Hopfield NN

CS385 Machine Learning – A.N.N.

Learning and memory: basic mechanism

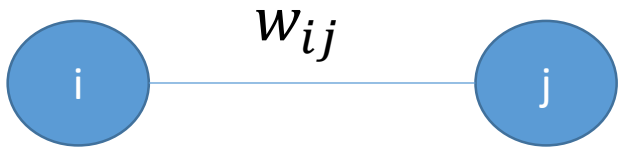
- In neuroscience, **Hebbian theory** is a theory that proposes an explanation for the adaptation of neurons in the brain during the learning process.

- Hebb rule/Hebbian learning
  - When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes place in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased (Hebb, 1949)

Hebbian learning – associative memory

- In other words:
  - If two neurons on either side of a synapse(connection) are activated simultaneously(i.e. synchronously), then the strength of that synapse is selectively increased.
- This rule is often supplemented by:
  - If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated. So that chance coincidences do not build up connection strengths.

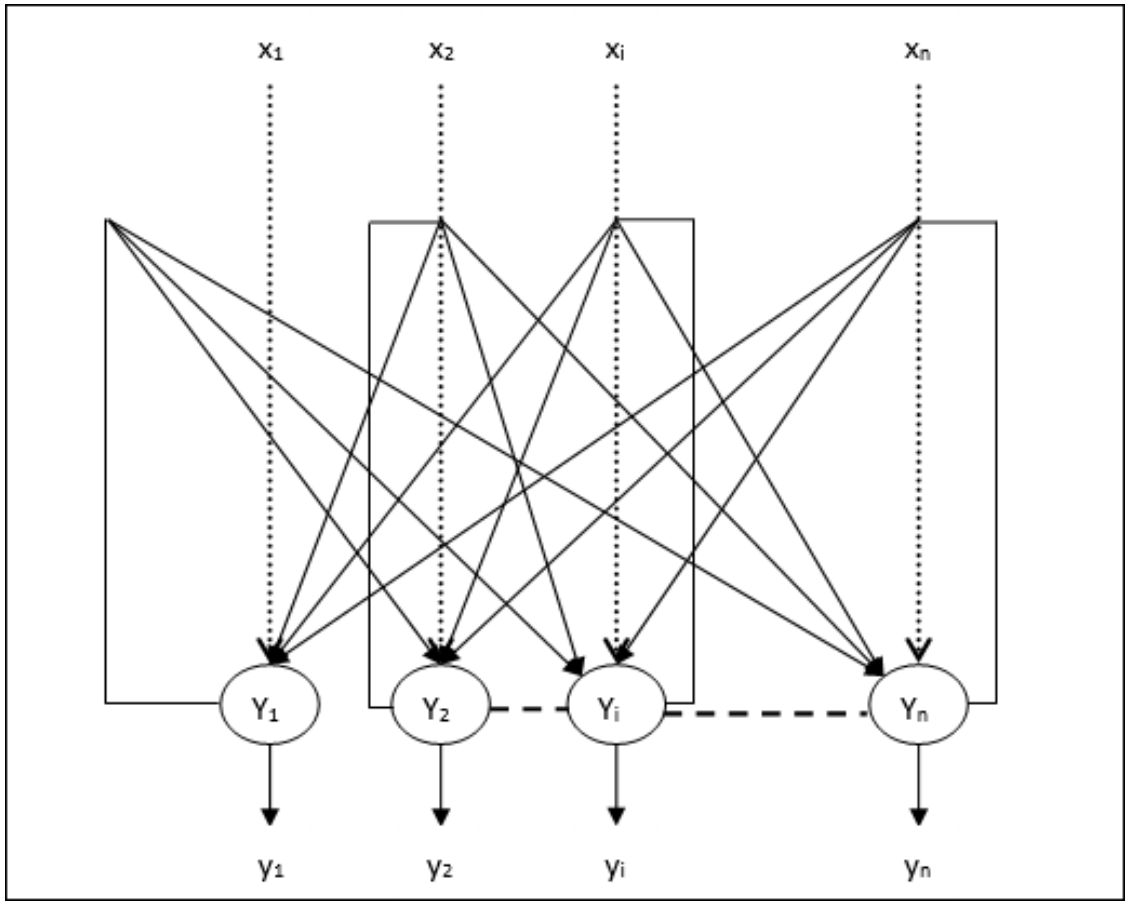
Inspired by Hebb Rule



- S_i=-1, S_j=-1 or S_i=1, S_j=1, the connection between them is strengthened
- S_i=-1, S_j=1, or S_i=1, S_j=-1, the connection between them is weakened

$$w_{ij} = S_i S_j$$

Hopfield Neural Network



It was invented by Dr. John J. Hopfield in 1982

- Discrete Hopfield Net
  - Single layer
  - Binary threshold linear/non-linear units
  - The output of each neuron should be the input of other neurons but not the input of self
- Very hard to analyze. They can behave in many different ways:
  - Settle to a stable state
  - Oscillate
  - Follow chaotic trajectories that cannot be predicted far into the future.

