Department of Computer Science
COSC 4P80 - Artificial Neural Networks

Examination of the B.A.M Artificial Intelligence Model

Prepared by: Kevin Olenic, ST#: 6814974
Instructor: Dave Bockus
Date: February 3, 2023

Abstract

As computer hardware and software become more advance, they can run complex artificial neural networks more efficiently and with more complex data. This paper will explore the workings of a Bidirectional Associative Memory (B.A.M) implementation of artificial intelligence by performing forward and backward encryption, the padding on the output to stabilize the B.A.M and examine its ability to error correct inputs that have undergone mutation. This report on the B.A.M will provide details into the workings of the B.A.M artificial intelligence system and discuss how it memorizes and corrects data.

Contents

1 Introduction **1**
1.1 Goal of this Report 1
1.2 Bidirectional Associative Memory (B.A.M) 1
1.3 Supervised Learning 3

2 Experimental Research **3**
2.1 Assign Part A (Available Memory) 3
2.2 Assign Part B (Filled Memory) 5
2.3 Assign Part C (Padding) 7
2.4 Assign Part D (Error Correction) 9

3 Conclusion **12**

1 Introduction

In this section, the goal of this paper and relevant topics such as bidirectional associative memory and supervised learning are explained to improve the reader's understanding of the concepts covered in this report.

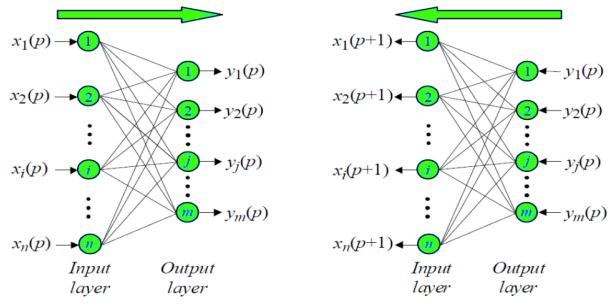
1.1 Goal of this Report

This report tests the B.A.M artificial intelligence system by experimenting with the volume of training data, the effects of padding on the training data and the effectiveness of the B.A.Ms error-correcting abilities. These experiments will provide an insight into the functionality of the B.A.M system and its ability to memorize and error-correct patterns.

1.2 Bidirectional Associative Memory (B.A.M)

Bidirectional associative memory is a recurrent neural network proposed in the early 1980s by Bart Kosko. It is a supervised training model containing two layers of memory capable of encoding patterns and performing pattern recognition operations.[1]

B.A.M uses a form of Hebbian learning, wherein the inputs generate a correlation matrix of weights connecting inputs to outputs and vice versa. Thus in the case of data loss or corruption, only half of the data set is needed as the B.A.M can find the associated data for even incomplete or distorted input data. Figure 1.1 depicts B.A.M.'s capability to determine the output from the input and vice versa, as shown in the image when the B.A.M receives a vector of size n from the set of input data, it outputs a vector of size m from the outputs and given a vector of size m from the outputs it outputs a vector of size n from the inputs, thus making it a bidirectional memorization model.[2]



(a) Forward direction. (b) Backward direction.

Figure 1.1: B.A.M associating in both directions

This artificial intelligence has a simple design, it consists of only two layers, the input layer and the output layer. The input layer of the B.A.M represents the number of neurons in the model, whereas the output layer represents the B.A.M's memory capacity.[2] Figure 1.2 depicts a B.A.M model with four neurons, the number of input nodes, and a memory size of three, the number of output nodes.

