Neuroengineering (I)
5. Artificial memories and recurrent networks

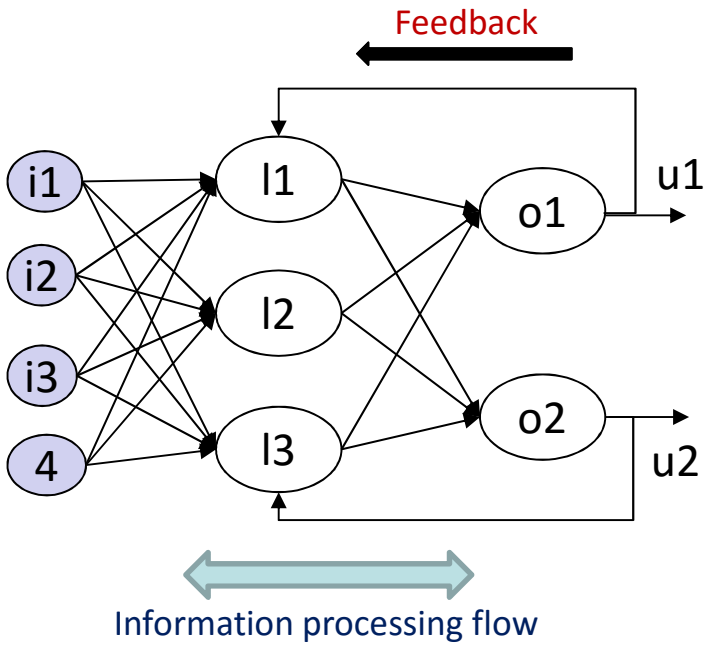
- **Scuola di Ingegneria Industriale e dell'Informazione** – Politecnico di Milano
- Prof. Pietro Cerveri

"Recall and storage". So far, FFNN, you don't need specifically to deal with time, for us is irrelevant the temporal size,you can assume the input as being concurrent with the output, is just a metter of computation for a pratical point of view. Time dealing is introduced only by computation, in principle they are concurrent. I'd like to address feedback processing now, we process information, structurally I have to get an output on any level of my network, output or hidden layer, and set back the output to other neurons at the input. If I'd like to enable this kind of processing I have to introduce a TIME DELAY in the computation. Indeed in the first class we saw that in the formal neuron I can see temporal seeing that the processing begins with time T computing the AP, and at the next time instant T+1 I'm computating the activation function and output. At time t I'm entering the neuron, at time t+1 I'm outputting. This enable us to allow the network to consider a temporal scale (Esplicitally). I have a clock, a time-stamp, at each timestamp I compute the output (the state of the neuron). And this output is becoming input for other neurons at the next time step. This is the basis to do feedback processing, simplest way.

Feedback and dynamics

This kind of topology, equipped with temporal dynamics allows us to emulate biological memories (episodic memories actually)

- FF Neural network as an input/output model
  - Pattern classification
  - Function mapping
  - Input encoding
  - Dimensionality reduction
- but .. what about other brain abilities as the memory?
  - Seeing how recurrent ANN can be used to simulate biological memories



Memory

- A primary function of the brain is the storage and recall of information.
  - We call this ability memory
- What?
  - Thought, action, behavior (things, experiences, programs, emotions)

Memory is an ability, or a function of the brain, with this ability I can store experiences, things, behaviours, motor command/control, emotions, sensetions ... .The memory is a very huge container that can address different kind of stuff that we can store.

Memory

- A primary function of the brain is the storage and recall of information.
  - We call this ability memory
- What?
  - Thought, action, behavior (things, experiences, programs, emotions)
- Making experience
  - Sensorial input
  - Attention
  - Repetition: principle of reinforcement learning

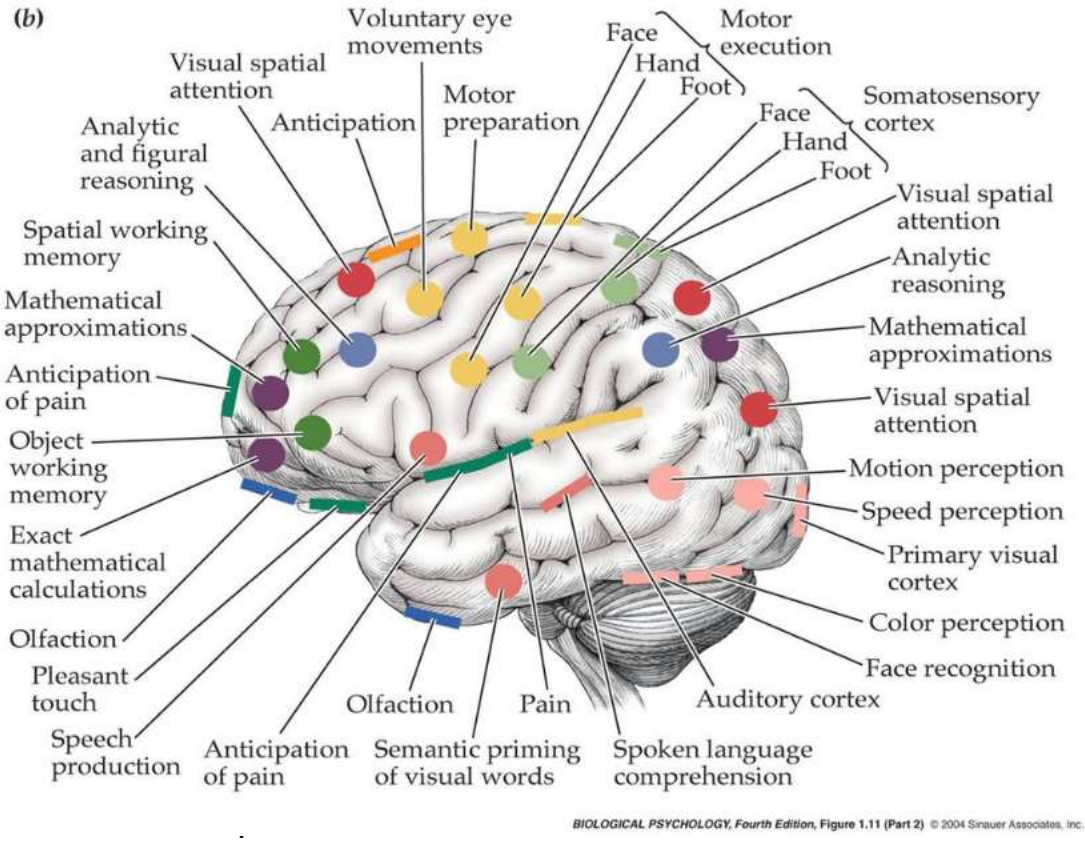
How to make memory?

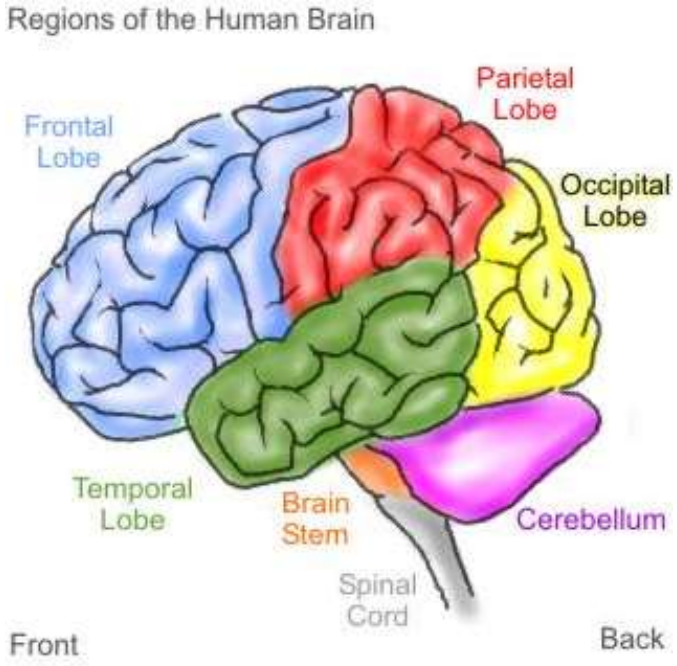
sensorial experience

We have to point out what we'd like to store in our memory.

Affidability: depends on how we do.. maybe if we repeat many times, to reinforce the learning, re-do the same experience many times, allows us to make our memory strong.

Strong: at high level it means it's easy to recall memory with few informations at the input. If I have a very confused input, maybe noise, maybe blurred, it could be hard to recall if the memory is not very strong. The opposite in the opposite case.





- Structure
  - Local/Distributed
  - Temporal lobe (1940 - 1950)
  - Association

Assumption: localized memory trace or *engram*

It was found that memory for simple classically conditioned responses was localized (in the cerebellum)

<u>but</u>

also found that maze-learning in rats was distributed throughout the brain

It has been difficult to link memories with changes in **individual** neurons or specific synaptic connections

Memory

- A primary function of the brain is the storage and recall of information.
  - We call this ability memory
- What?
  - Thought, action, behavior (things, experiences, programs, emotions)
- Making experience
  - Sensorial input
  - Attention
  - Repetition
- Structure
  - Local/Distributed
  - Temporal lobe (1950)
  - Association
- Memory processes
  - Encoding
  - Consolidation
  - Storage
  - Retrieval of memories

- Encoding--transforming information into a form that can be entered and retained in the memory system
- Consolidation--making stable memory traces
- Storage--retaining information in memory so that it can be used at a later time
- Retrieval--recovering information stored in memory so that we are consciously aware of it

Biological memory

- In Biological memories we do not know exactly
  - How the information is stored
  - Where the information is stored
- Encoding:
  - Sensorial experience
  - Explicit vs implicit
  - hierarchical, clustering, compression, …

Biological memory

- In Biological memories we do not know exactly
  - How the information is stored
  - Where the information is stored
- Encoding:
  - Sensorial experience
  - Explicit vs implicit
  - hierarchical, clustering, compression, …
- Recall
  - Quick vs slow
  - Conscious/Unconscious
  - Content-addressable memory

- Biological recall (memory access) means retrieving a specific pattern represented in the memory using information fragments, partial recall, with or without sensory input

Biological memory

- In Biological memories we do not know exactly
  - How the information is stored
  - Where the information is stored
- Encoding:
  - Sensorial experience
  - Explicit vs implicit
  - hierarchical, clustering, compression, …
- Recall
  - Quick vs slow
  - Conscious/Unconscious
  - Content-addressable memory
- Storage
  - Interference
  - Finite storage capacity?
  - Time-dependence (short and long term)
  - Loss of memory (depletion)

memories trying to occupy the same space

We have experience of loss memory, getting older and older: the efficiency of every memories decrease (probably due to loss of memory, or acting at a traumatic level (morphological modification of memory areas, trauma/accidents), or using drugs. This is called depletion.

Three Stages of Memory

MODEL FOR MEMORY, 3 stages. I have a sensory experience, I get into short term then LTM. [xxxxxx??]

Atkinson-Shiffrin model, 1968

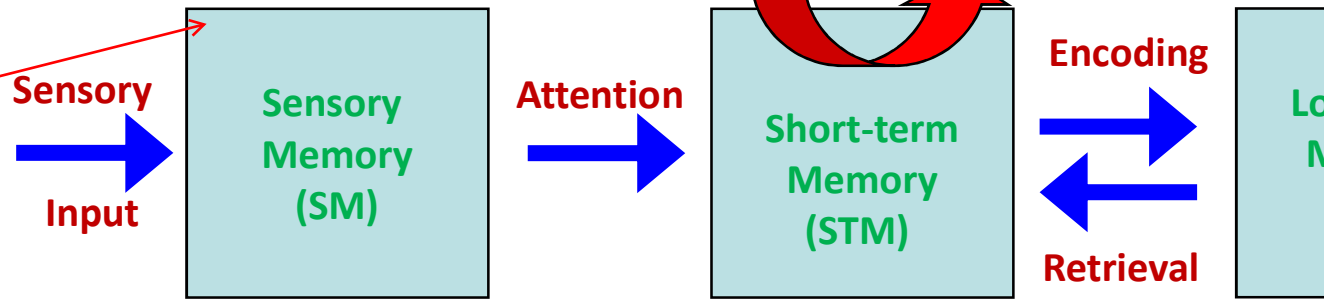
Three memory stores that differ in function, capacity and duration

guardo più di una volta, reharsal is a process to REINFORCE memory, to be put in LTM. If you study you can't ready just one time. And repeat. Repetition means, triggering between the two storages. The more you are trying to store in LTM a piece of information, the more you have to reharsal: reput in short term and then in LTM. Consolidation

Mental or verbal repetition of information allows information to remain in working memory longer than the usual 30 seconds

Maintenance Rehearsal

We are dealing with sensorial: each of our sensorial system has a sort of internal memory that allows to store input that lasts for like s, or ms. For example retina afterimages.The stimuls has stopped but you still ex perience. Your gangalial cells are still firing even without stimulus. It's a sort of very short memory.



