Associate Memory Network

Associative Memory Networks

An associative memory network can store a set of patterns as memories. When the associative memory is being presented with a key pattern, it responds by producing one of the stored patterns, which closely resembles or relates to the key panern. Thus, the recall is through association of the key pattern, with the help of informtion memorized. These types of memories are also called as **content-addressable memories (CAM)**. The CAM can also be viewed as associating data to address, i.e.; for every data in the memory there is a corresponding unique address. Also, it can be viewed as data correlator. Here inpur data is correlated with that of the stored data in the CAM. It should be nored that the stored patterns must be unique, i.e., different patterns in each location. If the same pattern exists in more than one locacion in the CAM, then, even though the correlation is correct, the address is noted to be ambiguous. Associative memory makes a parallel search within a stored data file. The concept behind this search is to Output any one or all stored items Which match the given search argumemt.

Training Algorithms for Pattern Association

There are two algorithms developed for training of pattern association nets.

1. Hebb Rule
2. Outer Products Rule

1. Hebb Rule

The Hebb rule is widely used for finding the weights of an associative memory neural network. The training vector pairs here are denoted as s:t. The weights are updated unril there is no weight change.

Hebb Rule Algorithmic

Step 0: Set all the initial weights to zero, i.e.,
$W_{ij} = 0$ (i = 1 to n, j = 1 to m)

Step 1: For each training target input output vector pairs s:t, perform Steps 2-4.

Step 2: Activate the input layer units to current training input, $X_i=S_i$ (for i = 1 to n)

Step 3: Activate the output layer units to current target output, $y_j = t_j$ (for j = 1 to m)

