Neural Associative Memories

Neural associative memories (NAM) are neural network models consisting of neuron-like and synapse-like elements. At any given point in time the state of the neural network is given by the vector of neural activities, it is called the *activity pattern*. Neurons update their activity values based on the inputs they receive (over the synapses). In the simplest neural network models the input-output function of a neuron is the identitiy function or a threshold operation. Note that in the latter case the neural activity state is binary: active or inactive. Information to be processed by the neural network is represented by activity patterns (for instance, the representation of a tree can an activity pattern where active neurons display a tree's picture). Thus, activity patterns are the *representations* of the elements processed in the network. A representation is called *sparse* if the ratio between active and inactive neurons is small.

The synapses in a neural network are the links between neurons or between neurons and fibers carrying external input. A synapse transmits presynaptic activity to the postsynaptic site. In the simplest neural network models each synapse has a weight value and the postsynaptic signal is simply the product of presynaptic activity and weight. The input of a neuron is then the sum of the postsynaptic signals of all synapses connecting the neuron. Thus, information is processed in a neural network by activity spread. How activity spreads, and by this, which algorithm is implemented in the network depends on how the synaptic structure, the matrix of synaptic weights in the network is shaped by learning. In neural associative memories the learning provides the storage of a (large) set of activity patterns during learning, the memory patterns.

Different memory functions are defined by the way how learned patterns can be selectively accessed by an input pattern. The function of *pattern recognition* means to classify input patterns in two classes, familiar patterns and the rest. *Pattern association* describes the function to associate certain input patterns with certain memory patterns, i.e., each stored memory consists of a pair of input and desired output pattern.

The advantage of neural associative memories over other pattern storage algorithms like lookup tables of hash codes is that the memory access can be fault tolerant with respect to variation of the input pattern. For pattern association this means that an output pattern can be produced for a set of input patterns that are (with respect to a metric like the Hamming distance) closest to the input pattern presented during learning. Fault tolerance is key in the function of content-addressed pattern retrieval or *pattern completion*. Here the pattern pairs used during learning have to consist of two identical patterns which one also calls autoassociative learning. In this case if the input patterns are noisy versions of the learned patterns, the pattern association can be seen as pattern completion, i.e., the noisy version is replaced by the noiseless version of the pattern.

One discerns two different retrieval mechanisms corresponding to feed-forward or feed-back architectures of the neural network: In one-step retrieval each network neuron evaluates its input just once. In *iterative retrieval* activity flows through feedback connections and neurons evaluate their inputs several times during the retrieval process.

Training or learning in the network means that neural activity patterns can influence the synaptic structure. The rules governing this influence are called synaptic learning rules. The simplest form is a *local learning rule* where synaptic weight change depends only on pre- and postsynaptic activity, i.e., the signals locally available at the synapse.

In associative memories many associations can be stored at the same time. There are different *schemes of superposition of the memory traces* formed by the different associations. The superposition can be simple linear addition of the synaptic changes required for each association (like in the Hopfield model) or nonlinear. The Willshaw model, for instance employs a clipped superposition (resulting in binary weights) that is nonlinear. There are different models of associative memory with respect to the learning procedure. It can either be one-shot where each learning pattern is presented once or recurrent. In recurrent training the pattern retrieval is checked and the patterns are presented multiple times until a desired level of retrieval quality is achieved.

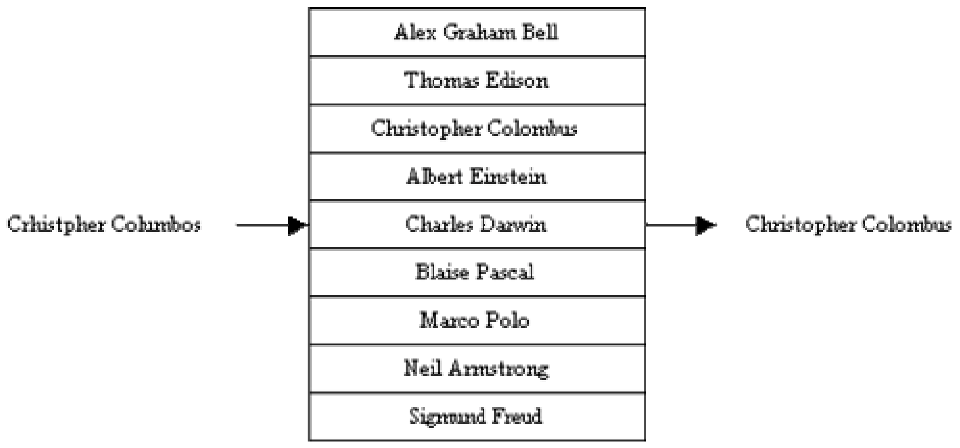
The performance of neural associative memories is usually measured by a quantity called *information capacity*, that is, the information content that can be learned and retrieved, devided by the number of synapses required.

