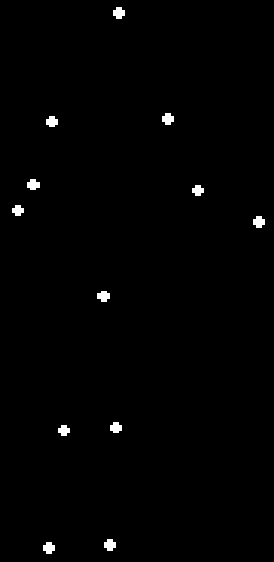
Gordon Pipa
Institute of Cognitive Science University of Osnabrück

Self-Organized Reservoir Computing

Lecture 1

The Brain: A gigantic orchestra without a conductor

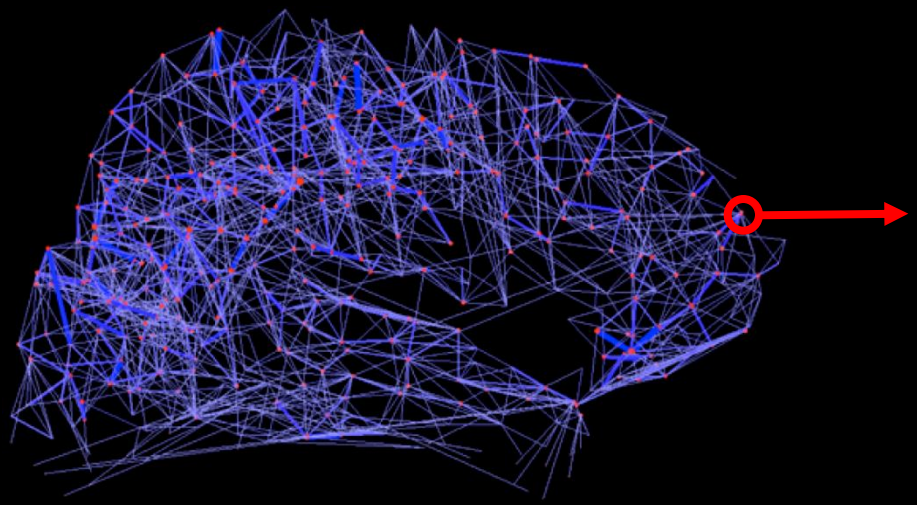


Blake 2001

- Network of ~10^{11} neurons with ~10^{14} links (synapses)
- Multi-scale structure:
  - microscopic to macroscopic: neurons – columns -- areas
  - fast to slow: spiking -- population activity -- plasticity

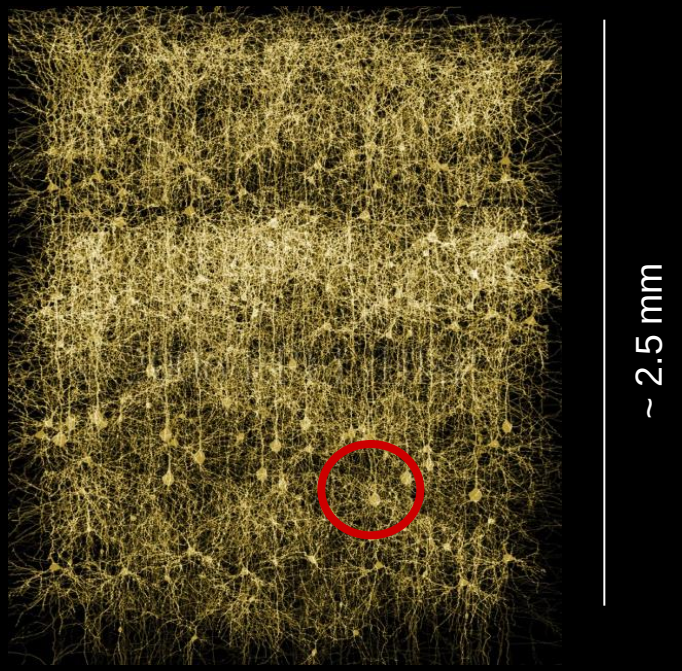
The brain: A multiscale complex neuronal network

Macroscopic network



Hagmann, et al. (2008), 'Mapping the structural core of human cerebral cortex' PLoS Biol 6(7): e159

Mesoscopic network

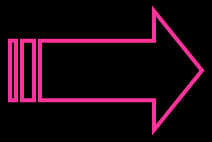


Modified figure from Blue Brain Project

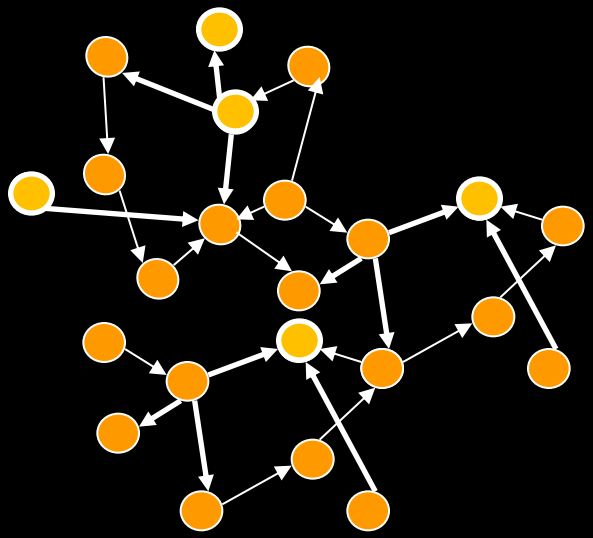
- Complex network with a high degree of randomness, and that is changing at all times
- Complex neuronal dynamics that are dominated by the intrinsic activity (Stimulus = Perturbation)

Fading memory, causal changes, and a model

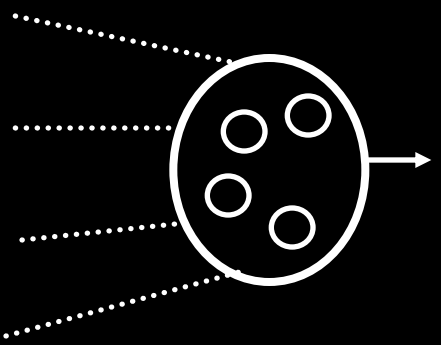
Reservoir computing



Perturbation
of the intrinsic
dynamics



task: predict past or future state or compute on input



Output

Echo State Networks ESN (Jager, 2002)

Liquid State Machines LSM (Maass et al 2003)

- Nonlinear mapping of the input signal into a high dimensional reservoir state
- Fading memory: System dynamics contain information about the past
- Reservoir computing can have universal computational properties

- H. Jaeger, Harald Haas. "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication." *Science* 304.5667 (2004): 78-80.
- W. Maass, T. Natschläger, H. Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations." *Neural computation* 14.11 (2002): 2531-2560.

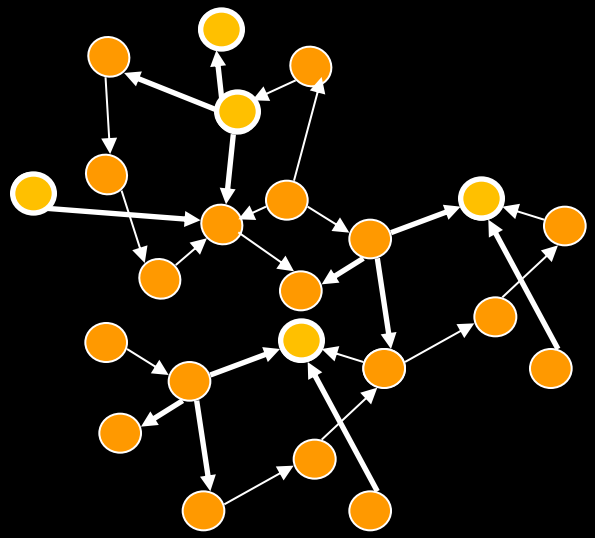
Prof. Herbert Jäger

Prof. Wolfgang Mass

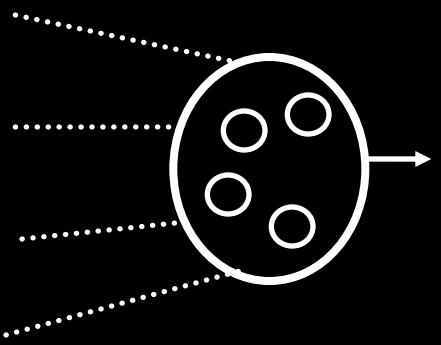
- Herbert. Jaeger, Harald Haas. "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication." *Science* 304.5667 (2004): 78-80.
- Jaeger, H., Lukoševičius, M., Popovici, D., & Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, *20*(3), 335-352.
- Yildiz, Izzet B., H. Jaeger, and S. J. Kiebel. "Re-visiting the echo state property." *Neural Networks* (2012).
- Waegeman, Schrauwen, Jaeger. "Technical report on hierarchical reservoir computing architectures." (2012).

- W. Maass, T. Natschläger, H. Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations." *Neural computation* 14.11 (2002): 2531-2560.
- Buesing, Lars, et al. "Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons." *PLoS computational biology* 7.11 (2011): e1002211.
- Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. Nature Reviews Neuroscience, 10(2), 113-125.

Perturbation
u(t)



Task specific



y(t)

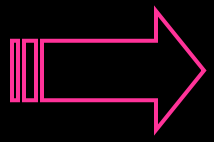
$$x^M(t) = (L^M u)(t) \qquad y(t) = f^M(x^M(t))$$

In mathematical terms, this liquid state is simply the current output of some operator or filter L^M that maps input functions $u(.)$ onto a liquid state $x^M(t)$:

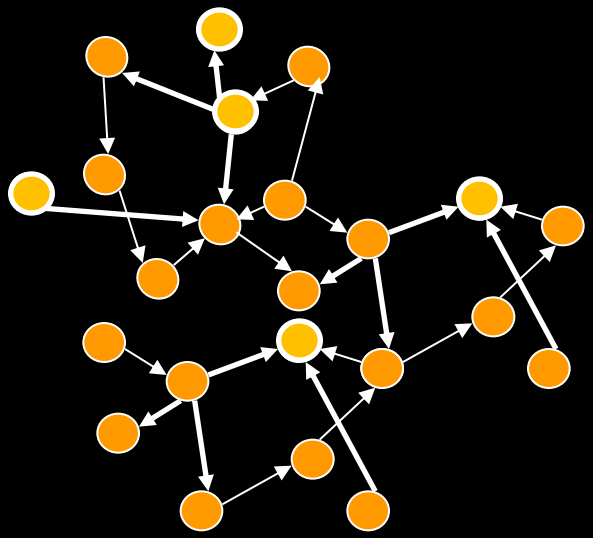
The second component is a memory-less readout map f^M that transforms, at every time t , the current liquid state $x^M(t)$ into the output

W. Maass, T. Natschläger, H. Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations." *Neural computation* 14.11 (2002): 2531-2560.

