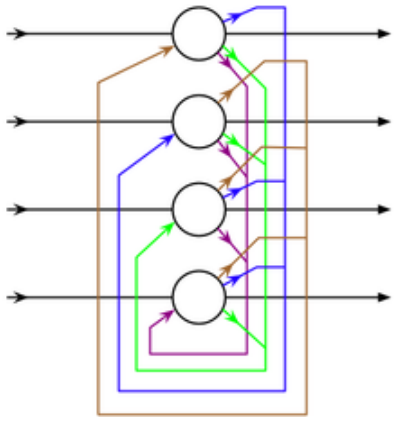
Hopfield network

A Hopfield network is a form of recurrent artificial neural network popularized by John Hopfield in 1982, but described earlier by Little in 1974. Hopfield nets serve as content-addressable ("associative") memory systems with binary threshold nodes. They are guaranteed to converge to a local minimum and, therefore, may converge to a false pattern (wrong local minimum) rather than the stored pattern (expected local minimum). Hopfield networks also provide a model for understanding human memory.

Structure



A Hopfield net with four units.

The units in Hopfield nets are binary threshold units, i.e. the units only take on two different values for their states and the value is determined by whether or not the units' input exceeds their threshold. Hopfield nets normally have units that take on values of 1 or -1, and this convention will be used throughout this article. However, other literature might use units that take values of 0 and 1.

Every pair of units *i* and *j* in a Hopfield network has a connection that is described by the connectivity weight w_{ij} . In this sense, the Hopfield network can be formally described as a complete undirected graph G=<V,f>, where V is a set of McCulloch-Pitts neurons and f is a function that links pairs of units to a real value, the connectivity weight.

The connections in a Hopfield net typically have the following restrictions:

- $w_{ii}=0$ (no unit has a connection with itself)
- $w_{ij}=w_{ji}$ (connections are symmetric)

The constraint that weights are symmetric guarantees that the energy function decreases monotonically while following the activation rules. A network with asymmetric weights may exhibit some periodic or chaotic behavior; however, Hopfield found that this behavior is confined to relatively small parts of the phase space and does not impair the network's ability to act as a content-addressable associative memory system.

Updating

Updating one unit (node in the graph simulating the artificial neuron) in the Hopfield network is performed using the following rule:

$$s_i \leftarrow \begin{cases} +1 & \text{if } \sum_j w_{ij} s_j \geq \theta_i, \\ -1 & \text{otherwise.} \end{cases}$$

where:

- w_{ij} is the strength of the connection weight from unit j to unit i (the weight of the connection).
- s_j is the state of unit j.
- θ_i is the threshold of unit i.

Updates in the Hopfield network can be performed in two different ways:

- **Asynchronous**: Only one unit is updated at a time. This unit can be picked at random, or a pre-defined order can be imposed from the very beginning.
- **Synchronous**: All units are updated at the same time. This requires a central clock to the system in order to maintain synchronization. This method is viewed by some as less realistic, based on an absence of observed global clock influencing analogous biological or physical systems of interest.

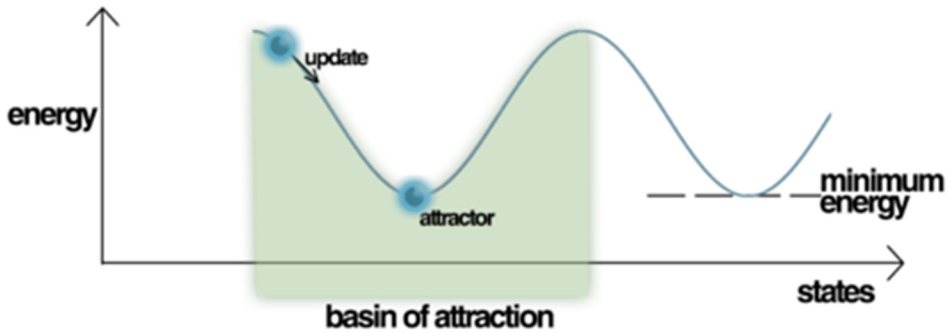
Neurons "attract or repel each other" in state-space