Associative Memories

Description

A *content-addressable memory* is a type of memory that allows for the recall of data based on the degree of similarity between the input pattern and the patterns stored in memory. It refers to a memory organization in which the memory is accessed by its content as opposed to an explicit address like in the traditional computer memory system. Therefore, this type of memory allows the recall of information based on partial knowledge of its contents.

Suppose we are given a memory of names of several people as shown in the figure below. If the given memory is content-addressable, using the erroneous string "Crhistpher Columbos" as key is sufficient to retrieve the correct name "Christopher Colombus." In this sense, this type of memory is robust and fault-tolerant, as this type of memory exhibits some form of error-correction capability.



A content-addressable memory in action

An *associative memory* is a content-addressable structure that maps specific input representations to specific output representations. It is a system that "associates" two patterns (***X***, ***Y***) such that when one is encountered, the other can be recalled. Typically, ***X***Î {-1, +1}^{m}, ***Y*** Î {-1, +1}^{n} and *m* and *n* are the length of vectors ***X*** and ***Y***, respectively. The components of the vectors can be thought of as pixels when the two patterns are considered as bitmap images.

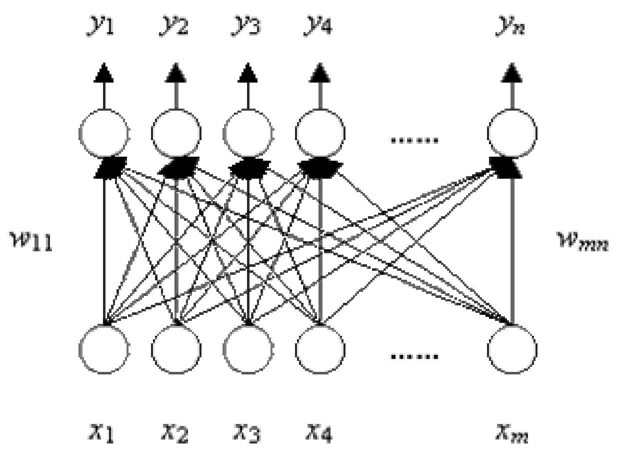
There are two classes of associative memory: *autoassociative* and *heteroassociative*. An autoassociative memory is used to retrieve a previously stored pattern that most closely resembles the current pattern, i.e., ***X*** = ***Y***. On the other hand, in a heteroassociative memory, the retrieved pattern is, in general, different from the input pattern not only in content but possibly also different in type and format, i.e., ***X*** ¹***Y***.

Artificial neural networks can be used as associative memories. One of the simplest artificial neural associative memory is the *linear associator*. The *Hopfield model* and *bidirectional associative memory* (BAM) models are some of the other popular artificial neural network models used as associative memories.

Associative Memories

Linear Associator

The *linear associator* is one of the simplest and first studied associative memory model. Below is the network architecture of the linear associator.



Linear Associator

It is a *feedforward type network* where the output is produced in a single feedforward computation. In the figure, all the m input units are connected to all the n output units via the connection weight matrix $\boldsymbol{W} = [w_{ij}]_{m \times n}$ where w_{ij} denotes the synaptic strength of the unidirectional connection from the ith input unit to the jth output unit. It is the connection weight matrix that stores the p different associated pattern pairs $\{(\boldsymbol{X}_k, \boldsymbol{Y}_k) \mid k = 1, 2, ..., p\}$ where $\boldsymbol{X}_k$ Î $\{-1, +1\}^m$ and $\boldsymbol{Y}_k$ Î $\{-1, +1\}^n$ in a distributed representation.