Permanent ability to store, capacity is not really infinite + competition interference means no ability to storage infinite knowledge

Function: process for basic physical characteristics

- Iconic, Echoic, Haptic, Olfactory, Taste

Capacity: large
Duration: less than 2s

Function: conscious processing of information where information is actively processed
Capacity: limited (~7±2 items: Miller's Law, 1956)
Duration: less than 30s

Function: organizes and stores information; more passive form of storage than working memory
Capacity: unlimited (in principle)
Duration: almost permanent

Reasoning, speech, talk, make computation.. This ability requires ATTENTION. So if I have some memories [xxxxxx] FOVIAL[xxx]? this is triggering

Despite the capacity (very big different in the two), it's also different the way the two storages FORGETS informations: STM: sort of NN composed by population cells triggering signals, and we can hypotesize that a piece of memory is a sort of pattern of signal produced by these. When the network ceeses this activation the corrisponding memory is lost. Just stopping the electrical activity is deplation. LTM is not like that, we have **plasticity,** this is related to the ability of the neurons to arrange connections in between. These connections can be reinforced. And the neurons that can learn have the ability to make connections stronger and stronger as I feed the network with the same information. Probabily a **synaptic connection** can be regarded as a bit of information. If you consolidate the bit, this synaptic connection is stronger and stronger as I repeat or reharsal. At the end the efficiency of the transduction is increased. "I need 100 unit of energy to trigger the other neuron"... plasticity allows this synaptic to increase sensitivty, I'll need less energy :). **Learning means increase sentivity**. It requires fewer and fewer energy. This means neurons are constituting an unique unit, triggering the first means immediataly triggering the second with a short energy (few spikes). Forgetting in LTM means to DISRUPT these connections, this can be a way of forget, but also there's another way, either: 1. lose synaptic interface 2.lose connection.

Working or Short-term Memory (STM)

VS

Long-term Memory (LTM)

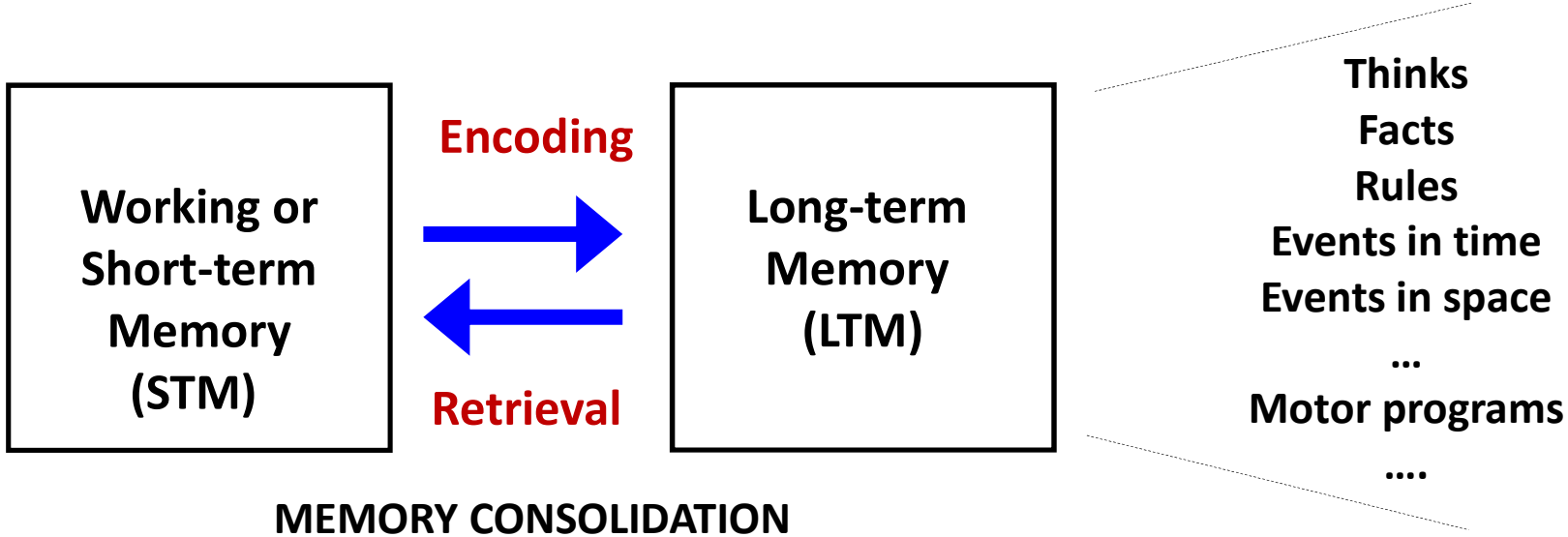
Capacity

Small ↔ Huge

Forgetting

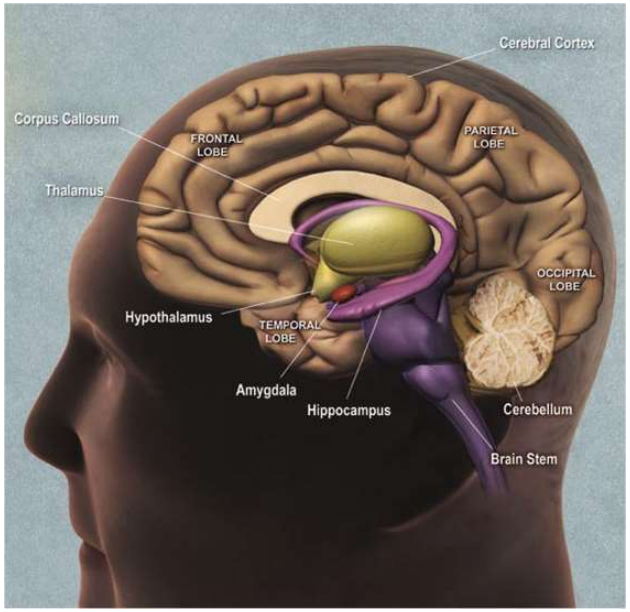
Ceasing transmission ↔ lowering synapse strengthen

CONSOLIDATION: process to make a memory robust in my brain (high level), at low level it means to exploit plasticity, make synaptic connection stronger. The hippocampus is a fondamental structure that allows our brain system to move memories from STM to LTM. If you have damages there you can't have LTM.Without it we CAN recall old memories tho', just not learning new. tl;dr demonstration they are separted



MEMORY CONSOLIDATION

Long-term Potentiation
Long-term Depression
(Hippocampus role)



Mental **repetition** improves transfer efficiency

Giving a **meaning** and **associating** it with previously acquired knowledge improves memorization

Concept of reinforcement learning. There's a correlation during your learning with electric signals: high freq. (spikes at high freq) is the mechanism enabling consolidation. If the same line is triggered with low freq the pathway that is being created is depressed. Plasticity means not only consolidation, but also open syn connections (free). not "open" but increasing the "" impedance"". Lowering the sensitivity.

Long-term Potentiation and Depression

Long-term potentiation is a process by which synchronous firing of neurons makes those neurons more inclined to fire together in the future. Long-term potentiation occurs when the same group of neurons fire together so often that they become permanently sensitized to each other. As new experiences accumulate, the brain creates more and more connections and pathways, and may **“re-wire”** itself by re-routing connections and re-arranging its organization.

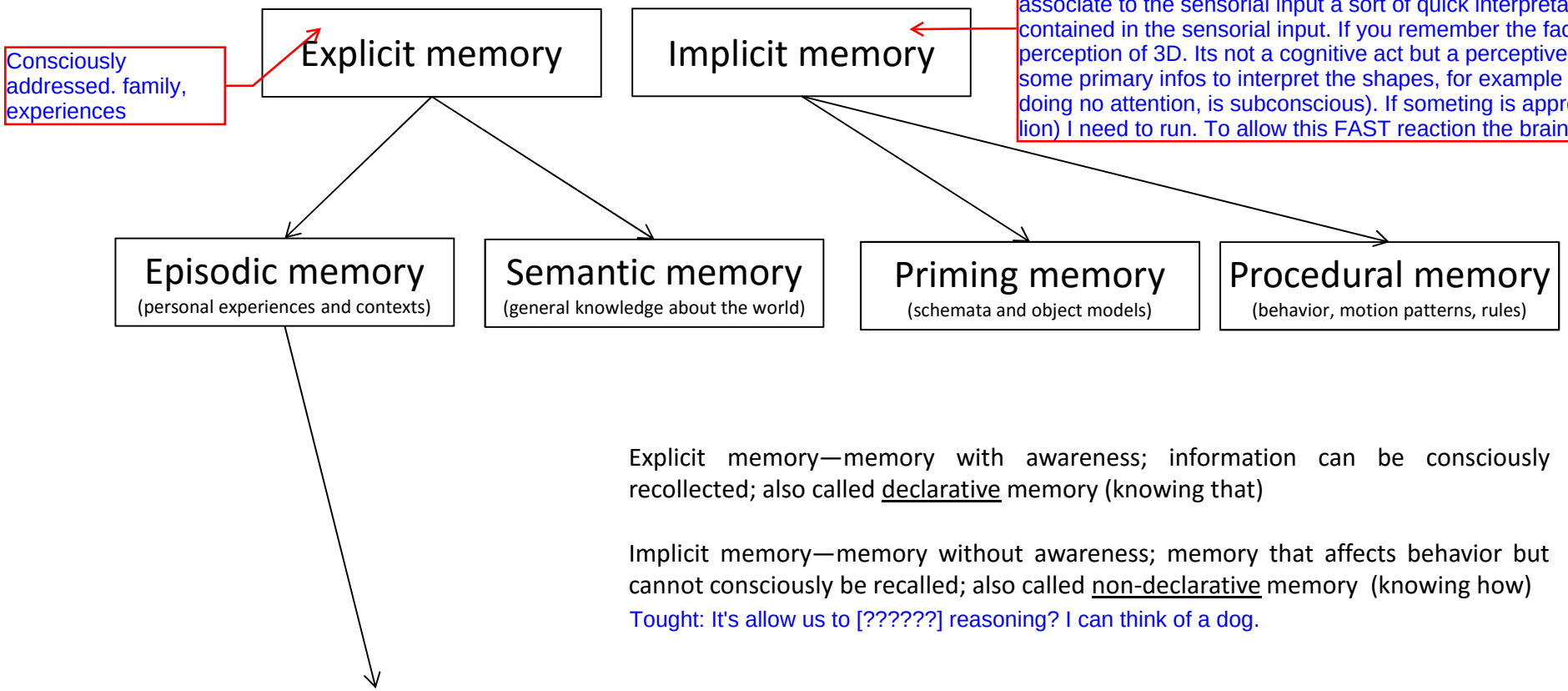
Triggered by **high-frequency** stimulation of the synapses

Long-term depression is produced by nerve impulses reaching the synapses at very **low frequencies**, leading them to undergo the reverse transformation from potentiation, and, instead of becoming more efficient, the synaptic connections are weakened

Ex. the neural networks involved in erroneous movements are **inhibited** by the silencing of their synaptic connections

LTM can be modeleded in 2 main substorage: explicit and implict. Pattern coming from sensorial input (looking at one paiting, seeing it) can be stored in episodic memory. This is a sort of AUTOASSOCIATION, I have in my LTM a sort of representation of the painting. This representation is related to the content, the shape of the cypress, but I have the shape for the cyphress already somehwere, not only for the one of van ghog. Se devo ricordare un dipinto che ha tipo un tavolo ed un albero allora ricordo queste cose, però sono disposte in punti diversi, non ho la mem. solo di quello. Autoassociation means that probably I have somewhere a sort of corresponding binary version of my image. Binary version, If I'm recalling Im reconstructing it. This process (recalling) means to reconstruct pixel by pixel the color. If a pixel is a neuron, the color is its output. Simplest way: pixel is on or off. Meaning, the neuron is outputting 1 or 0. If the image is actually corresponding to a population of neurons, I have to understand ther'es a topological relation of the neurons. Evolving dynamic system tho', if I'm in t it's like that, in t+1 the neuron output are changing... If I try to make the virtual image evolving in time.. recalling correctly means that after time, from t+k the output is no longer changing. Im in condition of stable recall. I don't know if the true recall of the painting, but could be.
How to start the evolution? I have to put into the network somehing that is omogeneous to the image. Autoassociation: input is homogen. to the [output?] network. So far we have seen FFN that are no able to associate,

LTM classification



Explicit memory—memory with awareness; information can be consciously recollected; also called declarative memory (knowing that)

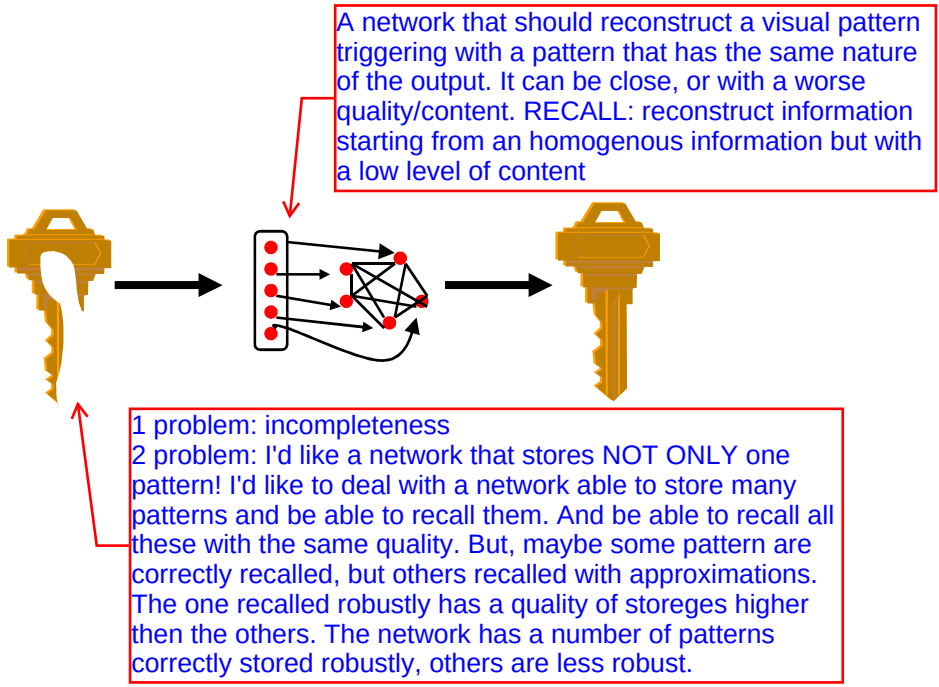
Implicit memory—memory without awareness; memory that affects behavior but cannot consciously be recalled; also called non-declarative memory (knowing how)

Tought: It's allow us to [??????] reasoning? I can think of a dog.

