Computation in Networks of Neurons in the Brain

Wolfgang Maass

Institute for Theoretical Computer Science
Graz University of Technology, Austria

Content

1. Theoretical results on the computational power of networks of spiking neurons

2. Features of BNNs that may provide clues about the way how they compute and learn:

--large trial-to-trial variability

--diversity of neurons and synapses

--characteristic firing patterns of simultaneously recorded neurons

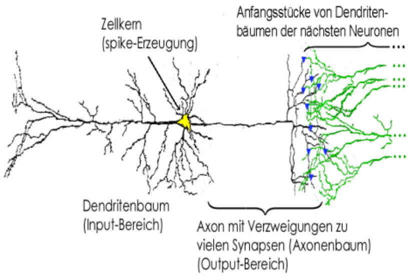
--unexpected properties of synaptic plasticity and neural codes

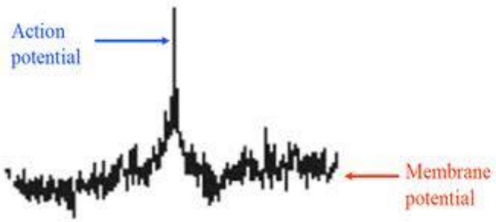
3. Methodological issues:
How should models for computation and learning in BNNs be chosen?

4. Summary

Color code: **red** for clues/constraints from experimental data
blue for open problems

The output of neurons consists of action potentials or „spikes“ in continuous time

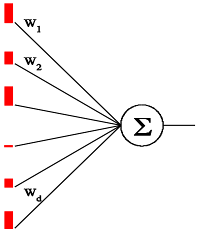

Zellkern
(spike-Erzeugung)
Anfangsstücke von Dendriten-
bäumen der nächsten Neuronen
Dendritenbaum
(Input-Bereich)
Axon mit Verzweigungen zu
vielen Synapsen (Axonenbaum)
(Output-Bereich)



A spike of the presynaptic neuron causes (with a certain probability) a temporary change of the postsynaptic membrane potential (EPSP or IPSP).

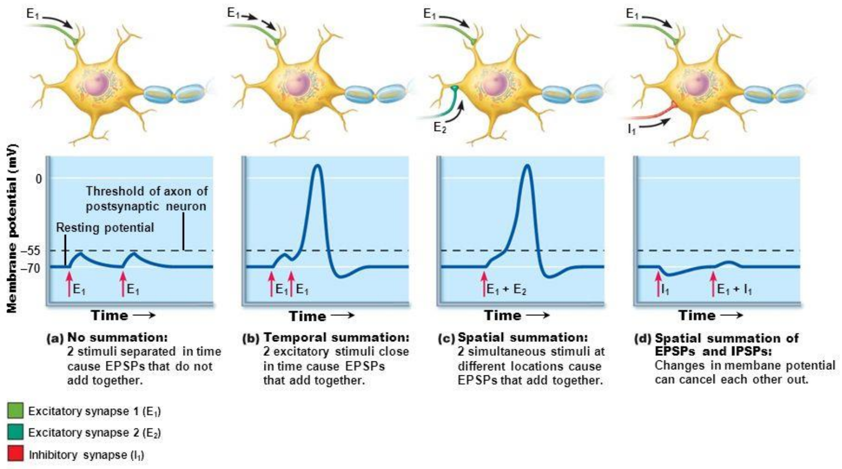
A simple neuron model:

The amplitude of EPSPs and IPSPs is scaled by synaptic weights w, and added linearly to model the membrane voltage.



The neurons emits a spike whenever its membrane potential reaches its „firing threshold“.

Figure 11.19 Neural integration of EPSPs and IPSPs.


E_1
E_2
I_1
Membrane potential (mV)
0
–55
–70
Threshold of axon of postsynaptic neuron
Resting potential
E_1
E_1 E_1
$E_1 + E_2$
I_1
$E_1 + I_1$
Time
🟩 Excitatory synapse 1 (E_1)
🟩 Excitatory synapse 2 (E_2)
🟥 Inhibitory synapse (I_1)

(a) No summation: 2 stimuli separated in time cause EPSPs that do not add together.

(b) Temporal summation: 2 excitatory stimuli close in time cause EPSPs that add together.

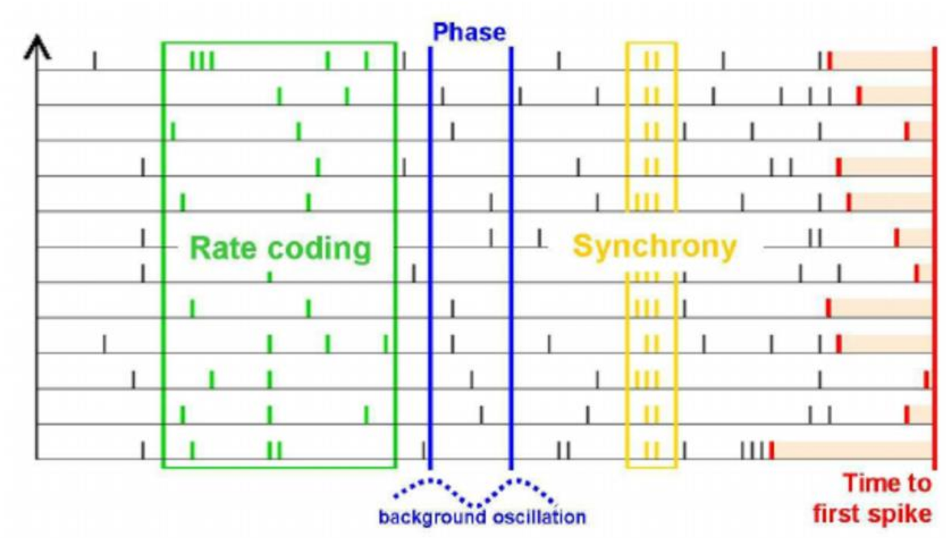
(c) Spatial summation: 2 simultaneous stimuli at different locations cause EPSPs that add together.

