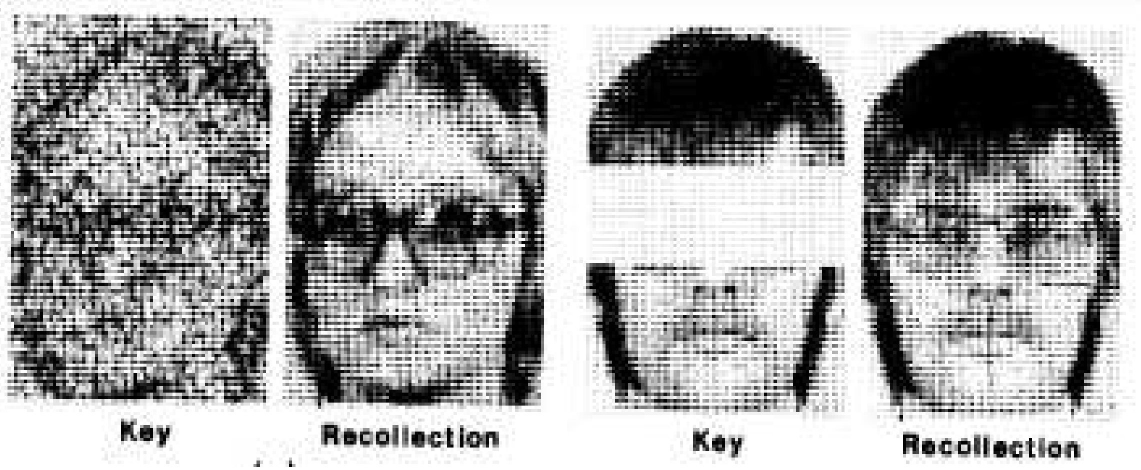
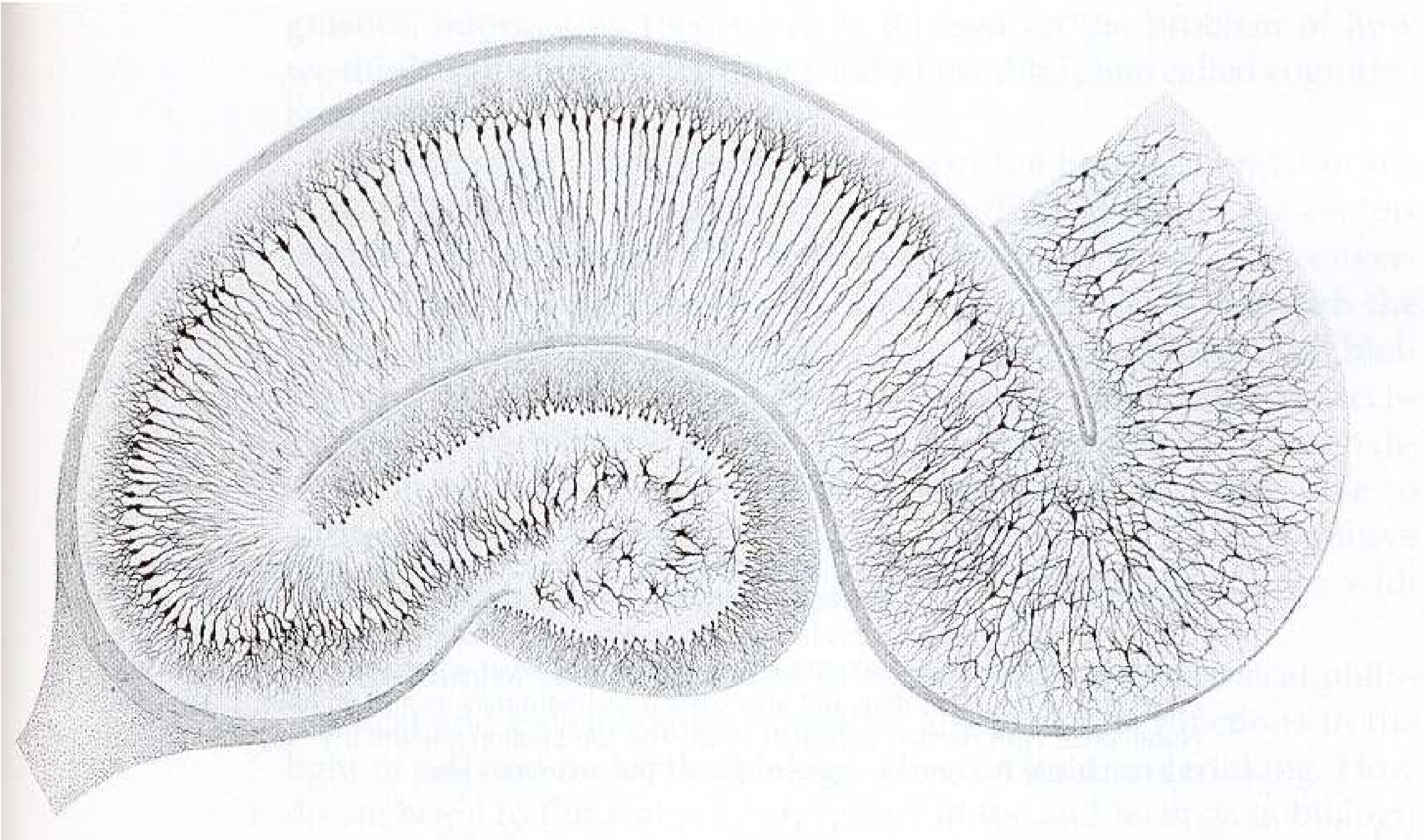
We recall memories based on cues. For example: Former president of the U.S.A, actor. These two keys are associated with a bunch of data, e.g., a name, a face, etc.

