# Hopfield Networks

1


- John Hopfield: Hopfield Networks
- Geoff Hinton: RBM, Deep models..

2

# Hopfield Networks

- Introduced by the physicist, professor of chemistry and biology, John Hopfield in 1982
- Used to represent content addressable memory: given a reasonable portion of the memorized information, given a partial or corrupted portion, it can be completed by recovering other associated information (pattern completion)
- The system preserves this propery also when some components are broken (fault tolerance)
- For instance.:
  - "H.A.Kramers and G.H.Wannier, *Phys. Rev.*, 60, 252 (1941)"
  - Enough "and Wannier (1941)" o addirittura "Vannier (1941)"

3

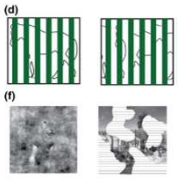
## Hopfield Networks

- Attractor networks, useful to memorize fundamental memories which attract other inputs

4

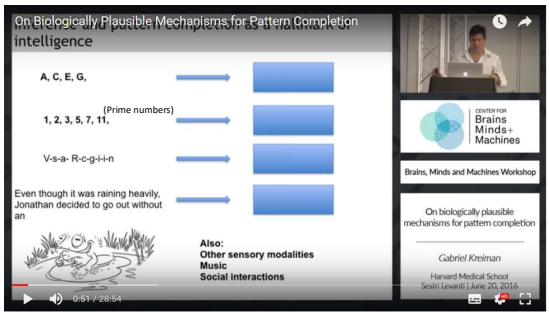
## Hopfield Networks

- Pattern completion characterizes a lot of our memory.
- For instance.:
- (Occluded) object recognition can also be seen as pattern completion



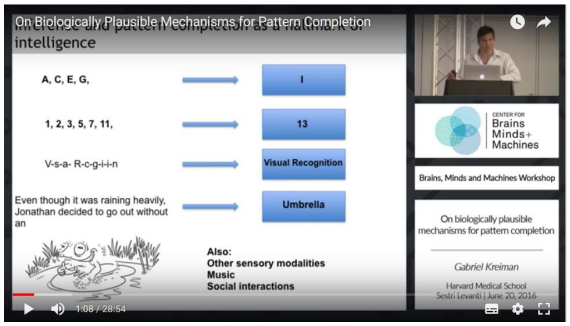

5

## Hopfield Networks




6

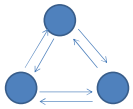
# Hopfield Networks




7

# Hopfield Networks

- Network of McCulloch-Pitts neurons interconnected.
- Each neuron can have two states: 1, -1.

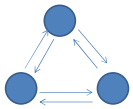


- Given N units:
  - Each neuron is connected with all the other neurons but itself.
  - Weights are symmetrical: $w_{ij} = w_{ji}$

8

# Hopfield Networks

- Network of McCulloch-Pitts neurons interconnected.
- Each neuron can have two states: 1, -1.



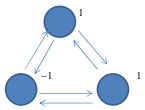
- Given N units:
  - Each neuron is connected with all the other neurons but itself.
  - Weights are symmetrical: $w_{ij} = w_{ji}$

Usually much bigger dimensions! This is a toy example!!!

9

# Hopfield Networks (**Withdrawal/Retrieval Phase**)

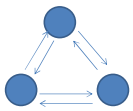
- State of a network with N neurons at a given iteration= configuration of a network with N neurons at a given iteration: activations of the N neurons (1,-1).




- Input to the network: vectors of 1 and -1 of dimension N. Each neuron represents an element of the input
- Then calculate new activation of a neuron at a time, randomly chosen.
- Until there is no change.

10

# Hopfield Networks



- Activation of j at iteration n= $y_j(n) = \varphi(v_j(n))$
- $v_j$= netinput to j = $\sum_{i=0}^{N} w_{ji} y_i(n)$
- $$\varphi(v_j)(n) = \begin{cases} 1 \text{ if } v_j(n) > 0 \\ -1 \text{ if } v_j(n) < 0 \\ \varphi(v_j)(n-1) \text{ if } v_j(n) = 0 \end{cases}$$

11

# Hopfield Networks. II phase: information withdrawal

1. Input. To the network is presented x*: configuration +1 and -1 of length N. For each neuron j, $y_j(0) = x^*(j)$
2. Iteration until convergence. Update the elements of the network in an asyncronous way by choosing a unit at random. The rule to update the elements is

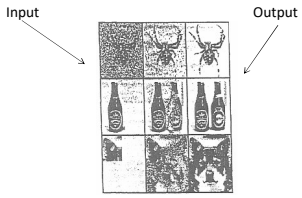
$$y_j(n) = \begin{cases} 1 \text{ if } \sum_{i=0}^{N} w_{ji} y_i(n) > 0 \\ -1 \text{ if } \sum_{i=0}^{N} w_{ji} y_i(n) < 0 \\ y_j(n-1) \text{ if } v_j(n) = 0 \end{cases}$$

3. Convergence. Repeat the operation until we find a **stable state**, s.t. for each unit j $y_j(n+1) = y_j(n)$.
4. Output. This stable state is the output of the network.