(d) Spatial summation of EPSPs and IPSPs: Changes in membane potential can cancel each other out.

How can one encode information with spikes?

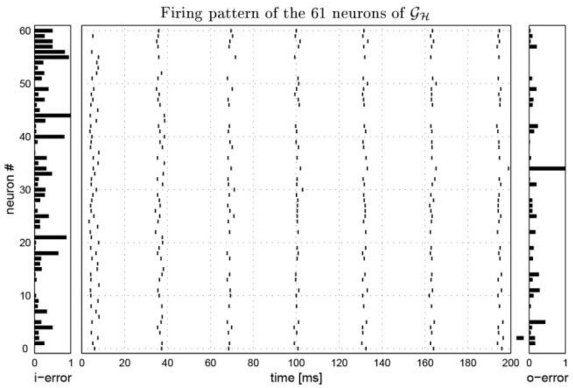
Different ways of encoding information with spikes are considered in theoretical models:

- Rate coding
- Firing time with regard to a reference signal (e.g. a background oscillation)
- Simultaneous firing of several neurons as a token of information
- Relative timing of spikes (the neuron that fires first conveys a more important signal)

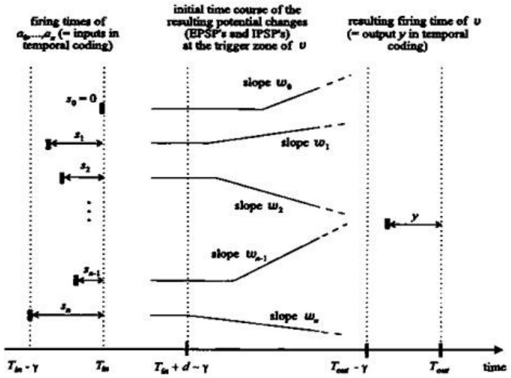


What is the theoretical computational power of a network of spiking neurons (SNN)?

One can construct a SNN that can simulate any **Turing machine** in real time, hence arguably any digital computation. This can be done with phase coding of spikes, where the time of a spike relative to a background oscillation encodes an analog (rational) number (e.g., the content of a Turing machine tape). (M., 1996)



With the same methods one can also show that any artificial neural network (ANN) with piecewise linear activation functions can be simulated in real time by a SNN (M., 1997)


firing times of $a_0,\dots,a_n$ (= inputs in temporal coding)
initial time course of the resulting potential changes (EPSP's and IPSP's) at the trigger zone of v
resulting firing time of v (= output y in temporal coding)
$s_0 = 0$
s_1
s_2
s_{n-1}
s_n
slope w_0
slope w_1
slope w_2
slope w_{n-1}
slope w_n
y
$T_{in} - \gamma$
T_{in}
$T_{in} + d - \gamma$
$T_{out} - \gamma$
T_{out}
time

These proofs assume that EPSPs and IPSPs contain some linear initial segment, and that there is no noise.

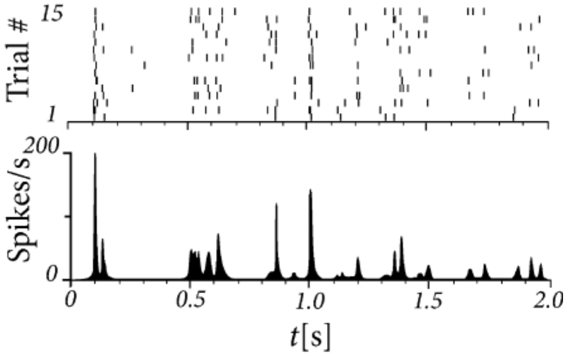
What happens if one allows noise in the SNN?

The computational power of SNNs drops to that of a deterministic finite automaton (M. and Orponen, 1997).

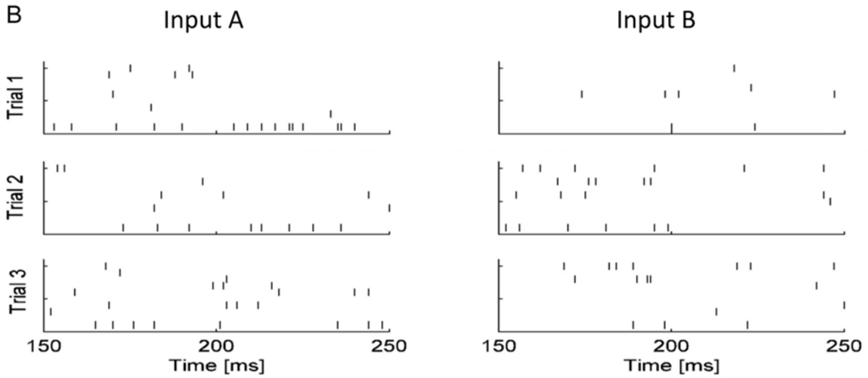
Is there noise in BNNs?