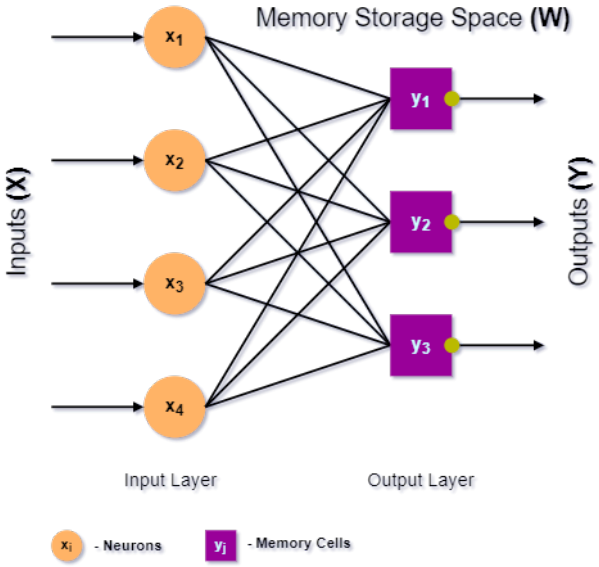


Figure 1.2: B.A.M with 4 neurons and memory capacity of 3

The B.A.M artificial intelligence possesses hetero-associative memory, meaning it can store related data regardless of its type and has a structure that does not need to perform continuous learning. B.A.M models are also efficient when used as a part of a decision-making process, as they can reason the answer to a specific data examination issue founded on various associations of multiple interrelated data sets.

The weight matrix (W) for the B.A.M is calculated by taking the sum of the products between the inputs and the outputs, represented as two bipolar matrices, meaning there are comprised entirely of either a one (1) or a negative one (-1). To calculate the matrix corresponding to an input (A) and the associated output (B), take A and multiply it against the transpose of B. This process is represented by the following equation:

$$W = \sum_{n=1}^{m} A_k B_k^t \tag{1}$$

The constructed weight matrix (W) provides the connections between the inputs and outputs and is used to calculate the associated data of a given input (A) or output (B) since the generated weight matrix is bidirectional, meaning we can feed either the input (A) to get the output (B) or vice versa. We can use the matrix to calculate the associated data of input by taking the desired outputs associated data and calculating its resulting matrix from being multiplied against the generated weight matrix. Then each element in the produced matrix is either changed to a negative one if its value is less than zero or is changed to a positive one if its value is zero or greater. This is outlined in the equations below:

$$\begin{aligned} B_k &= W^t A_k \\ A_k &= W B_k \end{aligned} \tag{2}$$

The below equation represents the threshold function used on the generate data associated

with the information entered into the B.A.M:

$$F(x) = \begin{cases} 1, x \geq 0 \\ -1, x < 0 \end{cases} \tag{3}$$

1.3 Supervised Learning

Supervised learning is a technique for training artificial intelligence systems involving input data associated with a designated output.

This learning technique teaches the artificial intelligence system by feeding it input and calculating the error generated (i.e. how much of a difference between the generated output and the actual output exists). By computing this value and applying it to the weight matrix, the learning technique allows the a.i to adjust the values and improve the results of future inputs.

Differing from unsupervised learning, which involves training the system through the use of input data not assigned to any specific output, but instead must use the data provided to determine what data belongs to which category.

2 Experimental Research

In this section, multiple experiments are performed, examining the B.A.M.'s abilities to memorize and error-correct a set of patterns, each consisting of either one or negative one, with each investigation discussing why the B.A.M. produced these results.

2.1 Assign Part A (Available Memory)

The first experiment will involve developing a B.A.M. capable of encoding pattern pairs and can associate data from both pattern directions (i.e. output from input and vice versa). The developed B.A.M. will have a memory capacity of four but shall receive only three sets of patterns to memorize, each consisting of an input and an associated output for training. Upon completion of training, the B.A.M. is tested, thus determining its ability to associate data for a particular input/output.