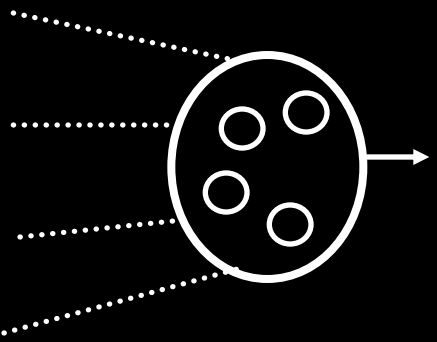
A First example of a reservoir computer



1) n-th order Markov process: A,B,C,A, …



task: predict past or future state or compute on input



Incoming drive | Threshold | Input

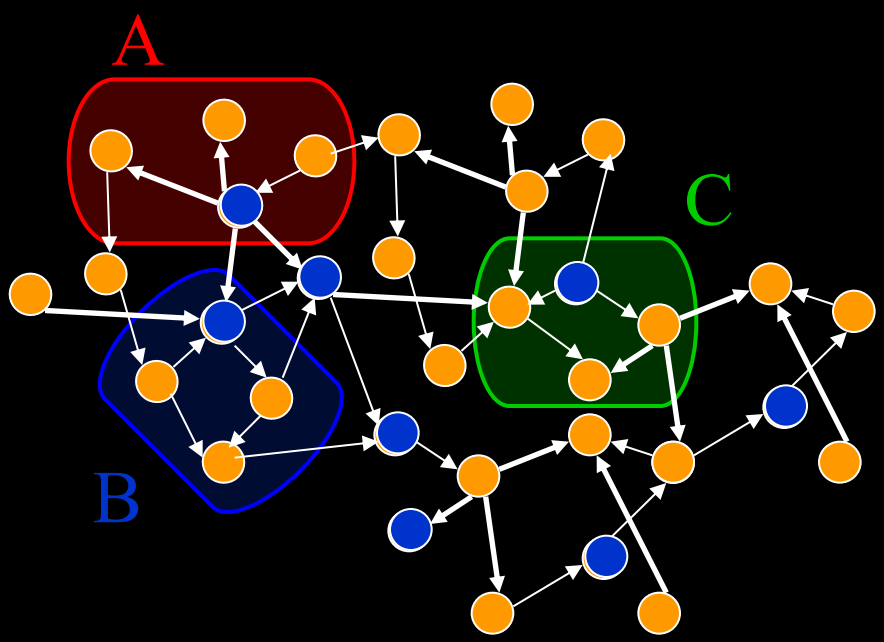
$$h_i(t+1) = f\left[\left(\sum_{j=1}^{N} W_{ij}(t)\,x_j(t)\right) - T_i(t) + I(t)\right]$$

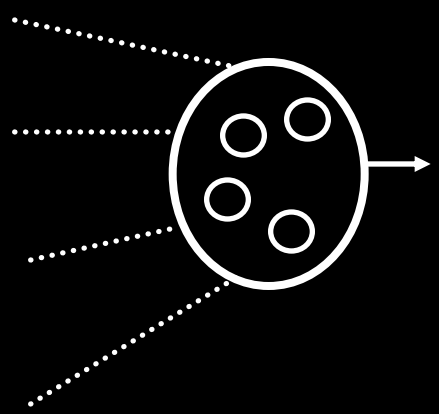
- A. Lazar*, G. Pipa*, and J. Triesch (*authors contributed equally), Neural Networks, 20(3):312--322, 2007
- A. Lazar, G. Pipa, and J. Triesch, Frontiers Computational Neuroscience 2009

Input time series:

input

$$h_i(t+1) = KWTA\left(\sum_{j=1}^{N} W_{ij}(t)\,x_j(t)\right) - T_i(t) + u_i(t)$$





Classification

A C B ...

Time series processing

time series: *n*-th order Markov process

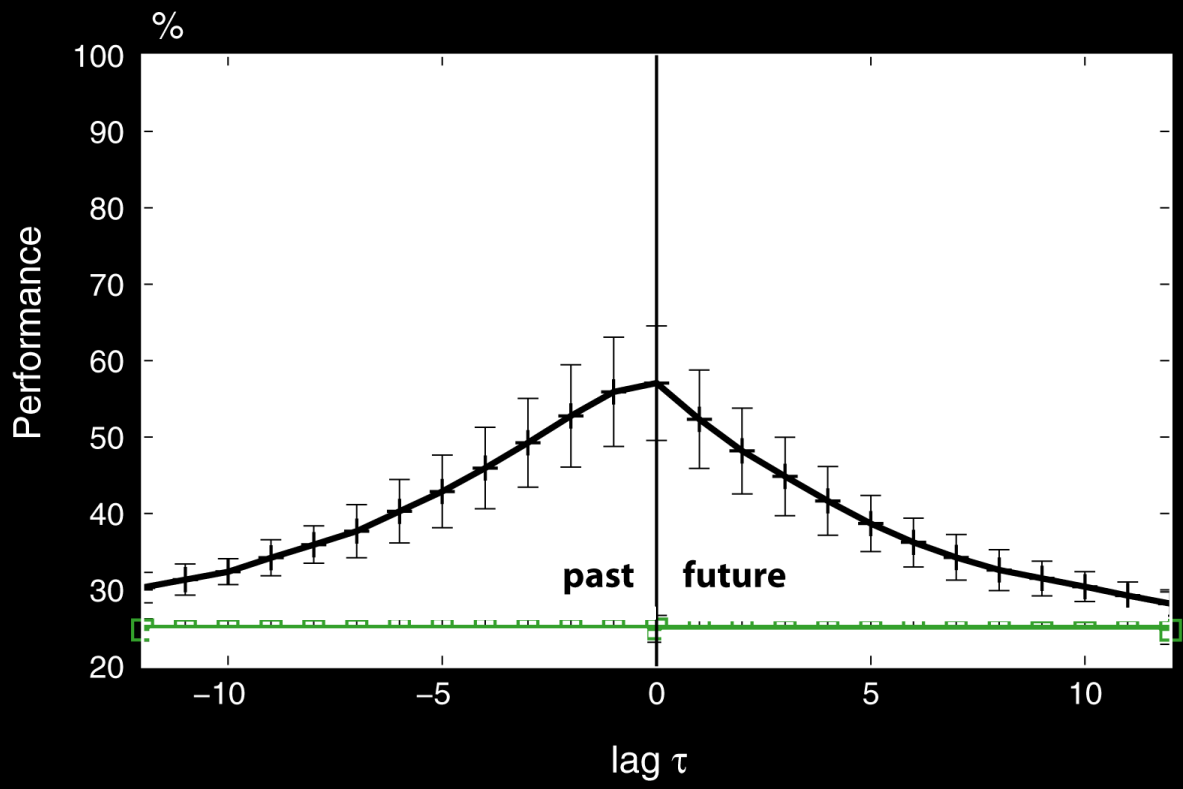
task: predict past or future state

1st order Markov process

Old new	A	B	C	D
A	0.03	0.03	0.03	0.91
B	0.91	0.03	0.03	0.03
C	0.03	0.91	0.03	0.03
D	0.03	0.03	0.91	0.03

Preferred sequence:

A → B → C → D → A



Fading memory and time series prediction in recurrent networks with different forms of plasticity
A. Lazar*, G. Pipa*, and J. Triesch (*authors contributed equally), Neural Networks, 20(3):312--322, 2007

Questionnaire: Q1

1. Tick off correct statements.

(a) (6 pts) There might be more than one correct statements.

- [x] Reservoir Computing has universal computational properties
- [x] With a reservoir computer any arbitrary function with fading memory can be approximated
- [ ] A reservoir computer relies on stable states like fix-point attractors
- [x] Reservoir computing is an any time computation
- [ ] Reservoir computing, like other attractor based methods, i.e. Hopfield networks, stores patterns by means of learned attractors
- [x] Reservoir computing relies on transient dynamics

✖ **correct**

✖ **wrong**

Questionnaire: Q1

2. Reservoir computing relies on two principle mechanism. First an operator that maps any input $u(\cdot)$ to an reservoir state $x^M(t)$. And second a mapping of reservoir states onto an output functions $y(t)$.

(a) (6 pts) There might be more than one correct statements.

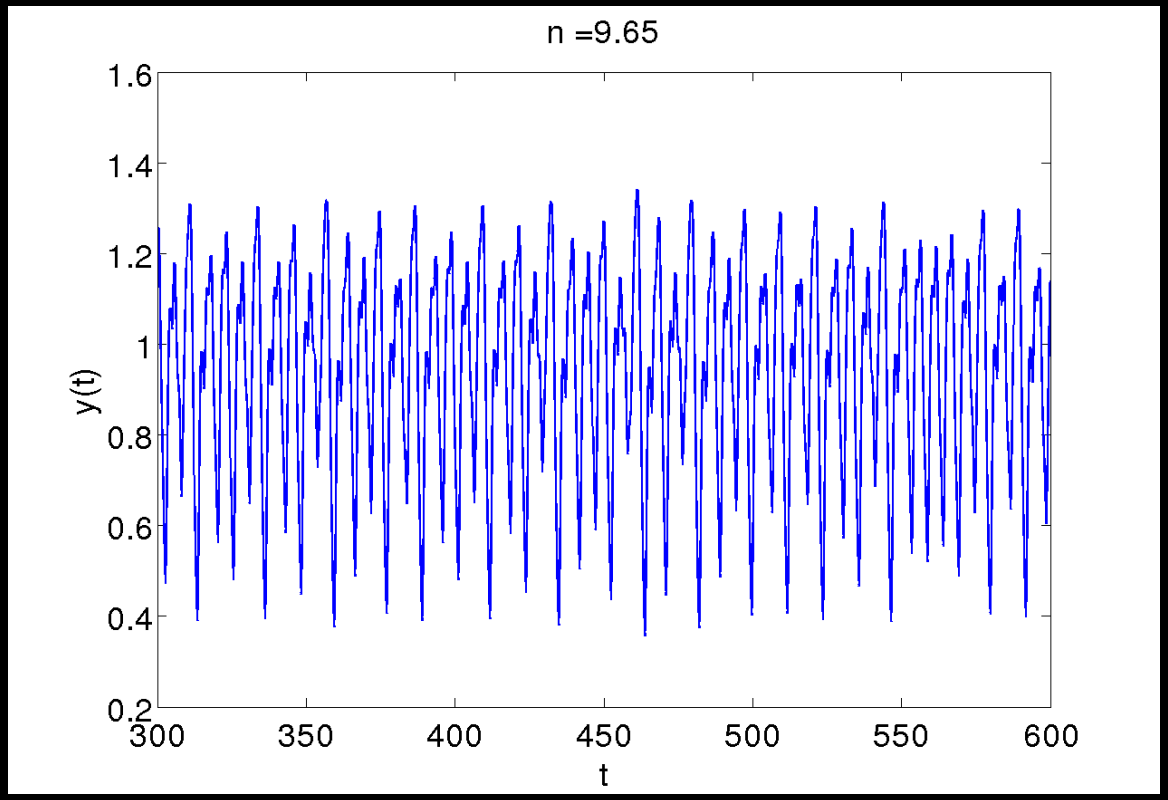
- ✗ The mapping of the reservoir state $x^M(t)$ onto an output function $y(t)$ is learned
- ✗ The mapping of the reservoir state $x^M(t)$ onto an output function $y(t)$ is memory free
- ✗ The operator that maps the input $u(\dot{)}$ onto reservoir states $x^M(t)$ is optimized via error back propagation
- ✗ The only memory containing element in a Reservoir computer is the operator that maps any input $u(\cdot)$ to an reservoir state $x^M(t)$
- ✗ Every Reservoir is optimized for a certain task
- ✗ Reservoirs are universal in the sense, that the same reservoir can be used for many different tasks at the same time.