The weight between two units has a powerful impact upon the values of the neurons. Consider the connection weight w_{ij} between two neurons i and j. If $w_{ij}>0$, the updating rule implies that:

- when $s_j=1$, the contribution of j in the weighted sum is positive. Thus, s_j is pulled by j towards its value $s_j=1$
- when $s_j=10$, the contribution of j in the weighted sum is negative. Then again, s_j is pushed by j towards its value $s_j=-1$

Thus, the values of neurons i and j will converge if the weight between them is positive. Similarly, they will diverge if the weight is negative.

Energy



Energy Landscape of a Hopfield Network, highlighting the current state of the network (up the hill), an attractor state to which it will eventually converge, a minimum energy level and a basin of attraction shaded in green. Note how the update of the Hopfield Network is always going down in Energy.

Hopfield nets have a scalar value associated with each state of the network, referred to as the "energy", E, of the network, where:

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

This quantity is called "energy" because it either decreases or stays the same upon network units being updated. Furthermore, under repeated updating the network will eventually converge to a state which is a local minimum in the energy function (which is considered to be a Lyapunov function). Thus, if a state is a local minimum in the energy function it is a stable state for the network. Note that this energy function belongs to a general class of models in physics under the name of Ising models; these in turn are a special case of Markov networks, since the associated probability measure, the Gibbs measure, has the Markov property.

Initialization and running

Initialization of the Hopfield Networks is done by setting the values of the units to the desired start pattern. Repeated updates are then performed until the network converges to an attractor pattern. Convergence is generally assured, as Hopfield proved that the attractors of this nonlinear dynamical system are stable, not periodic or chaotic as in some other systems. Therefore, in the context of Hopfield Networks, an attractor pattern is a final stable state, a pattern that cannot change any value within it under updating.

Training

Training a Hopfield net involves lowering the energy of states that the net should "remember". This allows the net to serve as a content addressable memory system, that is to say, the network will converge to a "remembered" state if it is given only part of the state. The net can be used to recover from a distorted input to the trained state that is most similar to that input. This is called associative memory because it recovers memories on the basis of similarity. For example, if we train a Hopfield net with five units so that the state (1, -1, 1, -1, 1) is an energy minimum, and we give the network the state (1, -1, -1, -1, 1) it will converge to (1, -1, 1, -1, 1). Thus, the network is properly trained when the energy of states which the network should remember are local minima.

- Note: In contrast to Perceptron training, the thresholds of the neurons are never updated.

Learning rules

There are various different learning rules that can be used to store information in the memory of the Hopfield Network. It is desirable for a learning rule to have both of the following two properties:

- *Local*: A learning rule is *local* if each weight is updated using information available to neurons on either side of the connection that is associated with that particular weight.
- *Incremental*: New patterns can be learned without using information from the old patterns that have been also used for training. That is, when a new pattern is used for training, the new values for the weights only depend on the old values and on the new pattern.[4]

These properties are desirable, since a learning rule satisfying them is more biologically plausible. For example, since the human brain is always learning new concepts, one can reason that human

learning is incremental. A learning system that were not incremental would generally be trained only once, with a huge batch of training data.

Hebbian learning rule for Hopfield networks

The Hebbian Theory was introduced by Donald Hebb in 1949, in order to explain "associative learning", in which simultaneous activation of neuron cells leads to pronounced increases in synaptic strength between those cells. It is often summarized as "Neurons that fire together, wire together. Neurons that fire out of sync, fail to link".

The Hebbian rule is both local and incremental. For the Hopfield Networks, it is implemented in the following manner, when learning n binary patterns:

$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^{n} \epsilon_i^\mu \epsilon_j^\mu$$

where ϵ_i^μ represents bit i from pattern μ.

