An Associative Memory Network for Memorization and Traversal of Multi-Output Pattern Associations

Sidharth Anupkrishnan
Bangalore, Karnataka, India
sanupkri@redhat.com

Arjun Sharma
Bangalore, Karnataka, India
arjsharm@redhat.com

Abstract

This paper introduces an associative memory network model that can efficiently store auto-associative and hetero-associative pattern associations for retrieval. This biologically inspired model can not only store multi-output associations for a single input but also permits the traversal of the stored memories for that particular input - both breadth-wise and depth-wise traversal in a highly connected graph of associated memories, paralleling that of the associative functions of biological memory, thus making the model a good candidate for applications like multi-modal information retrieval due to the variety and flexibility in its search. We show that the model gives a high average testing accuracy during recall of lower-dimensional pattern association vectors.

1 Introduction

The current Neural Network Models like Deep Neural Networks (DNNs) and Recurrent Neural Networks (RNN) have become the mainstream standard for machine learning because, given sufficient data, they give excellent results, even outperforming human experts on various contests and tasks. However, when these models are fed with highly associated input data, they fail to memorize these associations effectively, thereby proving ineffective for learning contextual information[1].

Linear associative neural networks[2, 3] and Hopfield Networks[5, 6] are fairly fundamental solutions for auto-associative and hetero-associative memory learning. These models are hence good candidates for content addressable information retrieval provided there is some measure of orthogonality among the stored patterns. Linear associative nets, however, fail to give a desired output when the input pattern is a mixture of different patterns. Hopfield nets and Boltzmann Machines[7] solve this problem by having an asynchronous and stochastic update model for the individual neurons. This guarantees convergence to a local minimum, albeit there can be plenty of them. One problem that both linear associative nets and Hopfield models fail to address efficiently is that they give unpredictable outcomes for multi output associations. Consider a pattern $S1$ with association mappings $S1 \rightarrow O1$ and $S1 \rightarrow O2$ that is trained on this network. The presentation of $S1$ during testing would yield an outcome which would be the combination of the output patterns $O1$ and $O2$, when what was desirable would have been either $O1$ or $O2$.

We propose a biologically inspired neural network model that achieves multi-output association via competitive learning and further introduce memory traversal semantics for multi-modal search.

In: Rudolph S., Marreiros G. (eds.): Proceedings of STAIRS, Santiago de Compostela, Spain, august 2020, published at http://ceur-ws.org

2 The Model

The proposed model consists of a 2-layered neural network with feed forward connections from layer 1 ($L1$) to layer 2 ($L2$) and the $L2$ neurons form a fully connected graph. Similar to biological neurons, the processing neurons will be either in the "firing" state ($y_i = 1$) or "not firing" state ($y_i = 0$)[4]. Unlike the asynchronous and stochastic update strategy employed by Hopfield nets and Boltzmann machines, the proposed model employs an update strategy that involves gradual increment or inhibition of the potential of each neuron (Z_i) in $L2$ until it crosses the threshold U_i and the neuron is activated. This draws inspiration from biological neurons and how continued trains of action potential spikes in pre-synaptic axonal terminals trigger proportional neurotransmitter release until the resulting post-synaptic depolarizations trigger an action potential. The synaptic weight between the two neurons determines the rate of neurotransmitter release. The more the weight, the faster the post-synaptic neuron gets depolarized and subsequently activated. The newly activated neurons would in turn help in activating the rest of the neurons corresponding to the stored pattern.

The model uses two weight matrices $W^1 = W^1_{ij}$, $\forall\ i \in L1$, $j \in L2$ and $W^2 = W^2_{ij}$ $\forall\ i \in L2$, $j \in L2$.

2.1 Neural Encoding

The user is required to specify the dimension d of the pattern vector space and the number of active neurons that should fire for every pattern. This global count m is used as the winning criteria for the model to find a winner set of neurons during the testing phase. This does restrict the model from using the full capacity of the d-dimensional input vector. Although letting the model run for a certain number of epochs (set by the user) may seem like a plausible strategy, it can incur excessive contamination in the output pattern depending on the number of epochs. For a d-dimensional input vector and count of active neurons m, the number of possible patterns that can be fed into the model would be $\binom{d}{m}$.

