Special Neural Network

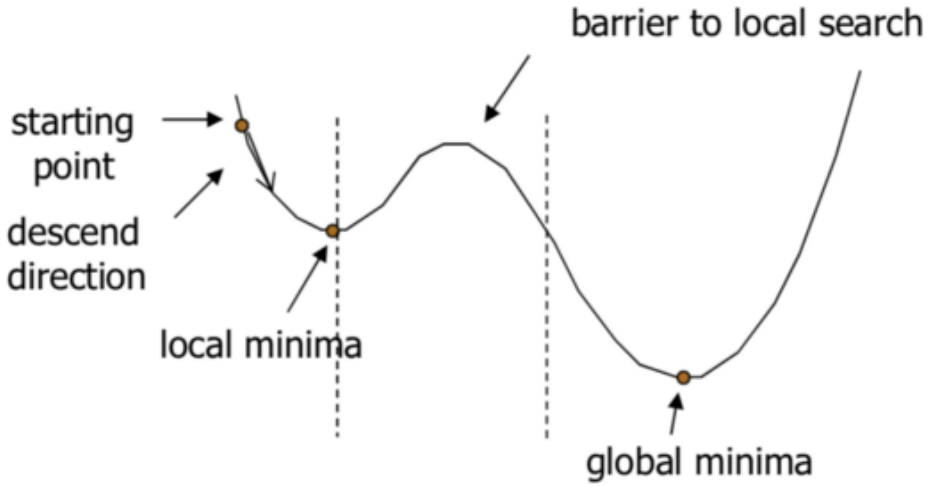
Simulated Annealing

- The Simulated Annealing algorithm is based upon Physical Annealing in real life. Physical Annealing is the process of heating up a material until it reaches an **annealing temperature** and then it will be **cooled down** slowly in order to change the material to a desired structure.
- When the material is hot, the molecular structure is weaker and is more susceptible to change.
- When the material cools down, the molecular structure is harder and is less susceptible to change.
- Another important part of this analogy is the following equation from Thermal Dynamics:

$$P(\Delta E) = e^{-\frac{\Delta E}{k*t}}$$

Simulated Annealing (cont.)

- Simulated Annealing (SA) mimics the Physical Annealing process but is used for optimizing parameters in a model.
- This process is very useful for situations where there are a lot of local minima such that algorithms like Gradient Descent would be stuck at.



Simulated Annealing (cont.)