If the bits corresponding to neurons i and j are equal in pattern μ, then the product $\epsilon_i^\mu \epsilon_j^\mu$ will be positive. This would, in turn, have a positive effect on the weight w_{ij} and the values of i and j will tend to become equal. The opposite happens if the bits corresponding to neurons i and j are different.

Spurious patterns

Patterns that the network uses for training (called *retrieval states*) become attractors of the system. Repeated updates would eventually lead to convergence to one of the retrieval states. However, sometimes the network will converge to spurious patterns (different from the training patterns). The energy in these spurious patterns is also a local minimum. For each stored pattern x, the negation -x is also a spurious pattern.

A spurious state can also be a linear combination of an odd number of retrieval states. For example, when using 3 patterns μ_1, μ_2, μ_3, one can get the following spurious state:

$$\epsilon_i^{\text{mix}} = \pm \operatorname{sgn}(\pm \epsilon_i^{\mu_1} \pm \epsilon_i^{\mu_2} \pm \epsilon_i^{\mu_3})$$

Spurious patterns that have an even number of states cannot exist, since they might sum up to zero

Capacity

The Network capacity of the Hopfield network model is determined by neuron amounts and connections within a given network. Therefore, the number of memories that are able to be stored is dependent on neurons and connections. Furthermore, it was shown that the recall accuracy between vectors and nodes was 0.138 (approximately 138 vectors can be recalled from storage for every 1000 nodes) (Hertz et al., 1991). Therefore, it is evident that many mistakes will occur if one tries to store

a large number of vectors. When the Hopfield model does not recall the right pattern, it is possible that an intrusion has taken place, since semantically related items tend to confuse the individual, and recollection of the wrong pattern occurs. Therefore, the Hopfield network model is shown to confuse one stored item with that of another upon retrieval. Perfect recalls and high capacity, >0.14, can be loaded in the network by Storkey learning method. Ulterior models inspired by the Hopfield network were later devised to raise the storage limit and reduce the retrieval error rate

Human memory

The Hopfield model accounts for associative memory through the incorporation of memory vectors. Memory vectors can be slightly used, and this would spark the retrieval of the most similar vector in the network. However, we will find out that due to this process, intrusions can occur. In associative memory for the Hopfield network, there are two types of operations: auto-association and hetero-association. The first being when a vector is associated with itself, and the latter being when two different vectors are associated in storage. Furthermore, both types of operations are possible to store within a single memory matrix, but only if that given representation matrix is not one or the other of the operations, but rather the combination (auto-associative and hetero-associative) of the two. It is important to note that Hopfield's network model utilizes the same learning rule as Hebb's (1949) learning rule, which basically tried to show that learning occurs as a result of the strengthening of the weights by when activity is occurring.

Rizzuto and Kahana (2001) were able to show that the neural network model can account for repetition on recall accuracy by incorporating a probabilistic-learning algorithm. During the retrieval process, no learning occurs. As a result, the weights of the network remain fixed, showing that the model is able to switch from a learning stage to a recall stage. By adding contextual drift they were able to show the rapid forgetting that occurs in a Hopfield model during a cued-recall task. The entire network contributes to the change in the activation of any single node.

McCulloch and Pitts' (1943) dynamical rule, which describes the behavior of neurons, does so in a way that shows how the activations of multiple neurons map onto the activation of a new neuron's firing rate, and how the weights of the neurons strengthen the synaptic connections between the new activated neuron (and those that activated it). Hopfield would use McCulloch-Pitts's dynamical rule in order to show how retrieval is possible in the Hopfield network. However, it is important to note that Hopfield would do so in a repetitious fashion. Hopfield would use a nonlinear activation function, instead of using a linear function. This would therefore create the Hopfield dynamical rule and with this, Hopfield was able to show that with the nonlinear activation function, the dynamical rule will always modify the values of the state vector in the direction of one of the stored patterns.