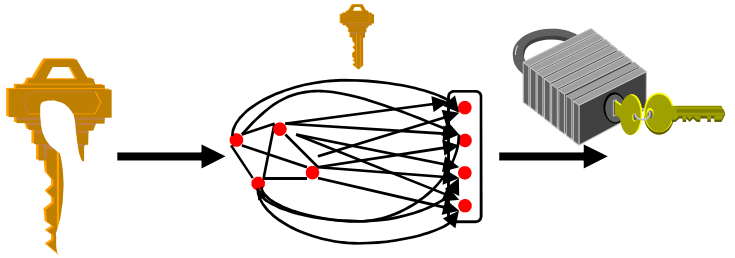
Pattern storage – Auto-associative and hetero-associative memories

Episodic memory representation

- Auto-associative memory
- It is a neural network the solve the following problem
  - Store in the network a set of p configurations, represented by p N-dimensional vectors, q^{μ}, $\mu=1,2,...,p$, in such a way that when in input is presented a new configuration, U_d, then the system answers producing in output the configuration q_d, among the stored configurations, that best approximate U_d.
  - mono or multi-dimensional vectors



- Hetero-associative memory
- It is a neural network the solve the following problem
  - The memory encodes sorted couples of patterns. When the input pattern is similar to the first pattern of a couple previously stored in the memory then the memory provides in output the second pattern of the couple



We need to define the structure, the interconnection, basically If I want an image being represented I have a number of neurons n=MxN, I have to define the connections. Fully connections: for this moment has not an explicit input, the output is from every neuron. pixel1, pixel2, ...Now I need to address the LEARNING: learning parameters to reperesent some patterns (in this case images). Then I'd like to do the EXTRAPOLATION, recall, try to make the network functioning. Im starting at time t, I'm assuming at time t=0 the x neuron is 1, the other 0 etc. (I'm guessing), at time t=1 ?! There's no feedback connection, I must put also a feedback connection. Greatest generalization of my topology: all neurons are fully connected + self-connection. I have many weights. I have activation function for each neuron + threshold for each neuron. The basic function of the neuron is the same seen so far. At time t=1 the output of the neuron will go towards everyneuron and itself.

Artificial associative memories

- Associative memories built as a neural network
  - STRUCTURE
    - Neurons and connections
  - LEARNING
    - Training process to store patterns
  - EXTRAPOLATION (RECALL)
    - Dynamic process (time-dependent)

The information recall is performed not by specifying the memory address but using a **pattern** similar to the content stored in memory

Content-addressable memory

Binary image: binary network. simplest way: heavyside. Or signum if I deal with +-1. Weights could be boundared as well.

Basic recurrent NN: binary output

- Threshold binary unit (MCP)
  - Boolean units
  - Activation potential $$P(t) = \sum_{i=1}^{t} w_i x_i(t) = w_1 x_1(t) + w_2 x_2(t) + w_3 x_3(t) + ..$$
  - the activation of each computing element consumes a unit of time.
  - u(t+1)=sgn(P(t)-S)
    - sgn(x) = 1 if x>0 , sgn(x) = -1 if x<=0
- Interconnection (potentially completely connected)
  - Each unit can get input from any other unit
- Bias S
  - -1<S<1
- Weights
  - from -1 (maximum inhibitory) to +1 (maximum excitatory)

we could boundary weights

- Discrete time: explicit representation

enable timescale for the dynamics

no

McCulloch-Pitts nets and temporal dynamics

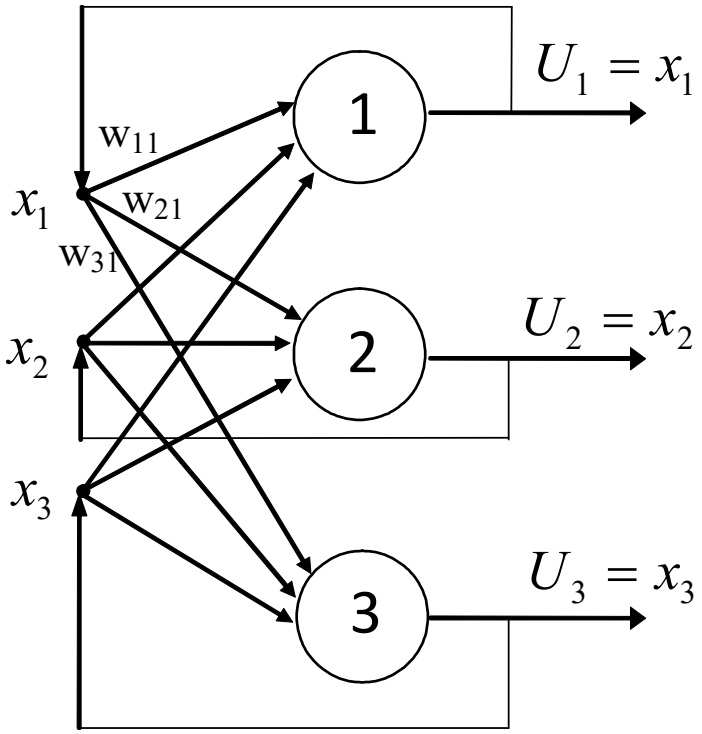
Theorem I: Each boolean unit is equivalent to an opportune McCulloch-Pitts network

Theorem II: Any McCulloch-Pitts network is finite automata and any finite automata is equivalent to an opportune McCulloch-Pitts network

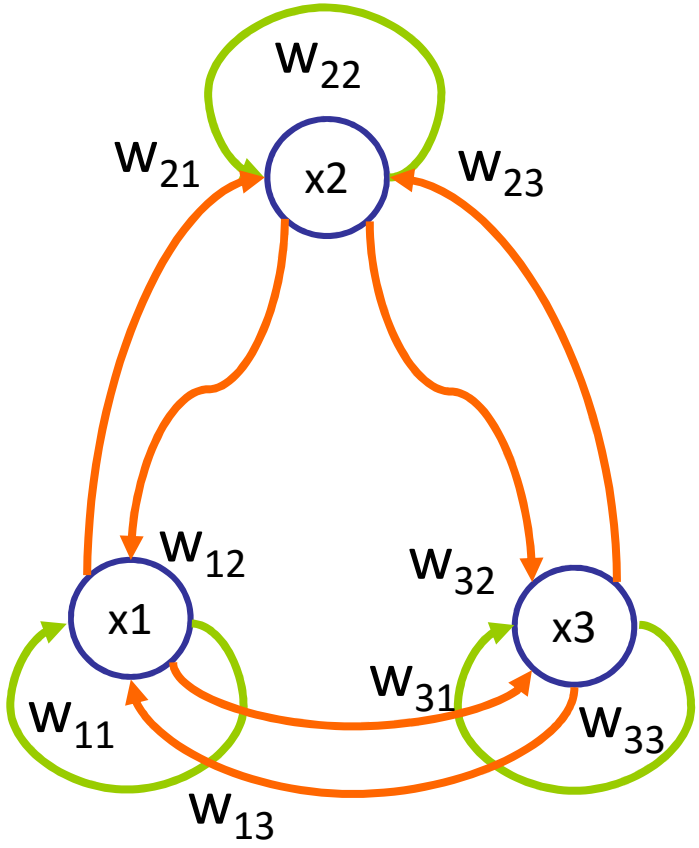
State≡pattern, Temporal dynamics

The two charts are equivalent. This allows me to define the state of each neuron, its output at certain timestamp, but I also have a state of the network, stacking all the outputs into an array, which can reconstruct the image.

Topology of the recurrent network

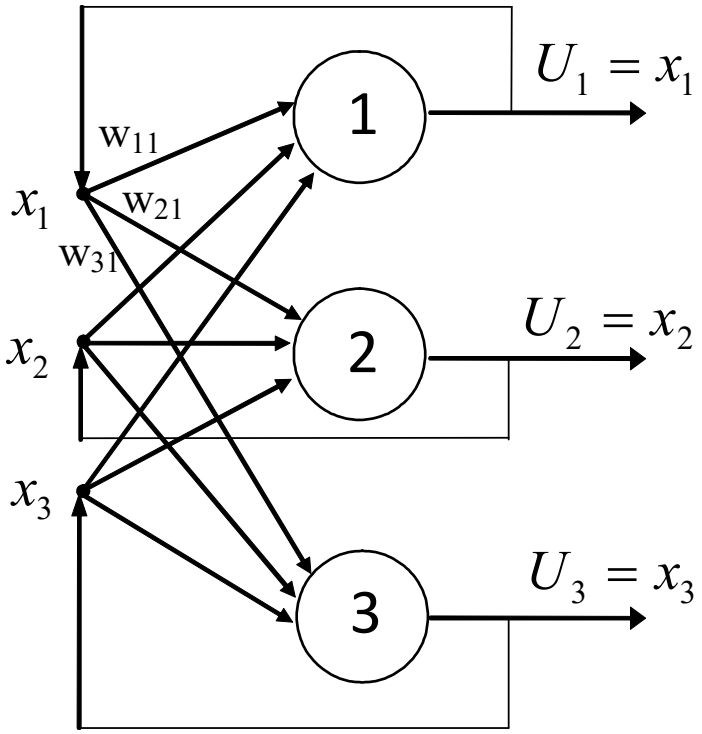


s = [x1,x2,x3] is the state of the network



Network graph

Topology of the recurrent network

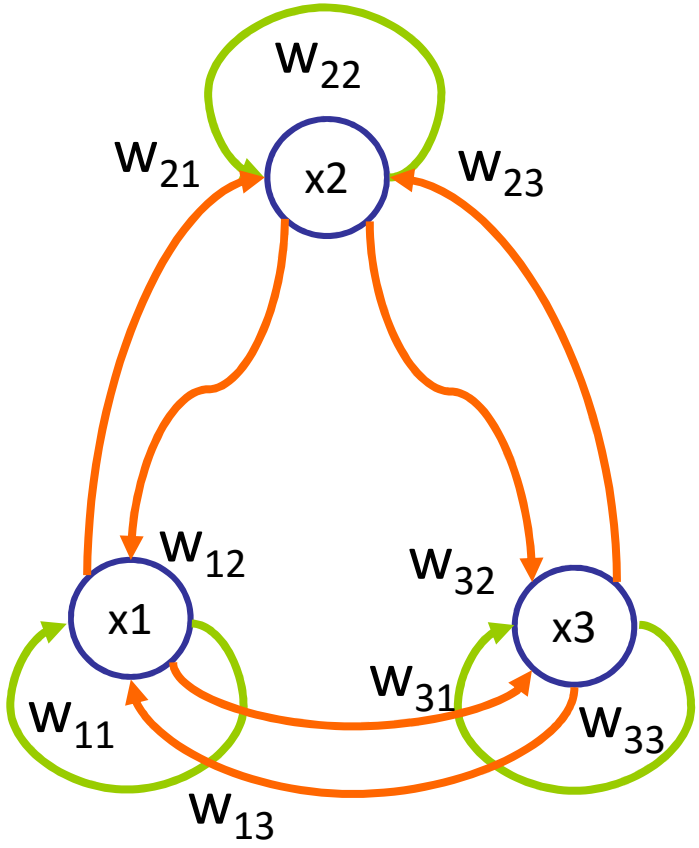


s = [x1,x2,x3] is the state of the network

$$U_i(t+1) = \text{sgn}\left(\sum_j w_{ij} x_j - S_j\right) \quad with \quad x_j = U_j(t)$$