2.2 Memorizing Associations

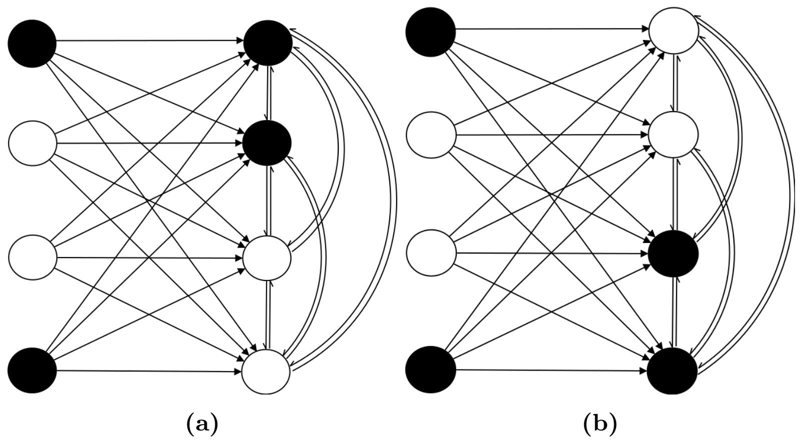

Figure 1: Training the model.
(a). The patterns S^p = [1 0 0 1] and O^p_1 = [1 1 0 0] are introduced for training the pattern association $S^p \rightarrow O^p_1$. The weights W^1_{11}, W^1_{12}, W^1_{41}, W^1_{42}, W^2_{12}, W^2_{21} are rewarded (incremented) due to association between firing neurons and W^1_{13}, W^1_{14}, W^1_{43}, W^1_{44}, W^2_{13}, W^2_{14}, W^2_{23}, W^2_{24} penalised (decremented) due to feed forward disassociation between the neurons (Note: We do not penalize W^2_{31}).
(b). The same input pattern S^p is trained with a different output vector O^p_2 so as to memorize the pattern association $S^p \rightarrow O^p_2$. The weight update follows the same methodology as (a).

The training of the model is dependent upon the time duration for which the pattern associations are presented. The patterns that need to be strongly remembered are given more training time. Consider the memorization of the association $S^p \rightarrow O^p$. During the training phase for $S^p \rightarrow O^p$, we will assume that both the firing neurons of S^p and O^p remain firing till its training duration and there is no decay with time. At the start, when there is no stored associations, the weight matrix is initialized as W_{ij} = 0, $\forall$ i,j.

At each training epoch, the weights between the pattern get updated as follows:

$$W_{ij}^1 = W_{ij}^1 + \frac{u_i \cdot v_j \cdot \eta \cdot \theta(t)}{D_{ij}} - \frac{(1 - v_j) \cdot u_i \cdot \gamma \cdot \theta(t)}{D_{ij}}$$

and,

$$W_{ij}^2 = W_{ij}^2 + \frac{v_i \cdot v_j \cdot \eta \cdot \theta(t)}{D_{ij}} - \frac{(1 - v_j) \cdot v_i \cdot \gamma \cdot \theta(t)}{D_{ij}}$$

where :

- u_i, $u_j \in S^p$ and v_i, $v_j \in O^p$
- η and γ represent the learning rate and the unlearning rate respectively
- θ(t) is the multiplier that increases at the start of each training session. The most recent training session will have the highest value of θ(t).
- D_{ij} here represent the inter-neuronal distance. One measure of distance could simply be $D_{ij} = |i - j| + 1$.
- $W_{ij} \in [0, \infty]$. The weights are constrained via a rectified linear unit (ReLU).

