Lecture 4: Associative memory

Jochen Braun

Otto-von-Guericke-Universität Magdeburg,
Cognitive Biology Group

Theoretical Neuroscience II, SS 2020

Lecture 4: Associative memory

A recurrent network may store several 'memory' patterns in its connectivity (synaptic weights). **Memory patterns** *are steady-state patterns of activity. Given any input pattern, the network converges to the most similar steady-state ("long-term memory"). Additionally, memory patterns may remain active after input has ceased ("working memory").*

Today we construct an associative memory network. To encode memory patterns, we must ensure that excitation spreads preferentially within the pattern (rather than 'leaking' to other patterns). To this end, we **choose connection weights** *such that arbitrary memory patterns become eigenvectors of connectivity. Additionally, we must ensure that memory patterns are steady-states. To this end, we* **fine-tune the activation function** *of units. Thus, functionality depends on properties of both network (connectivity) and units (activation function).*

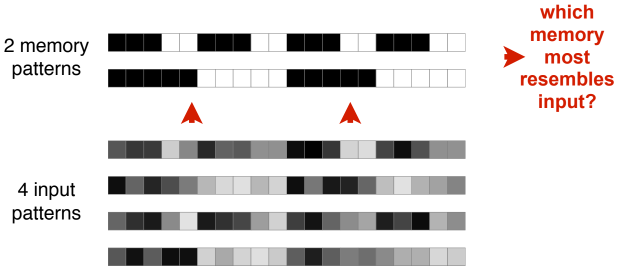
We visualize associative memory behaviour in terms of **energy landscapes**, *with local minima representing memory patterns.*

Organization of lecture

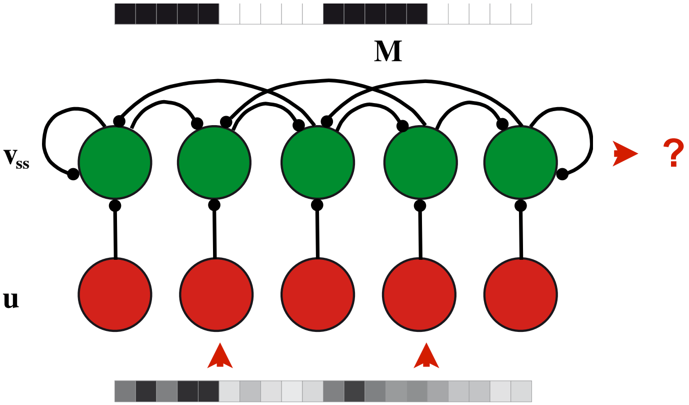
- 1. **Associative memory**
- 2. **Theoretical justification (optional)**
- 3. **Associative memory network**
- 4. **Inscribing memory patterns into connectivity**
- 5. **Steady-state condition**
- 6. **Simulated dynamics**

1. Associative memory

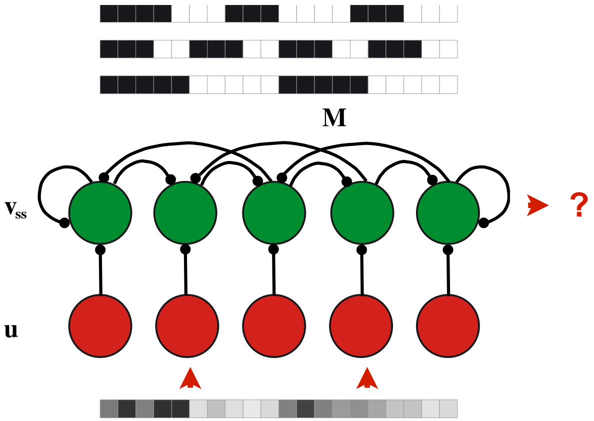
A recurrent network may function as an "associative" or "content-addressable" memory. Such a network contains several stored patterns ('memory patterns'). When challenged with an arbitrary input pattern, the network responds with the memory pattern that is most similar to the input.



We have already seen that recurrent networks can selectively amplify a particular pattern stored in its connectivity. We also know that the amplitude of the response to an input will depend on how well the input matches the stored pattern.



We now consider networks which store a set of arbitrary patterns, which we call memory patterns. When challenged with an input, the network will selectively amplify the memory pattern that most resembles the input.



Memory and persistent activity

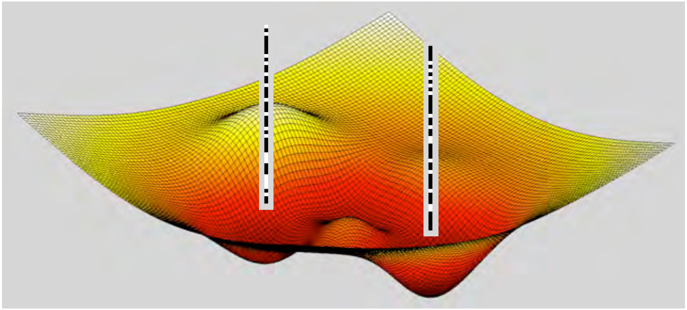
In essence, when challenged with an incomplete or corrupted pattern, the network restores the full pattern. This is thought to model mammalian memory systems that are characterized by recurrent connections, such as prefrontal cortex or area CA3 of hippocampus.

Note that memory patterns are stored in terms of connectivity (synaptic weights), not in terms of activity. However, memory patterns are, by definition, stable patterns of network activity.

Depending on details, memory patterns may remain active even after input has ceased ("persistent activity"). The persistent activity is thought to retain a "working memory" of the most recent input.

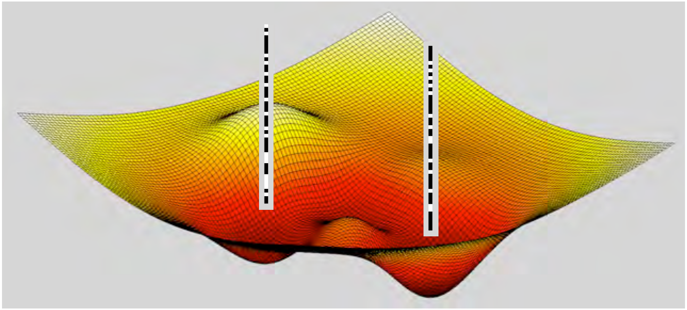
Energy landscape

Conceptually, we can imagine the possible activity patterns arranged on a plane and assign each of them an ‘potential energy’. The gradient of resulting ‘energy surface’ indicates the direction in which the activity will develop, when the network is allowed to relax. Local minima are fixed points and represent stable patterns of activity.