Recordings from BNNs show a large trial-to-trial variability

Recording of a single neuron in area MT during repeated presentations of the same stimulus
(randomly moving dots; data from Newsome Lab)



Simultaneous recording of several neurons during several trials
(static visual input patterns, data from Singer Lab)



A theoretical computer scientist might suggest that noise is not really a problem for computations in BNNs, since the brain just has to duplicate neurons (they all need to get the same input) and take the sum of their outputs.

But: Data on neural connectivity suggest that even „adjacent“ pyramidal cells have quite different sets of neurons from which they receive synaptic inputs, see e.g. (Yoshimura, Dantzker, Callaway. *Science* 2004).

In addition we have learnt that hardly any two neurons in the brain have the same inout/output behaviour.

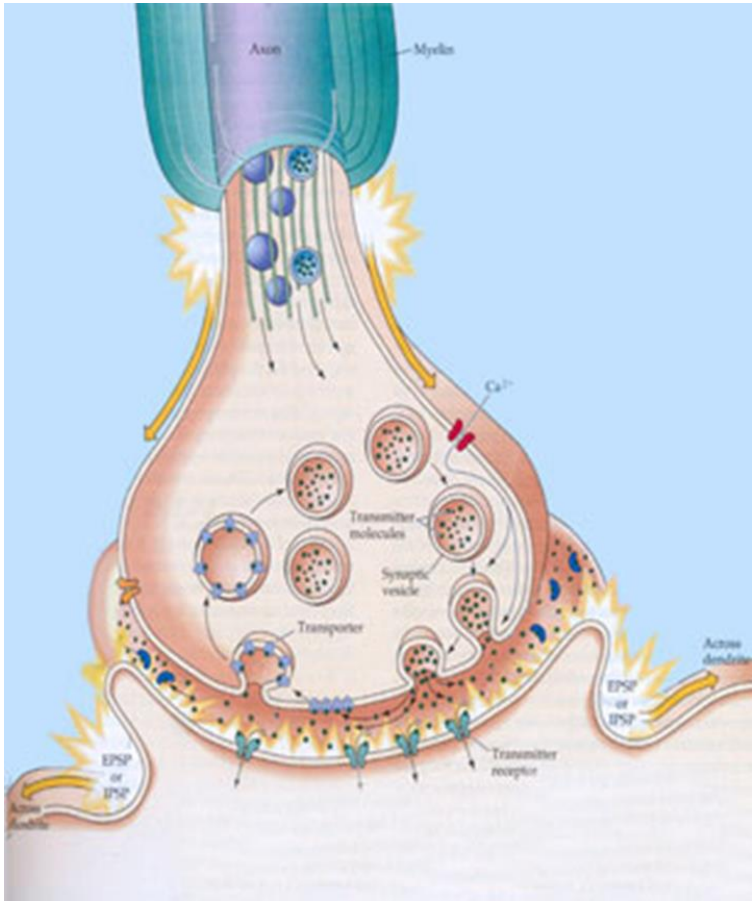
In fact, experimental data suggest that synapses in BNNs are designed to create noise

Common estimates of the probability of vesicle release in response to a presynaptic spike are around **0.5** for neocortex, see e.g.

(Branco, Staras, Nat. Rev. in Neurosci, 2009)

In addition vesicles are frequently released without a presynaptic spike

(Kavalali, Nat. Rev. in Neurosci., 2015).



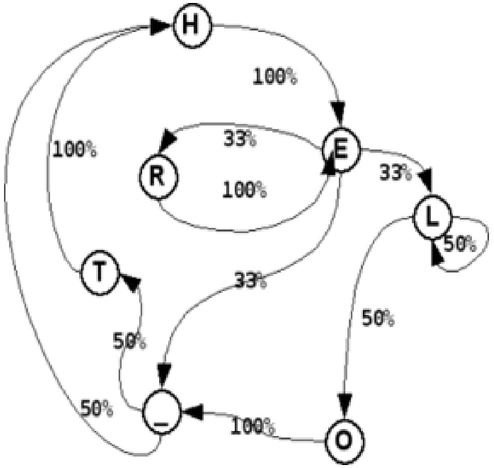
How can BNNs compute,
given that they have a large trial-to-trial variability?

It would be very difficult to emulate **deterministic** computational models by such networks. Are othert computing paradigms consistent with large trial-to-trial variability?

A prominent example: **Markov chains (MCs).**

A key property of MCs:

Under some mild assumptions they have a **unique stationary distribution** of network states, to which they converge from any initial state.