✗ **correct**

✗ **wrong**

A few examples for RC

Prediction of time series with Reservoir computation



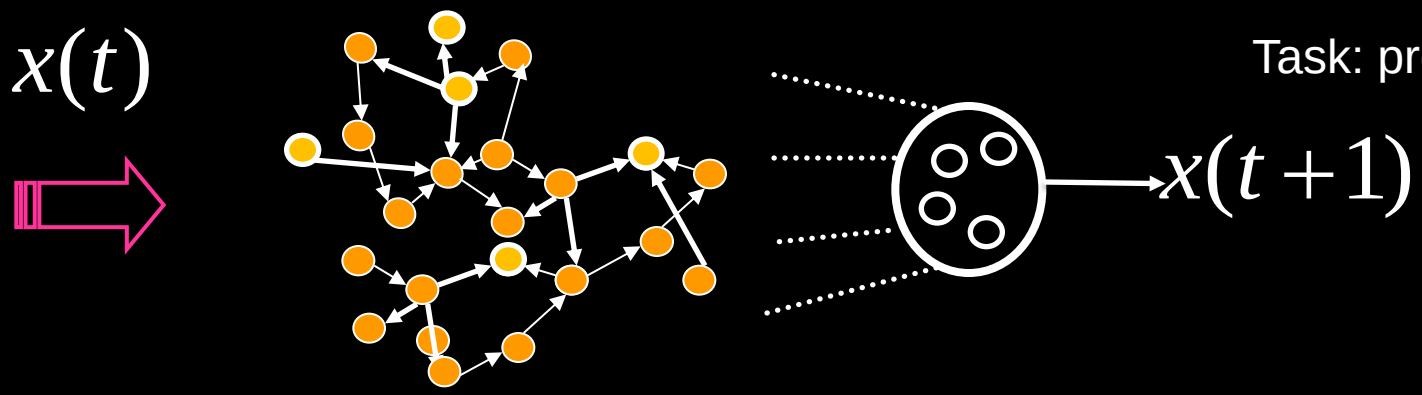
- complex dynamics including chaos.
- time- delay differential equations are infinite dimensional
- This makes ODE and DDE very different !

Mackey-Glass equation

$$\frac{dx}{dt} = \beta \frac{x(t-\tau)}{1+x(t-\tau)^n} - \gamma \cdot x(t)$$

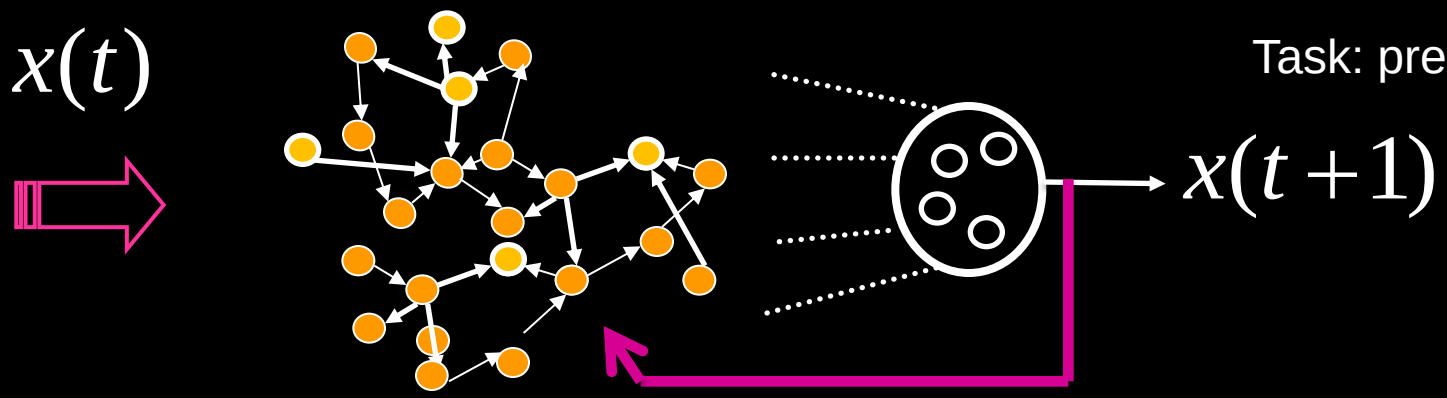
http://www.scholarpedia.org/article/Mackey-Glass_equation

Prediction of time series with Reservoir computation

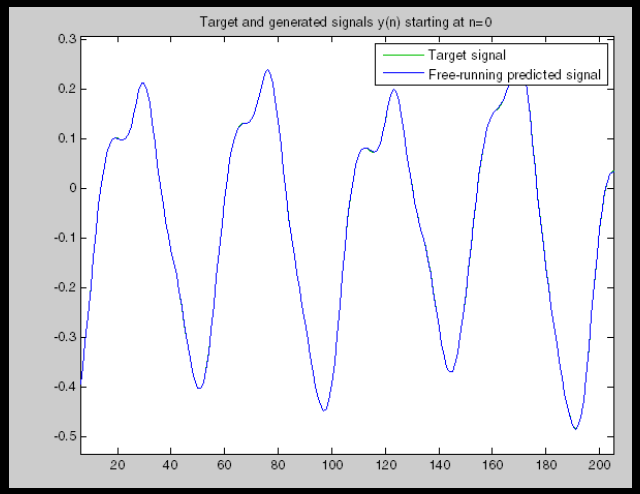


- H. Jaeger and H. Haas , 'Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication' *Science 2 April 2004: vol. 304 no. 5667 pp. 78-80*
- by Lukoševicius, M.http://minds.jacobs-university.de/mantas/code

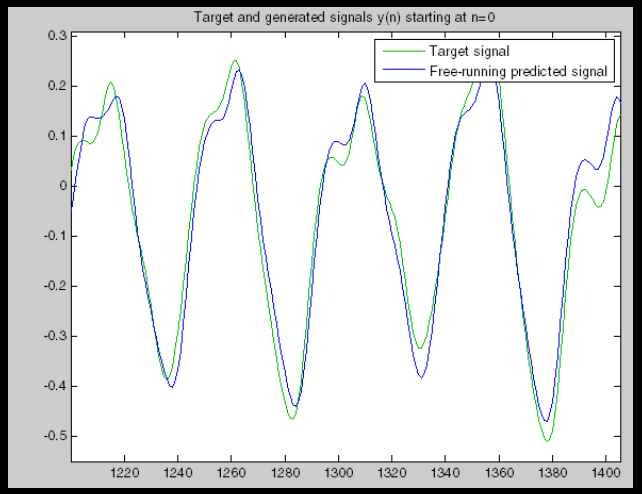
Prediction of time series with Reservoir computation



Task: predict values



[0, 200]



[1200, 1400]

Prediction of a mildly chaotic signal

Predicted and original plotted on top of each other

- H. Jaeger and H. Haas , 'Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication' *Science 2 April 2004: vol. 304 no. 5667 pp. 78-80*
- by Lukoševicius, M.http://minds.jacobs-university.de/mantas/code

Flight control based on Reservoir computation

Target Position

Position (now)

Bubble
Echo state network

6*200 nodes

Tilt

Rotor speed

H. Jaeger and H. Haas , 'Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication' *Science 2 April 2004: vol. 304 no. 5667 pp. 78-80*

Motion control Reservoir computation

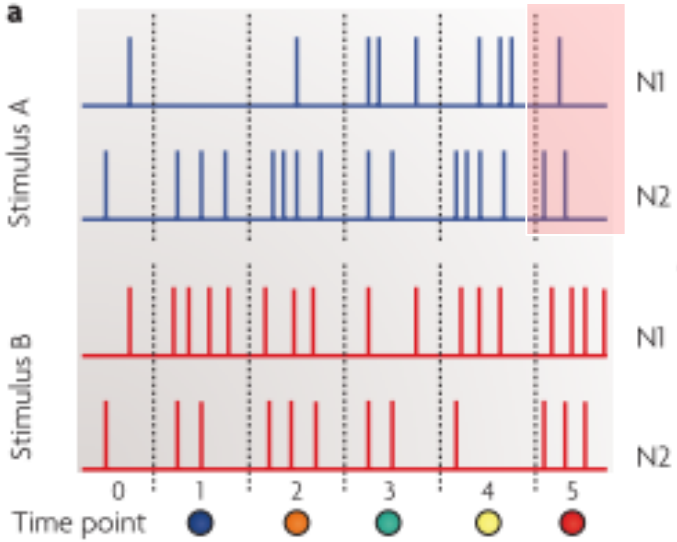
Study Project at the Institute of Cognitive Science, Kristoffer Appel, Witali Aswolinskiy, Patrick Bastien, Alexander, Lelais, Robert Pfeiffer, Leon René Sütfeld

Motion control Reservoir computation

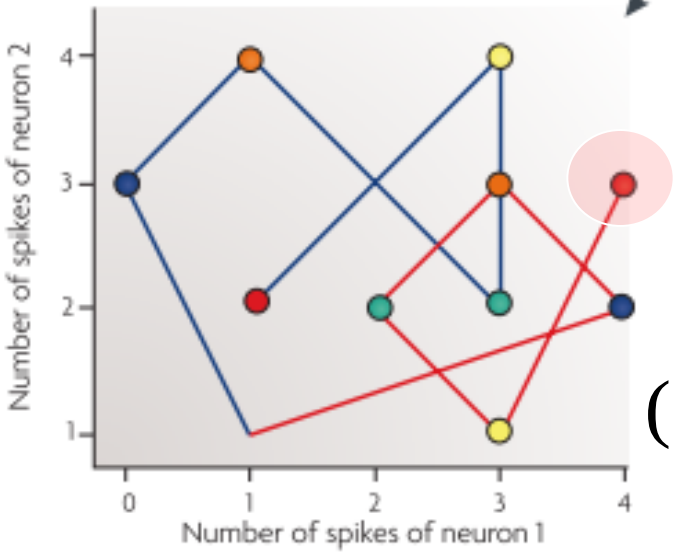
Study Project at the Institute of Cognitive Science, Kristoffer Appel, Witali Aswolinskiy, Patrick Bastien, Alexander Lelais, Robert Pfeiffer, Leon René Sütfeld

