Computational Intelligence
Lecture 6: Associative Memory

Farzaneh Abdollahi

Department of Electrical Engineering

Amirkabir University of Technology

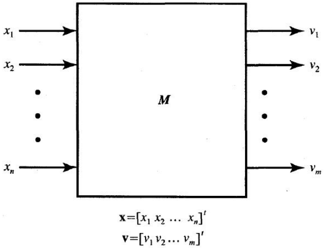
Fall 2011

Introduction

- Learning can be considered as a process of forming associations between related patterns.
- For example visual image may be associated with another visual image, or the fragrance of fresh-mown grass may be associated with a visual image of feeling
- Memorization of a pattern could be associating the pattern with itself
- Therefore, in such networks the input pattern cause an output pattern which may be similar to the input pattern or related to that.
- An important characteristic of the association is that an input stimulus which is similar to the stimulus for the association will invoke the associated response pattern.

- For example, if we learn to read music, so that we associate with fingering on a stringed instrument, we do not need to see the same form of musical note we originally learned
  - If the note is larger, or handwritten , we still can recognize and play.
  - So after learning it is expected to make a good guess and provide appropriate response
- Another example, ability to recognize a person either in person or form a photo even his/her appearance has been changed
- This is relatively difficult to program by a traditional computer algs.
- Associative memories belong to class of NN that learn according to a certain recording algs.
- They require information a priori and their connectivity matrices (weights) most often need to be formed in advance
- Writing into memory produces changes in the neural interconnections
- Reading of the stored info from memory named recall, is a transformation of input signals by the network

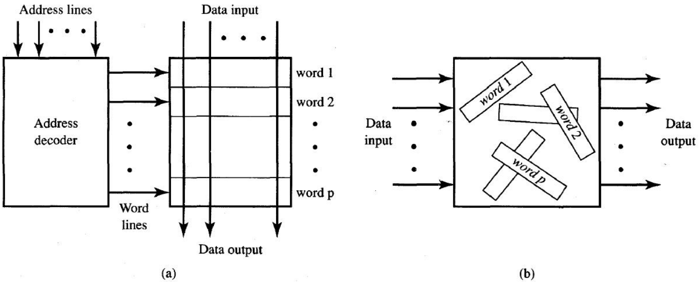
- Not usable addressing schemes exits in an associative memory
- All memory info is spatially distributed throughout the network
- The biological memory operates the same
- Associative memory enables a parallel search within a stored data
  - The purpose of search is to output one or all stored items that matches the search argument and retrieve it entirely or partially
- The fig. depicts a block diagram of an associative memory.



- The transformation is $v = M[x]$, M: a nonlinear matrix operator which has different meaning for each of memory models.

- For dynamic memories, M is time variable.
  - v is available in output at a later time than the input has been applied
- For a given memory model, M is usually expressed in terms of given prototype vectors that should be stored
- The algs of finding M are called recording or storage algs.
- The mapping in $v = M[x]$ preformed on x is called a retrieval.
- Retrieval may provide a desired/an undesired solution prototype
- To have efficient retrieval some mechanisms should be developed
- Assume there are p stored pairs: $x^{(i)} \rightarrow v^{(i)}$ for $i = 1, ..., p$
- If $x^{(i)} \neq v^{(i)}$ for $i = 1, ..., p$ it is called heteroassociative memory
- If $x^{(i)} = v^{(i)}$ for $i = 1, ..., p$ it is called autoassociative memory
- Obviously the mapping of a vector $x^{(i)}$ into itself cannot be of any significance
- A more realistic application of autoassociative memory is recovery of undistorted prototype in response to a distorted prototype vector.

- Associative memory which uses NN concepts may resemble digital computer memory
- Let us compare their difference:
  - Digital memory is address-addressable memory:
    - data have input and output lines
    - a word line access the entire row of binary cells containing word data bits.
    - activation takes place when the binary address is decoded by an address decoder.



Addressing modes for memories: (a) address-addressable memory and (b) content-addressable memory.