output at the neuron at the step before

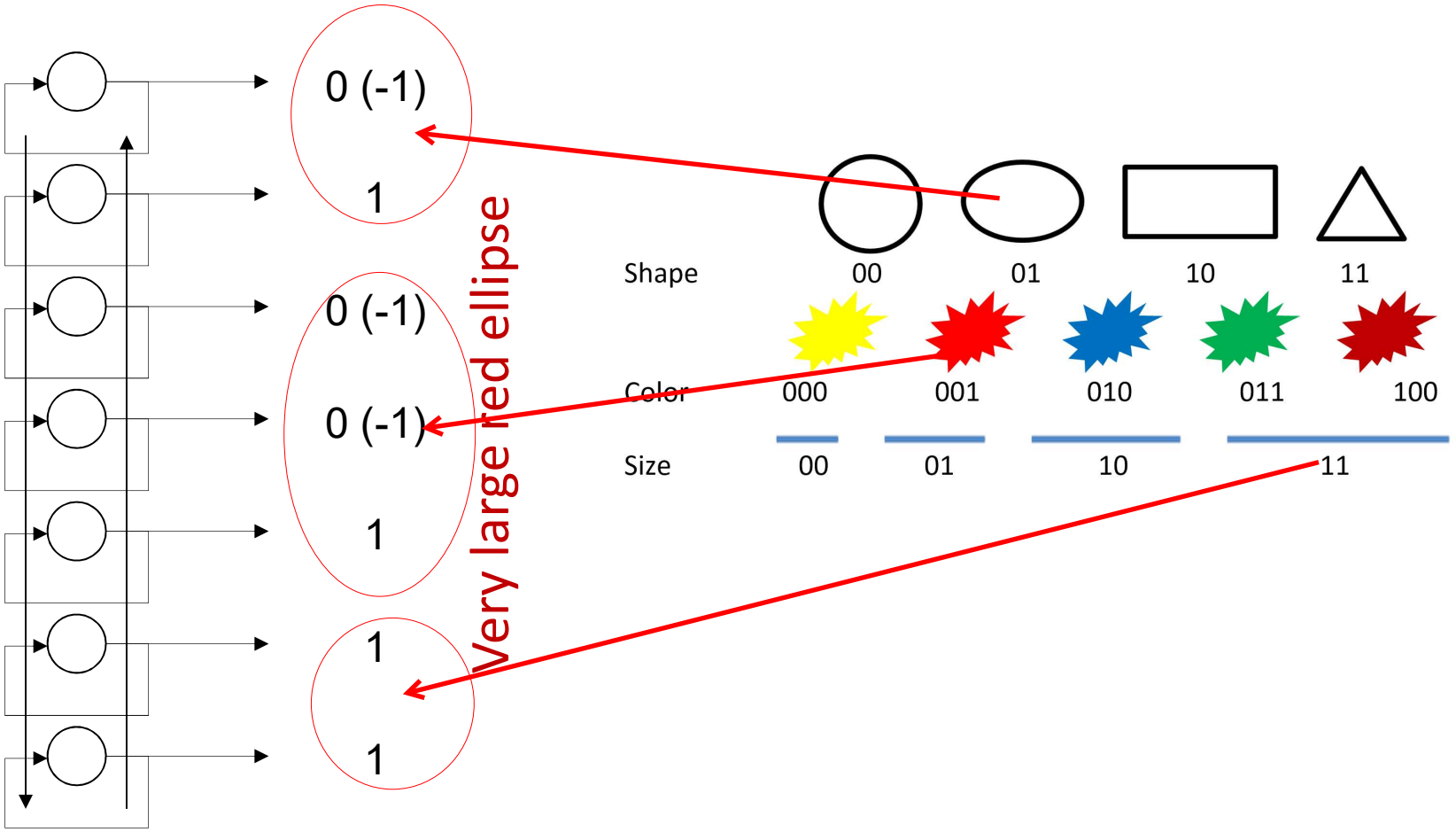
Read the time step



Network graph

In this case I'm modelling a shape (first bits) then the other bits the color, then the two following size. I'm encoding a lot of information!

Binary memory pattern – network state



2 fondamental questions: how to evolve the network? There are two ways of doing it.
1.) Parallel way: there's a sync among the neurons. Meaning: at specific time stamp t Im computing the output of all the neurons, producing the output at time t+1.
2.) Sequential: the neurons can be not sync. **At time t Im considering just one neuron**. Computing it, after that the output of ALL neurons. You can select randomly the neuron, there's no rule for the choosing. (Even again[?]) Or sequentially correctly. Or according to other rules: probabilistic rules.
Physiological way: 2nd. There's some sort of sync in the brain, but you really don't have a clock. This means the 2nd is more close to the brain physiology.

Dynamics of the network

- Parallel (synchronism)
  - At each time instant, all units simultaneously compute the value of action potential and update their own state

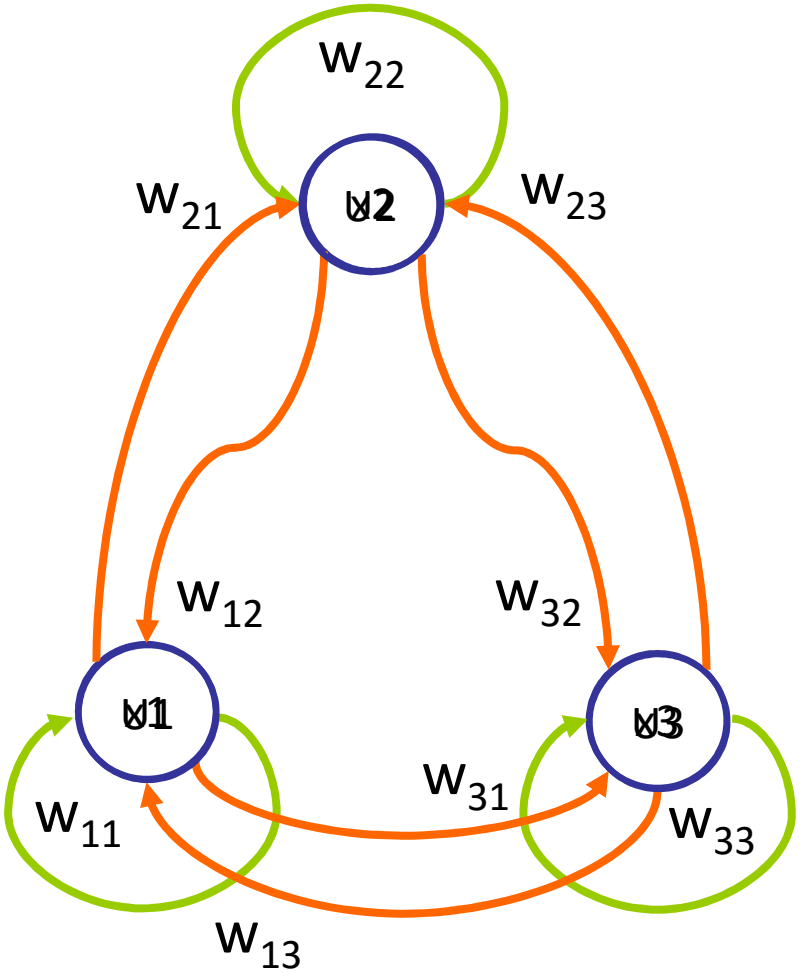
All neurons update their activity states **simultaneously** at discrete time steps t, where t = 1, 2, …, as if governed by a clock. The inputs of every neuron in the network are determined by the same activity state of the network in the time interval (t-1) < n < t. This choice requires a **central clock** to synchronize the neurons.

- Sequential (Hopfield: a-synchronism)
  - At each time instant, a specific unit computes its action potential and updates its own state (more natural for both brains and artificial networks)
  - The probability of two units firing simultaneously is zero

All neurons are updated one by one, where one can proceed in different ways:

a. the update sequence is deterministic (ex: x1 then x2, then x3 ,...)
b. the update sequence is obtained randomly (stochastic dynamic system)
c. let each unit independently choose to update itself, with some constant probability per unit, according to $U_i(t+1) = \mathrm{sgn}\left(\sum_j w_{ij} x_j - S_j\right)$

In this modality: every neuron coming up for a decision has full information about all the decisions of the individual neurons that have been updated before it.



Sequential

beginning: computing the output of the first neuron

1 $U_1=f(x_1,x_2,x_3)$

$$U_1 = x_1(t+1) = \mathrm{sgn}(w_{11}x_1(t) + w_{12}x_2(t) + w_{13}x_3(t))$$

second step: computing hte output of the 2nd, but using the NEW value of the first neuron! Keeping the old vals of the other

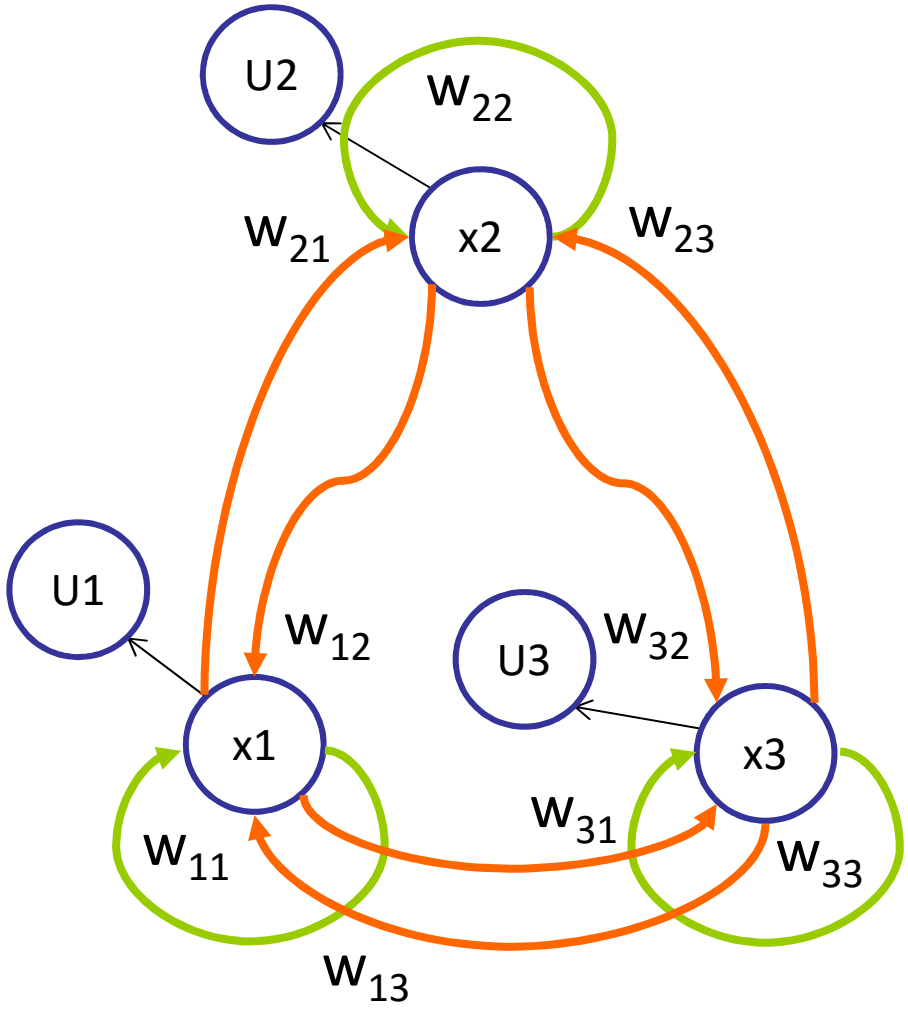
2 $U_2=f(U_1,x_2,x_3)$

$$U_2 = x_2(t+2) = \mathrm{sgn}(w_{21}x_1(t+1) + w_{22}x_2(t) + w_{23}x_3(t))$$

3 $U_3=f(U_1,U_2,x_3)$

$$U_3 = x_3(t+3) = \mathrm{sgn}(w_{31}x_1(t+1) + w_{32}x_2(t+2) + w_{33}x_3(t))$$

To update all the network you need at least the number of neurons step. But If I'm using a random selection you are not sure about the exact number (stocastic selection could pick up the same many times).



Sequential

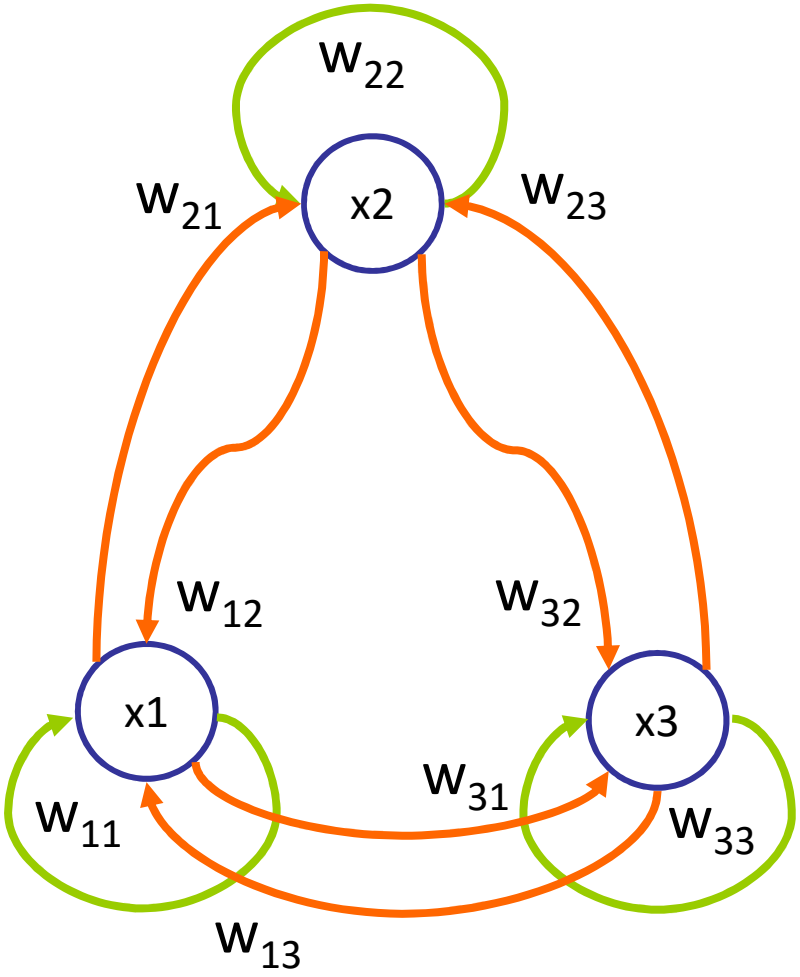
1 $U_1=f(x_1,x_2,x_3)$
2 $U_2=f(U_1,x_2,x_3)$
3 $U_3=f(U_1,U_2,x_3)$

Parallel

In just 1 time step I have updated all my network. It's like freezing the output

1 $U_1=f(x_1,x_2,x_3)$
2 $U_2=f(x_1,x_2,x_3)$
3 $U_3=f(x_1,x_2,x_3)$

$$U_1 = x_1(t+1) = \mathrm{sgn}(w_{11}x_1(t) + w_{12}x_2(t) + w_{13}x_3(t))$$
$$U_2 = x_2(t+1) = \mathrm{sgn}(w_{21}x_1(t) + w_{22}x_2(t) + w_{23}x_3(t))$$
$$U_3 = x_3(t+1) = \mathrm{sgn}(w_{31}x_1(t) + w_{32}x_2(t) + w_{33}x_3(t))$$



Sequential

1 $U_1=f(x_1,x_2,x_3)$
2 $U_2=f(U_1,x_2,x_3)$
3 $U_3=f(U_1,U_2,x_3)$

Parallel

1 $U_1=f(x_1,x_2,x_3)$
2 $U_2=f(x_1,x_2,x_3)$
3 $U_3=f(x_1,x_2,x_3)$

Time t+1

So, this kind of system can be seen like a sort of EVOLVING PROCESS end up into a STATE that can be STABLE, meaning, from a time stamp on the output is no longer changing. This final state is called equilibrium point. Or is an ATTRACTOR of the system. I'd need for example a certain number of step to converge to an equilibrium point. This number is variable, but I could have complete instability! I could enter with a pattern which doesn't change the dinamic of the network! It's a STABLE pattern for the network, it's something that was learnt by the network.