- Example – Stone and Hill (Textbook - siva

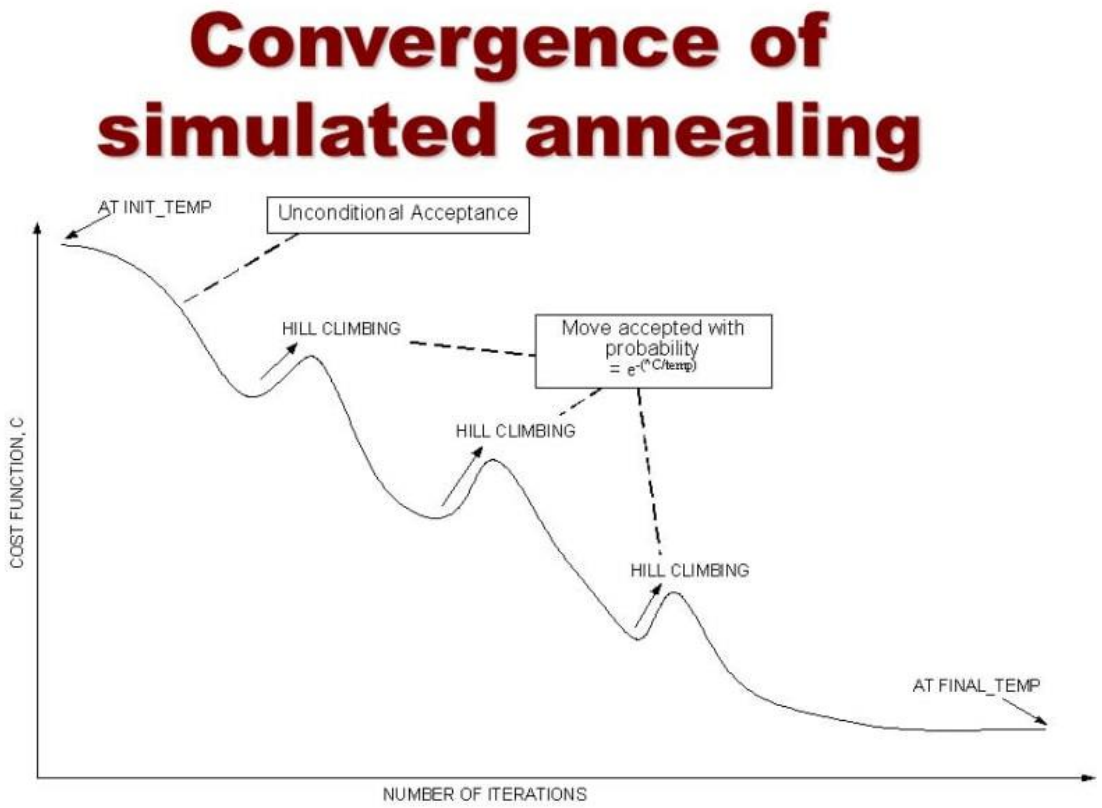


Fig- Reference - https://dev.to/swyx/unsupervised-learning-randomized-optimization-d2j

Boltzmann Machine

- When simulated annealing is applied to the Discrete Hopfield network then it becomes Boltzmann machine.
- It consists of as set of units Xi and Xj and set of bidirectional connections between pairs of units. Also, contains wij not equal to 0.
- This machine can be used as an associative memory then, wij = wji. Self-connection on weights also exist.
- For unit Xi, state xi may be 0 or 1.

Boltzmann Machine

- Objective is to maximize Consensus Function (CF);

$$CF = \sum_i \sum_{j \le i} w_{ij} x_i x_j$$

The maximum of the CF can be obtained by letting each unit attempt to change its state (alter between "1" and "0" or "0" and "1"). The change of state can be done either in parallel or sequential manner. However, in this case all the description is based on sequential manner. The consensus change when unit X_i changes its state is given by

$$\Delta CF(i) = (1 - 2x_i)\left(w_{ij} + \sum_{j \neq i} w_{ij} x_i\right)$$

where x_i is the current state of unit X_i. The variation in coefficient $(1 - 2x_i)$ is given by

$$(1 - 2x_i) = \begin{cases} +1, & X_i \text{ is currently off} \\ -1, & X_i \text{ is currently on} \end{cases}$$

Boltzmann Machine

If unit X_i were to change its activations then the resulting change in the CF can be obtained from the information that is local to unit X_i. Generally, X_i does not change its state, but if the states are changed, then this increases the consensus of the net. The probability of the network that accepts a change in the state for unit X_i is given by

$$AF(i, T) = \frac{1}{1 + \exp[-\Delta CF(i)/T]}$$

where T (temperature) is the controlling parameter and it will gradually decrease as the CF reaches the maximum value. Low values of T are acceptable because they increase the net consensus since the net accepts a change in state. To help the net not to stick with the local maximum, probabilistic functions are used widely.