- Associative memory is content addressable memory
  - the words are accessed based on the content of the key vector
  - When the network is excited by a portion of the stored data, the efficient response of autoassociative memory is the completed $x^{(i)}$ vector
  - In hetroassociative memory the content of $x^{(i)}$ provides the stored vector $v^{(i)}$
  - There is no storage for prototype $x^{(i)}$ or $v^{(i)}$ at any location of network
  - The entire mapping is distributed in the network.
  - The mapping is implemented through dense connections, feedback or/and a nonlinear thresholding operation
- Associative network memory can be
  - Static: networks recall an output response after an input has been applied in one feedforward pass, and, theoretically, without delay. They were termed instantaneous
  - Dynamic: memory networks produce recall as a result of output/input feedback interaction, which requires time.

- **Static memory**
  - implement a feedforward operation of mapping without a feedback, or recursive update, operation.
  - They are sometimes also called non-recurrent
  - The mapping can be expressed as f

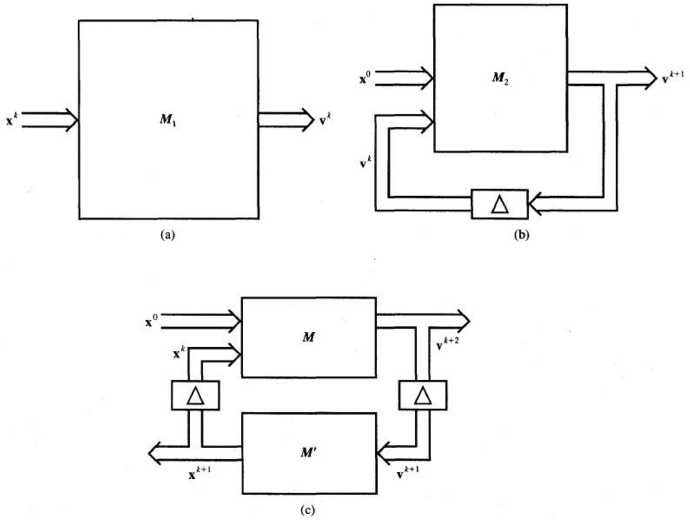
    $$v^k = M_1[x^k]$$

    where k: index of recursion, M_1 operator symbol
- **Dynamic memory**
  - exhibit dynamic evolution in the sense that they converge to an equilibrium state according to the recursive formula

    $$v^{k+1} = M_2[x^k, v^k]$$

  - This is a nonlinear difference equation.
  - Hopfield model is an example of a recurrent network for which the input x^0 is used to initialize v^o, i.e., $x^0 = v^0$, and the input is then removed.
  - So the formula will be simplified to

    $$v^{k+1} = M_2[v^k]$$



Block diagram representation of associative memories: (a) feedforward network, (b) recurrent autoassociative network, and (c) recurrent heteroassociative network.

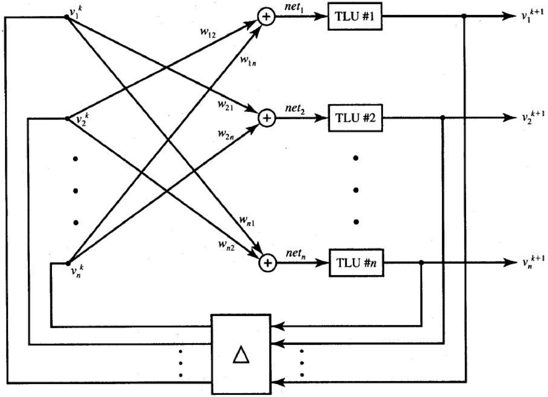
Hopfield Networks

- It is a special type of Dynamic Network that $v^0 = x^0$, i.e., $v^{k+1} = M[v^k]$
- It is a single layer feedback network which was first introduced by John Hopfield (1982,1988)
- Neurons are with either a hard-limiting activation function or with a continuous activation function (TLU)
- In MLP:
  - The weights are updated gradually by teacher-enforced which was externally imposed rather than spontaneous
  - The FB interactions within the network ceased once the training had been completed.
  - After training, output is provided immediately after receiving input signal

- In FB networks:
  - the weights are usually adjusted spontaneously.
  - Typically, the learning of dynamical systems is accomplished without a teacher.
  - i.e., the adjustment of system parameters does not depend on the difference between the desired and actual output value of the system during the learning phase.
  - To recall information stored in the network, an input pattern is applied, and the network's output is initialized accordingly.
  - Next, the initializing pattern is removed and the initial output forces the new, updated input through feedback connections.
  - The first updated input forces the first updated output. This, in turn, produces the second updated input and the second updated response.
  - The transition process continues until no new updated responses are produced and the network has reached its equilibrium.
- $\therefore$ These networks should fulfill certain assumptions that make the class of networks stable and useful, and their behavior predictable in most cases.