12

# Hopfield Networks

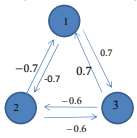
States= configurations of +1,-1




13

# Hopfield Networks

- Given a starting state, the states through which the network goes are calculated as follows:
- The activations of the neurons are calculated in an asyncronous way: each time a unit is chosen, its activity is calculated by using the equations:.
- Example:




$$v_j = \sum_{i=0}^{N} w_{ji} y_i(n)$$

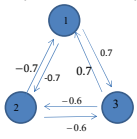
$$\varphi(v_j)(n) = \begin{cases} 1 \text{ if } v_j(n) > 0 \\ -1 \text{ if } v_j(n) < 0 \\ \varphi(v_j)(n-1) \text{ if } v_j(n) = 0 \end{cases}$$

Initial state [-1,-1,1].

14

# Hopfield Networks

- Given a starting state, the states through which the network goes are calculated as follows:
- The activations of the neurons are calculated in an asyncronous way: each time a unit is chosen, its activity is calculated by using the equations:.
- Example:




$$v_j = \sum_{i=0}^{N} w_{ji} y_i(n)$$

$$\varphi(v_j)(n) = \begin{cases} 1 \text{ if } v_j(n) > 0 \\ -1 \text{ if } v_j(n) < 0 \\ \varphi(v_j)(n-1) \text{ if } v_j(n) = 0 \end{cases}$$

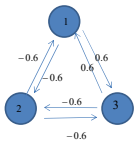
Initial state [-1,-1,1]. If I choose 1, $v_1 = 1.4, y_1 = \phi(v_1) = 1$ and I obtain [1,-1,1]

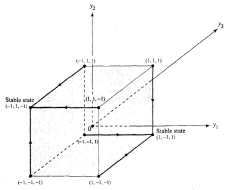
- If I then choose 2: $v_2 = -1.3, y_2 = \phi(v_2) = -1$ and I obtain [1,-1,1].
- If I then choose 3, it remains at +1.
- [1, -1, 1] is a stable state of the network, since $\forall i, y_i(n+1) = y_i(n)$

15

# Hopfield Networks

- [1, -1, 1] is one of the two stable states of the network (in this example). The other is [-1, 1, -1]



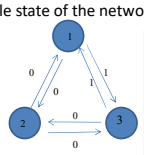




- The stable states act as attractors : given any state, it will converge in one of the two states.
- A convergence theorem guarantees that given a starting state the network will always arrive to a stable state: the activities cannot be changed forever.

16

# Hopfield Networks

- In general, finding a stable state is a non deterministic process. In case of networks with several units, it can require several iterations.
- Exercise: Find a stable state of the network starting from [1, 1, -1].



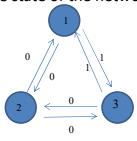

$$v_j = \sum_{i=0}^{N} w_{ji} y_i(n)$$

$$\varphi(v_j)(n) = \begin{cases} 1 \text{ if } v_j(n) > 0 \\ -1 \text{ if } v_j(n) < 0 \\ \varphi(v_j)(n-1) \text{ if } v_j(n) = 0 \end{cases}$$

17

# Hopfield Networks

- In general, finding a stable state is a non deterministic process. In case of networks with several units, it can require several iterations.
- Exercise: Find a stable state of the network starting from [1, 1, -1].




$$v_j = \sum_{i=0}^{N} w_{ji} y_i(n)$$

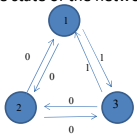
$$\varphi(v_j)(n) = \begin{cases} 1 \text{ if } v_j(n) > 0 \\ -1 \text{ if } v_j(n) < 0 \\ \varphi(v_j)(n-1) \text{ if } v_j(n) = 0 \end{cases}$$

Solution:
- If I update 1 first: [-1,1,-1]
- If I update 3 first: [1,1,1]

18

# Hopfield Networks

- In general, finding a stable state is a non deterministic process. In case of networks with several units, it can require several iterations.
- Exercise: Find a stable state of the network starting from [1, 1, -1].




$$v_j = \sum_{i=0}^{N} w_{ji} y_i(n)$$

$$\varphi(v_j)(n) = \begin{cases} 1 \text{ if } v_j(n) > 0 \\ -1 \text{ if } v_j(n) < 0 \\ \varphi(v_j)(n-1) \text{ if } v_j(n) = 0 \end{cases}$$

Solution:
- If I update 1 first: [-1,1,-1]
- If I update 3 first: [1,1,1]

19

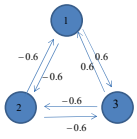
# Hopfield Networks (**Storage Phase**)

- Hopfield Networks are used to memorize information
- We want the memorized information to work as attractors, stable states towards which move the other states
- If we memorize images, we want:
  - The corresponding images to be stable states: if we show them to the network, this does not move
  - If we present to the network a corrupted version of the image, this converges towards the memorized image.