Building an associative memory is nothing but constructing the connection weight matrix $\boldsymbol{W}$ such that when an input pattern is presented, the stored pattern associated with the input pattern is retrieved. The process of constructing the connection weight matrix is called *encoding*. During encoding the weight values of the correlation matrix $\boldsymbol{W}_k$ for a particular associated pattern pair $(\boldsymbol{X}_k, \boldsymbol{Y}_k)$ are computed as:

$$(w_{ij})_k = (x_i)_k (y_j)_k$$

where $(x_i)_k$ represents the ith component of pattern $\boldsymbol{X}_k$, $(y_j)_k$ represents the jth component of pattern $\boldsymbol{Y}_k$ for $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$. Constructing the connection weight matrix $\boldsymbol{W}$ is then accomplished by summing up the individual correlation matrices, i.e.,

$$W = \alpha \sum_{k=1}^{p} W_k$$

where a is the proportionality or normalizing constant. a is usually set to $1/p$ to prevent the synaptic values from going too large when there are a number of associated pattern pairs to be memorized. The connection weight matrix construction above simultaneouly stores or remembers p different associated pattern pairs in a distributed manner.

After encoding or memorization, the network can be used for *retrieval*. The process of retrieving a stored pattern given an input pattern is called *decoding.* Given a stimulus input pattern ***X***, decoding or recollection is accomplished by computing the net input to the output units using:

$$input_j = \sum_{i=1}^{m} x_i w_{ij}$$

where *input$_j$* stands for the weighted sum of the input or activation value of node *j* for *j* = 1, 2, ..., *n*. Then determine the output of those units using the bipolar output function:

$$y_j = \begin{cases} +1 \text{ if } input_j \geq \theta_j \\ -1 \quad \text{otherwise} \end{cases}$$

where *q$_j$* is the threshold value of output neuron *j*. From the foregoing discussions, it can be seen that the output units behave like the *linear threshold units* (McCulloch and Pitts, 1943) and the *perceptrons*(Rosenblatt, 1958) that compute a weighted sum of the input and produces a -1 or +1 depending whether the weighted sum is below or above a certain threshold value.

However, the input pattern may contain errors and noise, or may be an incomplete version of some previously encoded pattern. Nevertheless, when presented with such a corrupted input pattern, the network will retrieve the stored pattern that is closest to actual input pattern. Therefore, the linear associator (and associative memories in general) is robust and fault tolerant, i.e., the presence of noise or errors results only in a mere decrease rather than total degradation in the performance of the network. Associative memories being robust and fault tolerant are the byproducts of having a number of processing elements performing highly parallel and distributed computations.

Traditional measures of associative memory performance are its *memory capacity* and *content-addressability*. Memory capacity refers to the maximum number of associated pattern pairs that can be stored and correctly retrieved while content-addressability is the ability of the network to retrieve the correct stored pattern. Obviously, the two performance measures are related to each other.

It is known that using Hebb's learning rule in building the connection weight matrix of an associative memory yields a significantly low memory capacity. Due to the limitation brought about by using Hebb's learning rule, several modifications and variants have been proposed to maximize the memory capacity. Some examples can be found in (Hassoun, 1993; Hertz, 1991; Patterson, 1996; Ritter, 1992).

Perfect retrieval is possible if the input patterns are *mutually orthogonal*, i.e.,

$$X_a X_b = \begin{cases} 1 \quad \text{if } a = b \\ 0 \text{ otherwise} \end{cases}$$

for *a* = 1, 2, ..., *k* and *b* = 1, 2, ..., *k*. If the stored input patterns are not mutually orthogonal, non-perfect retrieval can happen due *crosstalk* among the patterns. However, even if the input patterns are not mutually orthogonal, accurate retrieval can still be realized if the crosstalk term is small. The degree of correlation among the input patterns sets the limit on the memory capacity and content-addressability of an associative memory.

Another limitation of associative memories is the presence of *spurious memories*, i.e., meaningless memories or associated pattern pairs other than the original fundamental memories stored. Presence of spurious memories degrades the content-addressability feature of an associative memory. Another characteristic of associative memory is that the storage of the associated pattern pairs $\{(\boldsymbol{X}_k, \boldsymbol{Y}_k) \mid k = 1, 2, \ldots, p\}$ results in the storage of the associated pattern pairs $\{(\boldsymbol{X}_k^c, \boldsymbol{Y}_k^c) \mid k = 1, 2, \ldots, p\}$ where c stands for the complement of a vector, i.e., $\boldsymbol{X}_k^c = -\boldsymbol{X}_k$.