2.3 Memory Recall

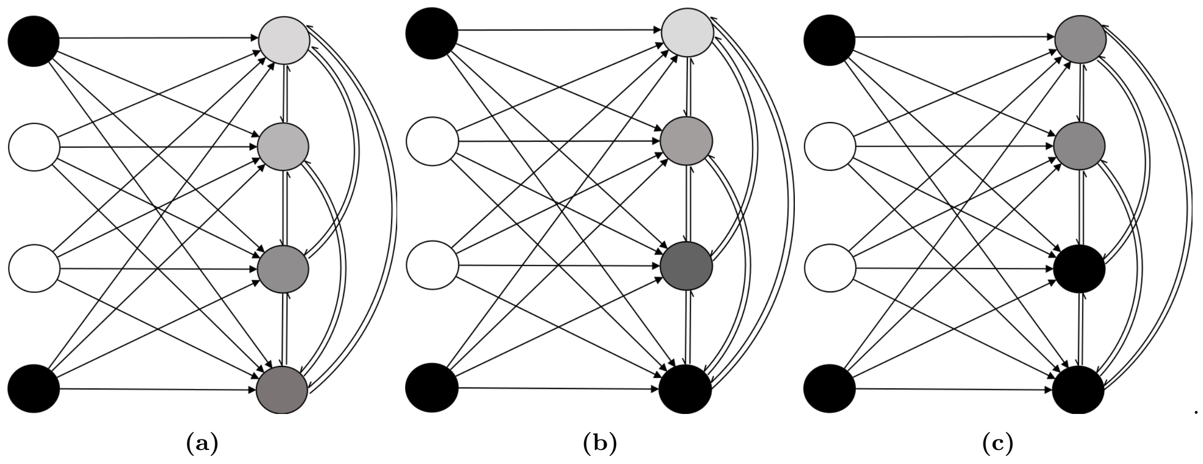

Figure 2: The Recall Process.
(a). $S^p = [1\ 0\ 0\ 1]$ is introduced into the model for testing and all the associated neurons in $L2$ start getting depolarized.
(b). After a few epochs, the first winner in the competition y_4 in $L2$, gets activated. The newly activated neuron helps in boosting the depolarization of y_3 due to high association of y_4 and y_3 (See Figure 1 (b)).
(c). When y_3 gets activated the competition ends due to number of activated neurons being equal to m and the output is $Y^p = [0\ 0\ 1\ 1]$.

The recall of stored associations is facilitated when the pattern S^p is introduced as input into the model. The recall process involves the triggering of depolarizations in the output vector Y^p. The depolarizations are recorded by the vector Z. A neuron y_i in Y^p is activated ($y_i = 1$) if the depolarization for that neuron (Z_i) exceeds the threshold (U_i). The d-dimensional output vector Y^p is continually updated as the recall phase goes on and the recall is completed when the number of active neurons in Y^p is m.
At the nth epoch:

$$Z_i^n = Z_i^{n-1} + \sum_j W_{ji}^1 \cdot u_j^{n-1} + \sum_j W_{ji}^2 \cdot y_j^{n-1}$$

$$y_i^n = \begin{cases} 1, & \text{if } Z_i^n \geqslant U_i \\ 0, & \text{if } Z_i^n < U_i \end{cases}$$

$$\text{where } u_j \in S^p \text{ and } y_i, y_j \in Y^p$$

2.4 Traversal of the Stored Memories

The model so far gives a single output when an input is presented. This output is determined by the winner set of neurons among the competing neurons. There seems to be a lot of wasted potential in the sense that other output patterns associated with that particular input pattern are forgotten or cannot be retrieved. To tackle this problem, we introduce an inhibition vector I^p that inhibits the set of winner neurons to let the other set of competing neurons win.

The inhibition vector I^p is implemented as a hash table. The entries of I^p are set when a recall is ended and the output is retrieved. For a set of active neurons $\{y_{i^1}, y_{i^2}, . ., y_{i^m}\}$, the hash table entry is filled by a hash function $h(i^1, i^2, .., i^m)$.

Algorithm 1 Breadthwise Traversal

```
1: procedure BT($S^p$, $num\_patterns$)
2:     $T \leftarrow \{\}$                                   ▷ A list of traversed patterns $T$
3:     $I^p \leftarrow (0, 0, ...0)_d$                       ▷ The inhibition vector $I^p$
4:     $i \leftarrow 1$
5:     while $i \neq num\_patterns$ do
6:         $Y \leftarrow (0, 0, ...0)_d$
7:         $keys \leftarrow \{\}$
8:         $Y \leftarrow$ Recall($S^p$, $I^p$)                ▷ Recall the stored pattern
9:         $T.add\{Y\}$
10:        $j \leftarrow 1$
11:        while $j \neq d$ do
12:            if $Y[j] == 1$ then
13:                $keys$.add($j$)                            ▷ Add index to list of keys
14:            $j \leftarrow j + 1$
15:        $I^p[h(keys)] \leftarrow 1$                        ▷ Set the entry in $I^p$
16:        $i \leftarrow i + 1$
17:    return $T$
```

During the end of a recall session i.e when $\sum_{i=1}^{d} y_i = m$, the inhibition is triggered and updates Z_i as:

$$Z_i = \begin{cases} 0, & \text{if } y_i \cdot I^p[h(keys)] = 1 \\ Z_i, & \text{if } y_i \cdot I^p[h(keys)] = 0 \end{cases}$$

The deactivation of the previously active neurons will let the other competing neurons cross the threshold and emerge as the winning set of active neurons.This algorithm when repeated over and over, facilitates a breadthwise traversal of stored patterns in the implicit pattern association graph (corresponding to the origin input pattern).