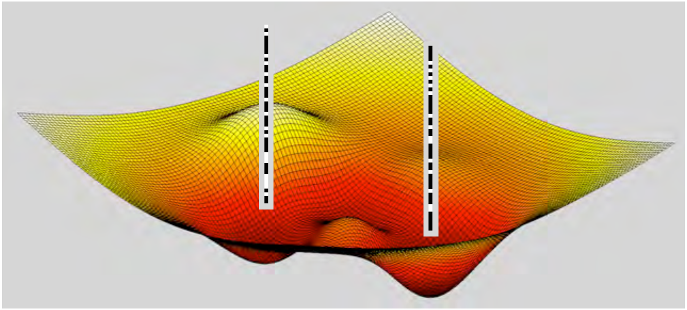
Relaxation dynamics

Finding the memory pattern that most closely matches an input ("pattern matching") can be visualized as follows: (i) initialize the network to a particular pattern (point on surface), (ii) let it 'relax' by following the gradient to the nearest fixed point, (iii) read out the memory pattern corresponding to the fixed point.



Basins of attraction

Each fixed point is surrounded by an area called its ‘basin of attraction’. Within this area, network activity will surely relax towards the fixed point. Outside this area, the probability less than one (and often close to zero).



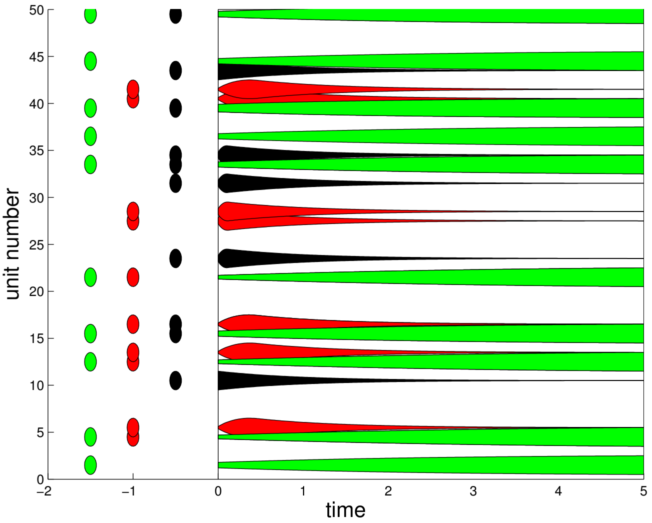
Goal of Lecture

The goal of this lecture is to construct an associative memory on the basis of a recurrent network.

Specifically, we wish to store several (possibly overlapping) memory patterns in the connectivity matrix.

We further wish the network to settle, after being challenged with an arbitrary input, to the memory pattern that most resembles the input.

We will discover that associative memory is a fragile feature, requiring delicate calibration of connectivity and responsiveness.



Here, three patterns are stored in memory (green, red, black). When challenged with a mixture of patterns, the network settles quickly on the green pattern.

2. Theoretical justification (optional)

Recall from Lecture 1 alternative formulations of firing rate model.

Synaptic current approaches steady-state more quickly:

$$\tau_r \frac{dv(t)}{dt} = - \underbrace{v(t)}_{output\ activity} + F \left[\underbrace{I_s(t)}_{synaptic\ current} \right], \qquad I_s \simeq w \underbrace{u(t)}_{input\ activity}$$

Activity approaches steady-state more quickly:

$$\tau_s \frac{dI_s(t)}{dt} = - \underbrace{I_s(t)}_{synaptic\ current} + w \underbrace{u(t)}_{input\ activity}, \qquad \underbrace{v(t)}_{output\ activity} \simeq F\left[I_s\right]$$

To justify ‘energy landscape’ metaphor, we choose the second formulation, in terms of synaptic input current $\boldsymbol{I}_s$, rather than output firing $\boldsymbol{v}$.

Recurrent network with output activity $\boldsymbol{v}$, synaptic input current $\boldsymbol{I}_s$, synaptic connectivity matrix $\boldsymbol{M}$, and feedforward input $\boldsymbol{h}$

$$\tau_s \frac{d\boldsymbol{I}_s}{dt} = -\boldsymbol{I}_s + \boldsymbol{h} + \boldsymbol{M} \cdot F(\boldsymbol{I}_s), \qquad \boldsymbol{v} = F(\boldsymbol{I}_s)$$

Dynamic state variable is $\boldsymbol{I}_s(t)$. Output activity follows as $\boldsymbol{v} = F(\boldsymbol{I}_s)$.

Lyapunov function

Assuming a symmetric $\boldsymbol{M}$ with zero diagonal, we formulate an expression $L_a(I_a, h_a)$ for each component h_a, I_a:

$$L_a(I_a, h_a) = \int_0^{I_a} i_a\, F'(i_a)\, di_a - h_a\, F(I_a) - \frac{1}{2} \sum_{a'} F(I_a)\, M_{aa'}\, F(I_{a'})$$

where $F'(i) = {}^{dF(i)}/_{di}$. Summing these expressions over all components, we obtain the *Lyapunov function*

$$L(\boldsymbol{I}_s) = \sum_a L_a(I_a, h_a)$$

As long as activity $\boldsymbol{I}_s$ changes with time, the Lyapunov function decreases monotonically:

$$\frac{d\boldsymbol{I}_s}{dt} \neq 0 \qquad \Rightarrow \qquad \frac{d\boldsymbol{L}}{dt} < 0$$