Step 4: Start the weight adjustment

$w_{ij}(new)=w_{ij}(old)+x_iy_j(i=1tonj=1tom)$

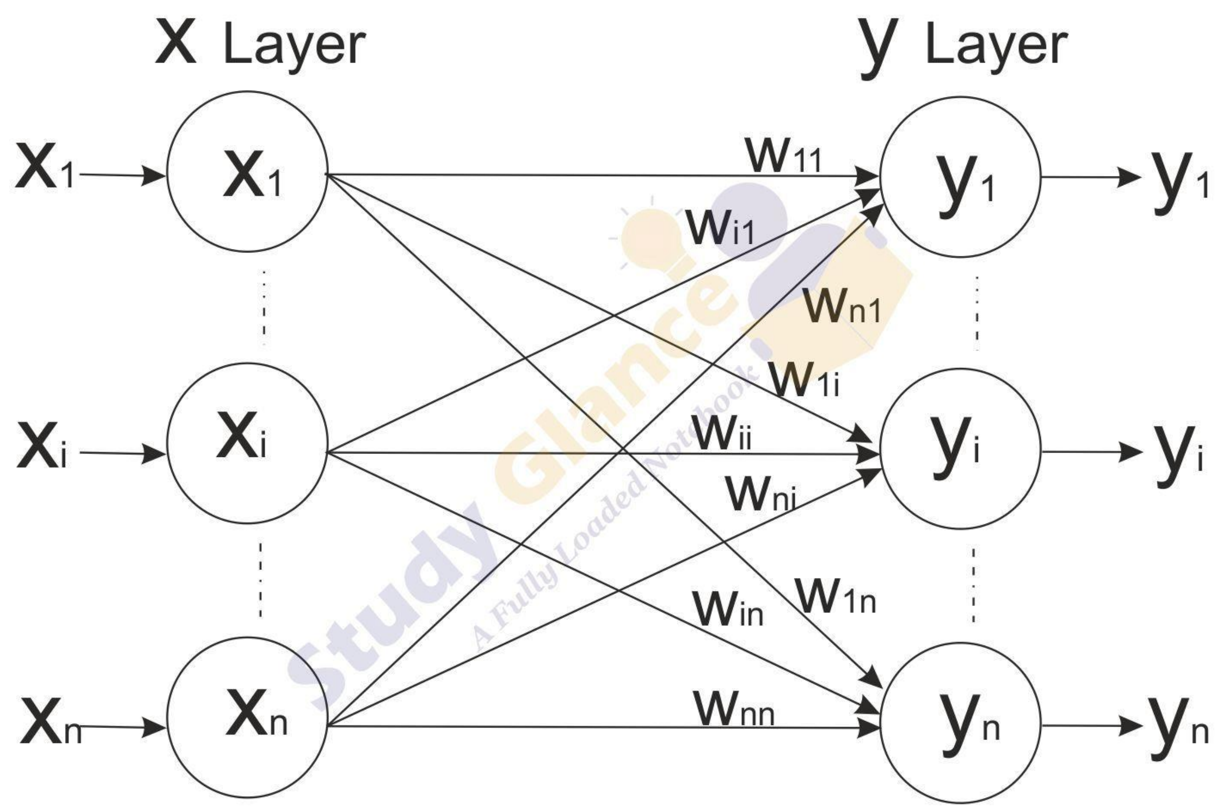
There two types of associative memories

- **Auto Associative Memory**
- **Hetero Associative memory**

Auto Associative Memory

An auto-associative memory recovers a previously stored pattern that most closely relates to the current pattern. It is also known as an auto-associative correlator.

In the auto associative memory network, the training input vector and training output vector are the same



Auto Associative Memory Algorithm

Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Step 1 − Initialize all the weights to zero as $\mathbf{w_{ij} = 0}$, $i = 1$ to n, $j = 1$ to n

Step 2 − Perform steps 3-4 for each input vector.

Step 3 − Activate each input unit as follows −

$$x_i = s_i \; (i = 1 \text{ to } n)$$

Step 4 − Activate each output unit as follows −

$$y_j = s_j \; (j = 1 \text{ to } n)$$

Step 5 − Adjust the weights as follows −

$$w_{ij}(new) = w_{ij}(old) + x_i y_j$$

Testing Algorithm

Step 1 − Set the weights obtained during training for Hebb's rule.

Step 2 − Perform steps 3-5 for each input vector.

Step 3 − Set the activation of the input units equal to that of the input vector.

Step 4 − Calculate the net input to each output unit $\mathbf{j = 1}$ **to** $\mathbf{n}$