- FB in the network
  - allows for great reduction of the complexity.
  - Deal with recalling noisy patterns
- Hopfield networks can provide
  - associations or classifications
  - optimization problem solution
  - restoration of patterns
  - In general, as with perceptron networks, they can be viewed as mapping networks
- One of the inherent drawbacks of dynamical systems is:
  - The solutions offered by the networks are hard to track back or to explain.

- w_{ij}: the weight value connecting the output of the jth neuron with the input of the ith neuron
- $W = \{w_{ij}\}$ is weight matrix
- $V = [v_1, ..., v_n]^T$ is output vector
- $net = [net_1, .., net_n]^T = Wv$
- $v_i^{k+1} = sgn(\sum_{j=1}^{n} w_{ij} v_j^k)$



- W is defined:

$$W = \begin{bmatrix} 0 & w_{12} & w_{13} & \dots & w_{1n} \\ w_{21} & 0 & w_{23} & \dots & w_{2n} \\ w_{31} & w_{32} & 0 & \dots & w_{3n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ w_{n1} & w_{n2} & w_{n3} & \dots & 0 \end{bmatrix}$$

- It is assumed that W is symmetric, i.e., $w_{ij} = w_{ji}$
- $w_{ii} = 0$, i.e., There is no self-feedback
- The output is updated asynchronously. This means that
  - For a given time, only a single neuron (only one entry in vector V) is allowed to update its output
  - The next update in a series uses the already updated vector V.

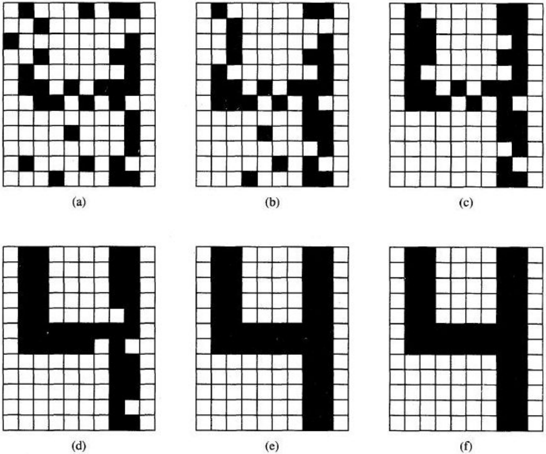
- Example: In this example output vector is started with initial value V^0, the updated by m, p and q respectively:

$$\begin{aligned} V^1 &= [v_1^0 \; v_2^0 \; \dots \; v_m^1 \; v_p^0 \; v_q^0 \; \dots \; v_n^0]^T \\ V^2 &= [v_1^0 \; v_2^0 \; \dots \; v_m^1 \; v_p^2 \; v_q^0 \; \dots \; v_n^0]^T \\ V^3 &= [v_1^0 \; v_2^0 \; \dots \; v_m^1 \; v_p^2 \; v_q^3 \; \dots \; v_n^0]^T \end{aligned}$$

- The vector of neuron outputs V in n-dimensional space.
- The output vector is one of the vertices of the n-dimensional cube $[-1, 1]$ in E^n space.
- The vector moves during recursions from vertex to vertex, until it stabilizes in one of the 2^n vertices available.
- To evaluate the stability property of the dynamical system of interest, let us study a so-called computational energy function.
- This is a function usually defined in n-dimensional output space v $E = -\frac{1}{2} v^T W v$

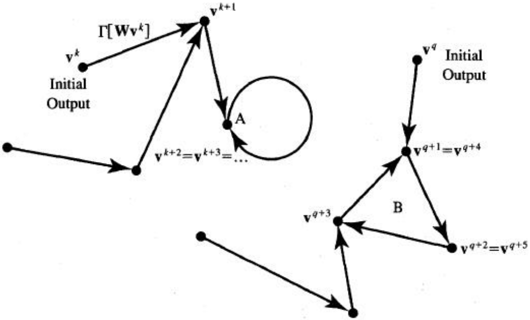
Example

- A 10×12 bit map of black and white pixels representing the digit 4.
- The initial, distorted digit 4 with 20% of the pixels randomly reversed.