One can argue that MCs are superior to deterministic computing devices for computations such as

- exploration for unsupervised and reward-based learning
- probabilistic inference through sampling: Markov chain Monte Carlo (MCMC) sampling
- solving constraint satisfaction problems

A frequently studied MC in the context of ANNs: Boltzmann machines (BMs)

- BMs are commonly used in machine learning (e.g. for „deep learning“) and for solving constraint satisfaction problems
- BMs are stochastic artificial neural networks, whose units output 1 or 0, with stochastic switches according to some global schedule:

  When unit i is allowed to switch, it assumes the state

  $x_i = 1 \, with \, probability \, \sigma \, (\frac{1}{T}(\sum_j w_{ij} x_j + \, b_i))$, else $x_i = 0$

  with the sigmoid activation function $\sigma(x) = 1/(1 + e^{-x})$

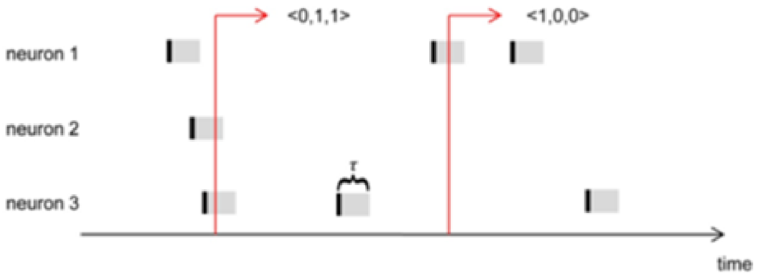
- The state of a BM with N units is at any time point some bit vector of length N

.

- Every Boltzmann distribution (i.e., distribution over binary random variables with at most 2nd order dependencies) is the stationary distribution of some BM.
- With „hidden units“ a BM can even represent distributions with higher order dependencies as marginal distributions.
- The stochastic dynamics of BMs is equivalent to Gibbs sampling, which is frequently used for probabilistic inference via MCMC sampling in ML

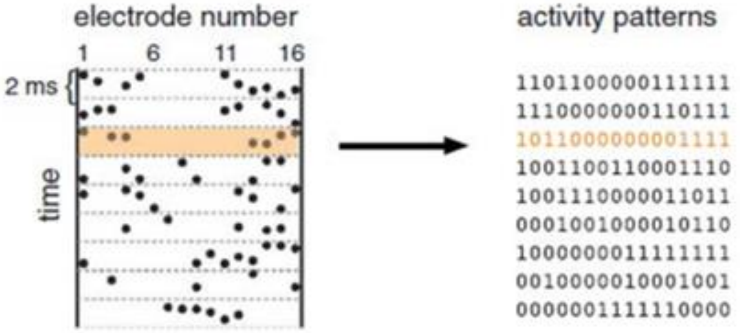
Relating spiking neurons to BM-units

Assume that each spike of a neuron sets some bit to 1 for a short time window:



Then the firing activity of a network of N spiking neurons can be encoded at any time point by a bit vector z of length N (like the state of a BM with N units).

Such transformation from spikes to bits is also considered by experimentalists, e.g. (Berkes et al., Science 2011). They proposed to view neural activity as sampling from a distribution that is encoded by the network:



```
1101100000111111
1110000000110111
1011000000001111
1001100110001110
1001110000011011
0001001000010110
1000000011111111
0010000010001001
0000001111110000
```

Note: Tenenbaum and other cognitive scientists had already earlier proposed that the **brain stores information** in the form of **probability distributions.**

If networks of N spiking neurons can be viewed as MCs, whose states z are bit vectors of length N, the question arises: *Which distributions over these states can be represented, or even learnt, by such SNNs?*

Known results:

1. When the weights are symmetric, SNNs (with an idealized neuron model) represent the same distribution as a BMs of the same size and with the same weights. Hence SNNs can represent all distributions with at most 2nd order dependencies. (Buesing et al., 2011)

1. With asymmetric weights, there is no a-priori limit for the type of distribution over discrete random variables which SNNs can represent. They also can learn quite arbitrary distribution from examples, using an idealized form of STDP (Pecevski et al., 2011, 2016).

These theoretical results hold **rigorously** for stochastic spiking neurons with an instantaneous firing probability