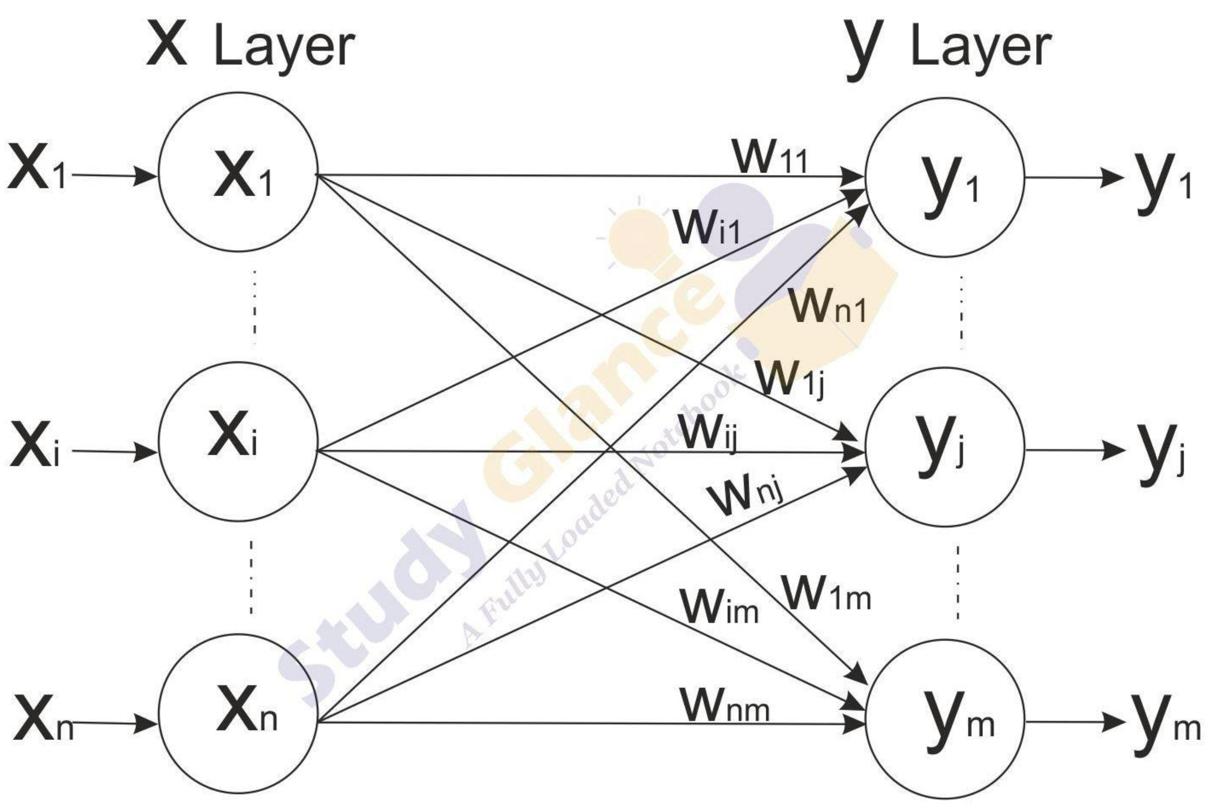
$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 − Apply the following activation function to calculate the output

$$y_j = f(y_{inj}) = \begin{cases} +1 \text{ if } y_{inj} > 0 \\ -1 \text{ if } y_{inj} \leqslant 0 \end{cases}$$

Hetero Associative memory

In a hetero-associate memory, the training input and the target output vectors are different. The weights are determined in a way that the network can store a set of pattern associations. The association here is a pair of training input target output vector pairs (s(p), t(p)), with p = 1,2,…p. Each vector s(p) has n components and each vector t(p) has m components. The determination of weights is done either by using Hebb rule or delta rule. The net finds an appropriate output vector, which corresponds to an input vector x, that may be either one of the stored patterns or a new pattern.



Hetero Associative Memory Algorithm

Training Algorithm

Step 1 – Initialize all the weights to zero as w_{ij} = 0 i= 1 to n, j= 1 to m

Step 2 – Perform steps 3-4 for each input vector.

Step 3 – Activate each input unit as follows –

$$x_i \;=\; s_i \; (i \;=\; 1 \, to \, n)$$

Step 4 – Activate each output unit as follows –

$$y_j \;=\; s_j \; (j \;=\; 1 \, to \, m)$$

Step 5 – Adjust the weights as follows –

$$w_{ij}(new) \;=\; w_{ij}(old) \;+\; x_i y_j$$

The weight can also be determine form the Hebb Rule or Outer Products Rule learning

$$\mathrm{w} \;=\; \sum_{p=1}^{n} S^{\mathrm{T}}(p) \,.\, \mathrm{S}(p)$$

Testing Algorithm

Step 1 – Set the weights obtained during training for Hebb's rule.

Step 2 – Perform steps 3-5 for each input vector.

Step 3 – Set the activation of the input units equal to that of the input vector.

Step 4 – Calculate the net input to each output unit **j = 1 to m**

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 – Apply the following activation function to calculate the output

$$y_j = f(y_{inj}) = \begin{cases} +1 \text{ if } y_{inj} > 0 \\ 0 \text{ if } y_{inj} = 0 \\ -1 \text{ if } y_{inj} < 0 \end{cases}$$

Bidirectional Associative Memory

Bidirectional Associative Memory (BAM)

Bidirectional associative memory (BAM), first proposed by Bart Kosko in the year 1988. The BAM network performs forward and backward associative searches for stored stimulus responses. The BAM is a recurrent hetero associative pattern-marching nerwork that encodes binary or bipolar patterns using Hebbian learning rule. It associates patterns, say from set A to patterns from set B and vice versa is also performed. BAM neural nets can respond to input from either layers (input layer and output layer).