Example of recursive asynchronous update of corrupted digit 4: (a) $k = 0$,

- State transition map for a memory network is shown
- Each node of the graph is equivalent to a state and has one and only one edge leaving it.
- If the transitions terminate with a state mapping into itself, A, then the equilibrium A is fixed point.
- If the transitions end in a cycle of states, B, then we have a limit cycle solution with a certain period.
  - The period is defined as the length of the cycle. (3 in this example)



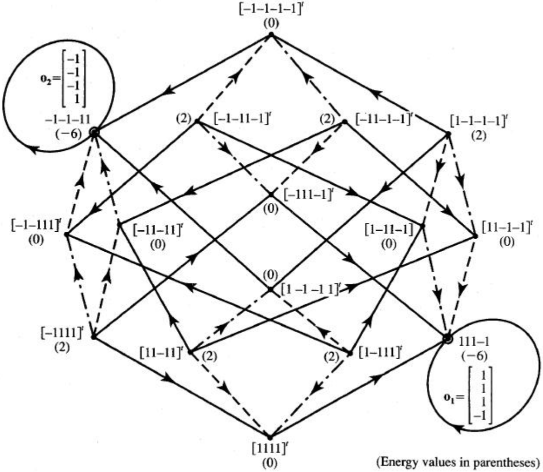
- Energy function was defined as $E = -\frac{1}{2}v^T Wv$
- In bipolar notation the complement of vector v is $-v$
- $\therefore E(-v) = -\frac{1}{2}v^T Wv$
- $E(v) = E(-v) \rightsquigarrow \min E(v) = \min E(-v)$
- The memory transition may end up to v as easily as $-v$
- The similarity between initial output vector and v and $-v$ determines the convergence.
- It has been shown that synchronous state updating algorithm may yield persisting cycle states consisting of two complimentary patterns (Kamp and Hasler 1990)

- Example 1: Consider $W = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}, v^0 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$
- $v^1 = sgn(Wv) = sgn(\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- $v^2 = sgn(Wv) = sgn(\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$
- $v^0 = v^1 \rightsquigarrow$ It provides a cycle of two states rather than a fix point
- Example 2: Consider $W = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$
- The energy function becomes

$$E(v) = -\frac{1}{2}[v_1 \ v_2 \ v_3 \ v_4] \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} =$$

$$-v_1(v_2 + v_3 - v_4) - v_2(v_3 - v_4) + v_3 v_4$$

- It can be verifying that all possible energy levels are $-6,\ 0,\ 2$
- Each edge of the state diagram shows a single asynchronous state transition.
- Energy values are marked at cube vertexes
- By asynchronous updates, finally the energy ends up to its min value -6.



Energy levels and state transitions for the network

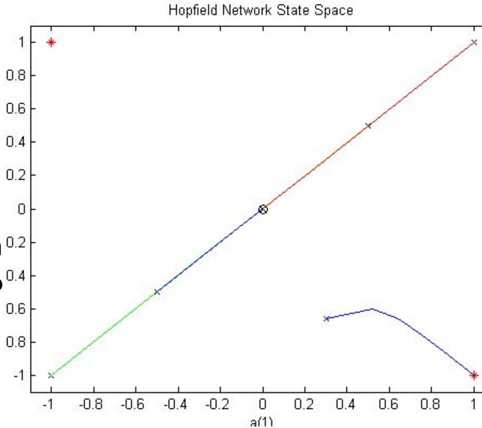
- Applying synchronous update:
  - Assume $v^0 = [1 \ -1 \ 1 \ 1]^T$
  - $v^1 = sgn(Wv^0) = [-1 \ 1 \ -1 \ -1]$
  - $v^2 = sgn(Wv^1) = [1 \ -1 \ 1 \ 1] = v^0$
- **Storage Algorithm**
  - For bipolar prototype vectors: the weight is calculated: $W = \sum_{m=1}^{p} s^{(m)} s^{(m)T} - PI$ or $w_{ij} = (1 - \delta_{ij}) \sum_{m=1}^{p} s_i^{(m)} s_j^{(m)}$
  - δ_{ij} is Kronecker function: $\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$
  - If the prototype vectors are unipolar, the the memory storage alg. is modified as $w_{ij} = (1 - \delta_{ij}) \sum_{m=1}^{p} (2s_i^{(m)} - 1)(2s_j^{(m)} - 1)$
  - The storage rule is invariant with respect to the sequence of storing pattern
  - Additional patterns can be added at any time by superposing new incremental weight matrices