Below is the Java code developed to train the neural network and generate the input and output from the developed weight matrix for this reports B.A.M:

```
public static void train(int[] input, int[] output){
        for(int y = 0; y < input.length; y++){
            for(int x = 0; x < output.length; x++){
                weightMatrix[y][x] += input[y] * output[x];
            }
        }
```

Listing 2.1: Java Implementation for B.A.M Weight Matrix Generator

```
private static int[] test(int[] input, boolean error){
    int[] result;
    int position = 0;
    if(input.length == weightMatrix.length) {// input length
    ↪  equals y-length of W
        result = new int[weightMatrix[0].length];// result will
        ↪  have length of W x-axis
        for (int x = 0; x < weightMatrix[0].length; x++) {// for
        ↪  each column of W
            int sum = 0;// reset the sum for the xth enrry in
            ↪  the result
            for (int y = 0; y < weightMatrix.length; y++) {//
            ↪  for each row
                sum +=  weightMatrix[y][x] * input[y];//
                ↪  calculate value of result matrix
            }
            if (sum >= 0) result[position] = 1;// apply
            ↪  threshold function to result
            else result[position] = -1;
            position++;// move to next position in result
        }
    }
    else{// we are generating an input
        result = new int[weightMatrix.length];
        for (int[] matrix : weightMatrix) {// for each row
            int sum = 0;// reset the sum
            for (int x = 0; x < weightMatrix[0].length; x++) {//
            ↪  for each column
                sum += matrix[x] * input[x];
            }
            if (sum >= 0) result[position] = 1;// apply
            ↪  threshold function to result
            else result[position] = -1;
            position++;// move to next position in result
        }
    }

   if(!error){// if we are not performing an error test print
    ↪  input and resulting matrix
       System.out.println("Input: " + Arrays.toString(input));//
       ↪  print input
       System.out.println("Output: " + Arrays.toString(result) +
       ↪  "\n");// print output
   }
    return result;// return result of test method
}//test
```

Listing 2.2: Java Implementation for B.A.M test method

These are the three data sets used to train and test the developed B.A.M systems ability to memorize patterns:

Table 1: Part A training data

Input	**Output**
-1, 1, 1, 1, -1	1, 1, -1, 1
-1, -1, -1 ,-1, 1	1, -1, -1, -1
-1, -1, -1, 1, 1	-1, -1, 1, 1

Below is the resulting output from the test performed on the trained B.A.M system using the above data:

Table 2: Part A output data

Input	**Output**
-1, 1, 1, 1, -1	1, 1, -1, 1
-1, -1, -1, -1, 1	-1, -1, 1, -1
-1, -1, -1, 1, 1	-1, -1, 1, 1
1, 1, -1, 1	-1, 1, 1, 1, -1
1, -1, -1, -1	-1, -1, -1, -1, 1
-1, -1, 1, 1	1, -1, -1, 1, 1

Based on the above output from the bidirectional association memory a.i, it is clear it can correctly associate most of the data. However, for the second and eighth inputs, it did not produce the correct association revealing an issue with the weight matrix generated from the above training data.

This issue is most likely due to the training data we provided the B.A.M. Since this report uses input with an of size five and an output of size four since this B.A.M has a topology of a 5x4 matrix, meaning it is capable of memorizing a maximum of four patterns. However, the B.A.M in this experiment gives only three to memorize. Thus the B.A.M was insufficiently trained and therefore contains issues in its weight matrix, causing it to associate data incorrectly or produce outputs that the B.A.M should not be able to output.

2.2 Assign Part B (Filled Memory)

The second experiment involves retraining the B.A.M system using an additional data set, filling this B.A.M's pattern memory capacity. Any effects the extra data set have

on the B.A.M's ability to memorize patterns will be examined to determine if filling the B.A.M's memory aids its ability to associate data.

