ONLY GOD

1402/2023

Neural Network & Deep Learning

Single-Layer Recurrent Network

CSE & IT Department

School of ECE

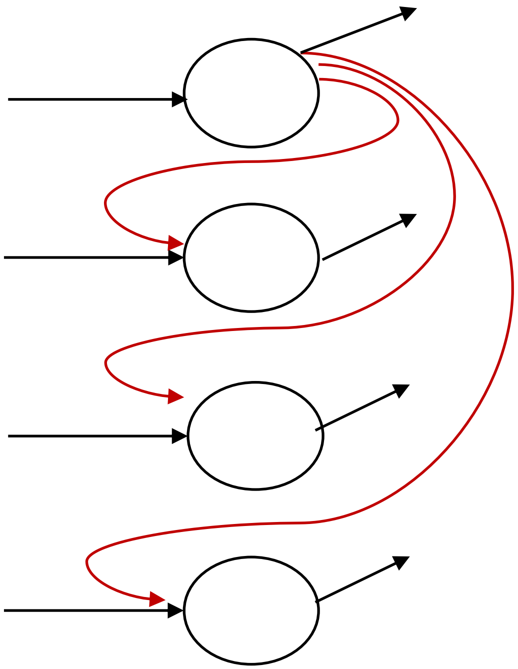
Shiraz University

Recurrent Neural Networks

Recurrent Neural Nets (RNN)

mansoori @shirazu.ac.ir

RNNs are characterized by



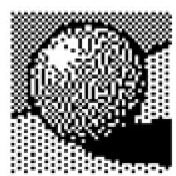
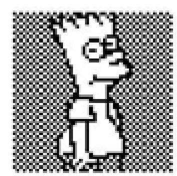
- Connection graph of network has cycles
  - Output of a neuron can influence inputs
- There are no natural input and output nodes
- Initially, each neuron has a given input state
- Neurons change state, using some update rules
- Network evolves until some stable situations reach
- Resulting state is output of network

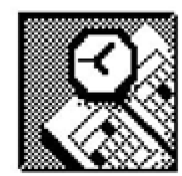
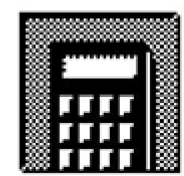
Pattern Recognition

mansoori@shirazu.ac.ir

RNNs can be used for pattern recognition

- Stable states represent patterns to be recognized
- Initial state is a noisy or mutilated version of one pattern
- Recognition process consists of network evolving from its initial state to a stable state
- Some patterns:

- Noisy pattern

Recognized pattern

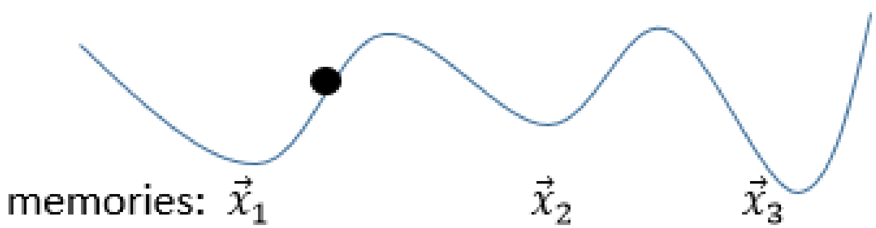
Physical Analogy

mansoori@shirazu.ac.ir

A Physical Analogy with Memory

- Bowl and ball: a system with a stable energy state
- Ball-bowl system settles in an energy minimum at equilibrium

- Resting state is a stable state because system remains there, so it remembers bottom of bowl
- Ball will come to rest at local memory closest to its initial position (with minimum energy)



- So, system can store a set of patterns (with minimum energy) and recalls that which is closest to its initial cue

Store and Retrieve in RNN

mansoori@shirazu.ac.ir

- In RNNs
  - System is completely described by
    - a state vector $\vec{y}(t) =< y_1(t), y_2(t), \dots, y_n(t) >$
    - a scalar variable $E(t)$ as energy function
  - A set of stable states $\vec{y}_1, \vec{y}_2, \dots, \vec{y}_m$ associated with energy minima, $E_1, E_2, \dots, E_m$ are defined as stored patterns or memories (storing process)
  - System evolves in time from any arbitrary starting state $\vec{y}(0)$ with initial energy $E(0)$ to one of stable state $\vec{y}_i$ with energy E_i when $E_i \leq E(0)$ (recall process)