Associative memory can be autoassociative or heteroassociative. The difference between autoassociative and heteroassociative memories lies in the retrieved pattern. An autoassociative memory retrieves the same pattern $\boldsymbol{Y}$ given an input pattern $\boldsymbol{X}$, i.e., $\boldsymbol{Y} = \boldsymbol{X}$. On the other hand, a heteroassociative memory retrieves the stored pattern $\boldsymbol{Y}$ given an input pattern $\boldsymbol{X}$ such that $\boldsymbol{Y}$ ¹$\boldsymbol{X}$.

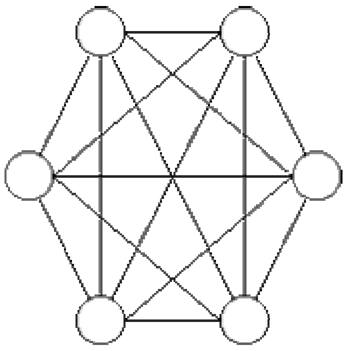
An autoassociative memory stores the associated pattern pairs $\{(\boldsymbol{X}_k, \boldsymbol{X}_k) \mid k = 1, 2, \ldots, p\}$. A heteroassociative memory is a generalization of an autoassociative memory with connection weight matrix equivalent to $\begin{bmatrix} 0 & W \\ W^T & 0 \end{bmatrix}$ and associated pattern pairs $\{(\boldsymbol{X}_k \mid \boldsymbol{Y}_k, \boldsymbol{X}_k \mid \boldsymbol{Y}_k) \mid k = 1, 2, \ldots, p\}$.

Another implementation of associative memory that differ significantly from the one discussed so far is the recurrent type networks where the output of the units are fed back to the input units. Recurrent type networks produce their output through recursive computations until they become stable. The popular *Hopfield Model* is of this type of network.

Hopfield Model

The *Hopfield model* was proposed by John Hopfield of the California Institute of Technology during the early 1980s. The publication of his work in 1982 significantly contributed to the renewed interest in research in artificial neural networks. He showed how an ensemble of simple processing units can have fairly complex collective computational abilities and behavior.

The dynamics of the Hopfield model is different from that of the *linear associator model* in that it computes its output recursively in time until the system becomes stable. Below is a Hopfield model with six units, where each node is connected to every other node in the network.


Hopfield model

Unlike the linear associator model which consists of two layers of processing units, one serving as the input layer while the other as the output layer, the Hopfield model consists of a single layer of processing elements where each unit is connected to every other unit in the network other than itself. The connection weight matrix $\boldsymbol{W}$ of this type of network is square and symmetric, i.e., $w_{ij} = w_{ji}$ for i, j = 1, 2, ..., m. Each unit has an extra external input I_i. This extra input leads to a modification in the computation of the net input to the units:

$$input_j = \sum_{i=1}^{m} x_i w_{ij} + I_j$$

for j = 1, 2, ..., m.

Unlike the linear associator, the units in the Hopfield model act as both input and output units. But just like the linear associator, a single associated pattern pair is stored by computing the weight matrix as follows:

$$\boldsymbol{W}_k = \boldsymbol{X}_k^{\mathrm{T}}\boldsymbol{Y}_k$$

where $\boldsymbol{Y}_k = \boldsymbol{X}_k$

$$W = \alpha \sum_{k=1}^{p} W_k$$

to store p different associated pattern pairs. Since the Hopfield model is an autoassociative memory model, patterns, rather than associated pattern pairs, are stored in memory.

After encoding, the network can be used for decoding. Decoding in the Hopfield model is achieved by a collective and recursive relaxation search for a stored pattern given an initial stimulus pattern. Given an input pattern $\boldsymbol{X}$, decoding is accomplished by computing the net input to the units and determining the output of those units using the output function to produce the pattern $\boldsymbol{X}'$. The pattern $\boldsymbol{X}'$ is then fed back to the units as an input pattern to produce the pattern $\boldsymbol{X}''$. The pattern $\boldsymbol{X}''$ is again fed back to the units to produce the pattern $\boldsymbol{X}'''$. The process is repeated until the network stabilizes on a stored pattern where further computations do not change the output of the units.

If the input pattern ***X*** is an incomplete pattern or if it contains some distortions, the stored pattern to which the network stabilizes is typically one that is most similar to ***X*** without the distortions. This feature is called *pattern completion* and is very useful in many image processing applications.