Bidirectional Associative Memory Architecture

The architecture of BAM network consists of two layers of neurons which are connected by directed weighted pare interconnecrions. The network dynamics involve two layers of interaction. The BAM network iterates by sending the signals back and forth between the two layers until all the neurons reach equilibrium. The weights associated with the network are bidirectional. Thus, BAM can respond to the inputs in either layer.

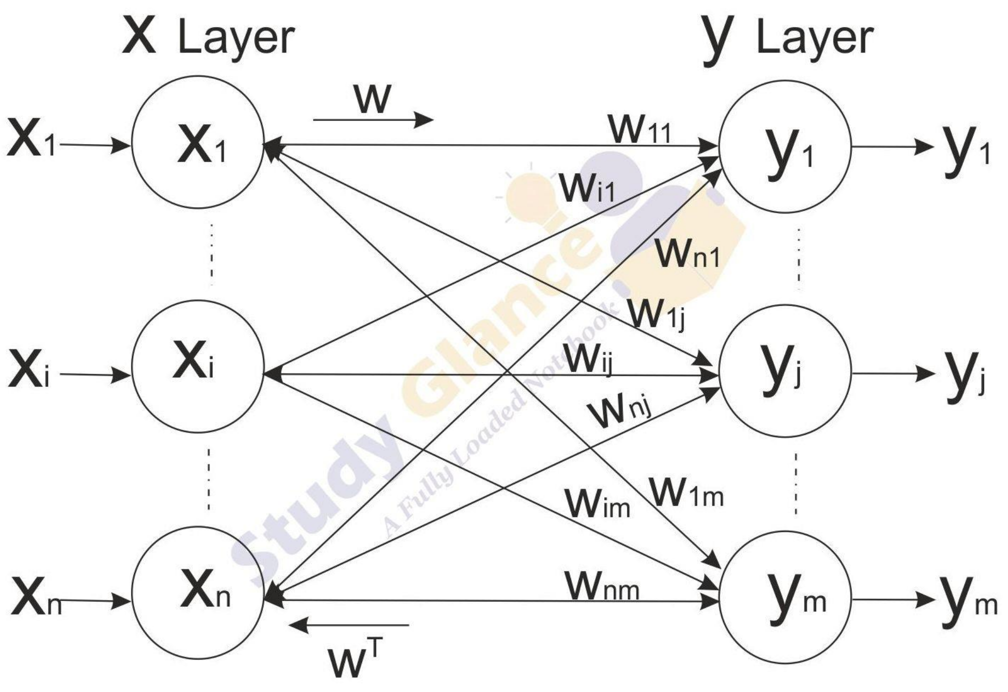


Figure shows a BAM network consisting of n units in X layer and m units in Y layer. The layers can be connected in both directions(bidirectional) with the result the weight matrix sent from the X layer to the Y layer is W and the weight matrix for signals sent from the Y layer to the X layer is W^T. Thus, the Weight matrix is calculated in both directions.

Recurrent Auto-associative Memory (RAAM)

A Recurrent Autoassociative Memory (RAAM) is a type of recurrent neural network (RNN) that is designed for associative memory tasks. Associative memory is a type of memory that allows the recall of a whole pattern when presented with a part of that pattern. In the case of autoassociative memory, the network is trained to recall an entire input pattern when given only a partial or corrupted version of that pattern.