Time derivative

The time-derivative of expression L_a:

$$\frac{dL_a}{dt} = I_a\, F'(I_a)\, \tfrac{dI_a}{dt} - h_a\, F'(I_a)\, \tfrac{dI_a}{dt} -$$
$$-\tfrac{1}{2}\tfrac{dI_a}{dt} \textstyle\sum_{a'}\; F'(I_a)\, M_{aa'}\, F(I_{a'}) - \tfrac{1}{2} \textstyle\sum_{a'}\; F(I_a)\, M_{aa'}\, F'(I_{a'})\, \tfrac{dI_{a'}}{dt}$$

$$\frac{dL}{dt} = \textstyle\sum_a \tfrac{dL_a}{dt} =$$
$$= \textstyle\sum_a (I_a - h_a) F'(I_a) \tfrac{dI_a}{dt} - \textstyle\sum_{aa'}\; F'(I_a)\, M_{aa'}\, F(I_{a'})\, \tfrac{dI_a}{dt} =$$
$$= - \textstyle\sum_a F'(I_a) \tfrac{dI_a}{dt} \left[-I_a + h_a + \textstyle\sum_{a'}\; M_{aa'}\, F(I_{a'}) \right] =$$
$$= -\tfrac{1}{\tau_s} \textstyle\sum_a F'(I_a) \left(\tfrac{dI_a}{dt}\right)^2 \quad \overset{!}{<} \quad 0$$

where we have used

$$\tau_s \frac{dI_a}{dt} = -I_a + h_a + \sum_{a'} M_{aa'}\, F(I_{a'})$$

$$\frac{dL(\boldsymbol{I})}{dt} = -\frac{1}{\tau_s} \sum_a F'(I_a) \left(\frac{dI_a}{dt} \right)^2$$

Since $F'(I_a)$ is monotonically increasing, the derivative is > 0 unless and until $\frac{dI_a}{dt} = 0$.

If $L(\boldsymbol{I}_s)$ has a lower bound, then it cannot decrease indefinitely, so that $\boldsymbol{I}_s$ must converge to a fixed point.

Since

$$\boldsymbol{I}_s = \boldsymbol{h} + \boldsymbol{M} \cdot \boldsymbol{v}$$

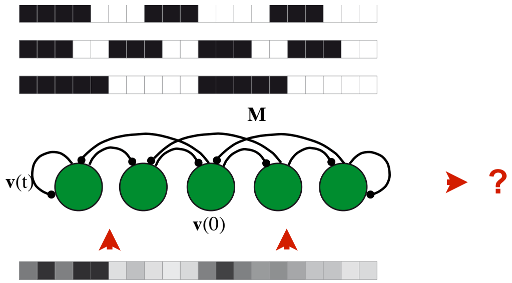
this means that $\boldsymbol{v}$ converges to a fixed point, too.

3. Associative memory network

We consider the following example network

$$\tau \frac{d\boldsymbol{v}}{dt} = -\boldsymbol{v} + F\left(\boldsymbol{M} \cdot \boldsymbol{v}\right)$$

with time-varying activity $\boldsymbol{v}(t)$, connectivity matrix $\boldsymbol{M}$, and relaxation time τ. Input sets initial activity $\boldsymbol{v}(0)$ (no explicit input).



Sigmoidal non-linearity

We use a sigmoidal non-linearity $F(v)$

$$F(v) = r_{max} \left[\tanh \left(\frac{v - \vartheta}{r_{max}} \right) \right]_{+}$$

with the following parameter values

$r_{max} = 150 Hz$	maximal activity
$\vartheta = -20 Hz$	background activity
$N = 50$	number of units

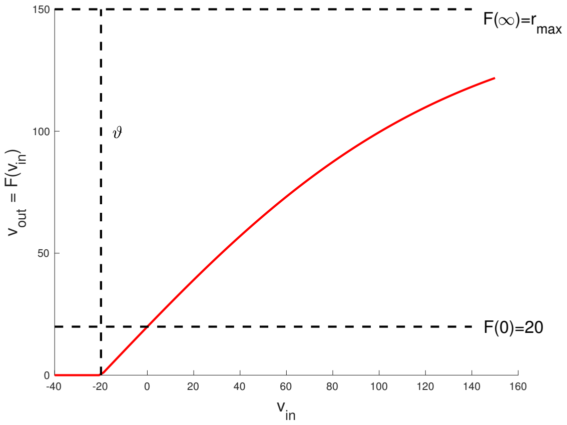
We will use our network to store $N_{mem} << N$ activity patterns $\boldsymbol{v}_m$

$$N_{mem} = 4 \qquad \text{stored patterns}$$

Note: Negative ϑ adds to activity v (acting like background activity). Positive ϑ would subtract from activity v (acting like threshold).

ctd

$$F(v) = r_{max} \left[\tanh \left(\frac{v - \vartheta}{r_{max}} \right) \right]_{+}$$



Memorized patterns

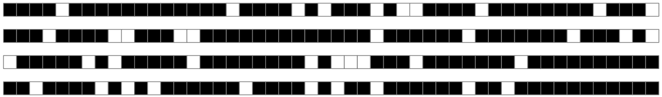
For simplicity, we consider patterns with αN components of value 1 and $(1-\alpha)N$ components of value 0. An example for such a pattern is

$$\boldsymbol{v}_1 = \left[\underbrace{1, \quad \ldots, \quad 1,}_{\alpha N} \underbrace{0, \quad \ldots, \quad 0}_{(1-\alpha)N} \right]$$

Sparseness α (fraction of active units) will turn out to be critical!

$\alpha = 0.2$	sparseness
$\alpha N = 10$	'on' units
$(1-\alpha)N = 40$	'off' units

Here is an illustration of $N_{mem} = 4$ patterns with $\alpha N = 10$ elements 1 (white squares) and $(1-\alpha)N = 40$ elements 0 (black squares).



Note that the sum of all components is always

$$trace(\boldsymbol{v}_m) = \sum_a v_m^a = \alpha N$$

For convenience of notation, we will pretend that elements 1 and 0 are sorted.