RC in neurobiology

Reservoir alike computation in a locust



$$(x, y) = (n_1(t), n_2(t))$$

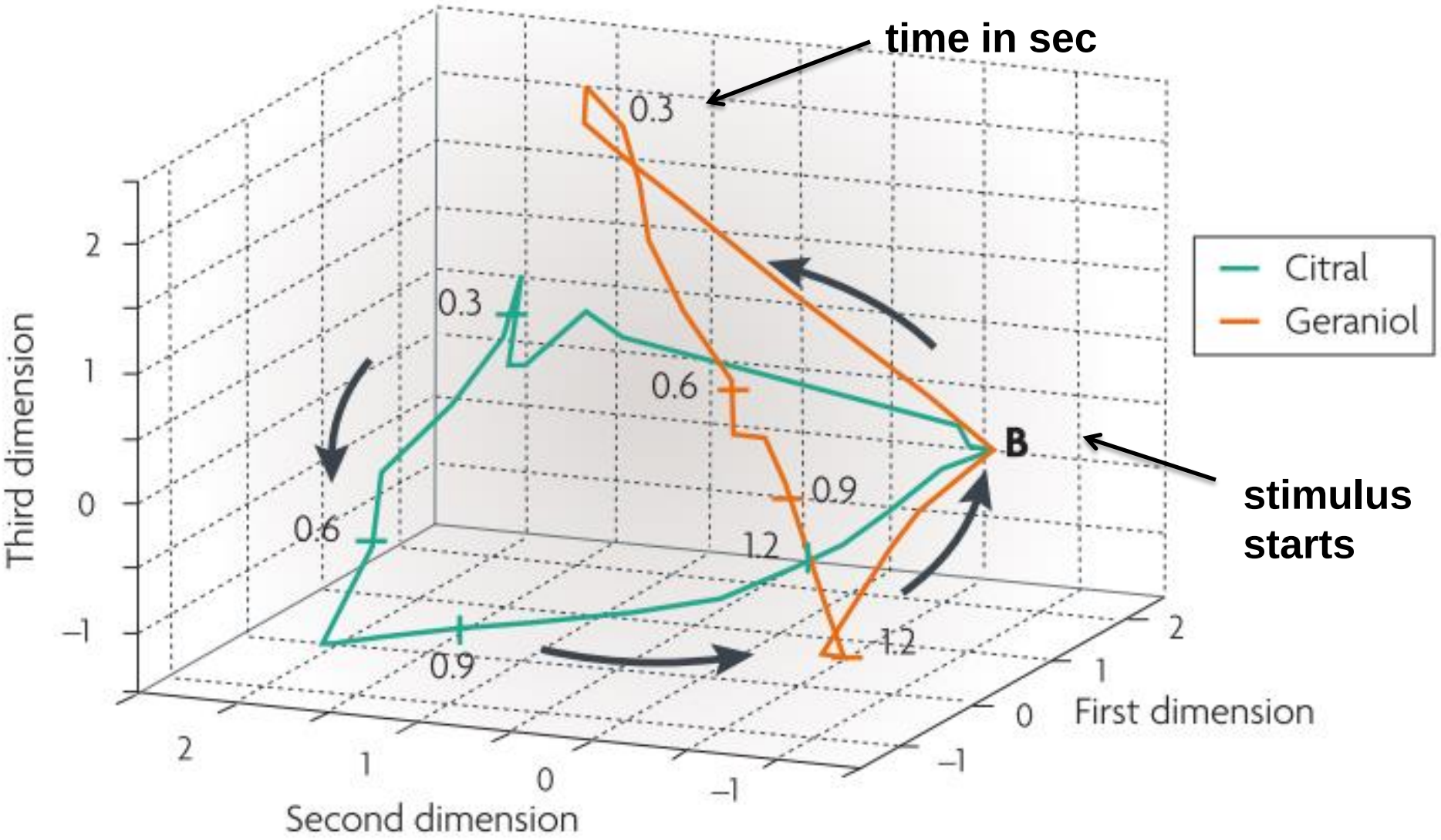


$$(n_1(t=5), n_2(t=5))$$

$$(n_1(t=4), n_2(t=4))$$

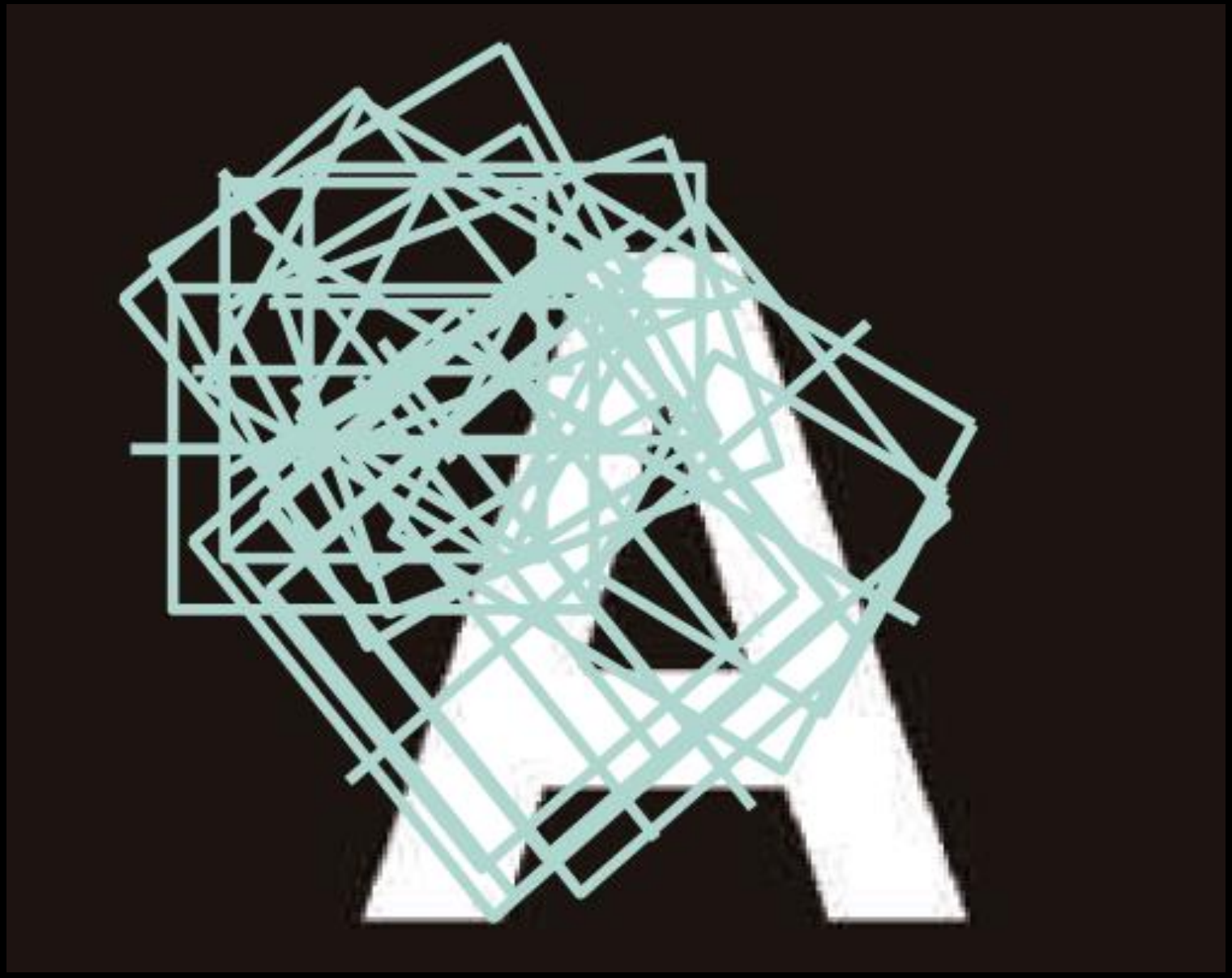
Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, *10*(2), 113-25.

Reservoir alike computation in a locust (87 projection neurons)

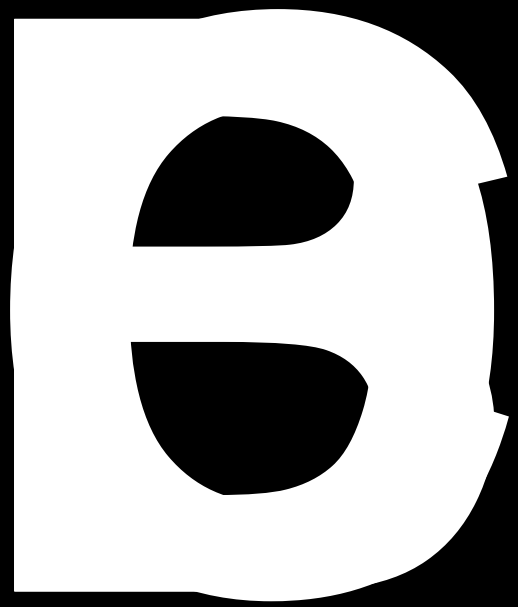


Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, *10*(2), 113-25.

Reservoir alike computation in V1 in cats

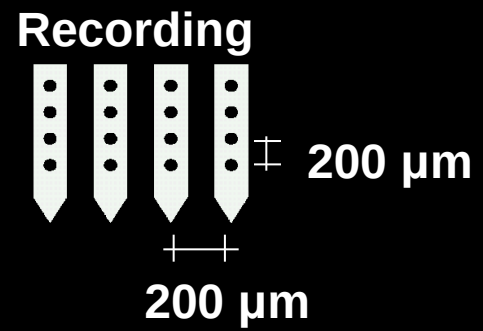


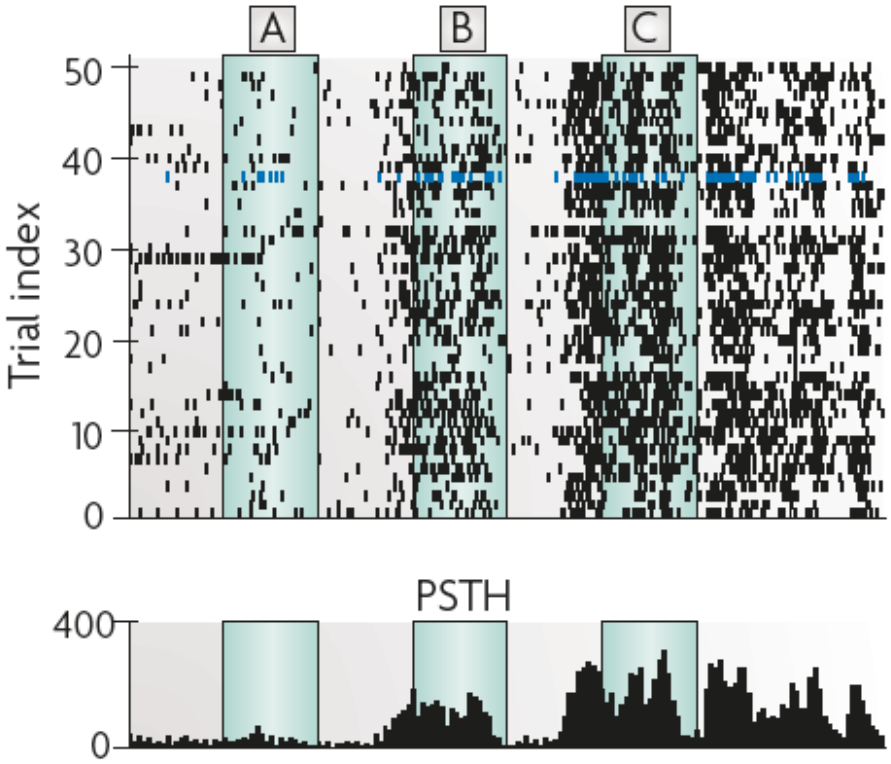
Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, *10*(2), 113-25.