20

# Hopfield Networks

- The problem is how to make the information we want to memorize, its fundamental memories, stable states.
- The states are stable thanks to weights.
- For instance [1,-1,1]

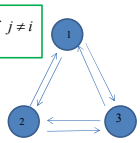



21

# Hopfield Networks

•Given M fundamental memories, $f_1..f_M$ vectors of dimension N, we memorize them by letting:

$$w_{ji} = \frac{1}{M}\sum_{k=1}^{M} f_k(i) \bullet f_k(j) \text{ if } j \neq i$$




• For each fundamental memory the product between the ith and the jth element

•If they have the same sign, $f_k(i) \bullet f_k(j) > 0$ this contributes to strengthening $w_{ji}$.

If they have different sign, this contributes to weakening $w_{ji}$.

22

# Hopfield Networks

- Given a set of fundamental memories, we must find adequate weights.
- The learning algorithm is based on Hebb's principle:
  - Strengthen the connections between units with the same activation
  - Weaken the connections between units with opposite activation

23

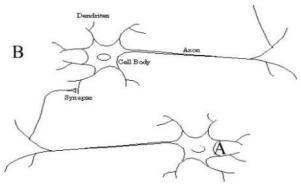
# Hopfield Networks

- Given a set of fundamental memories, we must find adequate weights.
- The learning algorithm is based on Hebb's principle:
  - Strengthen the connections between units with the same activation
  - Weaken the connections between units with opposite activation
  - Hebb's principle has a neurobiological counterpart: in the synapses between units which are often active at the same time are strengthened, while synapses between neurons which are not simultaneously active are weakened.

24

## Hebb's Principle

Neurons that fire together wire together




25

## Hopfield Networks

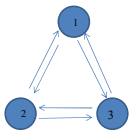
- WARNING: Following the tradition, we have seen the withdrawal phase before the storage phase.
- However, chronologically they come in the opposite order: First: storage of the fundamental memories. Then, withdrawal!

26

## Hopfield Networks. I phase: storage

- Esercise: apply the rule so that [1,-1,1] e [-1,1,-1] are stable states: these are fundamental memories to be memorized.

$$w_{ji} = \frac{1}{M}\sum_{k=1}^{M} f_k(i) \bullet f_k(j) \text{ se } j \neq i$$




27

## Hopfield Networks. I phase: storage

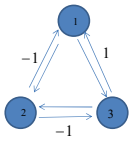
- Solution. If we want two fundamental memories: [1,-1,1] and [-1,1,-1]

$$w_{ji} = \frac{1}{M}\sum f_k(i) \bullet f_k(j) \text{ if } j \neq i$$

$$w_{21} = w_{12} = \frac{1}{2}\sum_{k=1}^{2} f_k(1) \bullet f_k(2) = \frac{1}{2}(1 \bullet (-1) + -1 \bullet (1)) = -1$$

$$w_{23} = w_{32} = \frac{1}{2}\sum_{k=1}^{2} f_k(2) \bullet f_k(3) = \frac{1}{2} \bullet (1 \bullet (-1) + (-1) \bullet 1) = -1$$

$$w_{31} = w_{13} = \frac{1}{2}\sum_{k=1}^{2} f_k(1) \bullet f_k(3) = \frac{1}{2}(1 \bullet 1 + (-1) \bullet (-1)) = 1$$




28

## Hopfield Networks. Convergence Theorem

- Convergence Theorem: Given any initial state, the network converges to a stable state.

29

## Hopfield Networks. Convergence Theorem

- Convergence Theorem: Given any state of the network, the network converges to a stable state.
  - Given N units, there are $2^N$ possibile states of the network
  - To each of these states is associated an Energy value
  - We prove that each change of state of the network leads to a lowering of the Energy.
  - There is a moment in which there is no reacheable state with lower Energy.