Uniform patterns

In addition, we will use uniform patterns such as

$$\boldsymbol{n} \equiv \left[\underbrace{1, \quad \ldots, \quad 1}_{N} \right]$$

and

$$\boldsymbol{v}_0 \equiv \left[\underbrace{\alpha, \quad \ldots, \quad \alpha}_{N} \right] = \alpha \boldsymbol{N}$$

The components of $\boldsymbol{v}_0$ sum to the same value as the memory patterns

$$trace(\boldsymbol{v}_0) = \sum_a v_0^a = \alpha N$$

Dot products

With these definitions, we have the following dot products:

$$\boldsymbol{n} \cdot \boldsymbol{v}_m = \alpha\, N$$

$$\boldsymbol{v}_m \cdot \boldsymbol{v}_m \quad = \alpha\, N \qquad \forall m$$

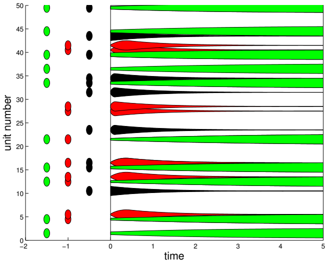
$$\boldsymbol{v}_n \cdot \boldsymbol{v}_m \quad \approx \alpha^2\, N \qquad \text{for} \quad n \neq m$$

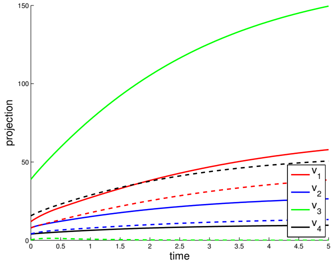
Note that α^2 is the probability that corresponding elements of $\boldsymbol{v}_n$ and $\boldsymbol{v}_m$ are both 1.

Stability condition

If initial activity $\boldsymbol{v}_0$ is proportional to $\boldsymbol{v}_m$, we want activity to develop towards $\boldsymbol{v}_m$. In other words, we want memory patterns $\boldsymbol{v}_m$ to be stable steady-states.

$$\boldsymbol{v}_0 \propto \boldsymbol{v}_m \quad \Rightarrow \quad \boldsymbol{v}(t) \rightarrow \boldsymbol{v}_m \quad \Rightarrow \quad 0 = -\boldsymbol{v}_m + F\left(\boldsymbol{M} \cdot \boldsymbol{v}_m\right)$$





Left: emergence of a stable pattern (green). Right: projection of network activity on different pattern vectors (blue, red, green, etc).

Summary so far

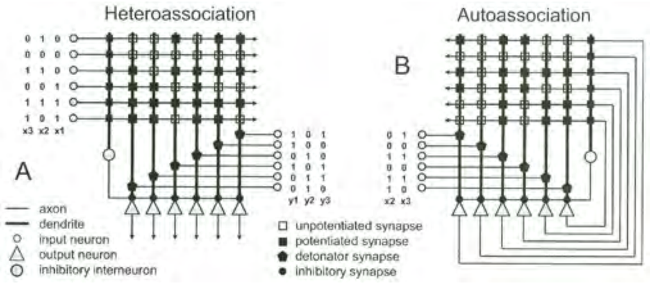
We have now defined our objective: a recurrent network with certain, predefined memory patterns as stable steady-states.

Next we need to construct a suitable connectivity matrix.

Although the principle is simple, the details are fiddly ...

4. Inscribing memory patterns into connectivity

Our basic intuition is that a memory pattern will be stable if the excitation propagates within active nodes, but does not 'leak out' to inactive nodes. To this end, the connections between active nodes should be strong, but connections between active and inactive nodes weak.



Conceptual cartoon of associative memory (from hippocampus lecture).

Sum of covariance matrices

To find a suitable connection matrix $\boldsymbol{M}$, we first consider the sum $\boldsymbol{K}$ of the covariance matrices of the to-be-stored patterns $\boldsymbol{v}_m$:

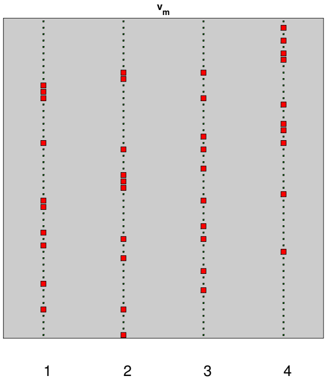
$$\boldsymbol{K} = \frac{1}{\alpha(1-\alpha)N} \sum_m \boldsymbol{C}^m$$

Covariance matrix (= product of deviations from mean)

$$\boldsymbol{C}^m \quad = (\boldsymbol{v}_m - \alpha\,\boldsymbol{n})\,(\boldsymbol{v}_m - \alpha\,\boldsymbol{n})$$

By inscribing the *covariance* between active members of a memory pattern into the connectivity, we ensure that excitation spreads preferentially *within* the pattern (rather than 'leaking' to other patterns). Of course, overlap is a problem, because any active member that also belongs to another pattern constitutes a 'leak'.

Illustration of $N_{mem} = 4$ patterns (left) and of first covariance matrix (right). Red squares indicate positive, green squares negative values. Size indicates absolute value.



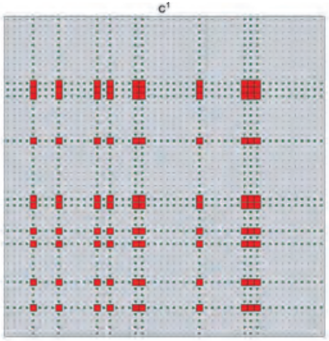
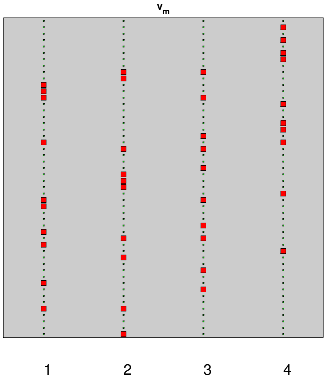


Illustration of $N_{mem} = 4$ patterns (left) and of second covariance matrix (right).



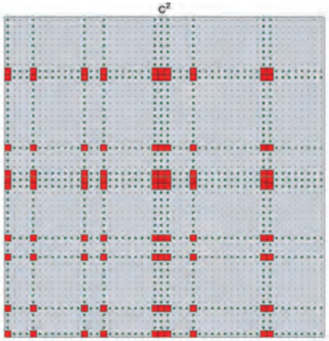
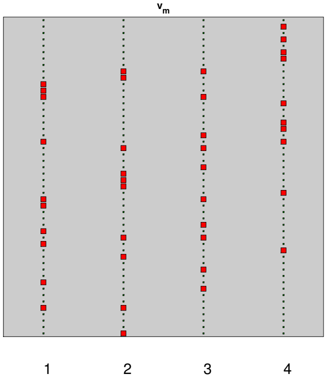


Illustration of $N_{mem} = 4$ patterns (left) and of third covariance matrix (right).