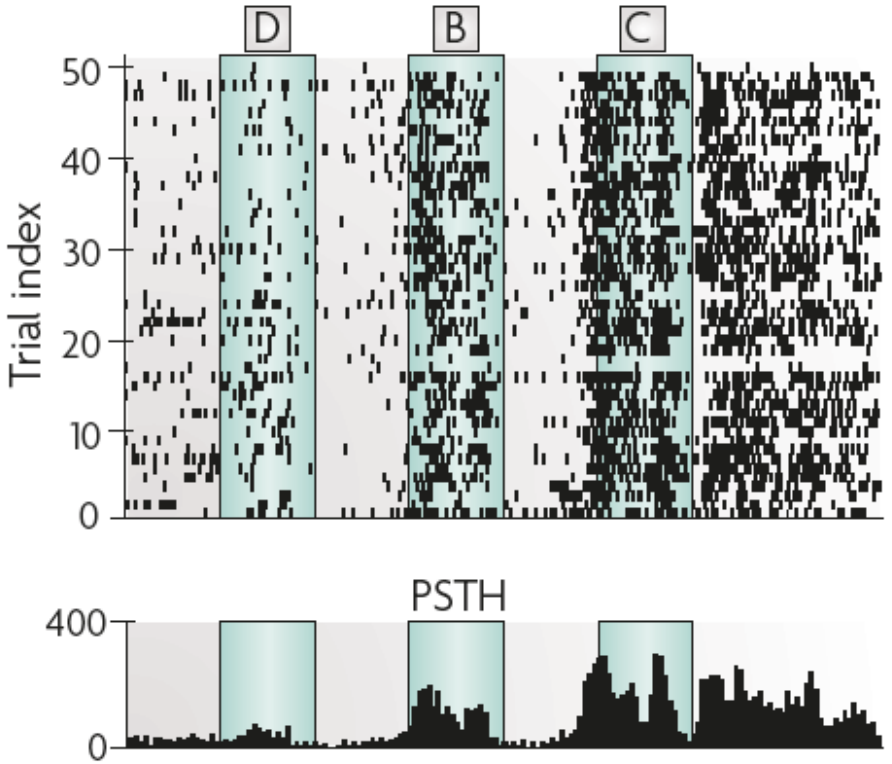


Reservoir alike computation in V1 in cats

Sequence 1: A→B→C
Sequence 2: D→B→C



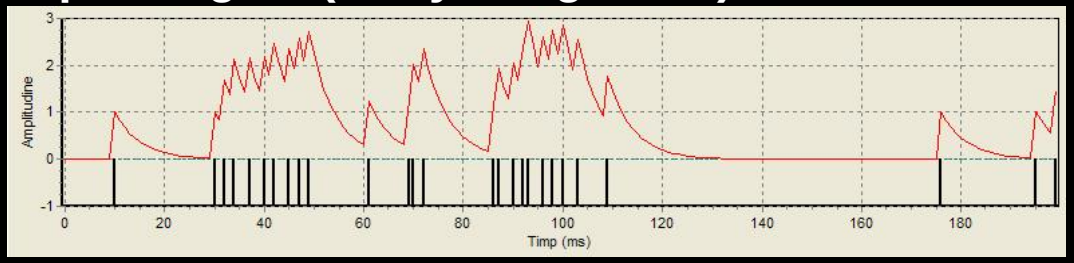




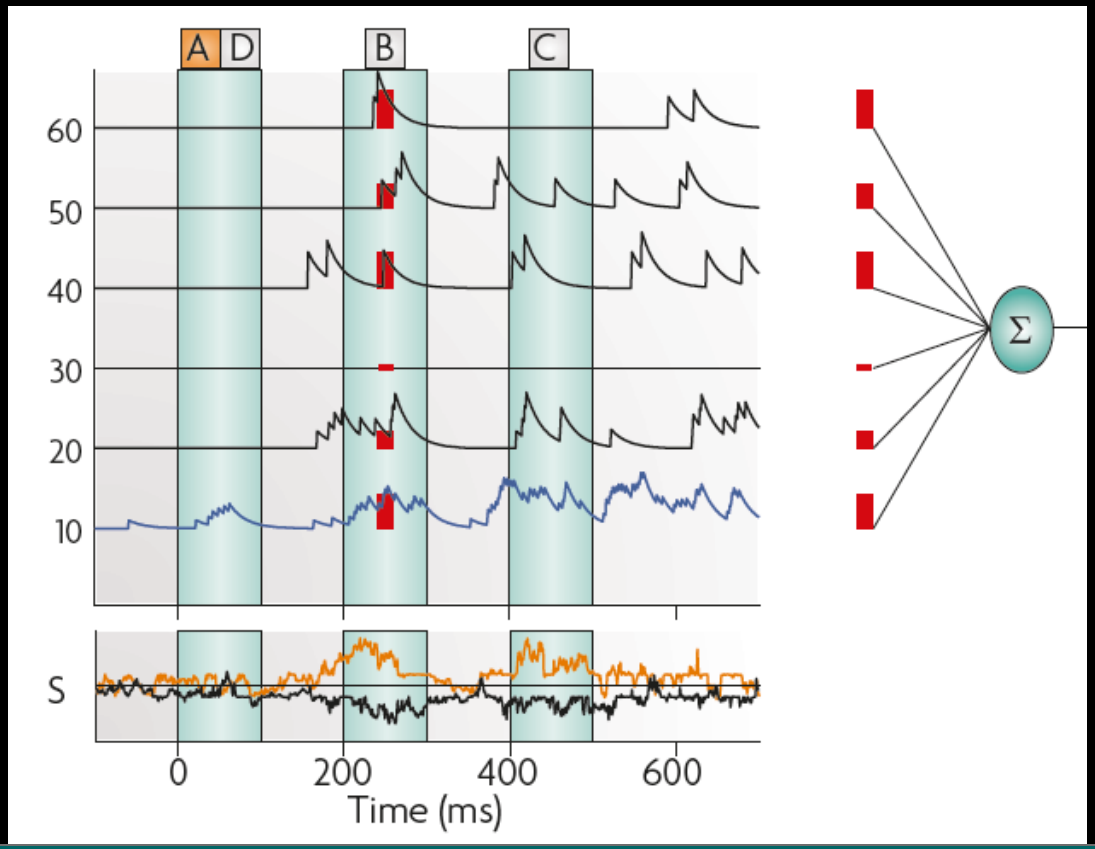
Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, *10*(2), 113-25.

Reservoir alike computation in V1 in cats

Spike signal (leaky integration)



Mapping of a spiking signal to an analog trace (filtered with an exponential kernel)

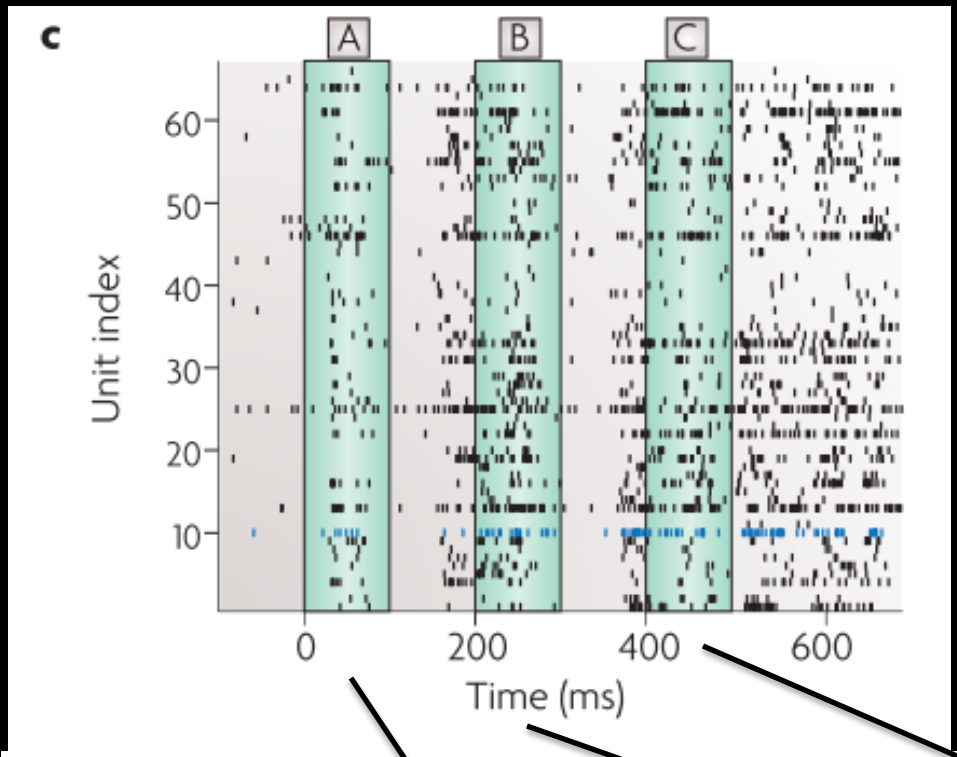


Sequence 1:
A→B→C

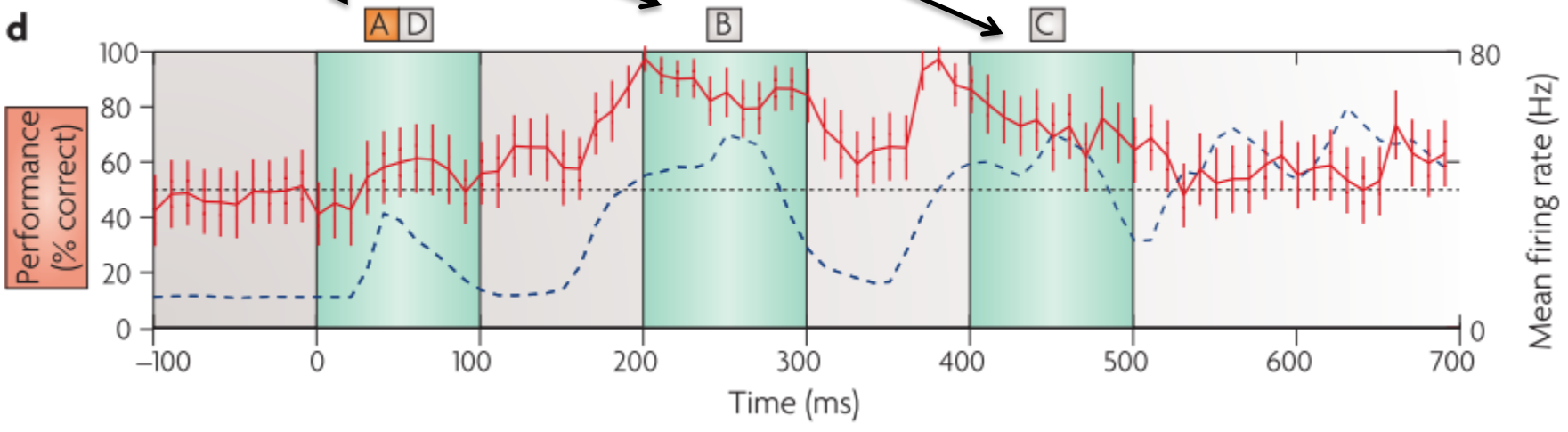
?