Here's a breakdown of the key components and concepts related to Recurrent Autoassociative Memory:

1. **Recurrent Neural Network (RNN):** RAAM is based on the architecture of recurrent neural networks, which have connections that form directed cycles. This cyclic structure allows information to be stored and recalled over time, making RNNs suitable for tasks involving sequential or temporal data.
2. **Autoassociative Memory:** An autoassociative memory is a type of neural network that is trained to reproduce its input patterns at the output. In the context of RAAM, the network is trained to remember and reconstruct complete patterns from partial or noisy inputs.
3. **Training Process:** During training, the RAAM is presented with input patterns, and the weights of the network are adjusted to minimize the difference between the input and the reconstructed output. This training process enables the network to learn the underlying patterns and associations in the data.
4. **Recall Mechanism:** After training, the RAAM can be used for recall. When presented with a partial or degraded version of a previously learned pattern, the network should be able to reconstruct and output the complete pattern.
5. **Applications:** RAAMs can be used in various applications where associative memory is crucial. For example, in image or pattern recognition, image restoration etc RAAM could be trained to recall complete images even when presented with only a portion of the image.

It's important to note that while RAAMs have been explored in the past, more recent advancements in neural network architectures, such as **Long Short-Term Memory (LSTM)** networks and **Gated Recurrent Units** (GRUs), have become popular choices for tasks involving sequential data due to their ability to capture long-term dependencies more effectively. However, depending on the specific requirements of a task, RAAMs or other memory-augmented networks may still have their applications.

Encoder-Decoder Recurrent Neural Network Models

What is seq2seq Model in Machine Learning?

Seq2seq was first introduced for machine translation, by Google. Before that, the translation worked in a very naïve way. Each word that you used to type was converted to its target language giving no regard to its grammar and sentence structure. Seq2seq revolutionized the process of translation by making use of deep learning. It not only takes the current word/input into account while translating but also its neighborhood.

Seq2Seq (Sequence-to-Sequence) is a type of model in machine learning that is used for tasks such as **machine translation, text summarization, and image captioning.** The model consists of two main components:

- Encoder
- Decoder

Seq2Seq models are trained using a dataset of input-output pairs, where the input is a **sequence of tokens** and the output is also a sequence of tokens. The model is trained to maximize the likelihood of the correct output sequence given the input sequence.

Seq2Seq models have been widely used in NLP tasks such as machine translation, text summarization, and image captioning, due to their ability to handle variable-length input and output sequences. Additionally, the **Attention mechanism** is often used in Seq2Seq models to improve performance and it allows the decoder to focus on specific parts of the input sequence when generating the output.

Nowadays, it is used for a variety of different applications such as image captioning, conversational models, text summarization, etc.

Encoder-Decoder Stack

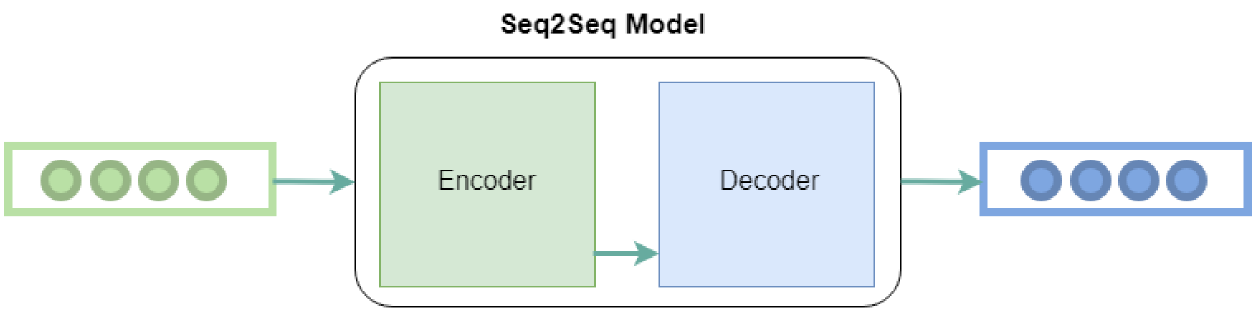
As the name suggests, seq2seq takes as input a sequence of words(sentence or sentences) and generates an output sequence of words. It does so by use of the recurrent neural network (RNN). Although the vanilla version of RNN is rarely used, its more advanced version i.e. LSTM or GRU is used. This is because RNN suffers from the problem of vanishing gradient. LSTM is used in the version proposed by Google. It develops the context of the word by taking 2 inputs at each point in time. One from the user and the other from its previous output, hence the name recurrent (output goes as input).
The encoder and decoder are typically implemented as Recurrent Neural Networks (RNNs) or Transformers.