Boltzmann Machine Architecture

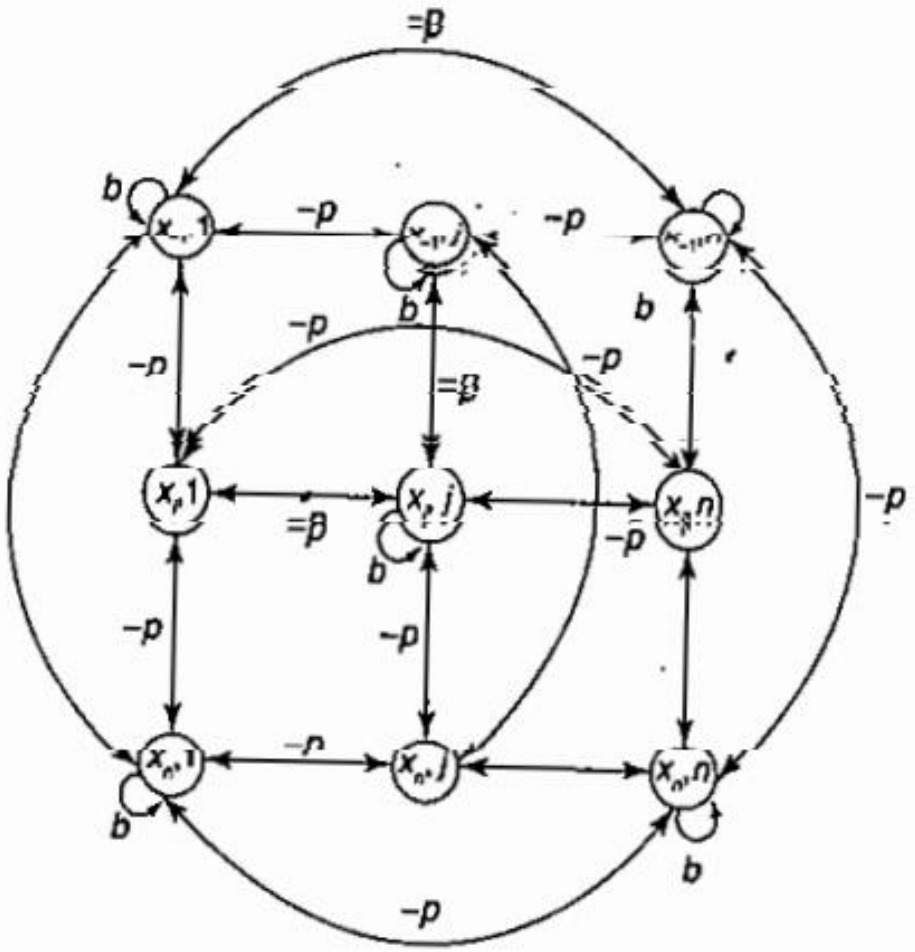
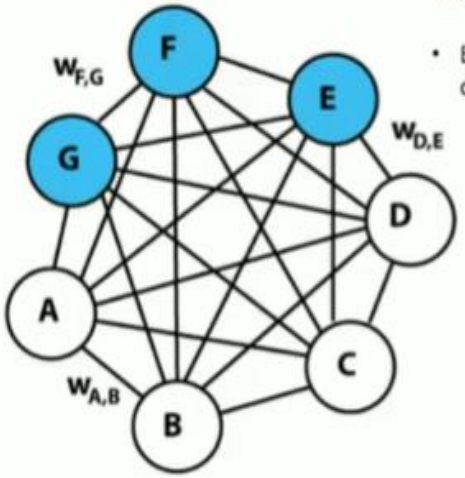


Figure 6-3 Architecture of Boltzmann machine.



- A few weights labelled.
- Each undirected edge represents dependency and is weighted with weight .
- In this example there are **3 hidden units** (blue) and **4 visible units** (white).

Boltzmann Machine Algorithm

It is assumed here that the units are arranged in a two-dimensional array. There are n^2 units. So, the weights between unit $X_{i,j}$ and unit $X_{I,J}$ are denoted by $w(i, j : I, J)$:

$$w(i,j : I,J) = \begin{cases} -p & \text{if } i \equiv I \text{ or } j \equiv J \text{ (but not both)} \\ b & \text{otherwise} \end{cases}$$

The testing algorithm is as follows.

Step 0: Initialize the weights representing the constraints of the problem. Also initialize control parameter T and activate the units.

Step 1: When stopping condition is false, perform Steps 2–8.

Step 2: Perform Steps 3–6 n^2 times. (This forms an epoch.)

Step 3: Integers I and J are chosen random values between 1 and n. (Unit $U_{I,J}$ is the current victim to change its state.)

Step 4: Calculate the change in consensus:

$$\Delta CF = (1 - 2X_{I,J}) \left[w(I,J : I,J) + \sum_{i,j \neq} \sum_{I,J} w(i,j : I,J) X_{I,J} \right]$$

Boltzmann Machine Algorithm

Step 5: Calculate the probability of acceptance of the change in state:

$$AF(T) = 1/1 + \exp[-(\Delta CF/T)]$$

Step 6: Decide whether to accept the change or not. Let R be a random number between 0 and 1. If $R < AF$, accept the change:

$X_{I,J} = 1 - X_{I,J}$ (This changes the state $U_{I,J}$.)

If $R \geq AF$, reject the change.

Step 7: Reduce the control parameter T.

$$T(\text{new}) = 0.95\ T(\text{old})$$

Step 8: Test for stopping condition, which is:

If the temperature reaches a specified value or if there is no change of state for specified number of epochs then stop, else continue.

The initial temperature should be taken large enough for accepting the change of state quickly. Also remember that the Boltzmann machine can be applied for various optimization problems such as traveling salesman problem.