Below is the new training data entered into the B.A.M. program. It is the same as the data set used in the previous experiment except for the inclusion of a fourth data set:

Table 3: Part B training data

Input	**Output**
-1,1,1,1,-1	1,1,-1,1
-1,-1,-1,-1,1	1,-1,-1,-1
-1,-1,-1,1,1	-1,-1,1,1
1,1,1,1,1	-1,1,1,-1

Below is the resulting output from adding the additional data set:

Table 4: Part B output data

Input	**Output**
-1, 1, 1, 1, -1	1, 1, -1, 1
-1, -1, -1, -1, 1	1, -1, -1, -1
-1, -1, -1, 1, 1	-1, -1, 1, 1
1, 1, 1, 1, 1	-1, 1, 1, -1
1, 1, -1, 1	-1, 1, 1, 1, -1
1, -1, -1, -1	-1, -1, -1, -1, 1
-1, -1, 1, 1	1, -1, -1, 1, 1
-1, 1, 1, -1	1, 1, 1, 1, 1

As shown in the above data, the B.A.M. correctly associated most of the data but failed to accurately correlate the output back to its input for the seventh input. Compared to the previously performed experiment, this is an improvement as the B.A.M. had incorrectly associated two data entries. Thus by adding the additional fourth data set to the B.A.M, the error produced was reduced by half.

The reason the B.A.M correctly associated more of the data during this experiment, but not all of it, is most likely due to the addition of the fourth data set. Adding the additional data set caused the orthogonality of the training data to increase as the B.A.M gained more training data, altering its weight matrix and changing the paths between

the input and output neurons, allowing the B.A.M to associate the data for input more accurately.

2.3 Assign Part C (Padding)

For the third experiment of this report, research into the effects of padding the output data with additional information. This experiment's goal, examine the results of increasing the neurons in the B.A.M. to determine which paddings aid in its ability to associate data, thus stabilizing the system.

For this experiment several different paddings are attempted to determine which causes the B.A.M's ability to associate data correctly increases and which result causes a decrease.

The following padding's were used: Padding 1:

- 1,1,-1,1 → 1,1,-1,-1,-1,-1,1
- 1,-1,-1,-1 → 1,-1,-1,1,1,1,-1
- -1,-1,1,1 → -1,-1,1,1,-1,-1,1
- -1,1,1,-1 → -1,1,1,-1,-1,-1,-1

Table 5: Part C output data with the above outlined padding

Input	**Output**
-1, 1, 1, 1, -1	1, 1, -1, -1, -1, -1, 1
-1, -1, -1, -1, 1	1, -1, -1, 1, 1, 1, -1
-1, -1, -1, 1, 1	-1, -1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1	-1, 1, 1, -1, -1, -1, -1
1, 1, -1, -1, -1, -1, 1	-1, 1, 1, 1, -1
1, -1, -1, 1, 1, 1, -1	-1, -1, -1, -1, 1
-1, -1, 1, 1, 1, 1, 1	-1, -1, -1, 1, 1
-1, 1, 1, -1, -1, -1, -1	1, 1, 1, 1, 1

Padding 2:

- 1,1,-1,1 → 1,1,1,1,1,-1,1
- 1,-1,-1,-1 → 1,-1,-1,-1,-1,-1,-1

- -1,-1,1,1 → -1,-1,1,1,1,1,-1
- -1,1,1,-1 → -1,1,1,1,1,1,-1

Table 6: Part C output data with the above padding

Input	**Output**
-1, 1, 1, 1, -1	1, 1, 1, 1, 1, -1, 1
-1, -1, -1, -1, 1	1, -1, -1, -1, -1, -1, -1
-1, -1, -1, 1, 1	-1, -1, 1, 1, 1, 1, -1
1, 1, 1, 1, 1	-1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, -1, 1	-1, 1, 1, 1, -1
1, -1, -1, -1, -1, -1, -1	-1, -1, -1, -1, 1
-1, -1, 1, 1, 1, 1, -1	1, 1, 1, 1, 1
-1, 1, 1, 1, 1, 1, -1	1, 1, 1, 1, 1

Padding 3:

- 1,1,-1,1 → 1,1,-1,-1,1,1,1
- 1,-1,-1,-1 → 1,-1,-1,1,-1,1,-1
- -1,-1,1,1 → -1,-1,1,1,1,1,1
- -1,1,1,-1 → -1,1,1,-1,-1,-1,-1