Associative Memory: After a stimulus S has been presented several times, the network should react in the same way on a noisy or an incomplete version of S.



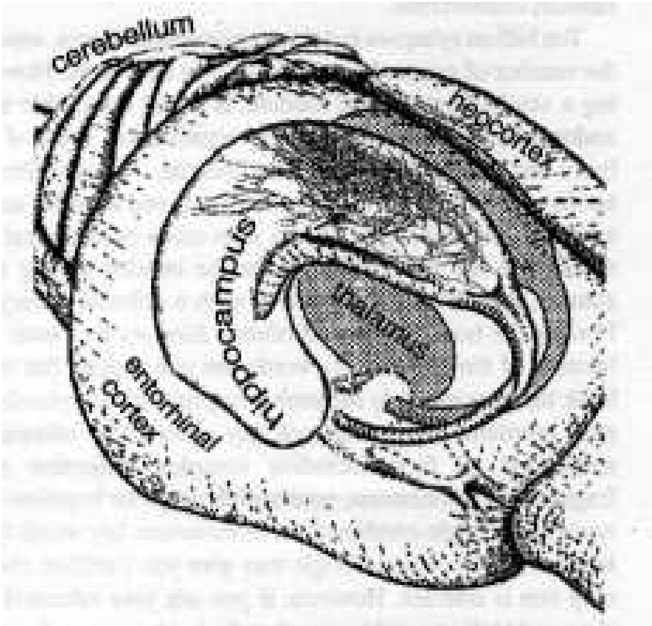
The hippocampus as a memory space



The hippocampus is known to be involved in the consolidation of memory.

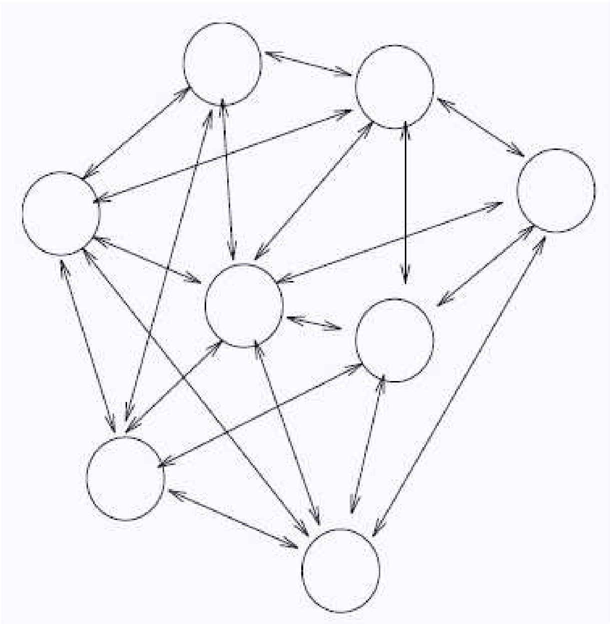
One of its specialities (as opposed to the neocortex) is its dense non-local connectivity.