Sequence 2:
D→B→C

Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, *10*(2), 113-25.

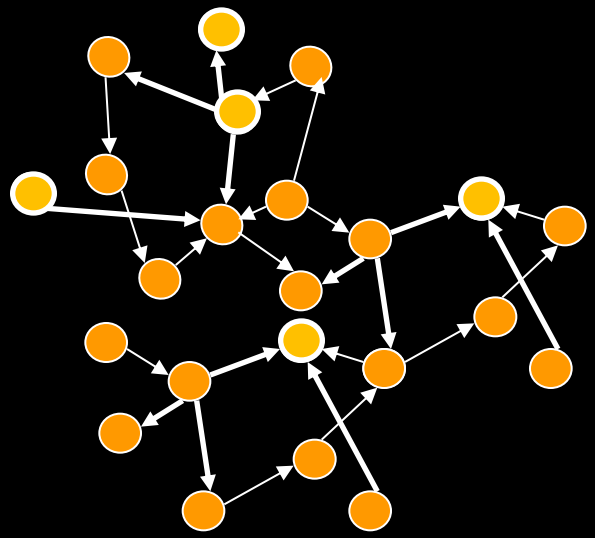

c
A
B
C
Unit index
60
50
40
30
20
10
0
200
400
600
Time (ms)

Time (ms)

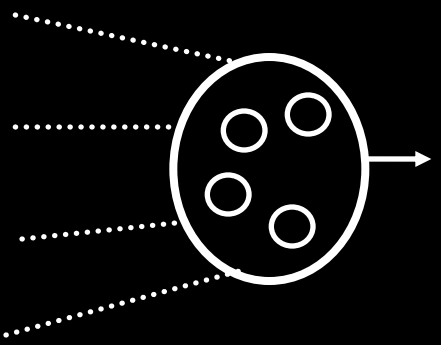

d
A D
B
C
Performance (% correct)
100
80
60
40
20
0
80
0
Mean firing rate (Hz)
–100
0
100
200
300
400
500
600
700
Time (ms)

Mathematical foundation of RC I

Perturbation
u(t)



Task specific



y(t)

$$x^M(t) = (L^M u)(t) \qquad y(t) = f^M(x^M(t))$$

In mathematical terms, this liquid state is simply the current output of some operator or filter L^M that maps input functions $u(.)$ onto a liquid state $x^M(t)$:

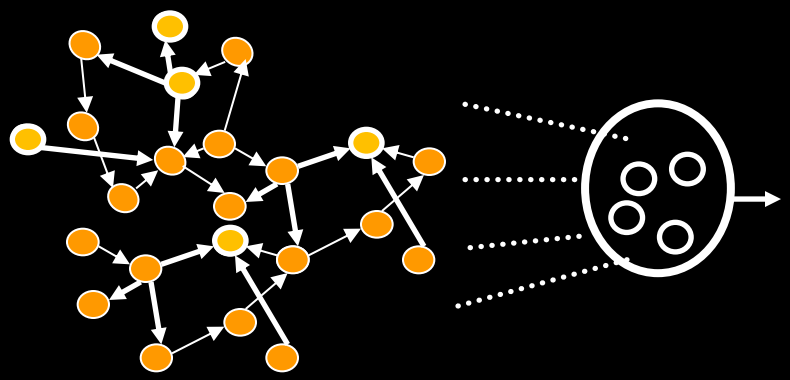
The second component is a memory-less readout map f^M that transforms, at every time t , the current liquid state $x^M(t)$ into the output

W. Maass, T. Natschläger, H. Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations." *Neural computation* 14.11 (2002): 2531-2560.

Non-linear Computation

What do we need for computation ?

Read out is usually a linear combination of reservoir states



→ Any non-linearity must come from the recurrent network

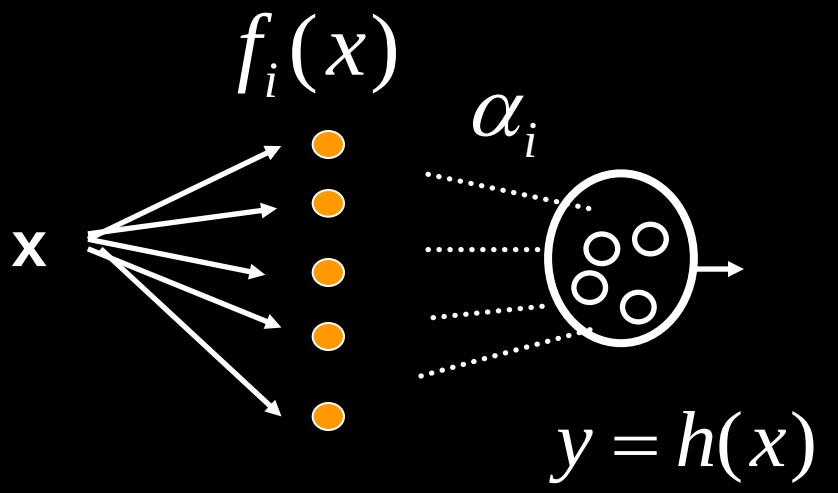
→ Recurrent network is composed of non-linear elements

→ Many different neuron types have been used in literature:

- threshold logic gates
- hyperbolic
- leaky integrator and spiking neurons (with or without synaptic models and dynamic synapses)
- ….

Non-linear Computation

$$y = h(x)$$



- We try to approximate any function by a finite sum of different non-linear mappings of the input
- The recurrent network is doing an feature expansion:
  That is just a set of different non-linear transforms of the input
- The parameters α is learned by a supervised learning given the target value y
- So we choose the α such that the resulting mapping is close to the given function h(x) (i.e. ML).

Questionnaire: Q1

3. Tick off correct statements.

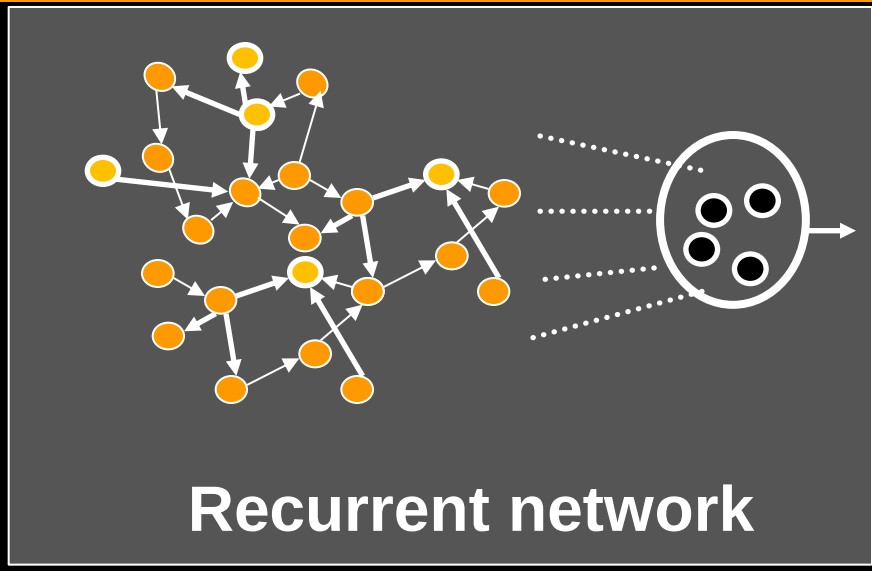
(a) (6 pts) There might be more than ones correct statements.

- ☒ A reservoir computer can act as a generative model one the readout is trained
- ☒ The output of the reservoir can be used as feedback into the reservoir
- ☒ Standard reservoir computing predicts receptive fields in V1
- ☒ Representations of memorized Stimuli in V1 are super sparse, and unique
- ☒ Rather complex activity patterns store past and recent stimuli in V1
- ☒ Patterns that form representations of past inputs have vary very little across trials

☒ **correct**

☒ **wrong**

Temporal unfolding (time discrete)



X(t-2)

$f_i(x_{t-2})$

X(t-1)

$f_i(x_{t-2}, x_{t-1})$

X(t)

$f_i(x_{t-2}, x_{t-1}, x_t)$

α_i

$y \approx h(x_{t-2}, x_{t-1}, x_t)$

Elman, J. L. Finding structure in time. Cogn. Sci. 14, 179–211 (1990).

Elman, J. L. & Zipser, D. Learning the hidden structure of speech. J. Acoust. Soc. Am. 83, 1615–1626 (1988).

- **There is an echo in the system that is given by the echo function**

$$f_i(x_{t-2}, x_{t-1}, x_t)$$

- **Past and recent states are combined by a non-linear function in every update**

  → Non-linear Kernel expansion of past and recent states
  → Non-linear computation across time

- **What are the necessary conditions to maintain information about the past and implement non-linear computation ?**

  - Echo state property
  - Separation property
  - Approximation property

echo state property

- **echo state property:**
  current activity is uniquely defined by the input sequence

  → Echo function $f_i(x_{t-2}, x_{t-1}, x_t)$

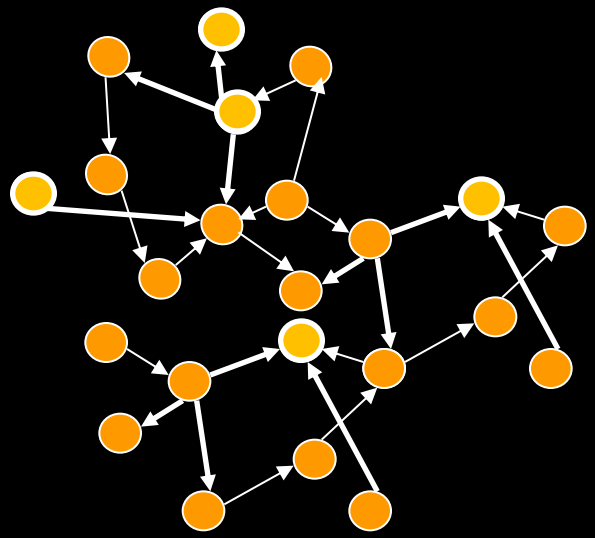
- The reservoir must asymptotically forget its input history, given that is driven by an input

- Or, the system must have fading memory

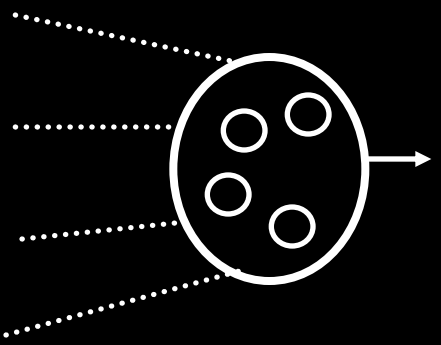
Fading memory (Boyd at al., 1985) for very $\varepsilon > 0$ there exist $\delta > 0$ and $T > 0$ so that $|(Fv)(t=0) - (Fu)(t=0)| < \varepsilon$ with $|u(t) - v(t)| < d$ for all $t \in [-T, 0]$.