Convergence -> Stability -> Pattern is stable for the network -> I have recalled correctly.
I have fed the network with an input pattern, and after k I evolved with one pattern that represent the stored pattern.

Cycle) Or it's not sure: it changes from one pattern to the other, both pattern could be coped with the input pattern, two **stored pattern** that could be matched with the input pattern.

Evolution of the dynamics

- Equilibrium point (attractor)
  - After a transitory, the network converges to a fixed state
  - $s(t)=s(t+1)=\ldots=\underline{s}[\underline{x}_1, \underline{x}_2, \underline{x}_3,\ldots, \underline{x}_n]$
- Cycle ← It could cycle in between two patterns (or three)
  - The network steps periodically into a predefined sequence of states. The number of state of the sequence represents the cycle length
  - $s(t) = s(t+\tau)$
- Chaotic behavior ← Patterns continously changing, no way to endup in stability
  - No regular state dynamics

Convergence ≡ Stability

The network can be regarded as system with points ATTRACTION that are storing patterns, the other points are state of the network that are not stable.

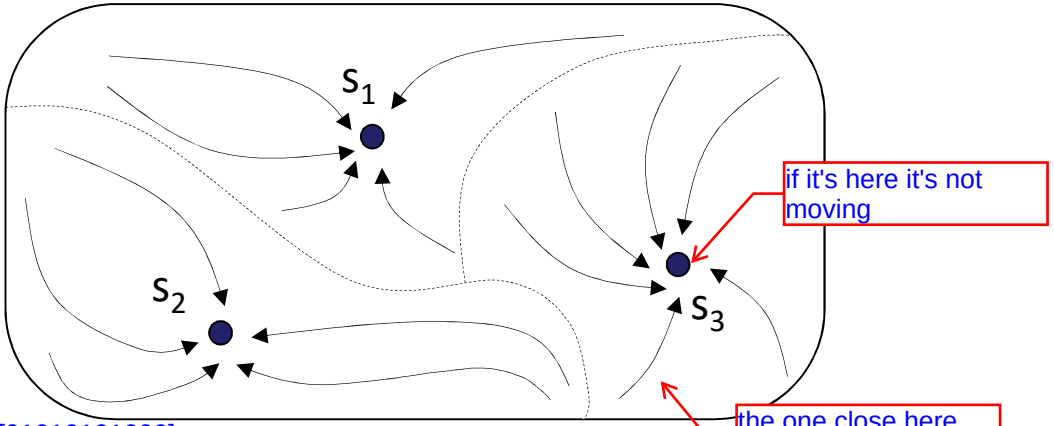
Information storage

- Binary networks store bit patterns
- A *n*-neuron binary network can represent/store 2^n different patterns

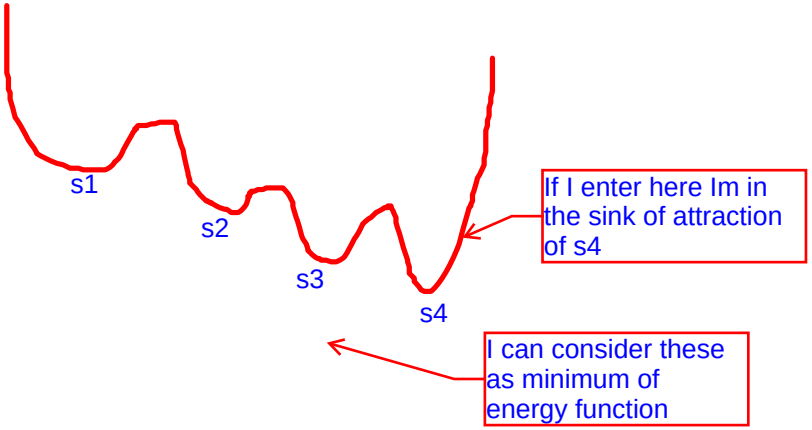
The space of all possible states of the network, is called the **configuration space.**

Sinks(basins) of attraction:
Division of the configuration space performed by the stored patterns

Stored patterns should be **stability states** for the network



s1 could be = [01010101000]
s2 could be = [00000011111]
....
what happens if Im entering the network with s=[01010101001]? Its smilar to s1.. so probably Im starting close to s1 and Im expecting if all its good I'll get a stable output s1 after a k. So, it means THE PATTERN BELONGS TO THE SINK OF ATTRACTION OF THE STORED PATTERN S1. That pattern is not in the sync of attraction of s2.



s1 has an energy lower than the input. All the patterns can have different level of energy, meaning that for example in this graph s4 is the lowest, meaning the most stable pattern.

The lowest the energy the more stable the pattern. There can be states of the network with energy level lower of some stored patterns. This is a problem tho', because these states can be confused? You should avoid these condition using a smart training but you can't avoid this condition [cit.]

Problem: does it depends on the way Im using to evolve the network? (Parallel or sequential) There are two principles demostrable. We can say that, **parallel evolution will ALWAYS end into a steady state, or a cycle. You are PREVENTED to have a CHAOTIC behaviour.**

Otherwise, If I use sequential dynamic, which is more biological founded... I can be affected by chaotic behaviours :(((. Meaning that there can be a starting point that prevets to get any of the stored patterns, jumping in between states that are not stored patterns of the network.

Cycle doenst mean that you switc immediatly from a stored to a stored

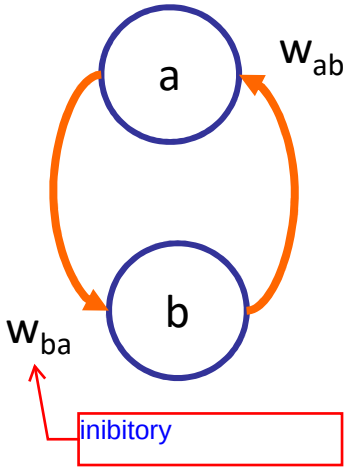
How is it possible to anticipate the evolution of a network?

- The evolution of a binary recurrent network with parallel dynamics terminates into a cycle or and equilibrium state
- The sequential dynamics implies that starting from particular initial states the evolution can be chaotic

Two-neuron network

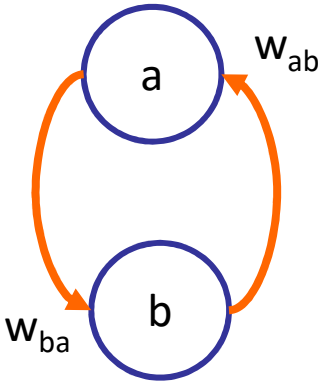
- Neurons a and b:
  - output range: -1 , +1
- The threshold is null
- $w_{ab}=1$, $w_{ba}=-1$
- Possible network states:
  (-1,-1), (1,-1), (-1,1), (1,1)

For a general network of n neurons you have 2^n states.



Transitions (parallel dynamics)

- Init: (-1,-1) (t=0)
  - **a** -> (-1) **b** -> (+1)
  - (-1,+1) (t=1)
  - **a** -> (+1) **b** -> (+1)
  - (+1,+1) (t=2)



$$a(t+1) = \mathrm{sgn}(w_{ab} b(t) - S_a) = \mathrm{sgn}(1*-1-0) =$$

$$= \mathrm{sgn}(-1) = -1$$

$$b(t+1) = \mathrm{sgn}(w_{ba} a(t) - S_b) = \mathrm{sgn}(-1*-1-0) = +1$$

$$a(t+2) = \mathrm{sgn}(w_{ab} b(t+1) - S_a) = \mathrm{sgn}(1*+1-0) = +1$$

$$b(t+2) = \mathrm{sgn}(w_{ba} a(t+1) - S_b) = \mathrm{sgn}(-1*-1-0) = +1$$

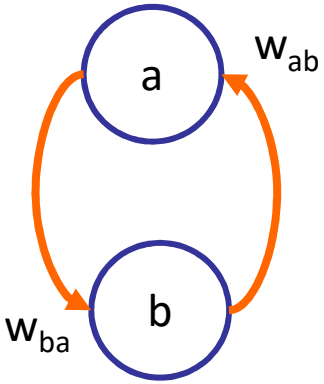
Transitions (parallel dynamics)

Cycle

- Init: (-1,-1) (t=0)
  - **a** -> (-1) **b** -> (+1)
  - (-1,+1) (t=1)
  - **a** -> (+1) **b** -> (+1)
  - (+1,+1) (t=2)
  - **a** -> (+1) **b** -> (-1)
  - (+1,-1) (t=3)
  - **a** -> (-1) **b** -> (+1)
  - (-1,-1) (t=4)

they are all the output lol

after 4 steps I got -1-1, I have a cycle



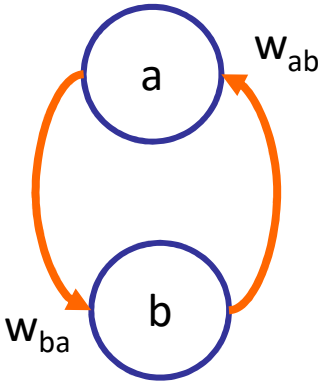
$$a(t+3) = \mathrm{sgn}(w_{ab} b(t+2) - S_a) = \mathrm{sgn}(1*+1-0) = +1$$
$$b(t+3) = \mathrm{sgn}(w_{ba} a(t+2) - S_b) = \mathrm{sgn}(-1*+1-0) = -1$$

$$a(t+4) = \mathrm{sgn}(w_{ab} b(t+3) - S_a) = \mathrm{sgn}(1*-1-0) = -1$$
$$b(t+4) = \mathrm{sgn}(w_{ba} a(t+3) - S_b) = \mathrm{sgn}(-1*+1-0) = -1$$

Transitions (sequential dynamics)

1st neuron then 2nd and so on

- Init: (-1,-1) (t=0)
  - **a** -> (-1,-1) (t=1)
  - **b** -> (-1,+1) (t=2)



$$a(t+1) = \mathrm{sgn}(w_{ab} b(t) - S_a) = \mathrm{sgn}(1 * -1 - 0) = -1$$

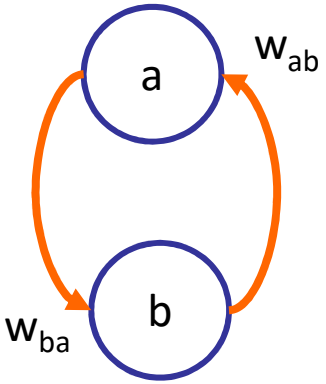
$$b(t+1) = b(t)$$

$$b(t+2) = \mathrm{sgn}(w_{ba} a(t+1) - S_b) = \mathrm{sgn}(-1 * -1 - 0) = +1$$

$$a(t+2) = a(t+1)$$

Transitions (sequential dynamics)

- Init: (-1,-1) (t=0)
  - **a** -> (-1,-1) (t=1)
  - **b** -> (-1,+1) (t=2)
  - **a** -> (+1,+1) (t=3)
  - **b** -> (+1,-1) (t=4)



$$a(t+3) = \mathrm{sgn}(w_{ab} b(t+2) - S_a) = \mathrm{sgn}(1*+1-0) = 1$$

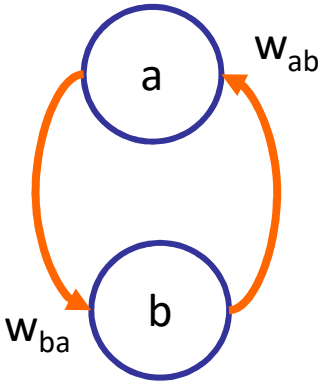
$$b(t+3) = b(t+2)$$

$$b(t+4) = \mathrm{sgn}(w_{ba} a(t+3) - S_b) = \mathrm{sgn}(-1*+1-0) = -1$$

$$a(t+4) = a(t+3)$$

Transitions (sequential dynamics)

- Init: (-1,-1) (t=0)
  - **a** -> (-1,-1) (t=1)
  - **b** -> (-1,+1) (t=2)
  - **a** -> (+1,+1) (t=3)
  - **b** -> (+1,-1) (t=4)
  - **a** -> (-1,-1) (t=5)
  - **b** -> (-1,+1) (t=6)



$$a(t+5) = \mathrm{sgn}(w_{ab} b(t+4) - S_a) = \mathrm{sgn}(1*-1-0) = -1$$

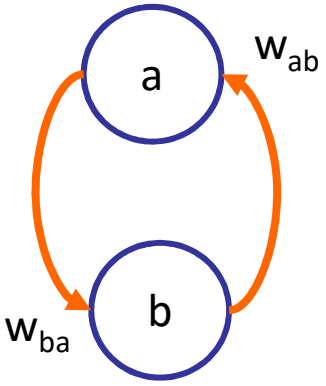
$$b(t+5) = b(t+4)$$

$$b(t+6) = \mathrm{sgn}(w_{ba} a(t+5) - S_b) = \mathrm{sgn}(-1*-1-0) = +1$$

$$a(t+6) = a(t+5)$$

Transitions (sequential dynamics)