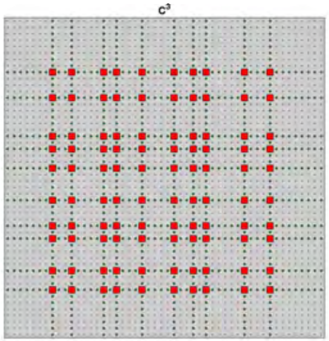
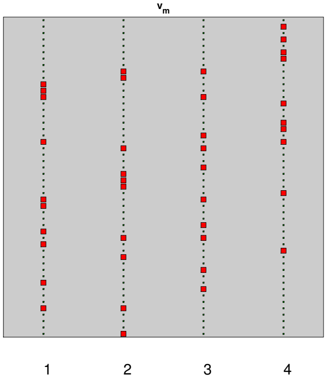
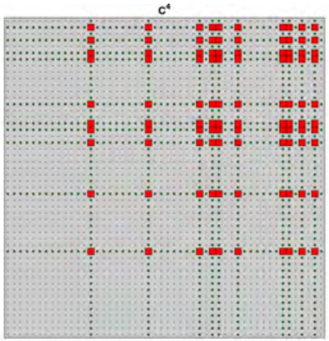
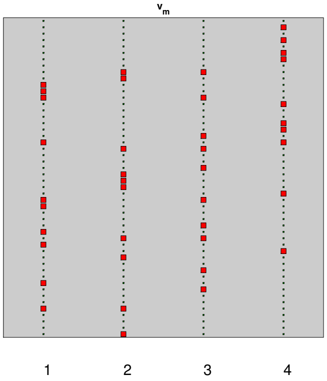


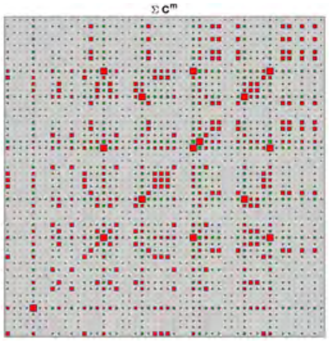
Illustration of $N_{mem} = 4$ patterns (left) and of fourth covariance matrix (right).





Sum of all four covariance matrices (right). Red squares indicate positive, green squares negative values. Size indicates absolute value.





Propagation of activity

To analyze propagation of memory pattern $\boldsymbol{v}_n$ through matrix $\boldsymbol{K}$, we consider the dot product between $\boldsymbol{K}$ and $\boldsymbol{v}_n$

$$\boldsymbol{K} \cdot \boldsymbol{v}_n = \frac{1}{\alpha(1-\alpha)N} \sum_m \boldsymbol{C}^m \cdot \boldsymbol{v}_n$$

which describes the outcome.

As it turns out, the outcome approximates (see proof below) a linear combination of memory pattern $\boldsymbol{v}_n$ and uniform pattern $\boldsymbol{n}$:

$$\boldsymbol{K} \cdot \boldsymbol{v}_n \approx \boldsymbol{v}_n - \alpha\, \boldsymbol{n}$$

While this is not exactly what we wanted, we can work with this!

Proof

$$
\begin{aligned}
\boldsymbol{C}_m \cdot \boldsymbol{v}_n &= (\boldsymbol{v}_m - \alpha\,\boldsymbol{n})\left[(\boldsymbol{v}_m - \alpha\,\boldsymbol{n}) \cdot \boldsymbol{v}_n\right] = \\
&= (\boldsymbol{v}_m - \alpha\,\boldsymbol{n})\left[\boldsymbol{v}_m \cdot \boldsymbol{v}_n - \alpha^2 N\right] = \\
&= (\boldsymbol{v}_m - \alpha\,\boldsymbol{n})\begin{cases} \left[\alpha N - \alpha^2 N\right] & \text{for } n = m \\ \left[\alpha^2 N \pm \sqrt{\alpha^2(1-\alpha^2)N} - \alpha^2 N\right] & \text{for } n \neq m \end{cases} \\
&= \alpha(1-\alpha)N\,(\boldsymbol{v}_m - \alpha\,\boldsymbol{n})\begin{cases} 1 & \text{for } n = m \\ \pm\sqrt{\frac{1-\alpha^2}{(1-\alpha)^2\,N}} & \text{for } n \neq m \end{cases}
\end{aligned}
$$

$$
\boldsymbol{K} \cdot \boldsymbol{v}_n = \frac{1}{\alpha(1-\alpha)N} \sum_m \boldsymbol{C}_m \cdot \boldsymbol{v}_n \approx \boldsymbol{v}_n - \alpha\,\boldsymbol{n}
$$

First pattern

To repeat this for one specific example, we consider the covariance matrix of pattern $\boldsymbol{v}_1$:

$$\boldsymbol{v}_1 - \alpha\, \boldsymbol{n} = \begin{bmatrix} 1 \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} \alpha 1 \\ \vdots \\ \alpha 1 \end{bmatrix} = \begin{bmatrix} 1-\alpha \\ \vdots \\ -\alpha \end{bmatrix}$$

$$\boldsymbol{C}_1 \quad = \begin{bmatrix} (1-\alpha) \\ \vdots \\ -\alpha \end{bmatrix} [(1-\alpha) \;\ldots\; -\alpha] = \begin{bmatrix} (1-\alpha)^2 & \ldots & -(1-\alpha)\alpha \\ \vdots & \ddots & \vdots \\ \underbrace{-(1-\alpha)\alpha}_{\alpha N} & \ldots & \underbrace{\alpha^2}_{(1-\alpha)N} \end{bmatrix}$$

Compute the dot product between pattern $\boldsymbol{v}_1$ and its covariance matrix:

$$\boldsymbol{C}_1 \cdot \boldsymbol{v}_1 = \left[\begin{array}{ccc} (1-\alpha)^2 & \dots & -(1-\alpha)\alpha \\ \vdots & \ddots & \vdots \\ \underbrace{-(1-\alpha)\alpha}_{\alpha N} & \dots & \underbrace{\alpha^2}_{(1-\alpha)N} \end{array} \right] \cdot \left[\begin{array}{c} 1 \\ \vdots \\ 0 \end{array} \right] =$$

$$= \left[\begin{array}{c} \alpha N(1-\alpha)^2 \\ \vdots \\ -\alpha^2 N(1-\alpha)^2 \end{array} \right] = \alpha N(1-\alpha) \left[\begin{array}{c} 1-\alpha \\ \vdots \\ -\alpha \end{array} \right] =$$

$$= \alpha N(1-\alpha)\,(\boldsymbol{v}_1 - \alpha\,\boldsymbol{n})$$

As expected, the result is a linear combination of $\boldsymbol{v}_1$ and uniform pattern $\boldsymbol{n}$.