- Yildiz, Izzet B., H. Jaeger, and S. J. Kiebel. "Re-visiting the echo state property." *Neural Networks* (2012).
- H. Jaeger, 'Reservoir Riddles: Suggestions for Echo State Network Research '

Perturbation
u(t)



Task specific



y(t)

$$x^M(t) = (L^M u)(t) \qquad y(t) = f^M(x^M(t))$$

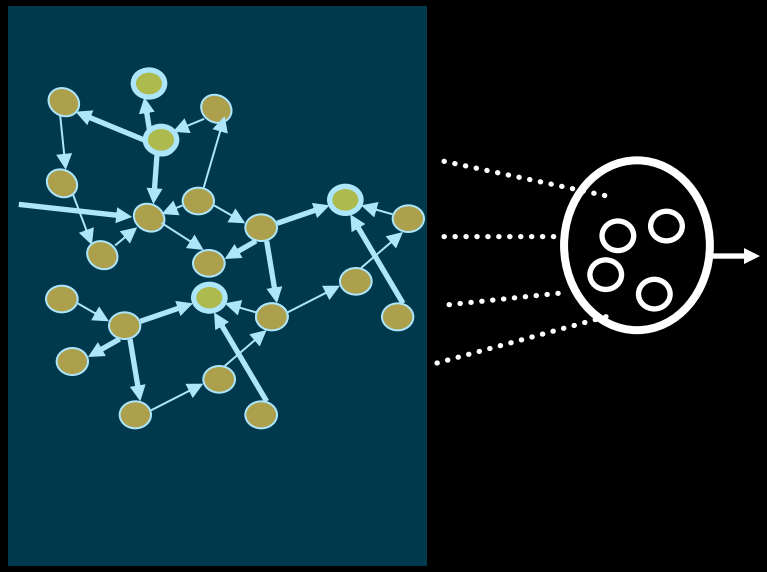
In mathematical terms, this liquid state is simply the current output of some operator or filter L^M that maps input functions $u(.)$ onto a liquid state $x^M(t)$:

The second component is a memory-less readout map f^M that transforms, at every time t , the current liquid state $x^M(t)$ into the output

W. Maass, T. Natschläger, H. Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations." *Neural computation* 14.11 (2002): 2531-2560.

Separation property

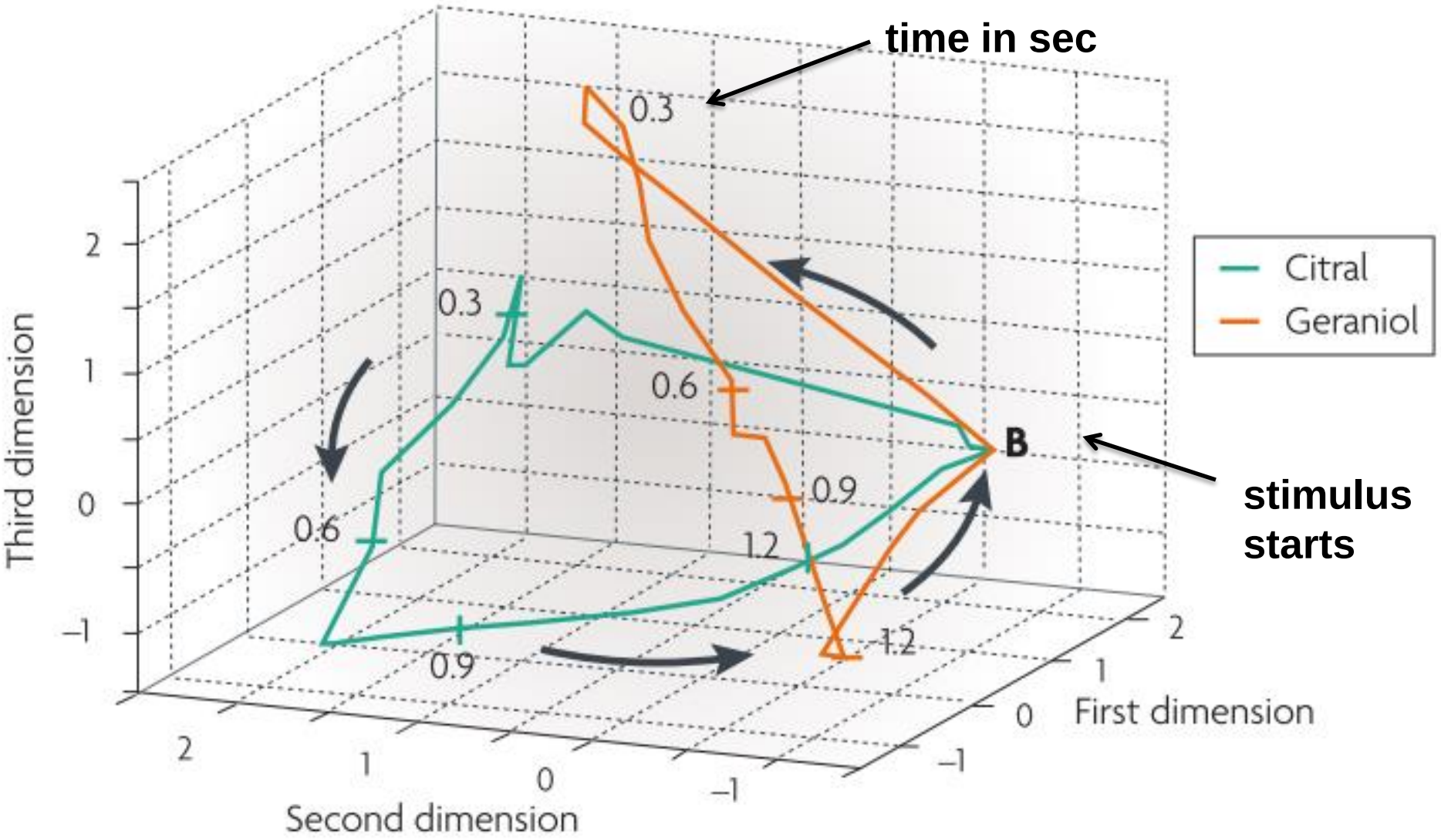
Separation property addresses the amount of separation between the trajectories of internal states of the system that are caused by two different input streams.



$$x^M(t) = (L^M u)(t)$$

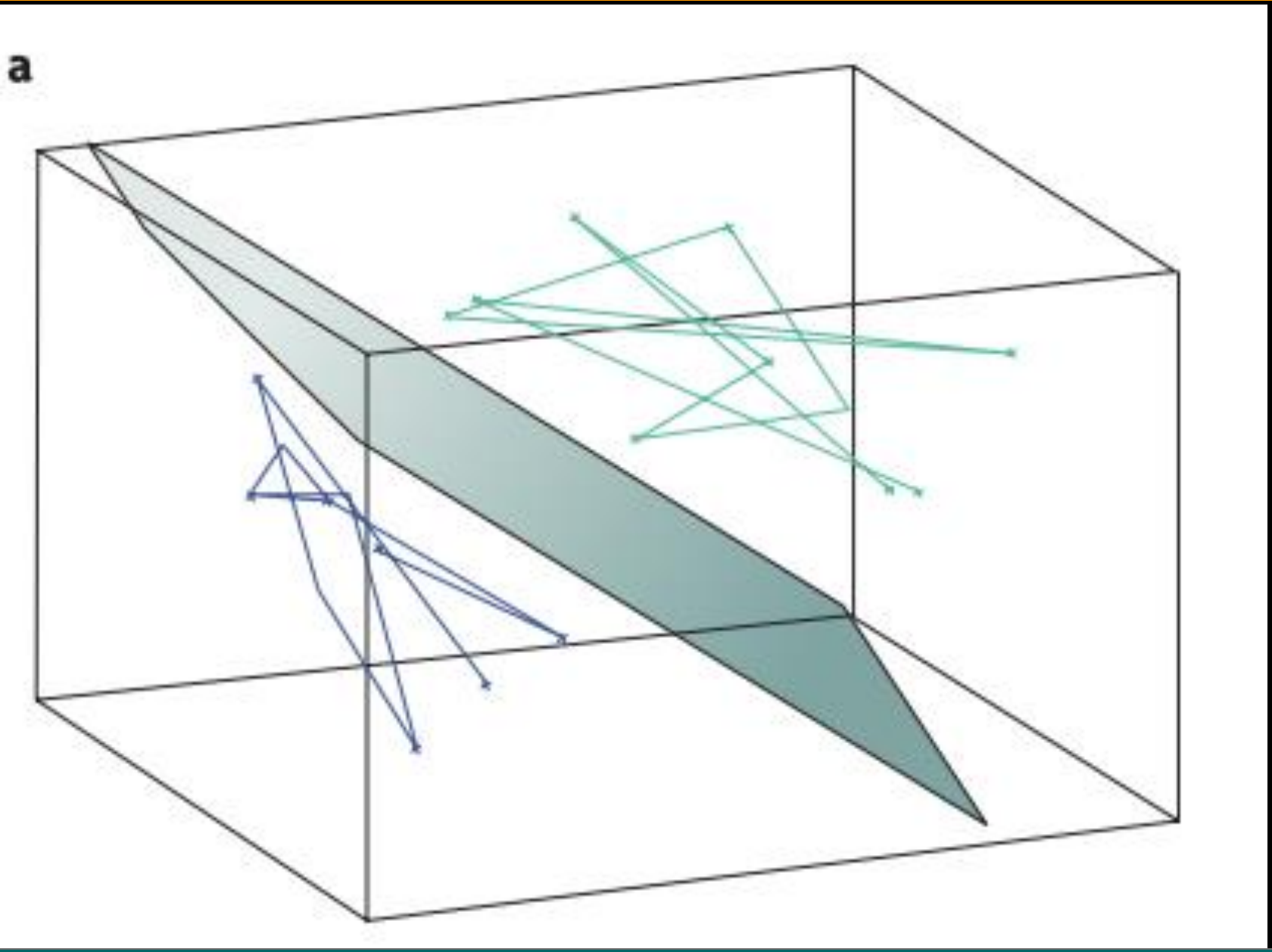
Mainly determined by the ‘complexity’ of the liquid