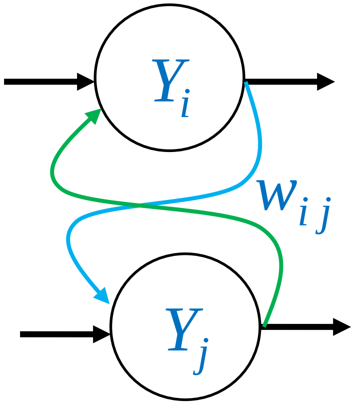
Store and Retrieve in RNN

mansoori@shirazu.ac.ir

- In RNNs
  - A state $\vec{y}_k$ is called a stable state, if and only if, update rule does not lead to a different state: $\vec{y}_k(t+1) = \vec{y}_k(t)$
  - In bipolar encoding, if $\vec{y}_k$ is stable, $-\vec{y}_k$ is also stable
  - Some combinations of stable states can also be stable
  - Moreover, there can be more complicated stable states that are not related to the stored states
  - Besides desired stable states, network can have additional undesired (spurious) stable states

Hebb rule in RNN

mansoori@shirazu.ac.ir



$$\text{Hebb rule: } \begin{cases} \text{if } w_{ij} > 0, \;\; Y_i \text{ and } Y_j \text{ tend to take } y_i = y_j \\ \text{if } w_{ij} < 0, \;\; Y_i \text{ and } Y_j \text{ tend to take } y_i \neq y_j \end{cases}$$

Internode energy: $e_{ij} = -w_{ij}\, y_i\, y_j$: for bipolar representation

$$\begin{cases} \text{if } w_{ij} > 0\,, \;\; e_{ij} = \begin{cases} -w_{ij} < 0\,, \text{when } y_i = y_j & \Rightarrow \;\text{energy : minimum} \\ w_{ij} > 0\,, \;\text{when } y_i \neq y_j & \Rightarrow \;\text{energy : maximum} \end{cases} \\ \text{if } w_{ij} < 0\,, \;\; e_{ij} = \begin{cases} w_{ij} < 0\,, \text{when } y_i \neq y_j & \Rightarrow \;\text{energy : minimum} \\ -w_{ij} > 0\,, \text{when } y_i = y_j & \Rightarrow \;\text{energy : maximum} \end{cases} \end{cases}$$

Energy in RNN

mansoori@shirazu.ac.ir

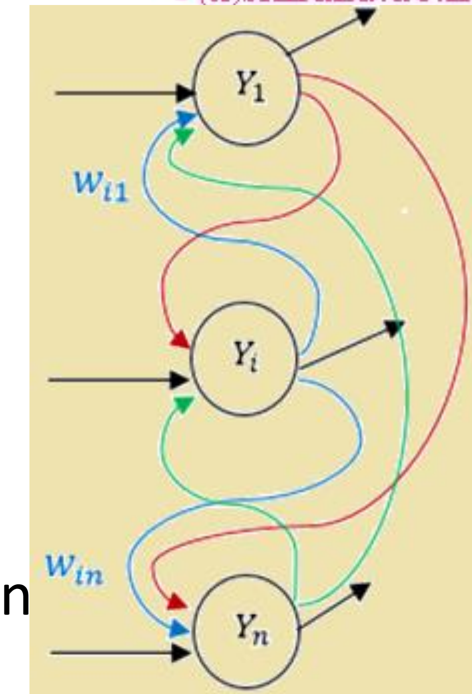
- Energy of whole network

$$E = \sum_i \sum_j e_{ij} = -\sum_i \sum_j w_{ij}\, y_i\, y_j \xRightarrow{w_{ij}=w_{ji}}$$

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij}\, y_i\, y_j = -\frac{1}{2} \vec{y}^T W \vec{y}$$

Since W is symmetric: $E = -\sum_i \sum_{j<i} w_{ij}\, y_i\, y_j$

At time t: $E(t) = -\sum_i \sum_{j<i} w_{ij}\, y_i(t)\, y_j(t)$



- To show existence of stable state, suppose Y_k is chosen to be fired:

Energy before firing Y_k

$$E(t) = -\sum_i \sum_{\substack{j<i \\ j\neq k}} w_{ij}\, y_i(t)\, y_j(t) - \sum_i w_{ik}\, y_i(t)\, y_k(t) = S(t) - y_k(t)\, y\_in_k(t)$$

Energy after firing Y_k

$$E(t+1) = S(t+1) - y_k(t+1)\, y\_in_k(t+1) = S(t) - y_k(t+1)\, y\_in_k(t)$$

So, $\Delta E = E(t+1) - E(t) = -y\_in_k(t)\,\{y_k(t+1) - y_k(t)\} = -y\_in_k(t)\, \Delta y_k$