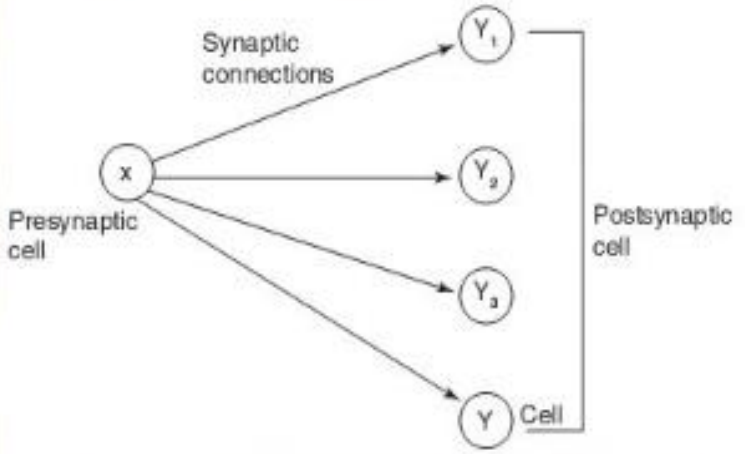
Cognitron Neural Network

Cognitron network was proposed by Fukushima in 1975.

The synaptic strength from cell X to cell Y is reinforced if and only if the following two conditions are true:

1. Cell X: presynaptic cell fires.
2. None of the postsynaptic cells present near cell Y fire stronger than Y.

The connection between presynaptic and postsynaptic cells is as follows:



"Principles of Soft Computing, 2nd Editio

Model of Cognitron Neural Network

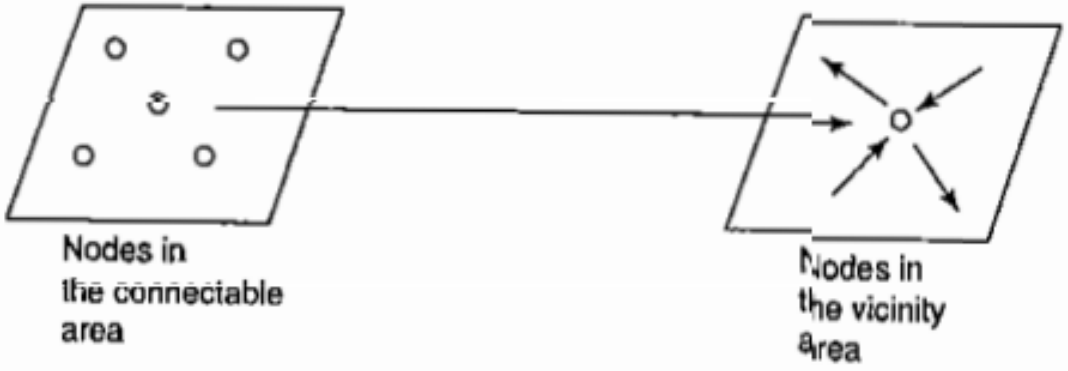


Figure 6-7 Model of a cognitron network.

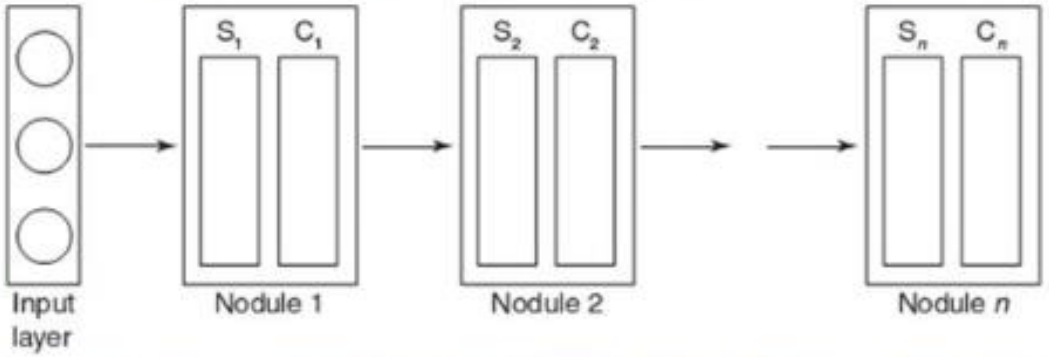
Application - Cognitron network can be used in neurophysiology and psychology, visual and auditory information processing system.