The hippocampus as a memory space



Some number for the CA3 region of the rat hippocampus:

- ca. 200k pyramidal cells in each hemisphere.
- ca. 2-10G synapses in each hemisphere.
- The distribution of synapses is spatially widespread. Neurons do not prefer neighbors.
- The CA3 region is a random graph with 2-5% connection probability.



An associative memory is realized in the so called *Hopfield network*.

- Each neuron is connected to every other neuron (in both directions, the neurons are not directly connected onto themselves).
- We consider the case of threshold neurons with outputs $\{-1, +1\}$.
- Weights are symmetric, i.e., $w_{ij} = w_{ji}$ for all i, j.
- Each unit is an input and output unit as well.

Neurons: n threshold gates with output from $\{-1, 1\}$ and with zero bias:

$$o_i(\mathbf{q}) = \begin{cases} 1, & \text{if } \sum_{j=1}^{n} w_{ij} q_j \geq 0 \\ -1, & \text{if } \sum_{j=1}^{n} w_{ij} q_j < 0. \end{cases}$$

Initialization with some vector $\mathbf{s} = (s_1, \ldots, s_n) \in \{-1, 1\}^n$:

- For all i, neuron i outputs s_i. This is the initial state $\mathbf{q}(0)$ of the network.

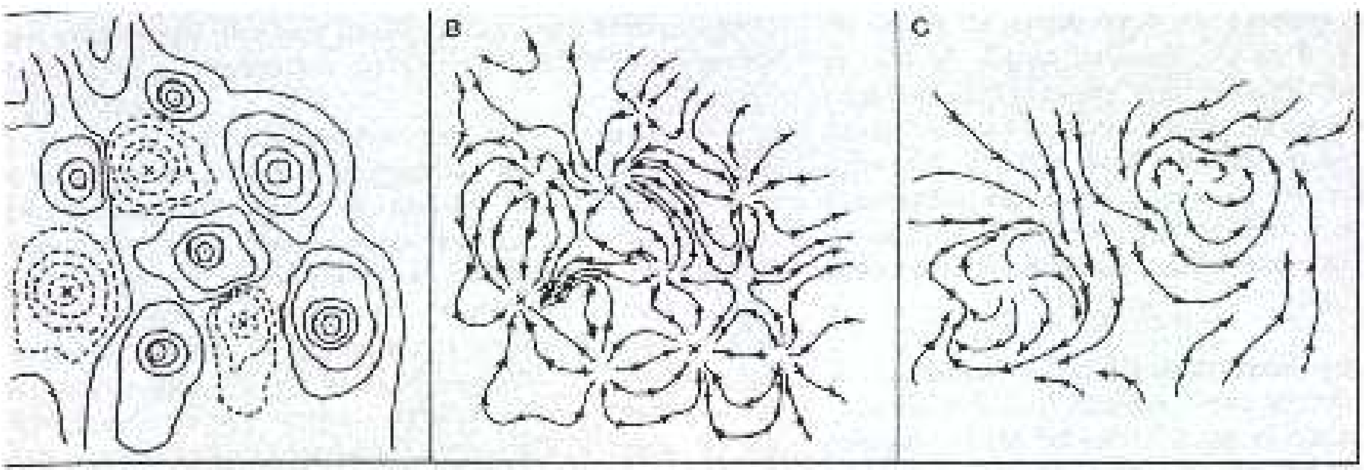
Computation (state-update):

- Let $\mathbf{q}(t) = (q_1(t), \ldots, q_n(t))$ denote the state of the network at time t.
- Randomly choose a neuron i and update the output of the neuron, i.e.,
- $q_i(t+1) = o_i(\mathbf{q}(t))$.
- Outputs of other neurons are not updated: $q_j(t+1) = q_j(t)$ for $j \neq i$.