Table 7: Part C output from the above padding

Input	**Output**
-1, 1, 1, 1, -1	1, 1, -1, -1, 1, 1, 1
-1, -1, -1, -1, 1	1, -1, -1, 1, -1, 1, -1
-1, -1, -1, 1, 1	-1, -1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1	-1, 1, 1, -1, -1, -1, -1
1, 1, -1, -1, 1, 1, 1	–1, 1, 1, 1, -1
1, -1, -1, 1, -1, 1, -1	-1, -1, -1, -1, 1
-1, -1, 1, 1, 1, 1, 1	-1, -1, -1, 1, 1
-1, 1, 1, -1, -1, -1, -1	1, 1, 1, 1, 1

As shown in the above data, some of the paddings resulted in the data associating with their given input and output, while others did not.

The reason the first and third paddings caused the B.A.M to associate the data correctly is that the paddings increased the overall orthogonality of the training sets such that $A_k = -B_k$.

However, the reason the second padding did not increase the B.A.Ms associative ability's, but instead decreased them, is because the paddings lowered the overall orthogonality of the training data, causing the encoding to improperly associate to the correct data.

2.4 Assign Part D (Error Correction)

The final experiment of this report will test the B.A.M's ability to error correct an altered output to determine whether or not it can ascertain what the input was original. For this test, the third encoding that created the stable network in part C will test the networks error correcting abilities.

During this examination, one input is chosen randomly for each of the twenty tests and bounced around the B.A.M neural network until convergence, with the input having a chance to mutate with a twenty percent (20%) mutation rate for each element. The mutation will cause either zero or five elements in the input to mutate, altering the memorized pattern to change into either an unrecognized pattern or a pattern stored in the B.A.M's memory.

The B.A.M's error correction works by taking an input X from the set of training data, an input or output, and performs a mutation on the data, altering the data into data not used to train the B.A.M.

The B.A.M takes this input X and generates its corresponding Y (output), which is then passed through the B.A.M to generate X_k. This data is then again passed through the B.A.M to generate X_{k+1}. If $X_k = X_{k+1}$, then the data has reached its error-corrected value. If however, $X_k \neq X_{k+1}$ then X_{k+1} becomes the new X_k, and the process repeats until the data converge (i.e. the data is ping-ponged back and forth in the B.A.M until the data no longer changes).

The Below code is the Java implementation of the above error-correcting ideology:

```
private static void errorTest(int[] input){
        int[] previnput = test(test(input,true),true);// array
        ↪  holding previous error-corrected input result
        int[] prevoutput = test(previnput,true);// array holding
        ↪  previous error-corrected output result
        while(!Arrays.equals(previnput, test(prevoutput, true)) ){
            previnput = test(prevoutput,true);
            prevoutput = test(test(prevoutput,true),true);
        }// error correct until input reach convergence
        System.out.println("Output: " + Arrays.toString(test(test(pr
        ↪  evinput,true),true))+"\n");// print error-corrected
        ↪  input
    }//errorTest
```

Listing 2.3: Java Implementation for B.A.M error-correcting method

The results of the B.A.M's error correcting from the stabilized network is shown below in table 8:

As shown in table 8, the B.A.M error-corrected most of the data that underwent mutation back to its original form. However, a few of the mutated data entries were incorrectly associated with different trained inputs and others were associated to input not used in the training of the B.A.M system.

The reason some of the B.A.Ms input error-corrected to different trained information because the data underwent such a mutation that the hamming distance between the original data and the mutated data is greater than the hamming distance between the error-corrected data and the mutated. Thus causing the B.A.M to error-correct the mutated data to a different trained data. An example of how the B.A.M error-corrected the mutated data into a different trained input with a smaller hamming distance is observable at run 2 and 14 in table 8.