The depthwise traversal of the memory graph can be achieved by simply using the recalled output as the new input to be recalled.

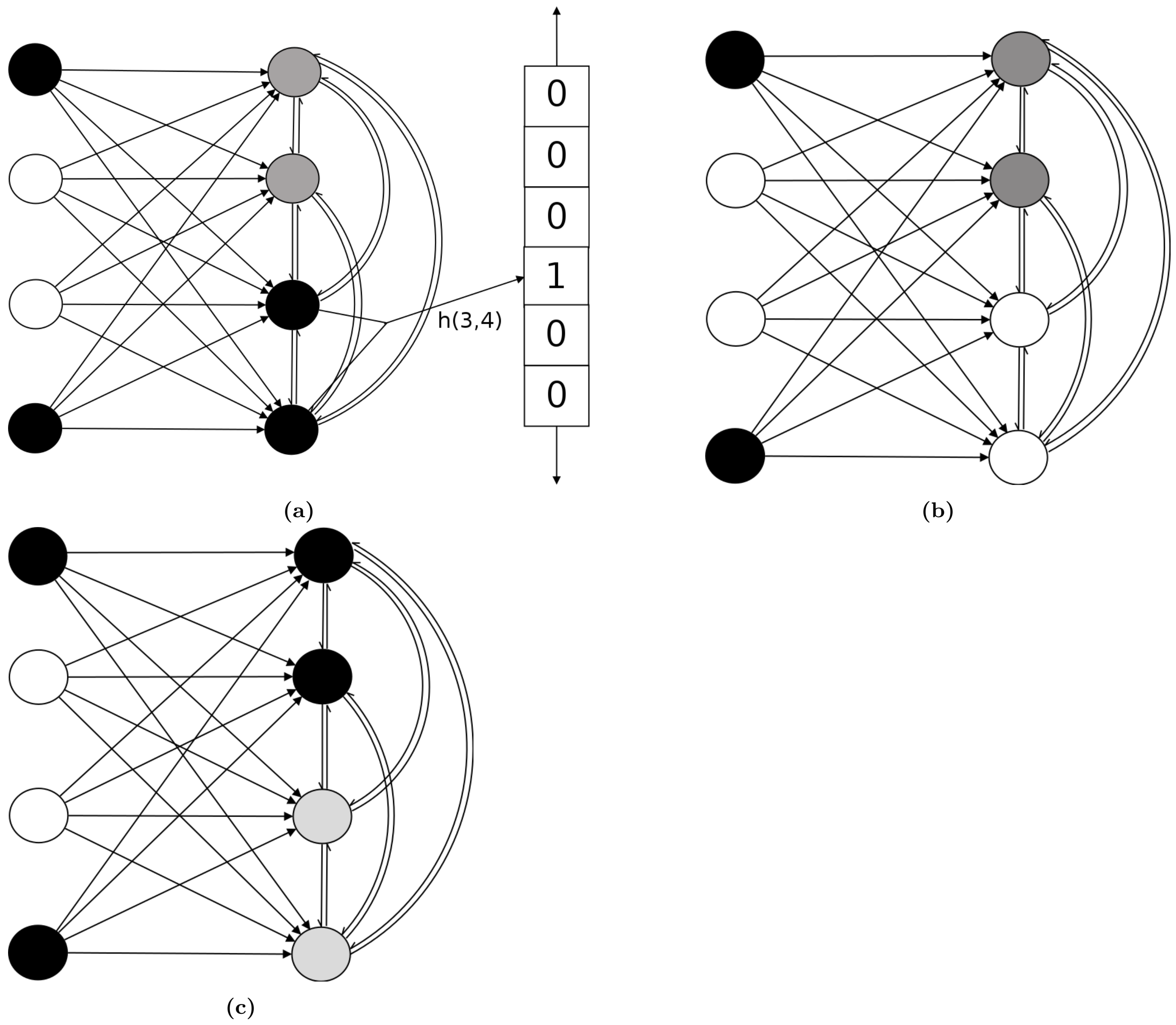

Figure 3: Change in Recall Process.
(a). This pattern output produced during recall is not retrieved due to the presence of a set entry in I^p for this particular set of active neurons.
(b). The active neurons are inhibited and the other neurons (y_1 and y_2) are allowed to compete.
(c). y_1 and y_2 is the winner set of active neurons and Y^p is retrieved.