Hopfield Networks - Operation

The space of all possible states is called the *state space*. Note that such a network has dynamics, i.e. the state of the network evolves over time due to the state update rules.

In general, a recurrent neural networks can have all types of weired dynamics, including state cycles and chaotic behavior.



However, the condition that weights are symmetric ensures that the behavior of the network is relatively simple, that is, the network will always converge to a stable state after finitely many update steps.

For a Hopfield-network consisting of n neurons $i = 1, \ldots, n$, we want the following *property*:

For

- a set of patterns $\mathcal{M} = \{\mathbf{p}(1), \ldots, \mathbf{p}(m)\}$ and
- initialization $\mathbf{q} \in \{-1, 1\}^n$,
- the network should - after a finite number of computational steps - converge to the *stored* pattern $\mathbf{p} \in \mathcal{M}$ which is closest to the initialization $\mathbf{q}$, i.e.
- $\|\mathbf{q} - \mathbf{p}\| \leq \|\mathbf{q} - \mathbf{p}'\|$ for all $\mathbf{p}' \in \mathcal{M}$.

To achieve this behavior, the weights w_{ij} from neuron j to neuron i is set to

$$w_{ij} = \frac{1}{n} \sum_{\mathbf{p} \in \mathcal{M}} p_i \cdot p_j$$

Note: This leads to the same weights as if one applies the *covariance rule* once for each example (with appropriate scaling).

The patterns $\mathbf{p} \in \mathcal{M}$ should be the *only fixed points* of the system:

$$\forall i : \quad p_i = o_i(\mathbf{p}) \quad \text{for all} \quad \mathbf{p} \in \mathcal{M}$$

However: If $\mathbf{p}$ is a fixed point, then $-\mathbf{p}$ is also a fixed point, and there are many more such *spurious fixed points*.

Furthermore, for a large number of patterns, one cannot guarantee that all patterns are stable fixed points (*crosstalk terms*).

The network works quite well for $|\mathcal{M}| \leq 0.138 \cdot n$ independent random patterns $\mathbf{p}$.

To be able to guarantee the stability of all patterns with high probability, it has to hold that $|\mathcal{M}| \leq \frac{n}{4 \cdot \log n}$.

From a technical point of view, a *Neirest Neighbor Algorithm* is much more efficient.

"Energy" E of a state $\mathbf{q}$:

$$E(\mathbf{q}) = -\sum_{i,j=1}^{n} w_{i,j} \cdot q_i \cdot q_j$$

Hopfield could show the following: If a neuron switches its state, then

- the energy drops,
- or it stays constant. In this case the neuron switched from -1 to 1.

Therefore, each update-sequence *converges* to a fixed point of the network after finitely many steps.

Hence, the network approximates the solution to an optimization problem. The goal of the optimization problem is to minimize the energy

$$E(\mathbf{q}) = -\sum_{i,j=1}^{n} w_{i,j} \cdot q_i \cdot q_j .$$

Lowest value of E, if:

$\mathbf{q}_i = \mathbf{q}_j$ for all i, j with $w_{i,j} > 0$

$\mathbf{q}_i \neq \mathbf{q}_j$ for all i, j with $w_{i,j} < 0$

Intuitively,

$w_{i,j} \gg 0$ emphasizes patterns with $q_i = q_j$, and

$w_{i,j} \ll 0$ emphasizes patterns with $q_i \neq q_j$.

Pros and Cons

Pros:

- very simple model
- nice mathematical analysis possible (also for capacity)
- the hippocampus seems to have a similar connection architecture
- plausible learning (Hebbian)

Cons:

- the dynamics of the system is constrained to fixed points
- no storage of time series
- the dynamics in the hippocampus are more complex
- low capacity