Hopfield N.N. - Energy

- But John Hopfield (and others) realized that if the connections are **symmetric**, i.e. $W_{ij} = W_{ji}$
- There is a global energy function.
  - Each binary "configuration" of the whole network has an energy.
  - The binary threshold decision rule causes the network to settle to a minimum of this energy function.

$$E = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}$$

Hopfield N.N. – properties

- Parallel computation, parallel input, parallel output
- Two processes
  - Memorizing (learning)
    - Update weights by using Hebb learning rule
  - Remembering (test)
    - Output a result most similar to memorizing example by calculating
    - To learn weights help to rest in the state with minimal energy

Memorizing – an example

- 4 nodes Hopfield NN

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix}$$

- 2 patterns needs to be memorized
  - P1=[1 1 1 1]
  - P2=[-1 -1 -1 -1]

$$w_{ij} = \sum_{k=1}^{m} S_{i,k} S_{j,k} \qquad m = 2$$

Memorizing – an example

$s_1\ s_2\ s_3\ s_4$

- P1=[1 1 1 1]
- P2=[-1 -1 -1 -1]

$$w_{ij} = \sum_{k=1}^{m} S_{i,k} S_{j,k}$$

$w_{ii} = 0$

$w_{12} = \underbrace{s_1 \cdot s_2}_{P_1} + \underbrace{s_1 \cdot s_2}_{P_2}$

$= 1 \times 1 + -1 \times -1$

$= 2$

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

Remembering – an example

Step1 input sample: [1 1 -1 1], or [-1 -1 -1 1]

Step2 calculate energy $E = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}$

E(1,1,-1,1)=0

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

Step3 repeat: randomly change status (neighbor status: only change 1 element)

- S1 = -1, [-1 1 -1 1], E(-1,1,-1,1)=4, increased, reject the change repeat step3
- S3 = 1, [1 1 1 1], E(1,1,1,1)=-12 decreased, accept the change, current configuration is [1 1 1 1], repeat step3

Step4 until convergence

Remembering – algorithm improvement

Input: [1 1 -1 1]

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

Step3 randomly change status

- S1’ = -1, [-1 1 -1 1]
- $\Delta E = (S1'-S1)*(S2*W_{12}+ S3*W_{13} + S4*W_{14})$
- If $\Delta E \geq 0$ reject
- Else accept

$$\Delta E = (S_1'S_2W_{12} + S_1'S_3W_{13} + S_1'S_4W_{14} + S_2S_3W_{23} + S_2S_4W_{24} + S_3S_4W_{34}) - (S_1S_2W_{12} + S_1S_3W_{13} + S_1S_4W_{14} + S_2S_3W_{23} + S_2S_4W_{24} + S_3S_4W_{34})$$

$$= (S_1' - S_1) \cdot (S_2 \cdot W_{12} + S_3 \cdot W_{13} + S_4 \cdot W_{14})$$

Hebb Rule – {0 1}

- Given m patterns, $X^1 \ldots X^m$

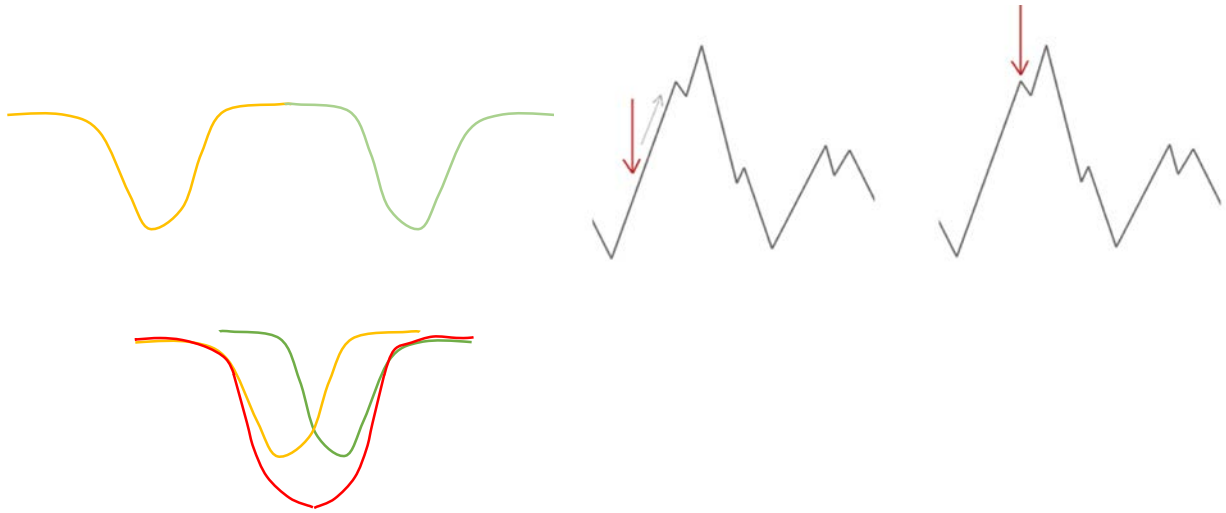
$$W_{ij} = \sum_{k=1}^{m} 4\left(x_i^k - \frac{1}{2}\right)\left(x_j^k - \frac{1}{2}\right)$$