Algorithm 2 Depthwise Traversal

```
1: procedure DT($S^p$, $num\_patterns$)
2:     $T \leftarrow \{\}$                          ▷ A list of traversed patterns $T$
3:     $i \leftarrow 1$
4:     while $i \neq num\_patterns$ do
5:         $Y \leftarrow$ Recall($S^p$)              ▷ Recall the stored pattern
6:         $T.add\{Y\}$
7:         $S^p = Y$
8:     return $T$
```

	n = 5			n = 10			n = 20		
	$m = 3$	$m = 5$	$m = 8$	$m = 3$	$m = 5$	$m = 8$	$m = 3$	$m = 5$	$m = 8$
$d = 10$	0.8785	0.7840	0.6802	0.7466	0.7187	0.5982	0.6017	0.5513	0.4502
$d = 30$	0.7167	0.6897	0.6277	0.6698	0.6623	0.5201	0.5926	0.5022	0.4003

Table 1: Average testing accuracies over different batches of training sessions for random pattern associations

	l = 5			l = 10			l = 20		
	$m = 3$	$m = 5$	$m = 8$	$m = 3$	$m = 5$	$m = 8$	$m = 3$	$m = 5$	$m = 8$
$d = 10$	0.8301	0.8012	0.7433	0.7306	0.7000	0.6302	0.6105	0.5801	0.5258
$d = 30$	0.7213	0.7051	0.6811	0.6899	0.6366	0.5330	0.5135	0.4992	0.4161

Table 2: Average testing accuracies over different batches of training sessions for multi-output associations of the same input vector

3 Results

The model was trained with random d-dimensional pattern vectors of m active neurons with a training time of 500 epochs per training sample. Table 1 shows the results of the average testing accuracy over varying numbers of stored associations (n), m and d on different training batches. For lower-dimensional pattern inputs ($d < 30$), the recall is almost perfect with an accuracy of around 0.8785. This metric was further bumped up when trained and tested against a set of mutually orthogonal pattern vectors and showed a mean testing accuracy of 0.9017 (The error only stemming from multi-output associations). For higher dimensional pattern inputs ($d \geqslant 30$), the testing accuracy dips as low as 0.4003 and the output suffers from contamination. This is effectively due to the saturation of the weight matrix which fails the competitive learning procedure. The accuracy, however, was high (0.8314) when a set of mutually orthogonal pattern vectors were chosen.

Further, we simulated the training of the model for a single input vector associated with multiple output vectors (l) and the results are shown in Table 2. The testing was done by utilizing breadthwise traversal to recall the multi-output pattern associations. The results were quite similar to that of the recall of random pattern associations. Lower-dimensional vectors ($d < 30$) achieved an average testing accuracy of 0.8785 whereas higher-dimensional vectors ($d \geqslant 30$) went as low as 0.4161

4 Conclusion and Future Work

Our results showed that the proposed model was able to efficiently store and retrieve pattern associations with minimal error (for lower dimensions). Further, in combination with the traversal schemes devised, the model is a great candidate for multi-modal information retrieval. Although restricted by the neural encoding scheme presented in this paper, there is still a high storage capacity (before excessive contamination).

However, there is much to be improved. We believe that the model can be made even better by introducing several layers of neurons and thus induce multi-stage competition among neurons that would further enhance learning. Like [8], future work needs to be done to introduce a framework for multi-modal information retrieval that utilizes this model and the flexibility in searching that it provides.

References

[1] H. He, Y. Shang, X. Yang, et al. Constructing an Associative Memory System Using Spiking Neural Network. *Front Neurosci. 2019;13:650. Published 2019 Jul 3. doi:10.3389/fnins.2019.00650*

[2] J. A. Anderson. A simple neural network generating an interactive memory. *Mathematical Biosciences, vol. 14, pp. 197-220, 1972.*

[3] B. Kosko. Adaptive bidirectional associative memories. *Appl. Opt. 26, 4947-4960 (1987)*

[4] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity *W. Bulletin of Mathematical Biophysics. (1943) 5: 115.*

[5] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences 79.8 (1982): 2554-2558.*

[6] Dmitry, John J. Hopfield and Krotov. Dense associative memory for pattern recognition. *Advances in neural information processing systems (pp. 1172-1180): 2016*

[7] Geoffrey E. Hinton. (2007-05-24). Boltzmann machine. *Scholarpedia. 2 (5): 1668.*

[8] P. Joshi, V.M. Ladwani, V. Ramasubramanian, R. Shriwas. Multi-modal Associative Storage and Retrieval Using Hopfield Auto-associative Memory Network. *Theoretical Neural Computation. ICANN 2019. Lecture Notes in Computer Science, vol 11727. Springer, Cham*