Energy in RNN

mansoori@shirazu.ac.ir

- $\Delta E = -y\_in_k(t)\,\Delta y_k$

- Since Y_k is chosen to be fired

$$\begin{cases} \text{if } y\_in_k(t) \geq 0 \;\Rightarrow y_k(t+1) = 1,\ \text{if } y_k(t) = \begin{cases} -1 \\ 1 \end{cases} \Rightarrow \Delta y_k \geq 0 \Rightarrow \Delta E \leq 0 \\ \text{if } y\_in_k(t) < 0 \;\Rightarrow y_k(t+1) = -1,\ \text{if } y_k(t) = \begin{cases} -1 \\ 1 \end{cases} \Rightarrow \Delta y_k \leq 0 \Rightarrow \Delta E \leq 0 \end{cases}$$

- So, net energy decreases or stays the same for any node selected to fire

- For lower bound of E (when $y_i = y_j$ for all nodes),

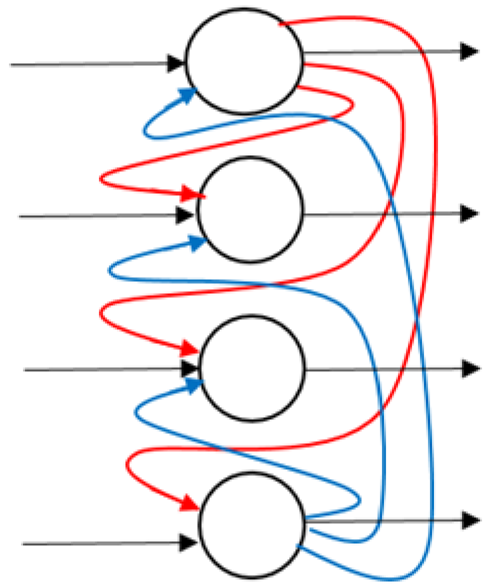
$E_{min} = -\sum_i \sum_{j<i} w_{ij}$

Hopfield Net

Discrete Hopfield Net

mansoori@shirazu.ac.ir

- Originally used as content-addressable memory
- A continues-valued version of it can be used for pattern association
- An iterative AAM net with bipolar encoding
- A fully interconnected recurrent net with no connection from a neuron to itself
- Each neuron continues to receive an external signal in addition to signals from other units



Discrete Hopfield Net

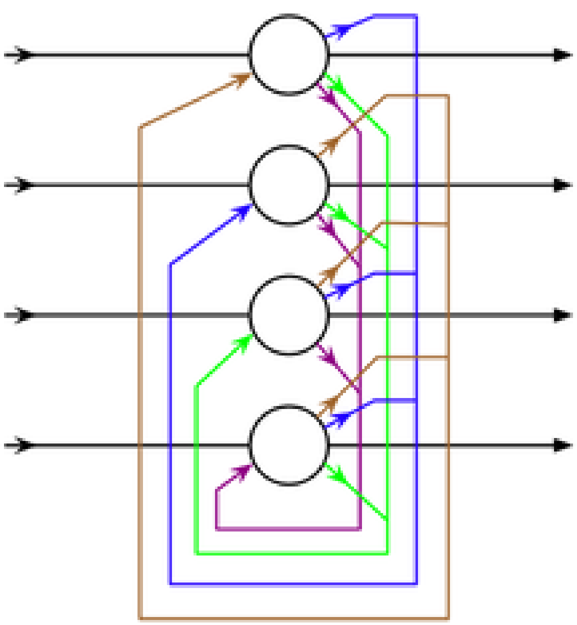
mansoori@shirazu.ac.ir

- If a Hopfield network has n neurons → state of network at time t is $\vec{y}(t) \in \{-1,1\}^n$ with components $y_i(t)$ that describe state of neuron Y_i at time t

- Since time is discrete ($t \in \mathcal{N}$), state of network at time $t+1$ will depend on sign of total input at time t

$$y_i(t+1) = f\big(y\_in_i(t)\big) = sgn(y\_in_i(t))$$

- Update strategies of net states:
  - Asynchronous update
  - Synchronous update