Problems

- Uncertain Recovery
  - Heavily overloaded memory ($p/n > 50\%$) may not be able to provide error-free or efficient recovery of stored pattern.
  - There are some examples of convergence that are not toward the closest memory as measured with the HD value
- Undesired Equilibria
  - Spurious points are stable equilibria with minimum energy that are additional to the patterns we already stored.
  - The undesired equilibria may be unstable. Therefore, any noise in the system will move the network out of them.

Example

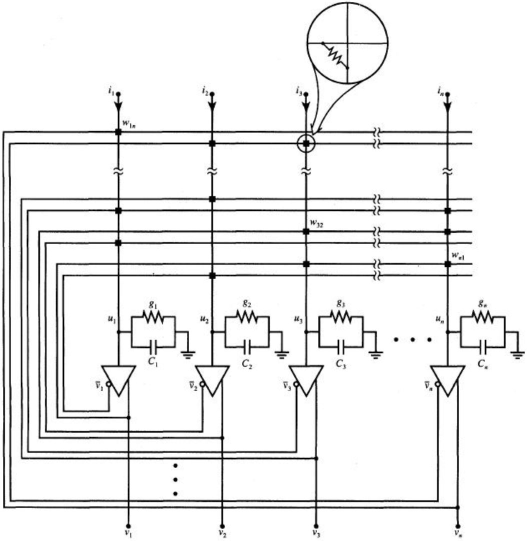
- Assume two patterns are stored in the memory as shown in Fig.
- The input vector converges to the closer pattern
- BUT if the input is exactly between the two stable points, it moves into the center of the state space!



Gradient-Type Hopfield Network

- Gradient-type neural networks are generalized Hopfield networks in which the computational energy decreases continuously in time.
- Gradient-type networks converge to one of the stable minima in the state space.
- The evolution of the system is in the general direction of the negative gradient of an energy function.
- Typically, the network energy function is equivalent to a certain objective (penalty) function to be minimized.
- These networks are examples of nonlinear, dynamical, and asymptotically stable systems.
- They can be considered as a solution of an optimization problem.

- The model of a gradient-type neural system using electrical components is shown in Fig.
- It has n neurons,
- each neuron mapping its input voltage u_i into the output voltage v_i through the activation function $f(u_i)$,
- $f(u_i)$ is the common static voltage transfer characteristic (VTC) of the neuron.



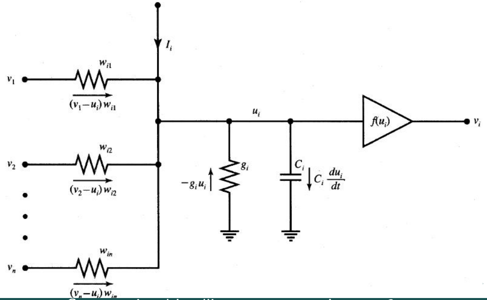
- Conductance w_{ij} connects the output of the jth neuron to the input of the ith neuron.
- The inverted neuron outputs $\bar{v}_i$ representing inverting output is applied to avoid negative conductance values w_{ij}
- Note that in Hopefield networks:
  - $w_{ij} = w_{ji}$
  - $w_{ii} = 0 \rightsquigarrow$, the outputs of neurons are not connected back to their own inputs
- Capacitances C_i, for $i = 1, 2, ..., n$, are responsible for the dynamics of the transients in this model.