However, while some of the mutated data error-corrected to data used to train the B.A.M, some error-corrected to data not used to condition the B.A.M system. This happened because the B.A.M's memory had not filled as the padding from part C increased the number of neurons in the output layer, causing the memory space in the B.A.M to expand. So when the training data received no additional data to fill the B.A.M's extra memory space, this caused the weight matrix to associate untrained patterns with those unused memory spaces, causing the B.A.M to error-correct to data not used in the training of the B.A.M model.

Table 8: Part D B.A.M Error correction

Run	Original (O)	Mutated (M)	Hdist(O,M)	Error Corrected (C)	HDist(O,C)
1	-1111-1	11-1-1-1	3	111-1-1	2
2	-1111-1	11111	2	11111	2
3	-1-1-1-11	-1-1-1-11	0	-1-1-1-11	0
4	-1-1-1-11	11-11-1	4	1111-1	5
5	-1111-1	-1111-1	0	-1111-1	0
6	-1111-1	1111-1	1	1111-1	1
7	-1-1-111	-1-1-111	0	-1-1-111	0
8	11111	11111	0	11111	0
9	11111	-11111	1	1111-1	1
10	11111	11111	0	11111	0
11	-1-1-1-11	-1-11-11	1	-1-1-1-11	0
12	11111	-111-1-1	3	-1111-1	2
13	-1111-1	-111-11	2	111-1-1	2
14	-1-1-1-11	-1-111-1	3	-1111-1	4
15	-1111-1	-1111-1	0	-1111-1	0
16	11111	-1-11-11	3	-1-1-1-11	4
17	-1111-1	1111-1	1	1111-1	1
18	-1-1-111	-1-1-111	0	-1-1-111	0
19	-1-1-111	-1-1-111	0	-1-1-111	0
20	-1-1-111	-1-1-1-1-1	2	-1-1-1-1-1	2

3 Conclusion

In conclusion, through experiments performed on Bart Kosko's implementation of the B.A.M model, numerous findings have shown the B.A.M's capabilities under certain operating conditions.

Based on the above experimentation, the following information on the B.A.M model has been observed.

First is when a training data set that does not fill the memory of the B.A.M model is fed into the B.A.M, there is not enough training information to associate the training data correctly.

Secondly, A training data set that does fill the B.A.M's memory increases the convergence rate of the data as the B.A.M can correctly associate more data. However, if the data does not have an encoding that maximizes the orthogonality between A_k and B_k such that $A_k = -B_K$, it will still make incorrect associations.

Thirdly, Padding the training data outputs works best when it increases the orthogonality between the input and output.

Finally, the effectiveness of the B.A.M.'s error-correcting abilities is limited. As seen in part D of this report's experimentation, the B.A.M had difficulties error-correcting mutated inputs.

Therefore based on the above experimentation, we see that the B.A.M model has several limitations, including limited memory space based on the number of neurons in the B.A.M and incorrect convergence when error-correcting data. However, even with these limitations outlined above the B.A.M model can still be used in identifying pictures or error-correcting sentences.

References

[1] A. V. Ratz, “”A Succient Guide To Bidirectional Associative Memory (BAM)”,” April 2022.

[2] D.Bockus, “”COSC 4P80 Artificial Neural Network Slides”,” January 2023.

List of Figures

1.1 B.A.M associating in both directions 1
1.2 B.A.M with 4 neurons and memory capacity of 3 2

List of Tables

1 Part A training data 5
2 Part A output data 5
3 Part B training data 6
4 Part B output data 6
5 Part C output data with the above outlined padding 7
6 Part C output data with the above padding 8
7 Part C output from the above padding 8
8 Part D B.A.M Error correction 11

List of Listings

2.1 Java Implementation for B.A.M Weight Matrix Generator 3
2.2 Java Implementation for B.A.M test method 4
2.3 Java Implementation for B.A.M error-correcting method 10