$$W_{ii} = 0$$

- Value range for each element in pattern {0 1}

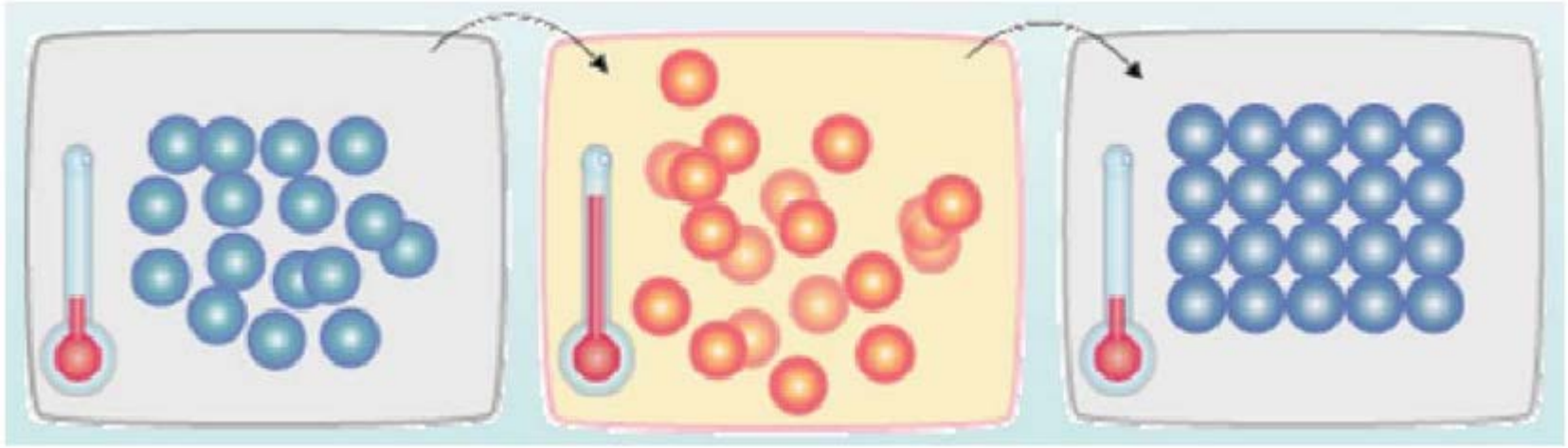
Local minimum &
Spurious minima limit capacity

- Each time we memorize a pattern, we hope to create a new energy minimum.
  - But what if two nearby minima merge to create a minimum at an intermediate location?
  - This limits the capacity of a Hopfield net.