- Init: (-1,-1) (t=0)
  - **a** -> (-1,-1) (t=1)
  - **b** -> (-1,+1) (t=2)
  - **a** -> (+1,+1) (t=3)
  - **b** -> (+1,-1) (t=4)
  - **a** -> (-1,-1) (t=5)
  - **b** -> (-1,+1) (t=6)
  - **a** -> (+1,+1) (t=7)
  - **a** -> (+1,-1) (t=8)



$$a(t+7) = \mathrm{sgn}(w_{ab} b(t+6) - S_a) = \mathrm{sgn}(1*+1-0) = +1$$

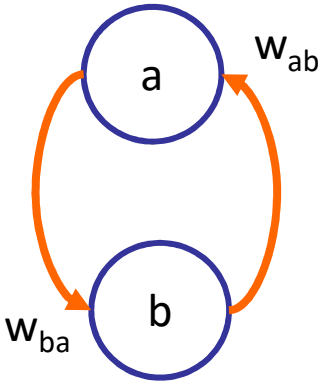
$$b(t+7) = b(t+6)$$

$$b(t+8) = \mathrm{sgn}(w_{ba} a(t+7) - S_b) = \mathrm{sgn}(-1*+1-0) = -1$$

$$a(t+8) = a(t+7)$$

Transitions (sequential dynamics)

- Init: (-1,-1) (t=0)
  - **a** -> (-1,-1) (t=1)
  - **b** -> (-1,+1) (t=2)
  - **a** -> (+1,+1) (t=3)
  - **b** -> (+1,-1) (t=4)
  - **a** -> (-1,-1) (t=5)
  - **b** -> (-1,+1) (t=6)
  - **a** -> (+1,+1) (t=7)
  - **a** -> (+1,-1) (t=8)



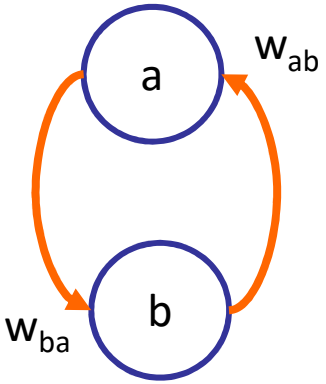
- Init: (-1,+1) (t=0)
  - **a** -> (+1,+1) (t=1)
  - **b** -> (+1,-1) (t=2)
  - **a** -> (-1,-1) (t=3)
  - **b** -> (-1,+1) (t=4)
  - **a** -> (+1,+1) (t=5)
  - **b** -> (+1,-1) (t=6)
  - **a** -> (-1,-1) (t=7)

Changing the initial state….

leads to different paths but … there is convergence to a cycle

What if with a random sequence

- Init: (-1,-1) (t=0)
  - **b** -> (-1,+1) (t=1)
  - **b** -> (-1,+1) (t=2)
  - **b** -> (-1,+1) (t=3)
  - **a** -> (+1,+1) (t=4)
  - **b** -> (+1,-1) (t=5)
  - **a** -> (-1,-1) (t=6)
  - **a** -> (-1,-1) (t=7)
  - **a** -> (-1,-1) (t=8)
  - **a** -> (-1,-1) (t=9)
  - **b** -> (-1,+1) (t=10)



we can also have no way of having stability , we won't find stability

The evolution can become erratic or even chaotic

Transitions, memory and stability

- Deterministic state selection
  - Predefined convergence into a subspace
- Random state selection
  - There can be convergence for particular initial states
  - Or neither equilibrium states nor cycles
  - How to avoid chaotic evolutions?

basic problem:

to store a set of R patterns $\{s_i\}$ in such a way that when presented with a new pattern s_k , the network responds by producing one of the stored patterns $\{s_i\}$ that most closely resembles s_k

or to check whether small deviations of the new pattern from the stored patterns are corrected as the network evolves.

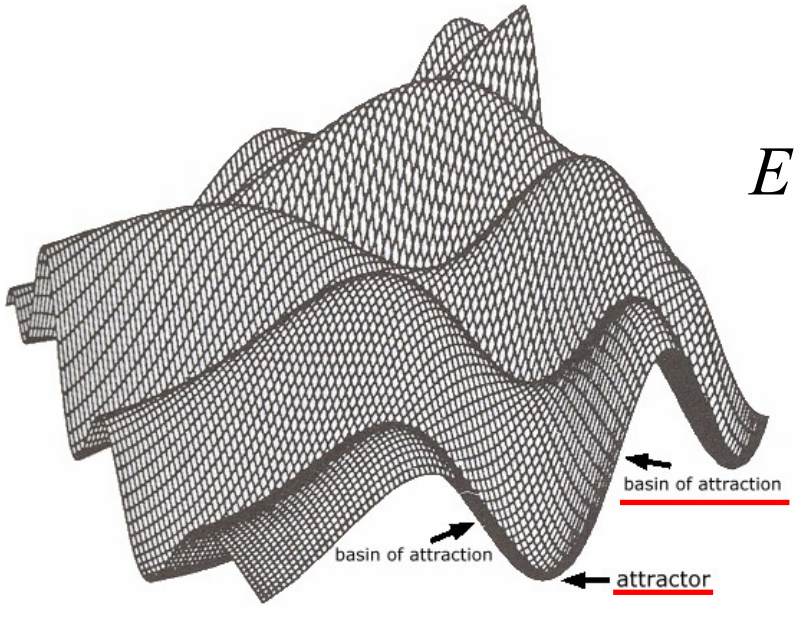
Study the network stability using the potential energy E (Lyapunov function) as a function of the network state s

Modelling the energy function

Sort of parabolic function, there's a dependance on neuron threshold (Sk).

THIS IS FOR MCP NETWORK WITH SEQUENTIAL

Energy (Lyapunov) function of the network



$$E(t) = -\frac{1}{2}\sum_{i,j}^{N} w_{ij} x_i(t) x_j(t) + \sum_{k}^{N} x_k(t) S_k$$

Quadratic function

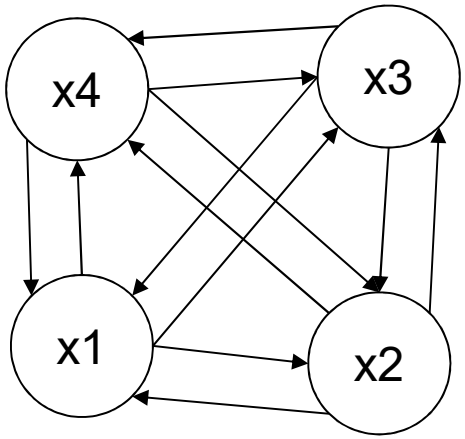
Minimum principle

Theorem V: If MCP network with sequential dynamic is characterized by a decreasing Liapunov function E(t) up to a minimum then the evolution of the network terminates necessarily into an equilibrium state

It can be demostrated that using sequential this will always converge to a stable point.

Hopfield network (1962)

- **Hopfield network (HN)**: MCP symmetric network with N units without self-connection
  - $w_{ij}=w_{ji}$ and $w_{ii}=0$
- **Activation potential** $\neq 0$
- **State**: **s =** [x1 x2 x3 x4]



Hopfield network, there are no self connections + The symmetry: If I am exciting you you are exciting me, and the opposite. With the same amplitude. (weight?)

- **Connectivity degree K**: number of inputs to a unit from other units
- **Theorem VI**: If the HN network evolves according to **sequential** dynamic then each evolution terminates into an **equilibrium state**

How to learn? we'll see HEBBIAN LEARNING, but it's only for fully connected (not this) [?].

Stability of HN and Lyapunov function

Energy $$E(t) = -\frac{1}{2}\sum_{i,j}^{N} w_{ij} x_i(t) x_j(t) + \sum_{k}^{N} x_k(t) S_k$$

NO

Select the unit *r* and compute $E(t)$ at time t (remember: $w_{ij}=w_{ji}$ and $w_{ii}=0$)

$$E(t) = -x_r(t)\sum_{j\neq r}^{N} w_{rj} x_j(t) + x_r(t) S_r - \frac{1}{2}\sum_{i,j\neq r}^{N} w_{ij} x_i(t) x_j(t) + \sum_{k\neq r}^{N} x_k(t) S_k$$

Compute $E(t+1)$

$$E(t+1) = -x_r(t+1)\sum_{j\neq r}^{N} w_{rj} x_j(t) + x_r(t+1) S_r - \frac{1}{2}\sum_{i,j\neq r}^{N} w_{ij} x_i(t) x_j(t) + \sum_{k\neq r}^{N} x_k(t) S_k$$

Δ E= $E(t+1) - E(t)$

$$\Delta E = -\left(x_r(t+1) - x_r(t)\right)\left(\sum_{j\neq r}^{N} w_{rj} x_j(t) - S_r\right)$$

Analysis

$$\Delta E = -\left(x_r(t+1) - x_r(t)\right)\left(\sum_{j \neq r}^{N} w_{rj} x_j(t) - S_r\right)$$

1. $x_r(t+1) = x_r(t)$ then $\Delta E=0$
2. $x_r(t+1) = 1$ $x_r(t) = -1$ then $\Delta E<0$
   a) $x_r(t+1) - x_r(t) = 2$
   b) as $x_r(t+1) = 1$ then $\Sigma_{j \neq r} w_{rj} x_j(t) - S_r > 0$
3. $x_r(t+1) = -1$ $x_r(t) = 1$ then $\Delta E<0$
   a) $x_r(t+1) - x_r(t) = -2$
   b) as $x_r(t+1) = -1$ then $\Sigma_{j \neq r} w_{rj} x_j(t) - S_r < 0$

Theorem VI is demonstrated

Goles networks

no

- Let HN a **symmetric** network without self-connection, constituted by threshold units so that the activation potentials cannot be null. If the network evolves according to parallel dynamic then each evolution terminates into a cycle of length 2 or an equilibrium state

- Let HN an **anti-symmetric** network (wij=-wji) without self-connection constituted by threshold units so that the activation potentials cannot be null. If the network evolves according to parallel dynamic then each evolution terminates into a cycle of length 4 (see former example).

Idea: exploit physiology of neurons, or exploiting the fact that if they are interacting with each other increasing the interaction you have to strenght the interconnection, meaning, from a bio point of view increase the senstivity of the interface, otherwise if they are not in phase, they are asyncronous, you have to weaken the relationship in between. And strength I means increasing the weights, and weak: decrease them. Princple of this rule.

Supervised Hebbian learning rule (D. Hebb, 1949)

1. If two neurons on either side of a synapse are activated **synchronously**, then the effectiveness of the synapse is selectively **strengthened**

2. If two neurons on either side of a synapse are activated **asynchronously**, then the effectiveness of the synapse is selectively **weakened**

- Hebbian:
  - Correlated signals cause strengthening
  - Un- or negatively correlated signals cause weakening
- Anti-Hebbian:
  - Correlated signals cause weakening
  - Un- or negatively correlated signals cause strengthening

- Hebb rule
  - Coherence between inputs
- Delta rule
  - Error in reproducing the output

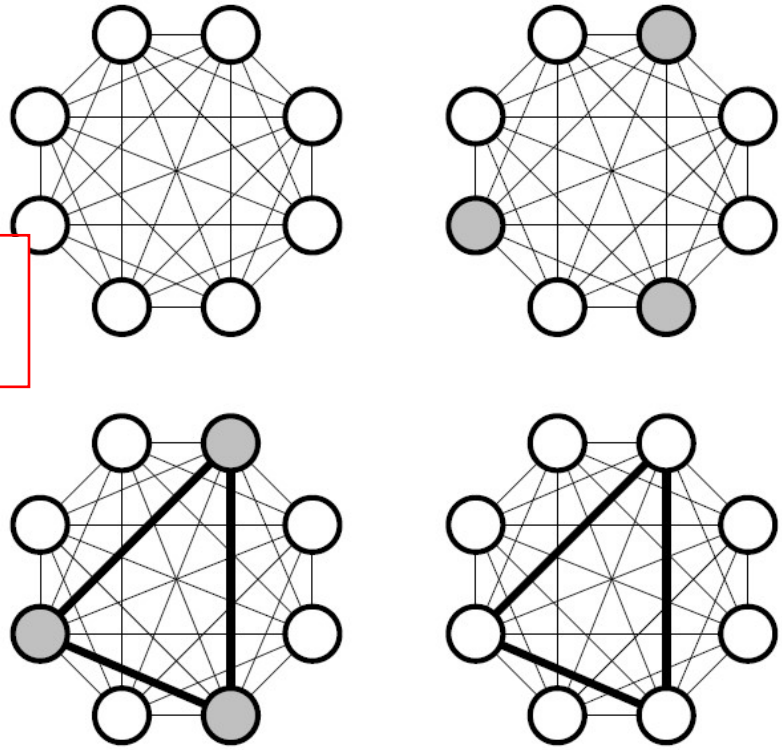
19-10
Tries to emulate the concept of plasticity. When two neurons are sync and are firing at the same time, this is an emulation of the "strengthing". We use the product of the two signal.
Coherence: increase the weight
Incoherence: decrease the weight.
We train the network with this principle. IT's an **UNSUPERVISED** technique. I have no ref. for the training part. I just see coherence and uncoherence.