$$\rho_k(t) = \frac{1}{\tau} exp(u_k(t))$$

for a standard definition of the membrane potential

$$u_k(t) = \sum_{i=1}^{n} w_{ki}\tilde{y}_i(t) + w_{k0}$$

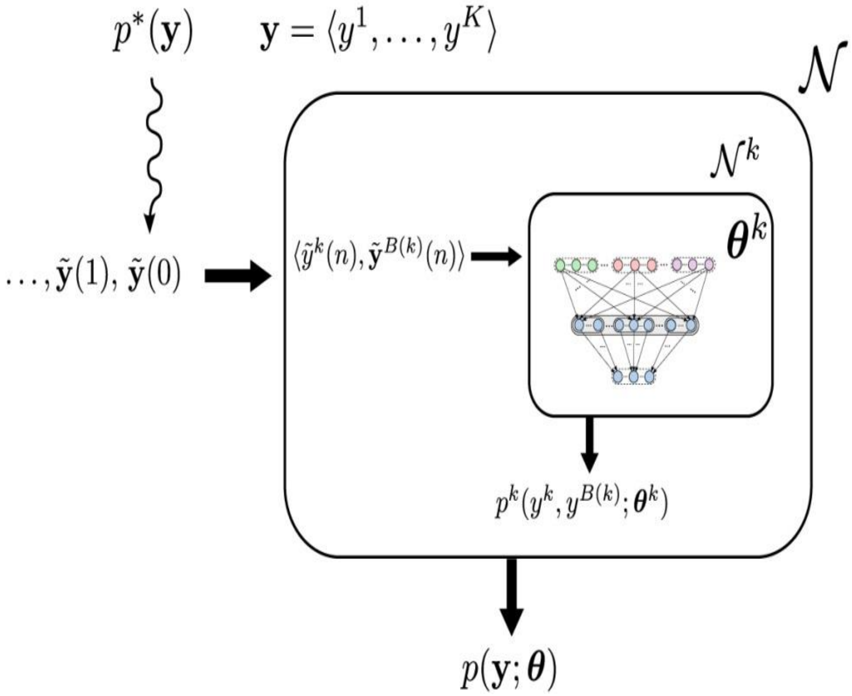
with simplified „rectangular“ EPSPs defined by step functions.

Method for unsupervised learning of an approximation to a given distribution over K binary variables through STDP in stochastic SNNs

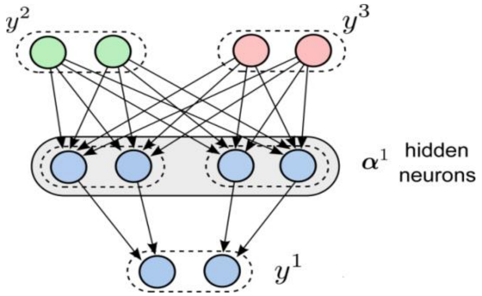
(Pecevski et al., 2016)

Assumption: Some external distribution p* generates samples **y,** given to the SNN through population coding.

Result: The network learns through STDP an internal model of p*, from which it can spontaneously produce samples through its stochastic dynamics.


$p^*(\mathbf{y})$
$\mathbf{y} = \langle y^1, \ldots, y^K \rangle$
$\mathcal{N}$
$\mathcal{N}^k$
$\boldsymbol{\theta}^k$
$\ldots, \tilde{\mathbf{y}}(1), \tilde{\mathbf{y}}(0)$
$\langle \tilde{y}^k(n), \tilde{\mathbf{y}}^{B(k)}(n) \rangle$
$p^k(y^k, y^{B(k)}; \boldsymbol{\theta}^k)$
$p(\mathbf{y}; \boldsymbol{\theta})$

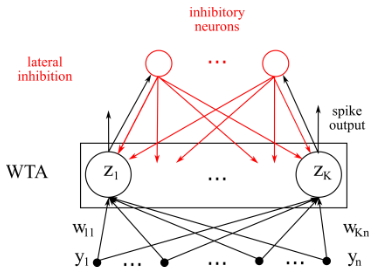
The network consists of 3-layer moduls, that each learn the probability table for one RV, conditioned on the RVs in its Markov blanket:

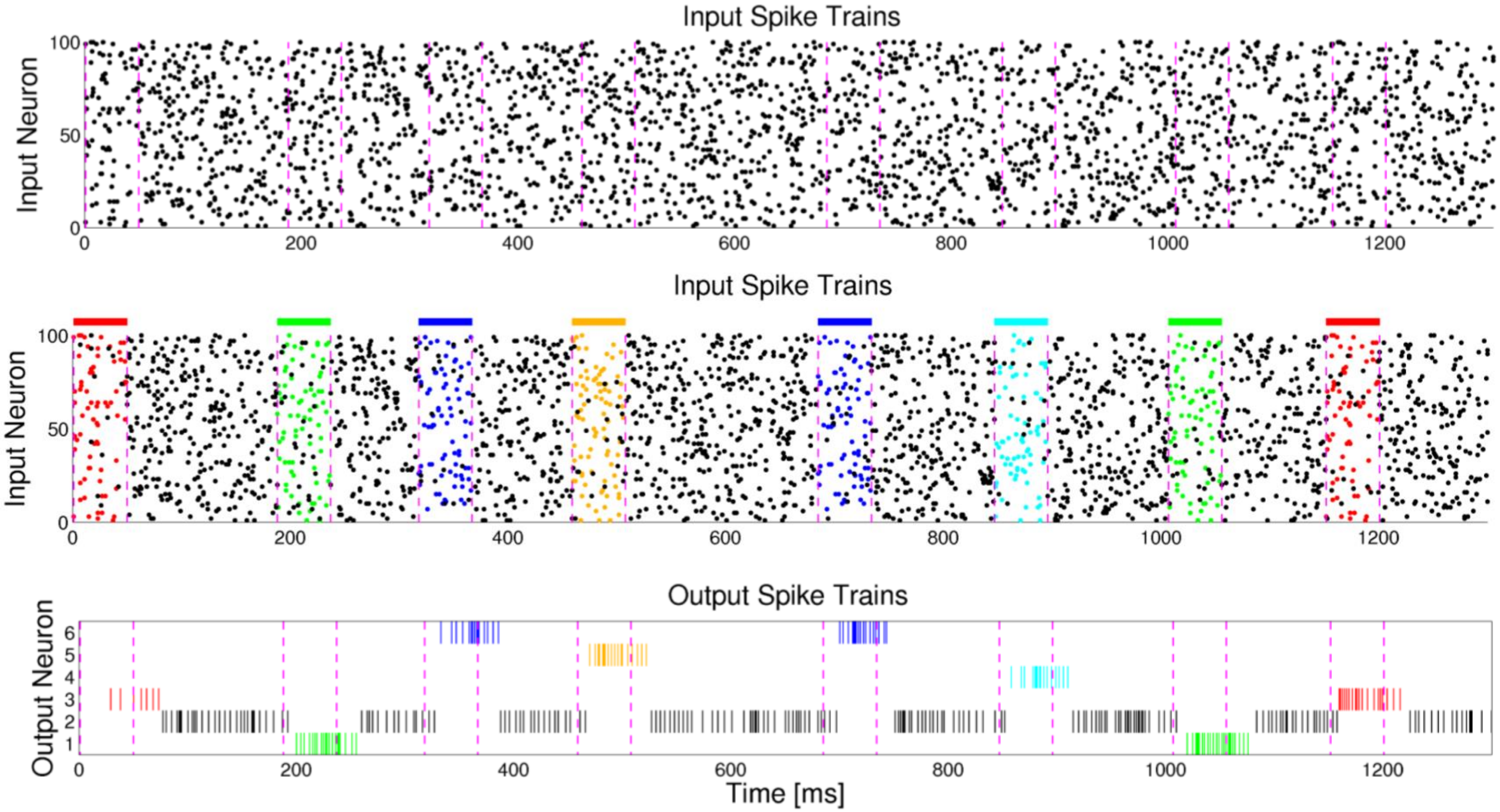


Learning takes place through STDP on synapses to hidden layer neurons, that are partitioned into WTA (winner-take-all) circuits.

Underlying theory: Expectation Maximization (EM), using stochastic sampling for the E-step.

This result makes use of an inherent unsupervised learning capability of WTA circuits (long known as „competitive Hebbian learning in ANNs), realized with STDP by SNNs



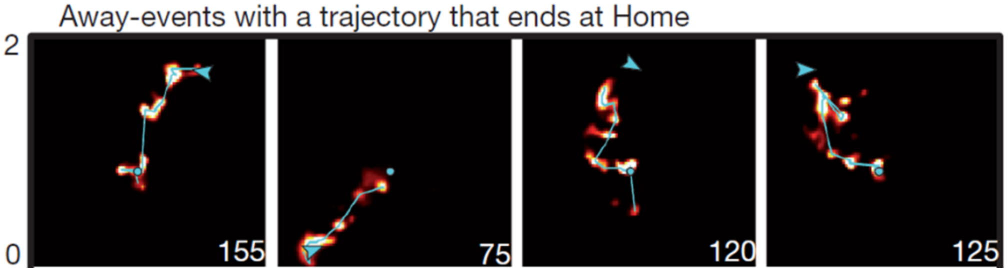


Theory can explain this process as fitting a mixture distribution to incoming spike streams via stochastic online EM (E-step realized via sampling)

Another possible functional role of stochastic computations of BNNs: Problem solving and planning, or abstractly: Solving Constraint Satisfaction Problems (CSPs)

These experimental data from rodents are hypothesized to reflect planning of a path to a „home“ location: through trajectories of network states in area CA1 during brief periods of rest. They are represented in this figure as sequences of decoded „places“ within an open 2x2m arena. Average duration of the trajectory: 103 ms.

Home location indicated by cyan circle. Cyan arrowhead indicates position and head direction at the time of the event



B. E. Pfeiffer and D. J. Foster. Hippocampal place-cell sequences depict future paths to remembered goals. Nature, 2013.

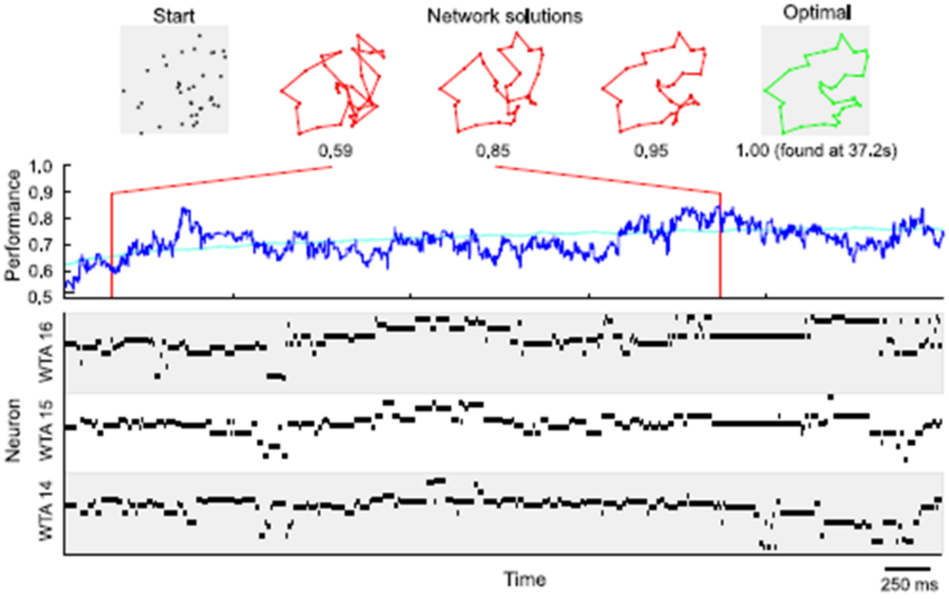
Actually, one can argue that the brain has to solve CSPs all the time, also just for sensory processing.