Another pattern

To consider the issue of 'leakage', we compare the covariance of another pattern $\boldsymbol{v}_2$, also comprising αN (randomly chosen) elements 1 and $(1-\alpha)N$ elements 0:

$$\boldsymbol{v}_2 - \alpha\, \boldsymbol{n} = \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix} - \begin{bmatrix} \alpha \\ \vdots \\ \alpha \end{bmatrix} = \begin{bmatrix} -\alpha \\ \vdots \\ 1-\alpha \end{bmatrix}$$

$$\boldsymbol{C}_2 = \begin{bmatrix} -\alpha \\ \vdots \\ 1-\alpha \end{bmatrix} [-\alpha \ \ldots \ 1-\alpha] = \begin{bmatrix} \alpha^2 & \ldots & -(1-\alpha)\alpha \\ \vdots & \ddots & \vdots \\ \underbrace{-(1-\alpha)\alpha}_{(1-\alpha)N} & \ldots & \underbrace{(1-\alpha)^2}_{\alpha N} \end{bmatrix}$$

To estimate 'leakage', we compute the dot product between the second covariance matrix $\boldsymbol{C}_2$ and the first pattern $\boldsymbol{v}_1$.

$$\boldsymbol{C}_2 \cdot \boldsymbol{v}_1 = \left[\begin{array}{ccc} \alpha^2 & \dots & -(1-\alpha)\alpha \\ \vdots & \ddots & \vdots \\ \underbrace{-(1-\alpha)\alpha}_{(1-\alpha)N} & \dots & \underbrace{(1-\alpha)^2}_{\alpha N} \end{array} \right] \cdot \left[\begin{array}{c} 1 \\ \vdots \\ 0 \end{array} \right] =$$

Of each sort of column, a fraction α is multiplied with 1 and contributes to the result. The others are multiplied with 0 and do not contribute:

$$\boldsymbol{C}_2 \cdot \boldsymbol{v}_1 = \left[\begin{array}{c} \alpha \left\{ (1-\alpha)N\alpha^2 + \alpha N[-(1-\alpha)\alpha] \right\} \\ \vdots \\ \alpha \left\{ (1-\alpha)N[-(1-\alpha)\alpha] + \alpha N(1-\alpha)^2 \right\} \end{array} \right] = \left[\begin{array}{c} \sim 0 \\ \vdots \\ \sim 0 \end{array} \right]$$

The average result is a vector with elements close to 0.

Sum of covariance matrices

Taking these findings together, we see that the dot product between $\boldsymbol{K}$ and memory pattern $\boldsymbol{v}_1$ approximates a linear combination of $\boldsymbol{v}_1$ and $\boldsymbol{n}$:

$$\boldsymbol{K} = \frac{1}{\alpha(1-\alpha)N} \sum_m \boldsymbol{C}_m$$

$$\begin{aligned}
\boldsymbol{K} \cdot \boldsymbol{v}_1 &= \frac{1}{\alpha(1-\alpha)N} \left[\boldsymbol{C}_1 \cdot \boldsymbol{v}_1 + \boldsymbol{C}_2 \cdot \boldsymbol{v}_1 + \ldots + \boldsymbol{C}_m \cdot \boldsymbol{v}_1\right] = \\
&\approx \frac{1}{\alpha(1-\alpha)N} \left[\alpha(1-\alpha)N\,(\boldsymbol{v}_1 - \alpha\,\boldsymbol{n}) + 0 + \ldots + 0\right] \\
&= \boldsymbol{v}_1 - \alpha\,\boldsymbol{n}
\end{aligned}$$

Covariance is balanced

Note that covariance matrices are balanced in that a uniform input $\boldsymbol{n}$ produces no net excitation or inhibition:

$$\boldsymbol{C}_m \cdot \boldsymbol{n} = \begin{bmatrix} \alpha^2 & \dots & -(1-\alpha)\alpha \\ \vdots & \ddots & \vdots \\ \underbrace{-(1-\alpha)\alpha}_{(1-\alpha)N} & \dots & \underbrace{(1-\alpha)^2}_{\alpha N} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} =$$

$$= \alpha(1-\alpha)N \begin{bmatrix} \alpha - \alpha \\ \vdots \\ -(1-\alpha) + 1 - \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

The same is true for random input (random subset of elements active, others inactive).

Connectivity matrix

The final connectivity matrix is obtained by scaling $\boldsymbol{K}$ with a factor κ and by nulling the $\alpha\boldsymbol{n}$ component with uniform inhibition between all units:

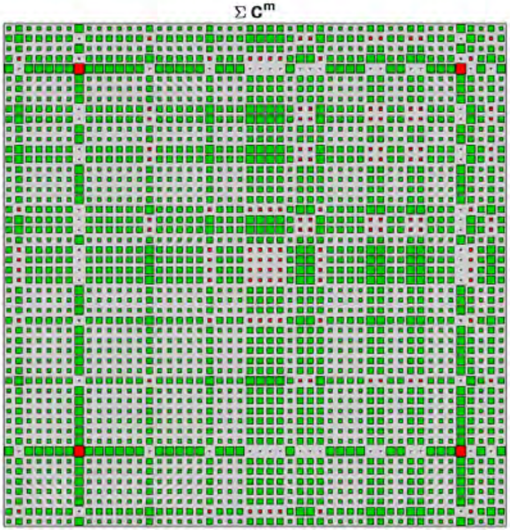
$$\boldsymbol{M} = \kappa\,\boldsymbol{K} - \frac{1}{\alpha N}\,\boldsymbol{n}\boldsymbol{n}$$

The parameter κ determines the relative strength of pattern-specific excitation.

The inhibition strength scales to unity when αN input units are active.

Note that the "vector outer product" $\boldsymbol{n}\,\boldsymbol{n}$ produces a matrix with all elements equal to unity.