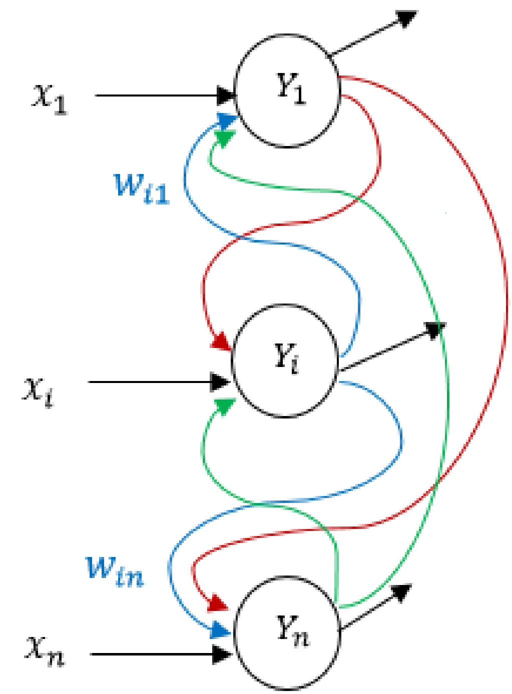


Weights of Hopfield Net

- Using Hebb rule for pattern association:

$$\vec{s}(p) =< s_1(p), \ldots, s_i(p), \ldots, s_n(p) >, \quad (p = 1, \ldots, P)$$

$$W = \{w_{ij}\}, \quad w_{ij} = \begin{cases} \sum_{p=1}^{P} s_i(p)\, s_j(p), & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$



- The weight matrix is symmetric ($w_{ij} = w_{ji}$) and has a zero diagonal

- Each neuron uses sign activation function:

$$y_i = f(y\_in_i) = sgn(y\_in_i) = \begin{cases} 1, & \text{if } y\_in_i > 0 \\ 0, & \text{if } y\_in_i = 0 \\ -1, & \text{if } y\_in_i < 0 \end{cases}$$

Hopfield Net State Update

mansoori@shirazu.ac.ir

- Asynchronous update rule
  - Only one neuron (selected randomly) at a time is allowed to change its state
  - Select neuron Y_i (randomly) for update

$$y\_in_i(t) = x_i + \sum_{j=1}^{n} y_j(t)\, w_{ji} \;= x_i + \vec{w}_{.i}{}^{T}\, \vec{y}$$

$$y_i(t+1) = \text{sgn}(y\_in_i(t)) = \text{sgn}(x_i + \vec{w}_{.i}{}^{T}\, \vec{y})$$

  - For each neuron Y_j except Y_i

$$y_j(t+1) = y_j(t)$$

- Synchronous update rule
  - All neurons are allowed to change their state simultaneously

$$\overrightarrow{y\_in}(t) = \vec{x} + W\, \vec{y}(t)$$

$$\vec{y}\,(t+1) = sgn\left(\overrightarrow{y\_in}(t)\right)$$

Recall in Hopfield Net

mansoori@shirazu.ac.ir

Recalling with Hopfield net (asynchronous update) for input vector $\vec{x}$

1. Set initial activations of net to external input vector

$$y_i = x_i \, , (i = 1, \ldots , n)$$

2. While activations of net are not converged
   2.1. For each neuron Y_i (randomly selected)
      2.1.1. Compute net input

$$y\_in_i = x_i + \sum_{j=1}^{n} y_i \, w_{ji} = x_i + \vec{w}_{.i}^{\,T} \, \vec{y}$$

      2.1.2. Determine output signal

$$y_i = f(y\_in_i) = sgn(y\_in_i)$$

      2.1.3. Broadcast value of y_i to all other neurons
3. Stop

- Input vector $\vec{x}$:
  - Is known if net converges to a stable state as a stored vector
  - Is unknown if net converges to a not stored vector

Recall in Hopfield Net

mansoori@shirazu.ac.ir

Example: $\vec{s} = \{<1,1,-1,-1>, <1,-1,1,-1>\}$

$$W = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 0 \\ 0 & -2 & 2 & 0 \\ -2 & 0 & 0 & 2 \end{bmatrix} \Rightarrow W = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \\ 0 & -2 & 0 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix}$$

An input vector with one missed component: $\vec{x} = <1,1,0,-1>$

$$\Longrightarrow \quad \vec{y}(0) = <1,1,0,-1>$$

Asynchronous update:

Updating Y_2: $y\_in_2(0) = 1 + [1 \;\; 1 \;\; 0 \;\; -1]\begin{bmatrix} 0 \\ 0 \\ -2 \\ 0 \end{bmatrix} = 1 \;\Longrightarrow y_2(1) = 1$