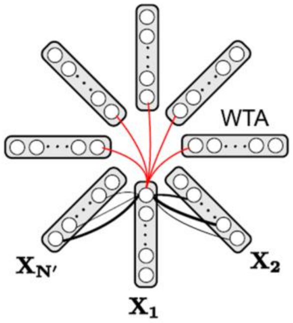
Open problem: How can BNNs solve CSPs?

This problem is of particular interest for Computer Science, since many CSPs are computationally hard.

Idealized SNNs can solve a hard CSP through sampling: Traveling Salesman Problem

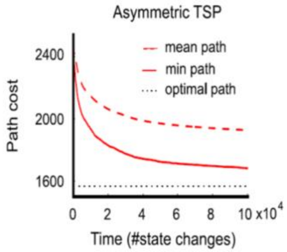
One WTA is used for each step of the trip, where the spiking neurons in each WTA circuit represent all the different „cities“:



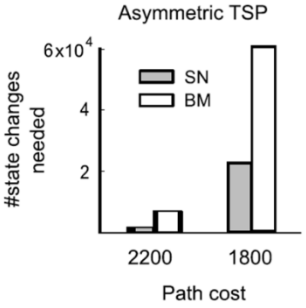


Results shown are for an instance with 38 cities.

Statistics of solution qualities found within some time span:



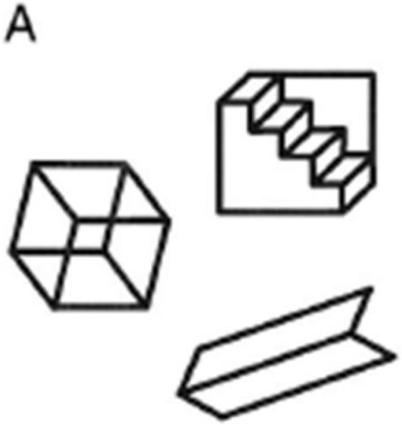
Number of state changes needed by SNNs and BMs to produce a solution of a given mox. cost:

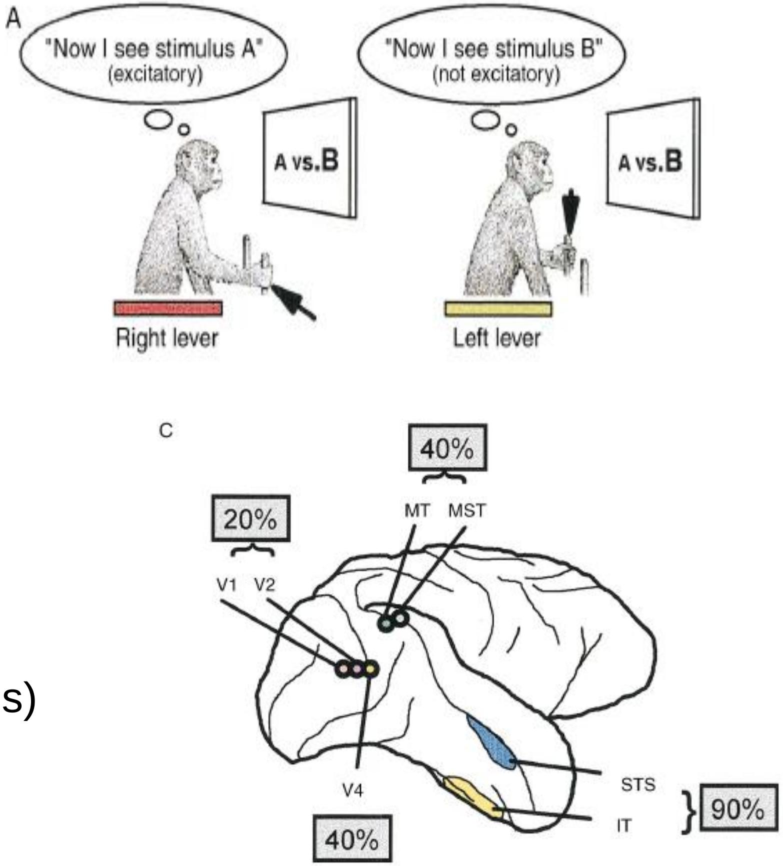


A strange feature is that stochastic SNNs tend to solve this problem faster than BMs that have the same stationary distribution.