During decoding, there are several schemes that can be used to update the output of the units. The updating schemes are *synchronous* (or parallel as termed in some literatures), *asynchronous* (or sequential), or a combination of the two (*hybrid*).

Using the synchronous updating scheme, the output of the units are updated as a group prior to feeding the output back to the network. On the other hand, using the asynchronous updating scheme, the output of the units are updated in some order (e.g. random or sequential) and the output are then fed back to the network after each unit update. Using the hybrid synchronous-asynchronous updating scheme, subgroups of units are updated synchronously while units in each subgroup updated asynchronously. The choice of the updating scheme has an effect on the convergence of the network.

Hopfield (1982) demonstrated that the maximum number of patterns that can be stored in the Hopfield model of *m* nodes before the error in the retrieved pattern becomes severe is around 0.15*m*. The memory capacity of the Hopfield model can be increased as shown by Andrecut (1972).

In spite of this seemingly limited memory capacity of the Hopfield model, several applications can be listed (Chaudhuri, Dai, deMenezes, Garcia, Poli, Soper, Suh, Tsai). Discussions on the application of the Hopfield model to combinatorial optimization problems can be found in (Siu, Suh).

Various widely available text and technical papers vy Hassoun, Hertz (et al), Hopfield, McEliece, Ritter, and Volk can be consulted for a mathematical discussion on the memory capacity of the Hopfield model.

Discrete Hopfield Model

In the discrete Hopfield model, the units use a slightly modified bipolar output function where the states of the units, i.e., the output of the units remain the same if the current state is equal to some threshold value:

$$x_i(t+1) = \begin{cases} +1 \text{ if } input_i > \theta_i \\ x_i(t) \text{ if } input_i = \theta_i \\ -1 \text{ if } input_i < \theta_i \end{cases}$$

for i = 1, 2, ..., m and where t denotes the discrete time.

An interesting property of recurrent type networks is that their state can be described by an *energy function*. The energy function is used to prove the stability of recurrent type networks. For the discrete Hopfield model with w_{ii} = 0 and w_{ij} = w_{ji} using the asynchronous updating scheme, the energy function E according to Hopfield (1982) is defined as:

$$E = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} x_i w_{ij} x_j - \sum_{i=1}^{m} x_i I_i + \sum_{i=1}^{m} x_i \theta_i$$

where the local minima of the energy function correspond to the energy of the stored patterns. Hopfield (1982) has shown that the energy of the discrete Hopfield model decreases or remains the same after each unit update. Therefore, the network will eventually converge to a local minimum that corresponds to a stored pattern. The stored pattern to which the network converges depends on the input pattern and the connection weight matrix.

The energy of the discrete Hopfield model is bounded from below by:

$$E = -\sum_{i=1}^{m}\sum_{j=1}^{m} | w_{ij} | - \sum_{i=1}^{m} | I_i | + \sum_{i=1}^{m} | \theta_i |$$

for all $\boldsymbol{X}_k$, k = 1, 2, ..., p. Since the energy is bounded from below, the network will eventually converge to a local minimum corresponding to a stored pattern.

Continuous Hopfield Model

The continuous Hopfield model is just a generalization of the discrete case. Here, the units use a continuous output function such as the sigmoid or hyperbolic tangent function. In the continuous Hopfield model, each unit has an associated capacitor C_i and resistance r_i that model the capacitance and resistance of real neuron's cell membrane, respectively. Thus the state equation of each unit is now:

$$C_j \frac{dinput_j}{dt} = \sum_{i=1}^{m} x_i w_{ij} - \frac{input_j}{R_j} + I_j$$

where

$$\frac{1}{R_j} = \frac{1}{\rho_j} + \sum_{i=1}^{m} w_{ij}$$

Just like in the discrete case, there is an energy function characterizing the continuous Hopfield model. The energy function due to Hopfield (1984) is given by:

$$E = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} x_i w_{ij} x_j - \sum_{i=1}^{m} x_i I_i + \sum_{i=1}^{m}\left(\frac{1}{R_i}\right)\int_0^{x_i} f^{-1}(x)dx$$

where f is the output function of the units.

It can be shown that dE/dt £ 0 when $w_{ij} = w_{ji}$. Therefore, the energy of the continuous Hopfield model decreases or remains the same. The minimum energy of the continuous Hopfield model also exists using analogous computation as that in the discrete Hopfield model.