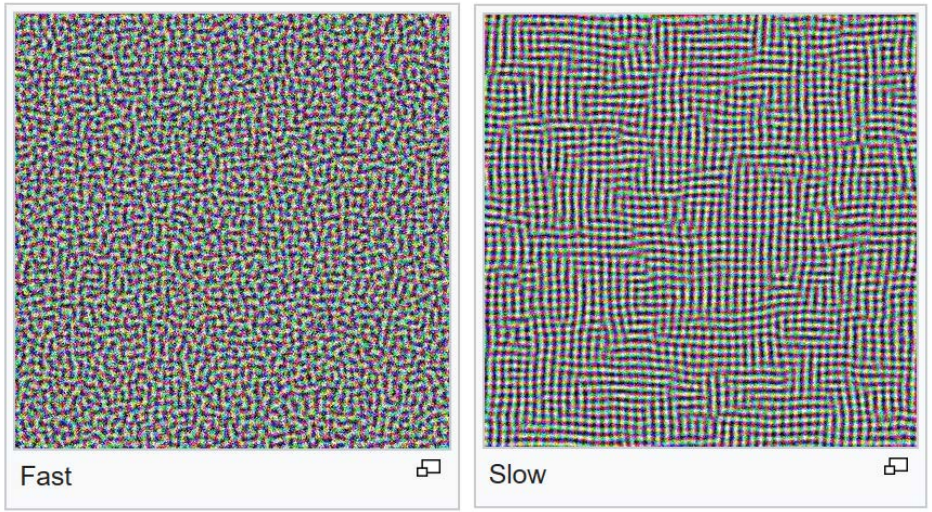


The state space is the corners of a hypercube. Showing it as a 1-D continuous space is a misrepresentation.

Local Minimum – Simulated Annealing

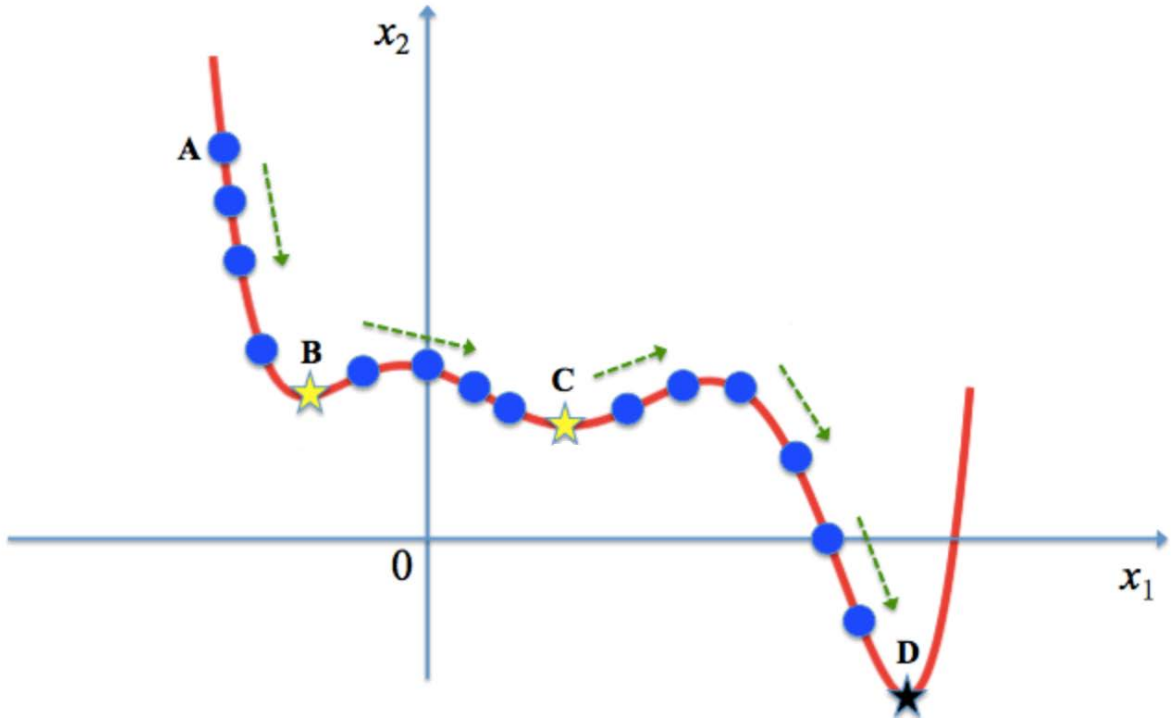


- Heat -- Cooled Slowly



Metropolis Rule

$$p = \begin{cases} 1 & if \quad E(x_{new}) < E(x_{old}) \\ \exp(-\dfrac{E(x_{new}) - E(x_{old})}{T}) & if \quad E(x_{new}) \geq E(x_{old}) \end{cases}$$



Simulated Annealing Example

$$p = \begin{cases} 1 & if \quad E(x_{new}) < E(x_{old}) \\ \exp(-\frac{E(x_{new}) - E(x_{old})}{T}) & if \quad E(x_{new}) \geq E(x_{old}) \end{cases}$$

- T=100
- ΔE=4 > 0 (ΔE=12 > 0)
- p = exp(-4/100) =0.96
- If p>random(0,1) Accept
  - (uniformly at random)
- Else reject

T=99
T=98
T=97
T=96
⋮

- T=1
- ΔE=4 > 0
- p = exp(-4/1) =0.02
- If p>random(0,1) Accept
  - (uniformly at random)
- Else reject