Reservoir computation locust (87 projection neurons)



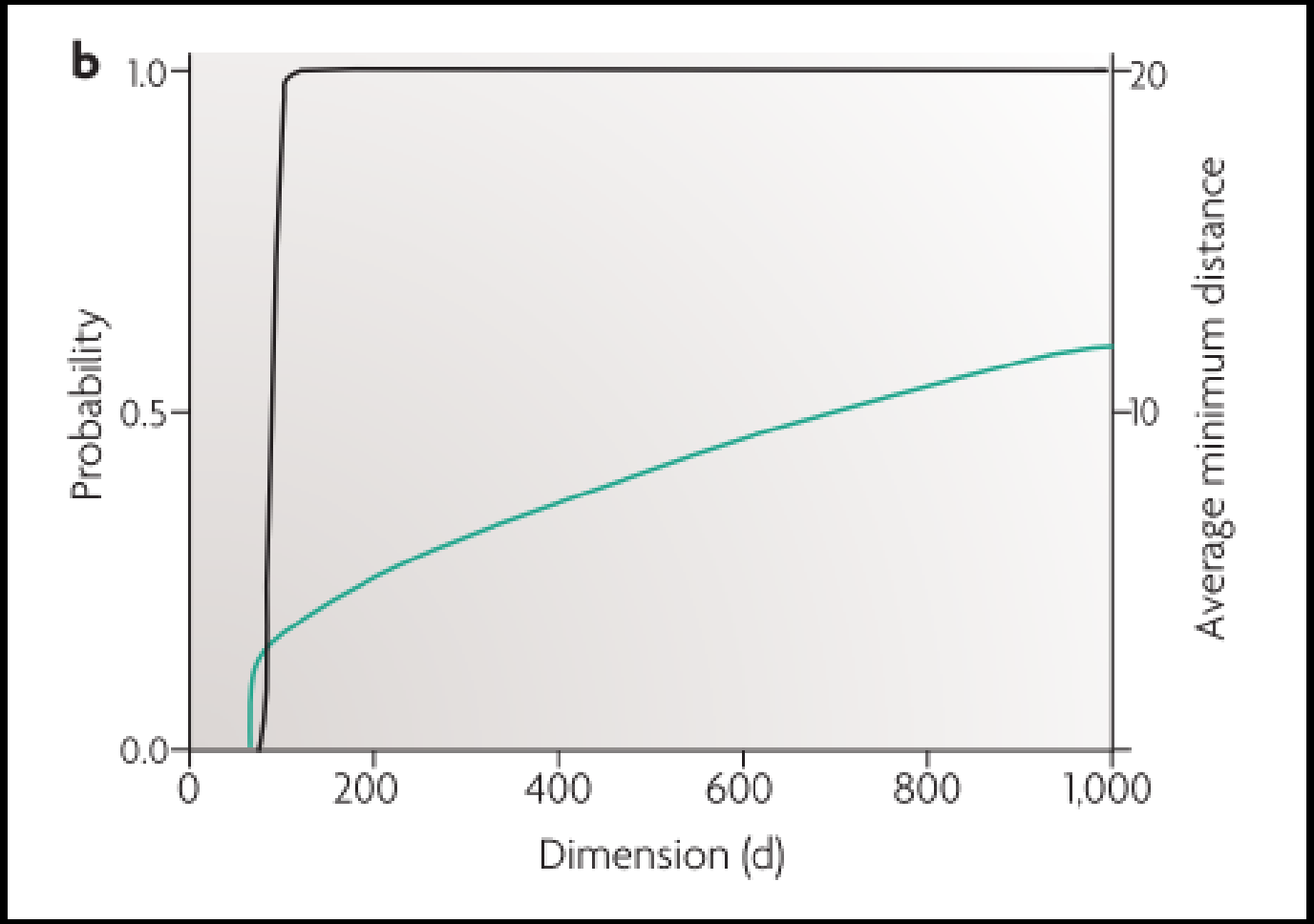
Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, *10*(2), 113-25.

Principle of RC computation



Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, *10*(2), 113-25.

Principle of RC computation



Black: probability that 2 trajectories that each linearly connect 100 randomly chosen points can be separated by a hyperplane

Green: average of the minimal Euclidean distance between pairs of trajectories

Buonomano, D. V., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, *10*(2), 113-25.

Separation property

Separation property addresses the amount of separation between the trajectories of internal states of the system that are caused by two different input streams.

Large computational power on functions of time can be realized even if all memory traces are continuously decaying.

→ 'Instead of worrying about the code and location where information about past inputs is stored, and how this information decays, it is enough to address the separation question: For which later time points *t* will any two significantly different input functions of time *u*(.) and *v*(.) cause significantly different liquid states *x*(*t*) *for input u and v ?*

- Yildiz, Izzet B., H. Jaeger, and S. J. Kiebel. "Re-visiting the echo state property." *Neural Networks* (2012).
- H. Jaeger, 'Reservoir Riddles: Suggestions for Echo State Network Research '

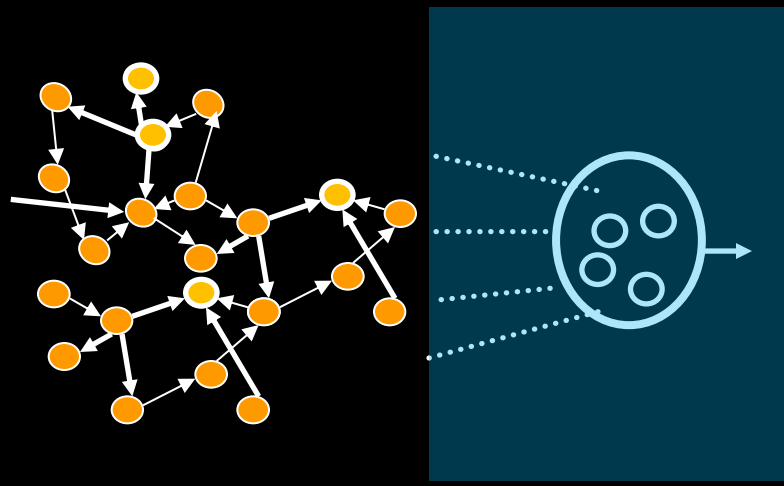
Separation property in practice

- We define a linear separation property of a circuit C. To evalute this linear separation property we form different inputs $u^1, \ldots, u^m$ (which are functions of time, i.e. input streams such as for example multiple spike trains) and compute the rank of the n×m matrix M whose columns are the reservoir states $x(t_0)$ resulting at some fixed time t_0 for the preceding input stream u^i.
- If this matrix has rank m, then it is guaranteed that any given assignment of target outputs $y_i \in R$ at time t_0 for the inputs u_i can be implemented by any trained a linear readout.
- If the rank of this matrix M has a value $r < m$, then this value r can still be viewed as a measure for the computational performance of this circuit C, since r is the number of “degrees of freedom” that a linear readout has in assigning target outputs y_i to these inputs u_i

Legenstein, R., & Maass, W. (2006). Edge of Chaos and Prediction of Computational Performance for Neural Circuit Models *Theoretical Computer Science*, 1–36.

Separation property

Approximation property
addresses the resolution and recoding capabilities of the readout mechanisms – more precisely its capability to distinguish and transform different internal states of the liquid into given target outputs



$$y(t) = f^M(x^M(t))$$

Mainly determined by the adaptability of the readout mechanism to the required task

Questionnaire: Q1

4. Tick off correct statements.

(a) (6 pts) There might be more than ones correct statements.

- [x] The reservoir states are non-linear combinations of past and recent states, and inputs
- [x] non-linearity arises by the non-linear links between linear nodes
- [x] non-linear nodes implement non linear computation
- [x] the network mediates the non-linear combination of states and inputs
- [x] nodes are always memory free
- [x] the readout is memory free

✖ **correct**

✖ **wrong**

Thanks you

Formal definition of separation property

We say that a class CB of filters has the *point-wise separation property* with regard to input functions from U^n if for any two functions $u(\cdot), v(\cdot) \in U^n$ with $u(s) \neq v(s)$ for some $s \leq 0$ there exists some filter $B \in CB$ that separates $u(\cdot)$ and $v(\cdot)$, i.e., $(Bu)(0) \neq (Bv)(0)$. Note that it is *not* required that there exists a filter $B \in CB$ with $(Bu)(0) \neq (Bv)(0)$ for any two functions $u(\cdot), v(\cdot) \in U^n$ with $u(s) \neq v(s)$ for some $s \leq 0$. Simple examples for classes CB of filters that have this property are the class of all delay filters $u(\cdot) \mapsto u^{t_0}(\cdot)$ (for $t_0 \in \mathbf{R}$), the class of all linear filters with impulse responses of the form $h(t) = e^{-at}$ with $a > 0$, and the class of filters defined by standard models for dynamic synapses, see (Maass and Sontag, 2000). A liquid filter L^M of a LSM M is said to be *composed* of filters from CB if there are finitely many filters $B_1, \ldots, B_m$ in CB – to which we refer as basis filters in this context – so that $(L^M u)(t) = \big\langle (B_1 u)(t), \ldots, (B_m u)(t) \big\rangle$ for all $t \in \mathbf{R}$ and all input functions $u(\cdot)$ in U^n. In other words: the output of L^M for a particular input u is simply the vector of outputs given by these finitely many basis filters for this input u.

Formal definition of separation property

We say that a class CB of filters has the *point-wise separation property* with regard to input functions from U^n if for any two functions $u(\cdot), v(\cdot) \in U^n$ with $u(s) \neq v(s)$ for some $s \leq 0$ there exists some filter $B \in CB$ that separates $u(\cdot)$ and $v(\cdot)$, i.e., $(Bu)(0) \neq (Bv)(0)$. Note that it is *not* required that there exists a filter $B \in CB$ with $(Bu)(0) \neq (Bv)(0)$ for any two functions $u(\cdot), v(\cdot) \in U^n$ with $u(s) \neq v(s)$ for some $s \leq 0$. Simple examples for classes CB of filters that have this property are the class of all delay filters $u(\cdot) \mapsto u^{t_0}(\cdot)$ (for $t_0 \in \mathbf{R}$), the class of all linear filters with impulse responses of the form $h(t) = e^{-at}$ with $a > 0$, and the class of filters defined by standard models for dynamic synapses, see (Maass and Sontag, 2000). A liquid filter L^M of a LSM M is said to be *composed* of filters from CB if there are finitely many filters $B_1, \ldots, B_m$ in CB – to which we refer as basis filters in this context – so that $(L^M u)(t) = \big\langle (B_1 u)(t), \ldots, (B_m u)(t) \big\rangle$ for all $t \in \mathbf{R}$ and all input functions $u(\cdot)$ in U^n. In other words: the output of L^M for a particular input u is simply the vector of outputs given by these finitely many basis filters for this input u.