Here is the final connectivity matrix for the $N_{mem} = 4$ patterns above. Due to the pervasive inhibition, it contains mostly negative values (green squares).



Summary so far

We have inscribed the desired memory patterns into connectivity: propagating a memory pattern (activity!) through the connection matrix, reproducing this pattern (as synaptic currents!).

To compensate for indiscriminate spreading of excitation, we have added some indiscriminate inhibition.

We now need to convince ourselves that the memory patterns (of activity) are indeed stable steady-states of the network.

Here, the details of the sigmoidal activation function will prove important.

5. Steady-state condition

A steady-state pattern of input activity $\boldsymbol{v}_{ss}$ must lead to a proportional pattern $\boldsymbol{M} \cdot \boldsymbol{v}_{ss}$ of synaptic currents and to identical pattern of output activity $\boldsymbol{v}_{ss}$:

$$0 \overset{!}{=} \tau \frac{d\boldsymbol{v}}{dt} = -\boldsymbol{v}_{ss} + \underbrace{F\left(\boldsymbol{M} \cdot \boldsymbol{v}_{ss}\right)}_{\textit{output activity}}$$

We want steady-state patterns to be scaled versions of memory patterns:

$$\boldsymbol{v}_{ss} = c\, \boldsymbol{v}_m$$

The steady-state condition thus becomes

$$0 \overset{!}{=} -c\, \boldsymbol{v}_m + F\left(c\, \boldsymbol{M} \cdot \boldsymbol{v}_m\right)$$

where c is some 'stable' scaling factor.

The steady-state condition

$$0 \overset{!}{=} -c\,\boldsymbol{v}_m + F\left(c\,\boldsymbol{M}\cdot\boldsymbol{v}_m\right)$$

can be rewritten as

$$c\,\boldsymbol{v}_m = F\left[c\left(\kappa\,\boldsymbol{v}_m - \gamma\,\boldsymbol{n}\right)\right], \qquad\qquad \gamma \equiv \kappa\alpha + 1$$

thanks to our earlier hard work (see previous section):

$$\begin{aligned}
\boldsymbol{M}\cdot\boldsymbol{v}_m &= \left[\kappa\,\boldsymbol{K} - \frac{\boldsymbol{n}\,\boldsymbol{n}}{\alpha\,N}\right]\cdot\boldsymbol{v}_m = \kappa\,\boldsymbol{K}\cdot\boldsymbol{v}_m - \frac{\alpha N}{\alpha N}\,\boldsymbol{n} = \\
&= \kappa\left(\boldsymbol{v}_m - \alpha\,\boldsymbol{n}\right) - \boldsymbol{n} = \\
&= \kappa\,\boldsymbol{v}_m - (\kappa\alpha + 1)\,\boldsymbol{n}
\end{aligned}$$

Taking $\boldsymbol{v}_1$ as an example and writing out the components

$$c\,\boldsymbol{v}_1 = \begin{bmatrix} c \\ \vdots \\ c \\ 0 \\ \vdots \\ 0 \end{bmatrix} = F\left(\kappa\, c\, \boldsymbol{v}_1 - \gamma\, c\, \boldsymbol{n}\right) = F\left(\kappa \begin{bmatrix} c \\ \vdots \\ c \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \gamma \begin{bmatrix} c \\ \vdots \\ c \\ c \\ \vdots \\ c \end{bmatrix} \right)$$

we obtain two steady-state conditions for scaling factor c, one from components $v_{1k} = 1$:

$$c = F(\kappa\, c - \gamma\, c)$$

and another from components $v_{1k} = 0$:

$$0 = F(-\gamma\, c)$$

Our non-linearity $F()$ satisfies (barely) both of these conditions:

$$\alpha = 0.2, \qquad \kappa = 1.25, \qquad \gamma = \kappa\alpha + 1 = 1.25$$

provided the scaling factor c takes a suitable value

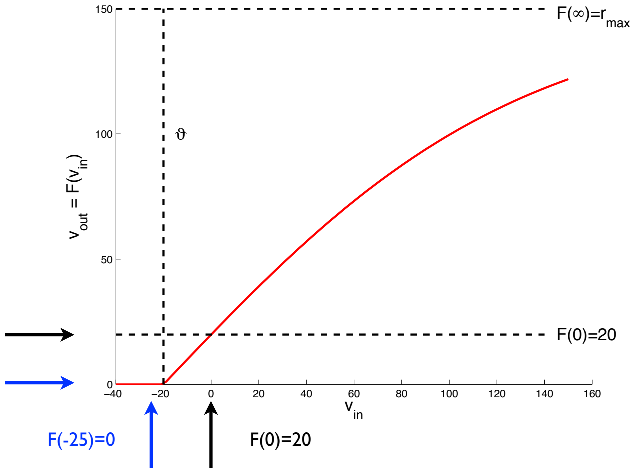
$$c = F[\kappa\, c - \gamma\, c] = F[0] \qquad \Rightarrow \qquad c = 20Hz$$

$$0 = F[-\gamma\, c] \qquad \Rightarrow \qquad c \geq 16Hz$$

since

$$-\gamma\, c \leq \vartheta = -20Hz \qquad \Leftrightarrow \qquad c \geq \frac{20Hz}{1.25} = 16Hz$$

$$F(v) = r_{max} \left[\tanh\left(\frac{v - \vartheta}{r_{max}} \right) \right]_+ , \qquad r_{max} = 150Hz \qquad \vartheta = -20Hz$$



Summary so far

We have convinced ourselves that the scaled memory patterns are indeed fixed points of the network.

Interestingly, the collective behaviour turned out to depend both on network features (connectivity) and on unit features (activation function).

Specifically, the activation function had to be just right (neither too steep, nor to flat) at input levels between zero activity and steady-state activity.

We can now validate this theoretical understanding of the collective dynamics by performing numerical simulations.