Hebbian rule

Synaptic effectiveness: if a neuron *i* produces an output towards a neuron *k* then *k* gets in input the signal from *i* and produces an output. The *i*-*k* connection is empowered

- Weights: $w_{ik} = \eta\, x_i(t)\, x_k(t)$
- $\eta > 0$ is the learning rate
- Typically $\eta = 1/N$ (number of neurons)

product of the " corrisponding bit" xi and xk. This is weighted by the learning rate



When three of the units are activated by an outside stimulus their mutual connections are strengthened. The next time some of them are activated they will activate each other

Rule for the weights:
I have a network with N neurons, with full and self connection. Considering a pattern s, I'd like to store the pattern with my network, like it could be s=[-1 1 ..... -1]. The activation is a signum f(), we can regard any input as a bit of this vector. Hebbian rule is computing the weight, by aking the scalar product of this vectory by itself, then it's weighted by the factor 1/N (learning rate). You don't need iteration, it's a closed solution. This means you don't need an initial weight set, or matrix. The idea is that: [-1 1 ..... -1]'[-1 1 ..... -1] this will be positive for -1*-1, negative for -1*1. So the first neuron is probably inibithing the second neuron (-1 vs 1 of the second neuron). **With this rule we consider also self-connection. [XXX] diagonal of the weights.**

Hebbian rule

$$s = \begin{bmatrix} \bar{x}_1 & \bar{x}_2 & \dots & \bar{x}_N \end{bmatrix} \qquad w_{ij} = \frac{1}{N} \bar{x}_i \bar{x}_j \qquad S_i = 0$$

If the bits corresponding to neurons *i* and *j* are equal in pattern *s*, then the product $\bar{x}_i \bar{x}_j$ will be positive. This would, in turn, have a positive effect on the weight w_{ij} and the values of *i* and *j* will tend to become equal. The opposite happens if the bits corresponding to neurons *i* and *j* are different.

Releasing Hopfield condition so that $w_{ii} \neq 0$

stable state = stable recall
other states = useless recall

Hopfield network

no self connection

- A **Hopfield net** is composed of binary threshold units with recurrent connections between them.
- Recurrent networks of non-linear units are generally very difficult to analyze. The y can behave in many different ways:
  - ✓ Settle to a stable state
  - ✓ Oscillate
  - ✓ Follow chaotic trajectories (unpredictable)
- **John Hopfield** first conceived that if the connections are symmetric, then there is a global energy function for the network
  - ➢ Any network state has an energy
  - ➢ The basic element of the energy depends on one single connection weight and the binary states of two neurons
  - ➢ Evolving the network leads to end up to a minimum of the network

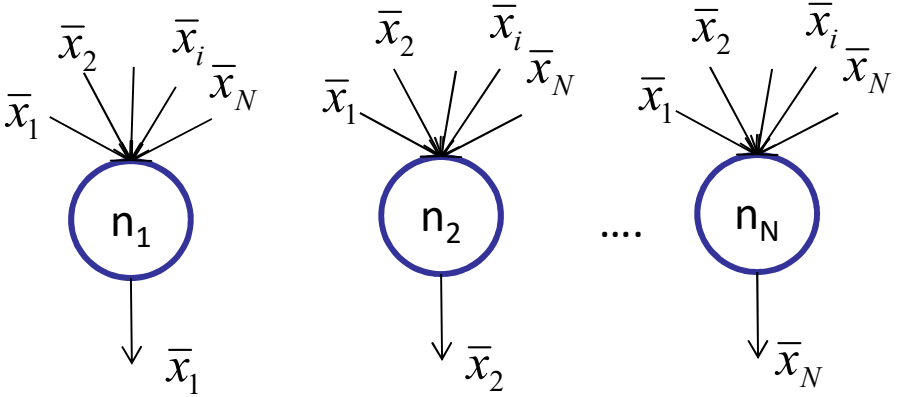
Now I consider a single pattern, and I'd like to use this rule to store patterns. The condition for succeding in storing is that I'd like the pattern to be a stable point of the system. If Im entering the network with the same pattern (assuming parallel dynamics), computing AF of all the neurons, with weights known (hebbian rule). I apply the signum function and compute the output of each neuron. If the pattern is correctly stored in the network, if I enter with a pattern used for computing the weights, I should obtain the same pattern.

Associative memory using Hebbian rule

Single pattern

The condition for one pattern $s = [\bar{x}_1 \quad \bar{x}_2 \quad \dots \quad \bar{x}_N]$ which should be memorized is:

$$\bar{x}_i = \operatorname{sgn}\left(\sum_j w_{ij}\bar{x}_j\right) \qquad \forall i$$



How to store the pattern?

Hebbian learning $w_{ij} = \frac{1}{N}\bar{x}_i\bar{x}_j$

$$W = \frac{1}{N}\begin{bmatrix}\bar{x}_1\\ \bar{x}_2\\ \dots\\ \bar{x}_N\end{bmatrix} * [\bar{x}_1 \quad \bar{x}_2 \quad \dots \quad \bar{x}_N]$$

$$W = \frac{1}{N}\begin{bmatrix} \bar{x}_1\bar{x}_1 & \bar{x}_1\bar{x}_2 & .. & \bar{x}_1\bar{x}_N \\ \bar{x}_2\bar{x}_1 & \bar{x}_2\bar{x}_2 & .. & \bar{x}_2\bar{x}_N \\ .. & .. & .. & .. \\ \bar{x}_N\bar{x}_1 & \bar{x}_N\bar{x}_2 & .. & \bar{x}_N\bar{x}_N \end{bmatrix}$$

w11 w12

symmetric

result, a matrix obtained by one single pattern (_s_).

As we are considering binary output (-1 and +1). This matrix is symmetric. If one is exiting a neuron, that neuron is exting the first. The opposite in the opposite case.

Example

Single pattern $w_{ij} = \frac{1}{N} \bar{x}_i \bar{x}_j$ $W = \frac{1}{N} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \dots \\ \bar{x}_N \end{bmatrix} * \begin{bmatrix} \bar{x}_1 & \bar{x}_2 & \dots & \bar{x}_N \end{bmatrix}$

Store the pattern *s* = [1 1 1 1 1 -1 -1]'

pattern I'd like to store in 7 neurons

W =1/7**s***s'*

cross product!!

first area positive: excitatory

$$W = \begin{bmatrix} 0.1429 & 0.1429 & 0.1429 & 0.1429 & 0.1429 & -0.1429 & -0.1429 \\ 0.1429 & 0.1429 & 0.1429 & 0.1429 & 0.1429 & -0.1429 & -0.1429 \\ 0.1429 & 0.1429 & 0.1429 & 0.1429 & 0.1429 & -0.1429 & -0.1429 \\ 0.1429 & 0.1429 & 0.1429 & 0.1429 & 0.1429 & -0.1429 & -0.1429 \\ 0.1429 & 0.1429 & 0.1429 & 0.1429 & 0.1429 & -0.1429 & -0.1429 \\ -0.1429 & -0.1429 & -0.1429 & -0.1429 & -0.1429 & 0.1429 & 0.1429 \\ -0.1429 & -0.1429 & -0.1429 & -0.1429 & -0.1429 & 0.1429 & 0.1429 \end{bmatrix}$$

[xxxxxxx] excitatory because 6 and 7 are negative??? [??]

second part inibitory contributions

If Im close to the pattern I should get it.. If I'm far I could fail in the recall.. let's check what happens

Example

Single pattern $w_{ij} = \frac{1}{N} \bar{x}_i \bar{x}_j$ $W = \frac{1}{N} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \dots \\ \bar{x}_N \end{bmatrix} * \begin{bmatrix} \bar{x}_1 & \bar{x}_2 & \dots & \bar{x}_N \end{bmatrix}$

Store the pattern *s* = [1 1 1 1 1 -1 -1]'

W =1/7**s***s'*

equivalent of computing in parallel dynamics. In just one cycle I'm computing all the ouputs

exactly the pattern I used: the network is not moving. I obtain it again.

it changed: from the input pattern I got the corrected one! The network reconstruct correctly the stored informations

Input	->	Output
s1 = [1 1 1 1 1 -1 -1]'	-> u1 = sgn(W*s1) =	[1 1 1 1 1 -1 -1]'
s2 = [1 1 1 1 **-1** -1 -1]'	-> u2 = sgn(W*s2) =	[1 1 1 1 1 -1 -1]'
s3 = [1 1 1 **-1 -1** -1 -1]'	-> u3 = sgn(W*s3) =	[1 1 1 1 1 -1 -1]'
s4 = [1 1 **-1 -1 -1** -1 -1]'	-> u4 = sgn(W*s4) =	[1 1 1 1 1 -1 -1]'

I change one bit of the pattern.

still reconstructed!

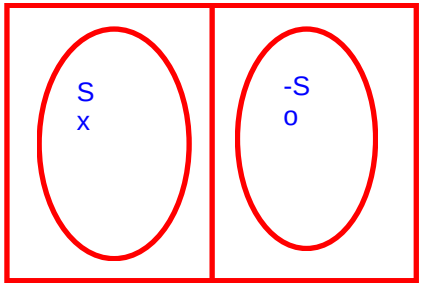
The network is correcting errors in the pattern and so the stored pattern is an attractor it's correcting the bits that are incoherent with the stored pattern. It makes the pattern close to the stored pattern. it corrects the errors.

s5 = [1 **-1 -1 -1 -1** -1 -1]' -> u5 = sgn(W*s5) = **[-1 -1 -1 -1 -1 1 1]'**

completely flipped stored pattern :(. If the network is providing a stable output, this is a stored pattern I never explicitly stored. So, storing a pattern stores also the complementary!!!

This gives me the same weight patterns. Because (-1*-1) = (1*1)

What happened?



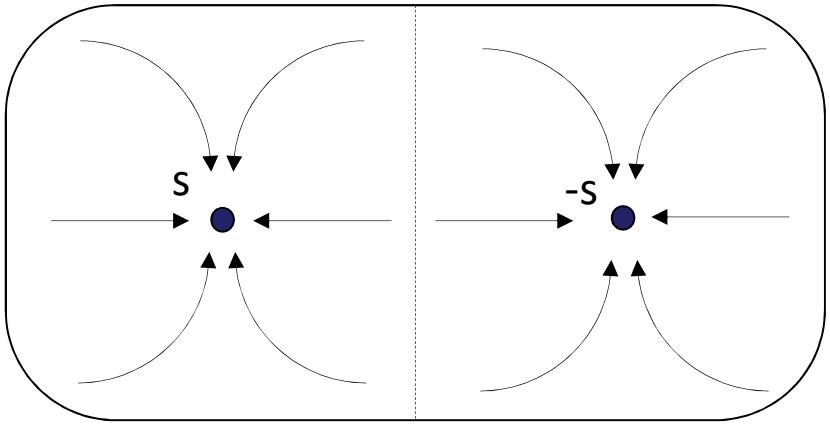
When I learn a pattern I always learn intrisically the opposite pattern. This causes another condition:

$$s = [\bar{x}_1 \quad \bar{x}_2 \quad \ldots \quad \bar{x}_N]$$

$$\bar{x}_i = \text{sgn}\left(\sum_j w_{ij}\bar{x}_j\right) \qquad \forall i \quad \text{with} \quad w_{ij} = \frac{1}{N}\bar{x}_i\bar{x}_j$$

but $w_{ij} = \frac{1}{N}(-\bar{x}_i)(-\bar{x}_j)$ therefore

$$-s = [-\bar{x}_1 \quad -\bar{x}_2 \quad \ldots \quad -\bar{x}_N]$$ is a stable attractor (implicit) for the net



If fewer than half of the bits of the starting patterns S_i are wrong they will be overwhelmed in the sum for the net input

All starting configurations with more than half the bits different from the original pattern will end up in the reversed state *-s*, which leads to a symmetrically divided configuration spaces into two basins of attraction.

Hebbian rules for multiple patterns

mathematical semplification of contributions to the final weight matrix

Multiple patterns $k = 1 \ldots R$ to be memorized

one by one

$$w_{ij}^{(k)} = w_{ij}^{(k-1)} + \frac{1}{N} \bar{x}_i^{(k)} \bar{x}_j^{(k)} \qquad w_{ij}^{(0)} = 0, \forall i, j$$

batch way

$$w_{ij} = \frac{1}{N} \sum_{r=1}^{R} \bar{x}_i^{(k)} \bar{x}_j^{(k)}$$

It is independent on the presentation order final result is the same, the computation is indepedent

Hebbian rules for multiple patterns

Storkey rule (Storkey et al. 1997)

no

$$w_{ij}^{(k)} = w_{ij}^{(k-1)} + \frac{1}{N}\overline{x}_i^{(k)}\overline{x}_j^{(k)} - \frac{1}{N}\overline{x}_i^{(k)}h_{ji}^{(k)} - \frac{1}{N}h_{ij}^{(k)}\overline{x}_j^{(k)}$$

$$h_{ij}^{(k)} = \sum_{r=1, r\neq i,j}^{N} w_{ir}^{(k-1)}\overline{x}_r^{(k)}$$

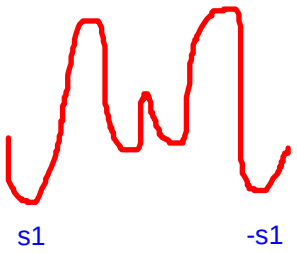
Local field at neuron *i*

$$w_{ij}^{(0)} = 0, \forall i, j$$

It is not independent on the presentation order

Learning rule features

- Locality
  - The update of a particular connection depends only on information available to the neurons on the either side of the connection
- Incremental
  - The learning process can modify an old network configuration to memorize a new pattern, without needing to refer to any of the previously learnt patterns (adaptive)
- Capacity
  - Measure of how many patterns can be stored in a network of a given size. Absolute capacity is the number of patterns that can be stored with correct recall
- Recall
  - Sinks of attraction: the set of states that are attracted to the stable state
  - Hopefully large, rounded, evenly distributed



Memory capacity with hebbian learning rule

[xxxxx? non chiede?] very short deomnstration about R and N to assure a certain accuracy.