Neocognitron Neural Network (1)

- Neocognitron is a multilayer feedforward network model for visual pattern recognition.
- It is an extension of cognitron network.
- Neocognitron net can be used for recognizing handwritten characters.
- The algorithm used in cognitron and neocognitron is same, except that neocognitron model can recognize patterns that are position-shifted or shape-distorted.
- The cells used in neocognitron are of two types:
  - S-cell: Cells that are trained suitably to respond to only certain features in the previous layer.
  - C-cell: A C-cell displaces the result of an S-cell in space, i.e., sort of "spreads" the features recognized by the S-cell.

Neocognitron Neural Network (2)

The model of neocognitron network is as given below:



Training is found to progress layer by layer. The weights from the input units to the first layer are first trained and then frozen. Then the next trainable weights are adjusted and so on. When the net is designed, the weights between some layers are fixed as they are connection patterns.

Neocognitron Neural Network (3)

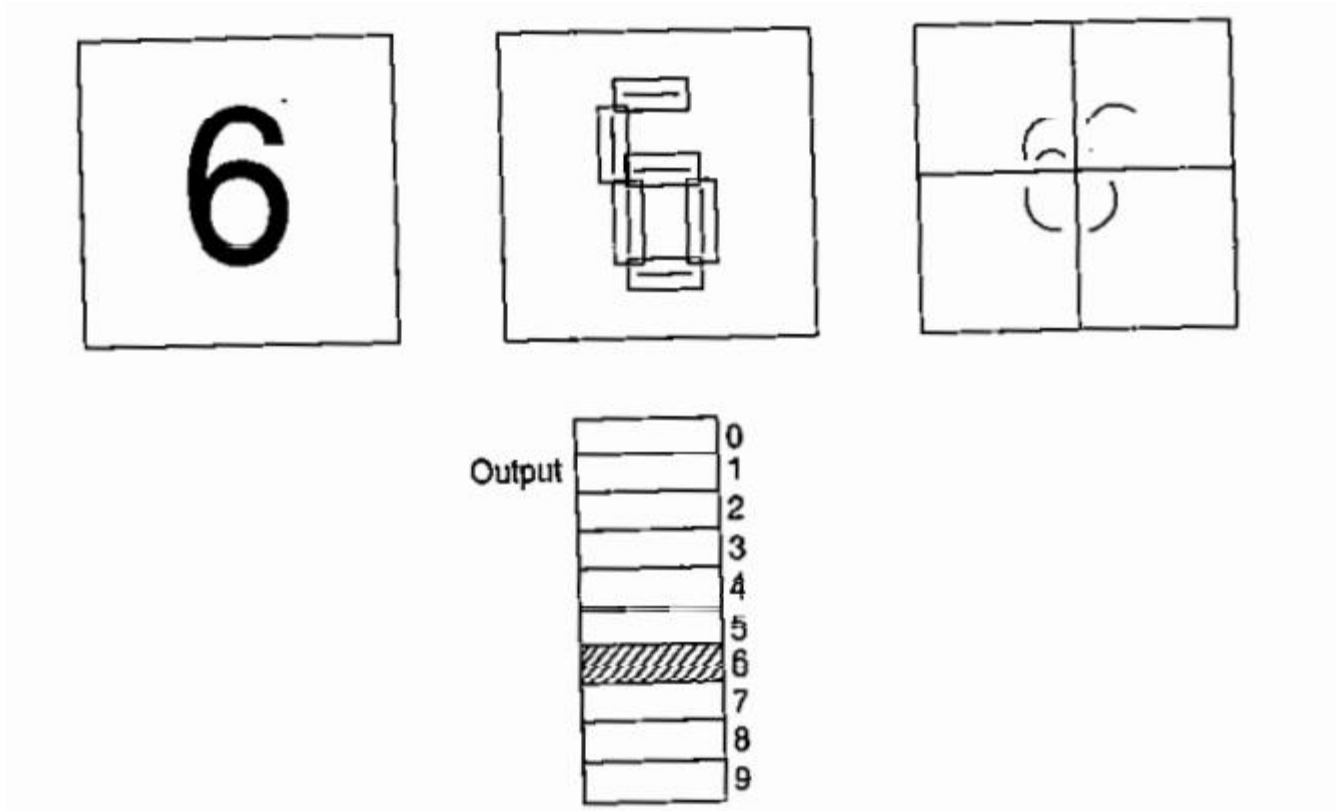


Figure 6-9 Spreading effect in neocognitron.