Encoder Stack

It uses deep neural network layers and converts the input words to corresponding hidden vectors. Each vector represents the current word and the context of the word. The encoder takes the input sequence, one token at a time, and uses an RNN or transformer to update its hidden state, which summarizes the information in the input sequence. The final hidden state of the encoder is then passed as the context vector to the decoder.

Decoder Stack

It is similar to the encoder. It takes as input the hidden vector generated by the encoder, its own hidden states, and the current word to produce the next hidden vector and finally predict the next word. The decoder uses the context vector and an initial hidden state to generate the output sequence, one token at a time. At each time step, the decoder uses the current hidden state, the context vector, and the previous output token to generate a probability distribution over the possible next tokens. The token with the highest probability is then chosen as the output, and the process continues until the end of the output sequence is reached.



Encoder and Decoder Stack in seq2seq model

Components of seq2seq Model in Machine Learning

Apart from these two, many optimizations have led to other components of seq2seq:

- **Attention:** The input to the decoder is a single vector that has to store all the information about the context. This becomes a problem with large sequences. Hence the attention mechanism is applied which allows the decoder to look at the input sequence selectively.
- **Beam Search:** The highest probability word is selected as the output by the decoder. But this does not always yield the best results, because of the basic problem of greedy algorithms. Hence beam search is applied which suggests possible translations at each step. This is done by making a tree of top k-results.
- **Bucketing:** Variable-length sequences are possible in a seq2seq model because of the padding of 0's which is done to both input and output. However, if the max length set by us is 100 and the sentence is just 3 words long it causes a huge waste of space. So we use the concept of bucketing. We make buckets of different sizes like (4, 8) (8, 15), and so on, where 4 is the max input length defined by us and 8 is the max output length defined.

STABILITY

Stability refers to the ability of the network to reliably store and retrieve patterns despite the presence of noise or partial information during recall. In other words, a stable associative memory network should be able to correctly recall the original patterns even when presented with incomplete or corrupted versions of those patterns.

Key Concepts Related to Stability in Associative Memory Networks:

1. **Pattern Storage:**
    - Associative memory networks are trained to store a set of patterns in their weights or connections. These patterns could represent memories or information that the network has learned during the training phase.
2. **Pattern Recall:**
    - During recall, the network is presented with a cue or a partial pattern, and it is expected to retrieve the complete, original pattern that is most closely associated with the given cue.
3. **Stability Criterion:**
    - A stable associative memory should exhibit a robust recall performance, meaning that it can accurately retrieve the stored patterns even in the presence of noise, distortions, or incomplete information.
4. **Error Correction:**
    - The stability of an associative memory network is often linked to its ability to correct errors or inaccuracies in the input patterns. This correction mechanism ensures that the retrieved pattern is close to the originally stored one.
5. **Capacity and Interference:**
    - The capacity of an associative memory network refers to the number of patterns it can successfully store and recall. Interference occurs when the cue for recall is similar to multiple stored patterns. A stable network should handle interference well and still produce accurate recalls.

Importance of Stability:

- Stability is crucial for the reliability and robustness of associative memory networks, especially in applications where noise or partial information is likely to be present.

- It ensures that the network can generalize well and recall patterns accurately, even when the input is not an exact match to the stored patterns.
- The study of stability in associative memory networks is essential for understanding their limitations, capacity, and performance in various practical applications, including pattern recognition, information retrieval, and cognitive modeling.