Updating Y_4: $y\_in_4(0) = -1 + [1 \;\; 1 \;\; 0 \;\; -1]\begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix} = -3 \Longrightarrow y_4(1) = -1$

Recall in Hopfield Net

mansoori@shirazu.ac.ir

Updating Y_1: $y\_in_1(0) = 1 + [1 \;\; 1 \;\; 0 \; -1]\begin{bmatrix} 0 \\ 0 \\ 0 \\ -2 \end{bmatrix} = 3 \;\; \Longrightarrow \;\; y_1(1) = 1$

Updating Y_3: $y\_in_3(0) = 0 + [1 \;\; 1 \;\; 0 \; -1]\begin{bmatrix} 0 \\ -2 \\ 0 \\ 0 \end{bmatrix} = -2 \;\; \Longrightarrow \;\; y_3(1) = -1$

$\Longrightarrow \vec{y}(1) =< 1, 1, -1, -1 >$, Updating $Y_1, Y_2, Y_3, Y_4 \Longrightarrow \vec{y}(2) =< 1, 1, -1, -1 >$

$\vec{x} =< 1, 1, 0, -1 > \;\; \Longrightarrow \;\; \vec{y}(0) =< 1, 1, 0, -1 >$

Synchronous update:

$$\overrightarrow{y\_in}\,(0) = \vec{x} + W\,\vec{y}(0) = \begin{bmatrix} 1 \\ 1 \\ 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \\ 0 & -2 & 0 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ -2 \\ -3 \end{bmatrix}$$

$$\Longrightarrow \vec{y}\,(1) = sgn\left(\overrightarrow{y\_in}(0)\right) = [1 \;\; 1 \;\; -1 \; -1]^T$$

States in Hopfield Net

mansoori @shirazu.ac.ir

Ex.: $\vec{s} = \{< 1, 1, -1, -1 >\} \implies -\vec{s} = \{< -1, -1, 1, 1 >\}$

$$W = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix}$$

$$y_i(t+1) = \operatorname{sgn}(x_i + \vec{w}_{.i}{}^T \vec{y})$$

$y_1(t)$	$y_2(t)$	$y_3(t)$	$y_4(t)$	state	$y_1(t+1)$	$y_2(t+1)$	$y_3(t+1)$	$y_4(t+1)$	state
-1	-1	-1	-1	0	1	1	-1	-1	12
-1	-1	-1	1	1	-1	-1	1	1	3
-1	-1	1	-1	2	-1	-1	1	1	3
-1	-1	1	1	3	-1	-1	1	1	3
-1	1	-1	-1	4	1	1	-1	-1	12
-1	1	-1	1	5	1	1	-1	-1	12
-1	1	1	-1	6	1	1	-1	-1	12
-1	1	1	1	7	-1	-1	1	1	3
1	-1	-1	-1	8	1	1	-1	-1	12
1	-1	-1	1	9	-1	-1	1	1	3
1	-1	1	-1	10	-1	-1	1	1	3
1	-1	1	1	11	-1	-1	1	1	3
1	1	-1	-1	12	1	1	-1	-1	12
1	1	-1	1	13	1	1	-1	-1	12
1	1	1	-1	14	1	1	-1	-1	12
1	1	1	1	15	-1	-1	1	1	3

States in Hopfield Net

mansoori@shirazu.ac.ir

Ex.: $\vec{s_1} = \{< 1, 1, -1, -1 >\} \implies -\vec{s_1} = \{< -1, -1, 1, 1 >\}$

$\vec{s_2} = \{< 1, 1, 1, -1 >\} \implies -\vec{s_2} = \{< -1, -1, -1, 1 >\}$

$$W = \begin{bmatrix} 0 & 2 & 0 & -2 \\ 2 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 \\ -2 & -2 & 0 & 0 \end{bmatrix}$$

