arises as more vectors are stored, representing interference between overlapping patterns.
3) Saturation limits storage capacity: once too many vectors are stored, recall becomes unreliable.
4) Error correction under noise is possible: BAM often restores mutated inputs back to their original state when the mutation is small. However, large mutations can lead to miscorrection, where the network converges to the wrong pattern.

Overall, this experiment demonstrated both the strengths and weaknesses of Bidirectional Associative Memory. BAM is powerful as a heteroassociative memory system but is constrained by vector orthogonality, storage capacity, and crosstalk.