Multiple patterns

The condition for R patterns $s_k = \left[\bar{x}_1^{(k)} \quad \bar{x}_2^{(k)} \quad \ldots \quad \bar{x}_N^{(k)}\right]$ which should be memorized is:

Hebbian learning: $w_{ij} = \frac{1}{N}\sum_k^R \bar{x}_i^{(k)}\bar{x}_j^{(k)}$

Stability of the pattern s_k

remember that $\bar{x}_i = \mathrm{sgn}\left(\sum_j w_{ij}\bar{x}_j\right) \quad \forall i$ for stability

skipped, described in the document

and the definition of the net input $h_i = \sum_j w_{ij}x_j$

the stability condition generalizes to $\mathrm{sgn}\left(h_i^{(v)}\right) = \bar{x}_i^{(v)} \quad \forall i, v$

Taking $w_{ij} = \frac{1}{N}\sum_{k}^{R} \bar{x}_i^{(k)} \bar{x}_j^{(k)}$

skipped

the net input $h_i^{(v)}$ to unit *i* in pattern *v* is: $h_i^{(v)} = \frac{1}{N}\sum_{j}^{N}\left(\sum_{k}^{R} \bar{x}_i^{(k)} \bar{x}_j^{(k)}\right)\bar{x}_j^{(v)}$

the recall is the pattern affected by a noise, each neuron can be affected by flipping. Embdeed in the intrinsic uncertainty in the recall operation

separating the sum on *k* into the special term $k = v$

meaning

! $h_i^{(v)} = \bar{x}_i^{(v)} + \frac{1}{N}\sum_{j}^{N}\left(\sum_{k\neq v}^{R} \bar{x}_i^{(k)} \bar{x}_j^{(k)}\right)\bar{x}_j^{(v)}$

↓

Crosstalk term *c*

If the second term were zero, one can conclude that pattern number *v* is stable according to

skipped

$\mathrm{sgn}\left(h_i^{(v)}\right) = \bar{x}_i^{(v)} \qquad \forall i$

This is still true if the second term is small enough:
if its magnitude is smaller than 1 it cannot change the sign of $\bar{x}_i^{(v)} + c$

Storage Capacity

$$h_i^{(v)} = \bar{x}_i^{(v)} + \frac{1}{N}\sum_j^N \left(\sum_{k \neq v}^R \bar{x}_i^{(k)} \bar{x}_j^{(k)} \right) \bar{x}_j^{(v)}$$

Let consider the quantity $C_i^{(v)} \equiv -\bar{x}_i^{(v)} * \frac{1}{N}\sum_j^N \left(\sum_{k \neq v}^R \bar{x}_i^{(k)} \bar{x}_j^{(k)} \right) \bar{x}_j^{(v)}$

If $C_i^{(v)}$ is negative the crosstalk term has the same sign as the desired $h_i^{(v)}$ and does no harm. But if its is positive and larger than 1, it changes the sign of $h_i^{(v)}$ and makes the bit *i* of pattern ν unstable

skipped

So $C_i^{(v)}$ depends on the patterns we try to store

For *random* patterns and with equal probability for the values +1 and –1 we can estimate the probability p_{error} that any chosen bit is unstable:

$$p_{error} = prob\left(C_i^{(v)} > 1\right)$$

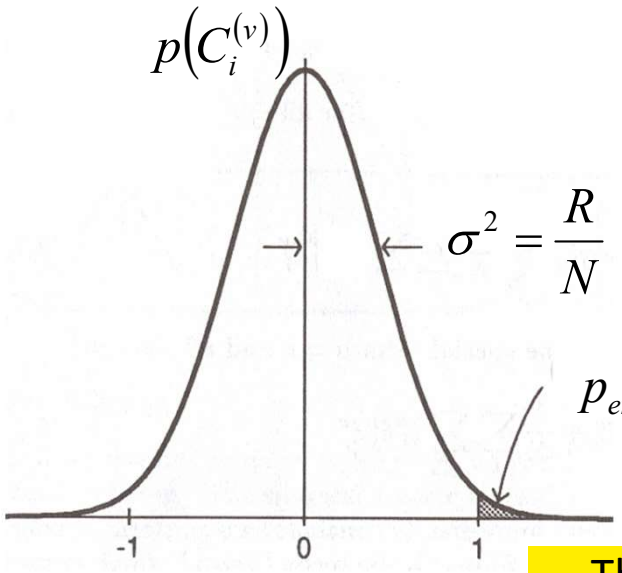
p_{error} increases as we increase the number *R* of patterns. Choosing a criterion for acceptable performance (e.g. p_{error} <0.01) we can try to determine the **storage capacity** of the network: the maximum number of patterns that can stored without unacceptable errors.

Storage Capacity $$C_i^{(v)} \equiv -\bar{x}_i^{(v)} * \frac{1}{N} \sum_j^N \left(\sum_{k \neq v}^R \bar{x}_i^{(k)} \bar{x}_j^{(k)} \right) \bar{x}_j^{(v)}$$

For *R* random patterns and *N* units $C_i^{(v)}$ is a binomial distribution with zero mean and variance $\sigma^2 = \frac{R}{N}$

For large values of *N**R, we can approximate this distribution with a Gaussian distribution

The uncertainty of the network can be formalized



sort of variance of the uncertainty. If I try to store many pattern this variance increases, I have more uncertainty.The opposite in the opposite.

$$p_{error} = prob(C_i^{(v)} > 1)$$

The shaded area is p_{error}, the probability of error per bit

My recall is the pattern but affected by noise. The nosie can be regarded as flipping, each neuron can be affected by flipping. This is an intrinsic uncertainty in the recall operation. [wont look at the demonstration]
There's a function taht describe the uncertainty of the network, it can be formalized with a probabilty density function that is a sort of semplification of the relation between the number of patterns I'd like to store and of neurons. This can be regarded as a sort of varince regarding the uncertainty. As the ration R/N increases the variances increases, meaninig that I have more uncertainty, and I decrease R I'm closing the distribution, meanining that I reduce the uncertainty of the recall operation.

can define the **stored capacity**... sort of probability of flipping. I have learned a number of pattern, then I'm feeding the net with a pattern, and I'd like to recall one of the stored pattern. The uncertainty is the probabilty that this feeding, generate a flipping of the bits, that is not actually a [xxstored?] but is a flipping produced by the noise, by the uncertainty of the network. This probabilty is related directly to the ratio. When you try to recover a pattern you have that uncertainty per bit. If I increase more and more pattern I lose this ability.
[riascolta, ~54.00]

Storage Capacity

$$C_i^{(v)} \equiv -\bar{x}_i^{(v)} * \frac{1}{N} \sum_j^N \left(\sum_{k \neq v}^R \bar{x}_i^{(k)} \bar{x}_j^{(k)} \right) \bar{x}_j^{(v)}$$

This calculation tells us only about the *initial* stability of the patterns. If we choose R <0.185N, it tells us that no more than 1% of the pattern bits will be unstable initially.

But if the system starts in a particular pattern h_i^{ν} and about 1% of the bits flip, what happens next? It may be that the first few flips will cause more bits to flip. In the worst case we will have an avalanche phenomenon. So, our estimates of $p_{\max}$ are really upper bounds. We may need smaller values of R to keep the final attractors close to the desired patterns.

If I have 100 bits, 1 bit can be flipped

Capacities

p_{error}	R/N
0.001	0.105
0.0036	0.138
0.01	0.185
0.05	0.37
0.1	0.61

You are loosing the pattern you wanted to store and instead you end up recalling false memories. You are obtaining spurious pattern.

In summary, the capacity $p_{\max}$ is proportional to N (but never higher than 0.138N) if we are willing to accept a small percentage of errors in each pattern. It is proportional to N / log(N) if we require that most of the patterns be recalled perfectly

Lyapunov function and Hebbian learning

$$E(s) = -\frac{1}{2}\sum_{i,j}^{N} w_{ij} x_i x_j$$

$$w_{ij} = \frac{1}{N}\sum_{k}^{R} \overline{x}_i^{(k)} \overline{x}_j^{(k)}$$

Multiple patterns

no

$$E(s) = -\frac{1}{2}\sum_{i,j}^{N}\left(\frac{1}{N}\sum_{k}^{R} \overline{x}_i^{(k)} \overline{x}_j^{(k)}\right) x_i x_j$$

$$\Delta E = -\left(x_r(t+1) - x_r(t)\right)\left(\sum_{j\neq r}^{N} w_{rj} x_j(t)\right)$$

Sequential dynamics and $w_{ij}=w_{ji}$

$$\Delta E = -\left(x_r(t+1) - x_r(t)\right)\left(\sum_{j\neq r}^{N}\left(\frac{1}{N}\sum_{k}^{R} \overline{x}_r^{(k)} \overline{x}_j^{(k)}\right) x_j(t)\right)$$

$\Delta E <= 0$

Working hypotheses

The Hopfield network output will tend to converge to the stored prototype patterns (among other possible equilibrium points)

Where the prototype patterns are orthogonal, every prototype pattern will be an equilibrium point of the network. However there exists spurious equilibrium states

no

The supervised Hebbian rule does not work well if there is significant correlation between the prototype patterns

When using the Hebbian rule, the number of stored patterns can be no more than 15% of the number of neurons

Spurious states

Reversed states

intrinsic states

Mixture states

stable states which are not equal to any single pattern but instead correspond a linear combinations of an *odd* (+1,-1) number of patterns

Uncorrelated states

for large R, local minima that are not correlated with any finite number of the original patterns

no

So the memory does not work perfectly; there are all these additional minima in addition to the ones we want.

These have in general smaller basin of attraction than the explicit memorized states

Working hypotheses for associative memories

- N number of units is big
- R number of patterns is small with respect to N
- No correlation among patterns

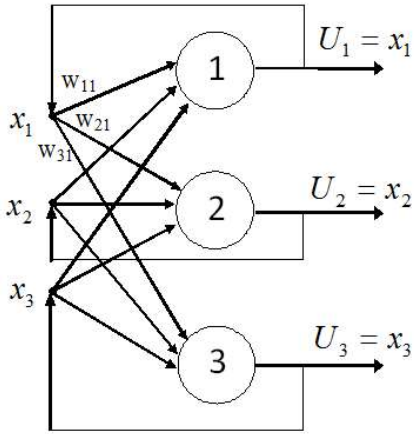
no

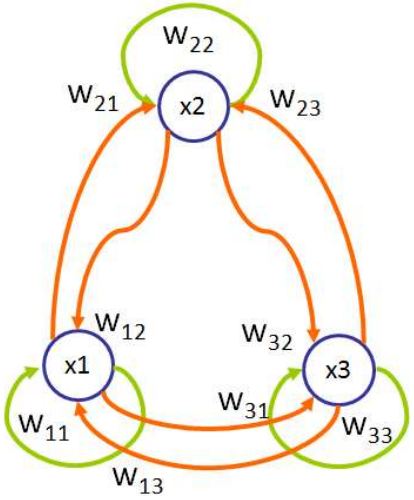
$\Rightarrow$ The R patterns are equilibrium states both in sequential and parallel dynamic

$\Rightarrow$ if s is stored then also -s is an equilibrium state

$\Rightarrow$ There exist equilibrium states other than that stored

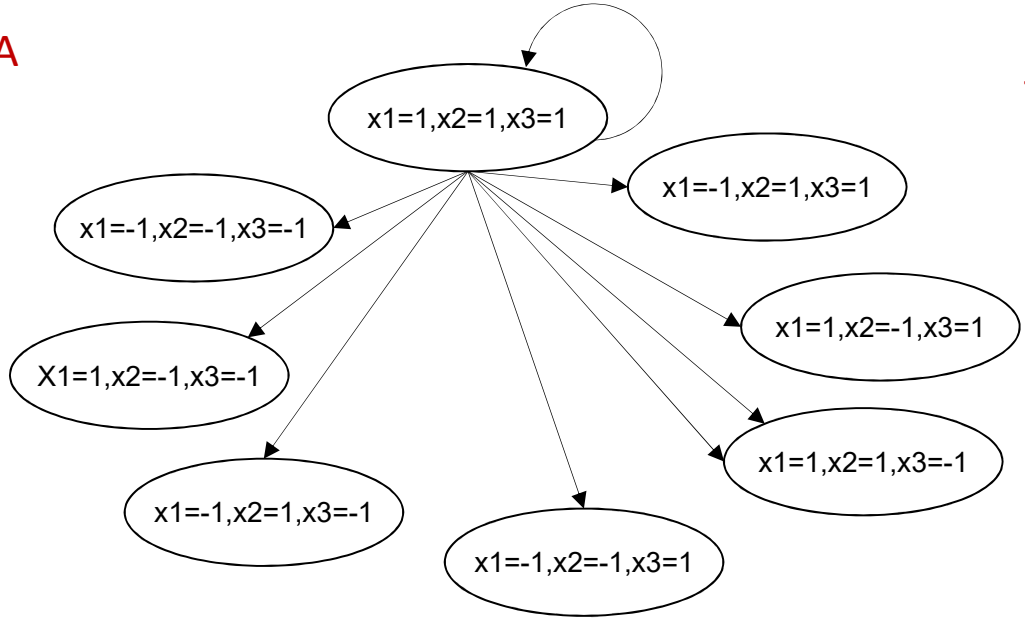
Local minima of the Liapunov function (spurious attractors)





no

AUTOMATA



Transition probs