- KCL equ. for each node is

$$i_i + \sum_{j \neq i}^{n} W_{ij} v_j - u_i(\sum_{j \neq i}^{n} w_{ij} + g_i) = C_i \frac{du}{dt} \tag{1}$$

- Considering $G_i = \sum_{j=1}^{n} w_{ij} + g_i$, $C = diag[C_1, C_2, ..., C_n]$, $G = [G_1, ..., G_n]$, the output equ. for whole system is

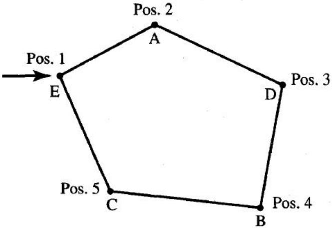
$$\begin{aligned} C\frac{du}{dt} &= Wv(t) - Gu(t) + I \\ v(t) &= f[u(t)] \end{aligned} \tag{2}$$



- The energy to be minimized is
  $E(v) = -\frac{1}{2}v^t Wv - iv + \frac{1}{\lambda}\sum_{i=}^{n} G_i \int_0^{v_i} f_i^{-1}(z)dz$
- The Hopfield networks can be applied for optimization problems.
- The challenge will be defining W and I s.t. fit the dynamics and objective of the problem to (2) and above equation.

Example: Traveling Salesman Tour Length [1]

- The problem is min tour length through a number of cities with only one visit of each city
- If n is number of cities $(n-1)!$ distinct path exists
- Let us use Hopefiled network to find the optimum solution
- We are looking to find a matrix shown in the fig.
  - n rows are the cities
  - n columns are the position of the salesman
  - each city/position can take 1 or 0
  - $v_{ij} = 1$ means salesman in its jth position is in ith city
- The network consists n^2 unipolar neurons
- Each city should be visited once $\rightsquigarrow$ only one single 1 at each row and column



City, y \ Position, x	1	2	3	4	5	
A	0	1	0	0	0	5
B	0	0	0	1	0	4
C	0	0	0	0	1	3
D	0	0	1	0	0	2
E	1	0	0	0	0	1

- We should define w and i such that the energy of the Hopfield network represent the objective we are looking for
- Recall the energy of Hopfiled network:
  $E(v) = -\frac{1}{2}\sum_{Xi}\sum_{Yj} w_{Xi,Y_j} v_{Xi} v_{Yj} - \sum_{Xi} i_{Xi} v_{Xi}$
  - The last term is omitted for simplicity
- Let us express our objective in math:
  $E_1 = A\sum_X \sum_i \sum_j v_{Xi} v_{Xj}$ for $i \neq j$
  $E_2 = B\sum_i \sum_X \sum_Y v_{Xi} v_{Yi}$ for $X \neq Y$
- E_1 be zero $\rightsquigarrow$ each row has at most one 1
- E_2 be zero $\rightsquigarrow$ each column has at most one 1
- $E_3 = C(\sum_X \sum_i v_{Xi} - n)^2$
- E_3 guarantees that there is at least one 1 at each column and row.
- $E_4 = D\sum_X \sum_Y \sum_i d_{XY} v_{Xi}(v_{Y,i+1} + v_{Y,i-1}), \quad X \neq Y$
- E_4 represents minimizing the distances
- d_{XY} is distance between city X and Y

- Recall the energy of Hopfiled network:
  $E(v) = -\frac{1}{2}\sum_{Xi}\sum_{Yj} w_{Xi,Y_j} v_{Xi} v_{Yj} - \sum_{Xi} i_{Xi} v_{Xi}$
- The weights can be defined as follows
- $\therefore$
  $W_{Xi,Yj} = -2A\delta_{XY}(1-\delta_{ij}) - 2B\delta_{ij}(1-\delta_{XY}) - 2C - 2Dd_{xy}(\delta_{j,i+1} + \delta_{j,j-1})$
- δ_{ij} is Kronecker function: $\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$
- $i_{Xi} = 2Cn$
- Positive consts A, B, C, and D are selected heuristically

J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biolog. Cybern*, vol. 52 , pp. 141–154, 1985.