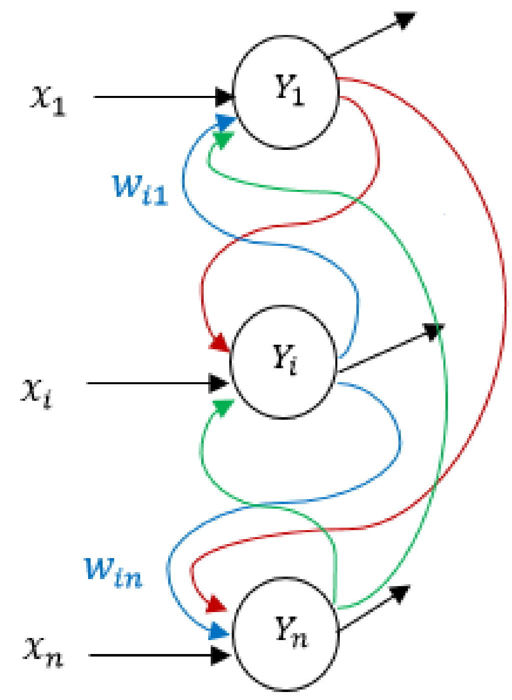
$y_1(t)$	$y_2(t)$	$y_3(t)$	$y_4(t)$	state	$y_1(t+1)$	$y_2(t+1)$	$y_3(t+1)$	$y_4(t+1)$	state
-1	-1	-1	-1	0	-1	-1	-1	1	1
-1	-1	-1	1	1	-1	-1	-1	1	1
-1	-1	1	-1	2	-1	-1	1	1	3
-1	-1	1	1	3	-1	-1	1	1	3
-1	1	-1	-1	4	1	1	-1	-1	12
-1	1	-1	1	5	-1	-1	-1	1	1
-1	1	1	-1	6	1	1	1	-1	14
-1	1	1	1	7	1	1	1	-1	14
1	-1	-1	-1	8	1	1	-1	-1	12
1	-1	-1	1	9	-1	-1	1	1	3
1	-1	1	-1	10	-1	-1	-1	1	1
1	-1	1	1	11	-1	-1	1	1	3
1	1	-1	-1	12	1	1	-1	-1	12
1	1	-1	1	13	1	1	-1	-1	12
1	1	1	-1	14	1	1	1	-1	14
1	1	1	1	15	1	1	1	-1	14

Energy in Hopfield Net

mansoori@shirazu.ac.ir

- In Hopfield net (asynchronous updating), an energy function is defined to prove that net will converge to a stable set of activation, provided diagonal weights are set to zero

$$E = -\frac{1}{2}\sum_i \sum_j w_{ij}\, y_i\, y_j = -\sum_i \sum_{j<i} w_{ij}\, y_i\, y_j$$



Energy in Hopfield Net

mansoori@shirazu.ac.ir

Example:

$$\vec{s} = <-1,-1,1,1> \quad , \quad W = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix}$$

$$E = -w_{12}\, y_1\, y_2 - w_{13}\, y_1\, y_3 - w_{14}\, y_1\, y_4 - w_{23}\, y_2\, y_3 - w_{24}\, y_2\, y_4 - w_{34}\, y_3\, y_4$$
$$= -\, y_1\, y_2 + y_1\, y_3 + y_1\, y_4 + y_2\, y_3 + y_2\, y_4 - y_3\, y_4$$

$$\vec{y}(0) =< -1,-1,-1,1 > \;\Longrightarrow\; E(0) = -1 + 1 - 1 + 1 - 1 + 1 = 0$$
$$\vec{y}(1) =< -1,-1,1,1 > \;\Longrightarrow\; E(1) = -1 - 1 - 1 - 1 - 1 - 1 = -6$$

$$\vec{y}(0) =< -1,-1,-1,-1 > \;\Longrightarrow\; E(0) = -1 + 1 + 1 + 1 + 1 - 1 = 2$$
$$\vec{y}(1) =< 0,1,-1,-1 > \;\Longrightarrow\; E(1) = 0 + 0 + 0 - 1 - 1 - 1 = -3$$
$$\vec{y}(2) =< 1,1,-1,-1 > \;\Longrightarrow\; E(2) = -1 - 1 - 1 - 1 - 1 - 1 = -6$$

Capacity of Hopfield Net

mansoori@shirazu.ac.ir

- Not guaranteed a Hopfield net with this weight matrix has vectors $\vec{s}(p) =< s_1(p), \ldots, s_i(p), \ldots, s_n(p) >, (p = 1, \ldots, P)$ as its stable states

$$W = \{w_{ij}\}, \; w_{ij} = \begin{cases} \sum_{p=1}^{P} s_i(p)\, s_j(p) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

- Disturbance caused by other vectors is called crosstalk
- Closer vectors are, larger crosstalk is
- So, how many vectors can be stored in a Hopfield net before crosstalk gets overhand

Storage capacity of Hopfield net with n neurons:

- For binary patterns: $P \approx 0.15\, n$
- For bipolar patterns: $P \approx \frac{n}{2 \ln n}$

$$n = 100 \quad \Longrightarrow \quad \begin{cases} \text{binary: } P \approx 15 \\ \text{bipolar: } P \approx 11 \end{cases}$$