6. Simulated dynamics

Finally, we are in a position to simulate the network dynamics:

$$\tau \frac{d\boldsymbol{v}(t)}{dt} = -\boldsymbol{v}(t) + F\left[\boldsymbol{M} \cdot \boldsymbol{v}(t)\right]$$

Over a sufficiently small interval Δt, we can consider $\boldsymbol{v}(t) = \boldsymbol{v}^{(i)}$ as constant and define

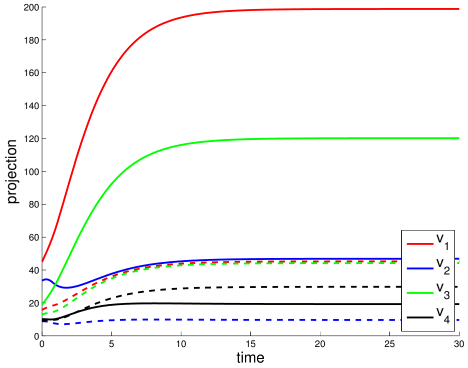
$$\boldsymbol{v}_{ss}^{(i)} \equiv F\left[\boldsymbol{M} \cdot \boldsymbol{v}^{(i)}\right]$$

During this interval, the activity will relax from $\boldsymbol{v}^{(i)}$ towards $\boldsymbol{v}_{ss}^{(i)}$:

$$\boldsymbol{v}^{(i+1)} = \boldsymbol{v}_{ss}^{(i)} + \left[\boldsymbol{v}^{(i)} - \boldsymbol{v}_{ss}^{(i)}\right] \exp\left(-{}^{\Delta t}\!/_{\tau}\right)$$

Projection on memory and non-memory patterns

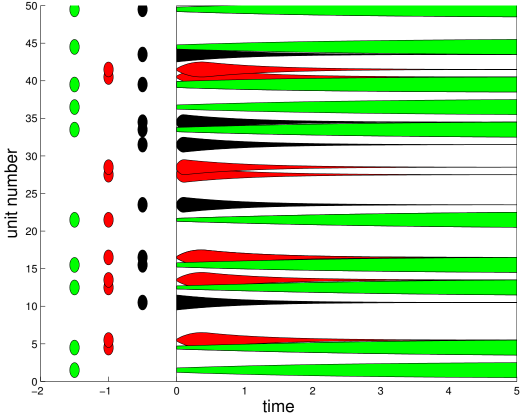
Beginning with a combination of two memory patterns, we follow the evolution of network activity in terms of its projections (dot products) on N_{mem} memory patterns and on an equal number of randomly generated non-memory patterns.



The red pattern emerges dominant. The others survive only to the extent that they overlap the red. In this simulation, $\kappa = 2$.

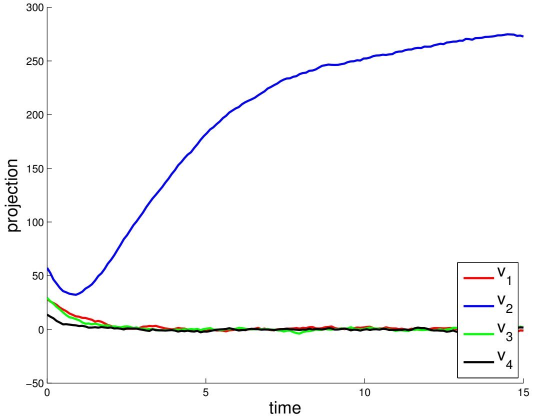
Individual units

This illustration shows the evolution of individual units, focussing on the green, red, and black patterns (indicated by colored circles on the left). We see that initial activity in the black pattern dies out quickly. The only remaining red activity is hidden beneath (i.e., overlaps with) the green activity.



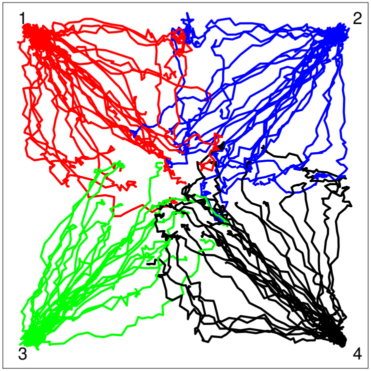
Avoiding overlap

Network performance becomes even more selective when overlap between memory patterns is avoided. In this case, one pattern dominates and all others are suppressed.

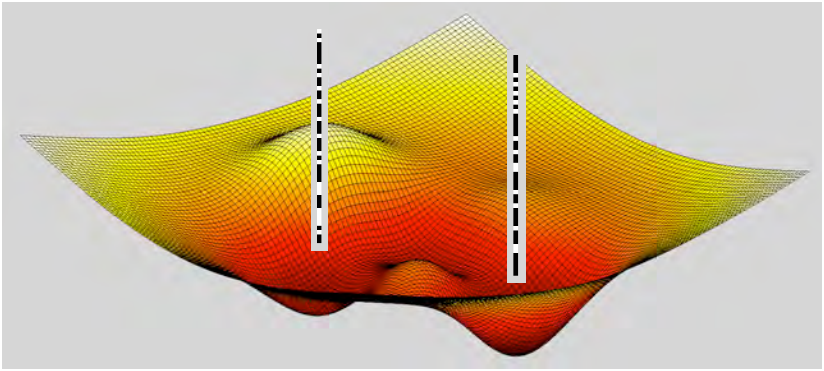


Basins of attraction

Network trajectories from random initial conditions towards one of the four memory patterns. Location in state space represents proximity to each pattern. The central region contains states that may evolve to any of the pattern. The corners contain states that reliably evolve to one pattern. The latter regions may be considered 'basins of attraction'.



ctd



Final thoughts

A recurrent network performs as an associative memory if the connectivity propagates excitation preferentially *within* memory patterns, so that memory patterns are stable fixed points of the dynamics.

This works only if (i) the memory patterns are sparse ($\alpha << 1$) and (ii) there are far fewer patterns than units ($N_{mem} << N_v$).

If these conditions are violated, too much excitation ‘leaks’ from one pattern to other patterns so that the memory function is lost.

The covariance prescription used here is far from optimal, more efficient methods can be devised.

Overall summary

- We have introduced the principle of ‘content-addressable’ or ‘associative’ memory.
- A recurrent network can perform this function, provided the desired memory patterns are fixed points of its activity.
- A simple way to achieve this is to base connectivity on the covariances of the desired memory patterns.
- In addition, the sigmoidal activation function must be just right (neither too steep, nor too flat).
- When initiated with an arbitrary activity pattern, such a recurrent network settles on the memory pattern that most resembles the input.
- Intuitively, we can visualize this in terms of an ‘energy landscape’.

Next:
State-space analysis