30

## Hopfield Networks. Convergence Theorem

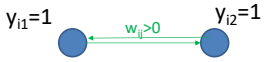
- Energy function E = $-\frac{1}{2}\sum_i\sum_j w_{ij}y_iy_j$
- The energy E measures the propension of the network to change state (opposite of the harmony) : ENERGY IS BAD!
- We prove that each change in state diminishes E

- Roughly speaking, Harmony increases with the number of positive $w_{ij}y_iy_j$ (where $y_i$ and $y_j$ have the same sign and $w_{ij}$ is positive or they have opposite sign, and $w_{ij}$ is negative). Higher harmony= less propension to change state. Lower harmony= more propension to change state.

31

## Hopfield Networks. Convergence Theorem

- Energy function E = $-\frac{1}{2}\sum_i\sum_j w_{ij}y_iy_j$
- Each product $w_{ij}y_iy_j$ is positive if $y_i$ and $y_j$ have the same sign and $w_{ij}$ is positive or they have opposite sign, and $w_{ij}$ is negative.
- Example:

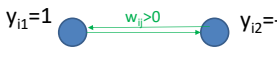



Roughly speaking: The activations enforce each other: Energy Low: Low predisposition to change!

32

## Hopfield Networks. Convergence Theorem

- Energy function E = $-\frac{1}{2}\sum_i\sum_j w_{ij}y_iy_j$
- Each product $w_{ij}y_iy_j$ is negative if $y_i$ and $y_j$ have the same sign and $w_{ij}$ is negative or they have opposite sign, and $w_{ij}$ is positive.
- Intuitively:




Instable!
Energy high! High propension to change!

33

# Hopfield Networks. Convergence Theorem

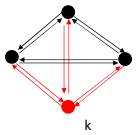
- Consider how E changes with the change of the activation of the kth unit.
- E and E' denote energy before and after the change of $y_k$

$$E = -\frac{1}{2}\sum_i\sum_j w_{ij}y_i y_j = -\frac{1}{2}\left(\sum_{i\neq k}\sum_{j\neq k} w_{ij}y_i y_j + 2\sum_{j\neq k} w_{kj}y_k y_j\right)$$

$$E' = -\frac{1}{2}\left(\sum_{i\neq k}\sum_{j\neq k} w_{ij}y_i y_j + 2\sum_{j\neq k} w_{kj}y'_k y_j\right)$$

$$E - E' = -\frac{1}{2}\left(2\sum_{j\neq k} w_{kj}y_k y_j - 2\sum_{j\neq k} w_{kj}y'_k y_j\right) =$$

$$= -\sum_{j\neq k} w_{kj}y_j(y_k - y'_k)$$

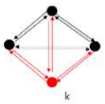



34

# Hopfield Networks. Convergence Theorem

This is $v_k$!

$$E - E' = -\sum_j w_{kj}y_j(y_k - y_k')$$




We distinguish 2 cases:

1. $y_k$ passes from +1 to -1. Hence $y_k - y'_k > 0$ and $\sum_j w_{kj}y_j < 0$

$$-\sum_j w_{kj}y_j(y_k - y_k') > 0 \text{ and } E > E'$$

2. $y_k$ passes from -1 to +1. Hence $y_k - y'_k < 0$ and $\sum_j w_{kj}y_j > 0$. Then again

$$-\sum_j w_{kj}y_j(y_k - y_k') > 0 \text{ and } E > E'$$

At each change E lowers. Since the states are finite, at a given point we get to a state that does not change: a **stable state**

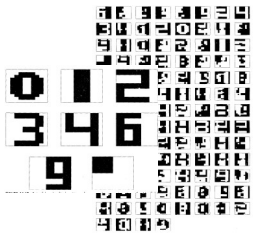
35

# Hopfield Networks. Qualities

- They complete partial patterns
- They generalize: given an input similar to what has been memorized, they recover the corresponding information
- They are fault tolerant: if some synapses get broken (brain damage) the output is still reasonable
- They allow the extraction of prototypes: if the network learns several similar informations, it creates their prototype and this is the explicitely presented state.
- The learning rule is Hebb like, which is biologically plausible and there is evidence that it exists in the brain
- The networks can account for context effects: we remember better what is learned if we are put in the same context

36

## Hopfield Networks. Example



37

## Hopfield Networks. Drawbacks

- Unfortunately, not all the stable states are fundamental memories memorized during storage. There are spurious states:
  - The opposite of a stable state is a stable state
  - Combinations of stable states are stable states
- Not all the fundamental memories are stable states
- Storage capacity with few errors given N units = 0.14 N

38

## Hopfield Networks. Drawbacks

- Limited storage capacity
- Errors
- In the brain synapses are not symmetrical
- In the brain there are no stable states but states transitorily stable, which evolve in successive states. There are extensions of Hopfield networks that learn sequences of states.

39

## Hybrid models

- Some reserachers today (e.g., G. Kreiman) use hybrid models:
- Standard convolutional poor at pattern completion
- Ex: convolutional + Hopfield for pattern completion

40