(Zeno, Habenschuss, M., 2016)

Experimental evidence for sampling in the cortex on the behavioural time scale of seconds (and in terms of firing rates): Perception of ambiguous sensory stimuli



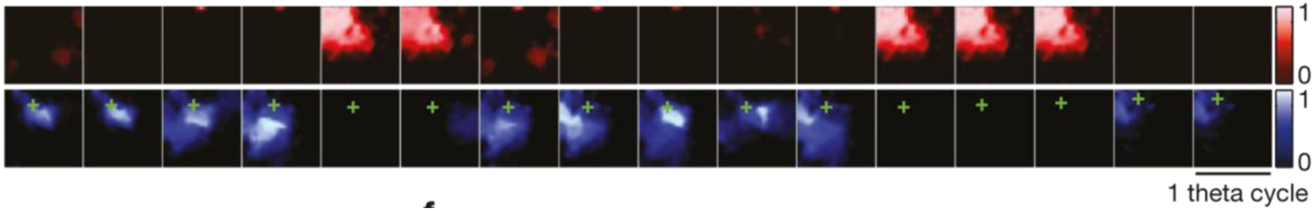


Both the perception of the stimulus and brain states (defined by neural firing rates) switch on a time scale of seconds.

Leopold, Logothetis, Trends in Cogn. Sci., 1999

Evidence for sampling in the hippocampus on the time scale of about 125ms per sample

The rodent was uncertain whether it currently was in the „blue“ or „red“ environment. Its place-cell map switched abruptly between the maps for the two environments (each indicated by red or blue color) during transitions between cycles of the theta rhythm (about 8 Hz), rather than forming a temporally stable fusion of the two maps:



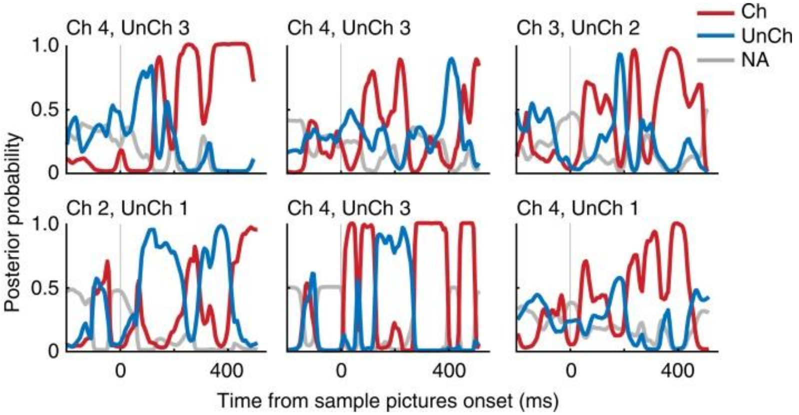
Jezek, K., Henriksen, E. J., Treves, A., Moser, E. I., & Moser, M. B. (2011). Theta-paced flickering between place-cell maps in the hippocampus. *Nature*, *478*(7368), 246-249.

Experimental evidence for sampling in the orbitrofrontal cortex (less than 100ms per “sample”)

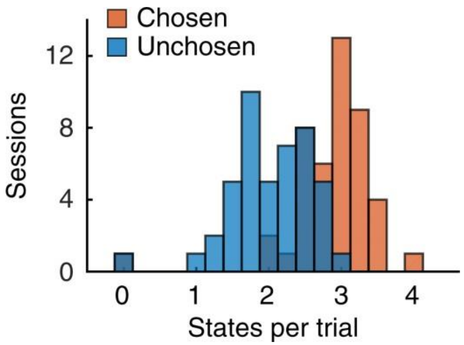
Rich, Wallis, Nature Neuroscience 2016

Monkeys could choose between two different rewards.

Neural networks in the orbitro-frontal cortex (OFC) switch during the choice period between different network states z that represent the two currently available options



The network state z that corresponds to the subsequently chosen option usually occurs with higher probability, as predicted by the previously sketched model for problem solving in the brain through sampling.



Further reading

Links between models which propose that stochastic BNNs carry out probabilistic inference through sampling and experimental data:

Orbán, G., Berkes, P., Fiser, J., & Lengyel, M. (2016). Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron*, *92*(2), 530-543.

An alternative model for probabilistic inference through sampling in BNNs:

Aitchison, L., & Lengyel, M. (2016). The Hamiltonian brain: efficient probabilistic inference with excitatory-inhibitory neural circuit dynamics. *PLoS computational biology*, *12*(12), e1005186.

Review of models for probabilistic inference in BNNs that do not rely on sampling:

Lochmann, T., & Deneve, S. (2011). Neural processing as causal inference. *Current Opinion in Neurobiology*, *21*, 774-781.

Open problems with regard to stochastic computations of BNNs

- Have BNNs other methods besides stochastic sampling for solving CSPs?

- Have BNNs other methods besides stochastic sampling for probabilistic inference?

- How can the brain make use of potential solutions to a CSP that are produced through sampling? E.g., how can it recognize the relative quality of a stochastically produced solution, and return to a very good previously produced solution?

- How can BNNs separate different samples in time? One possibility: each sample is produced in a separate cycle of a background brain rhythm.

- Current models for stochastic computations in NNs are based on homogeneous models for neurons and synapses. Are there also models for stochastic NN computations that are compatible with, or even benefit from, a diversity of neuron- and synapse-types?

- Can we produce a theory of stochastic computation in BNNs where salient random variables are represented by **assemblies** of neurons, rather than by single neurons?

Large trial-to-trial variability is just one salient clue/constraint for modelling computations in BNNs

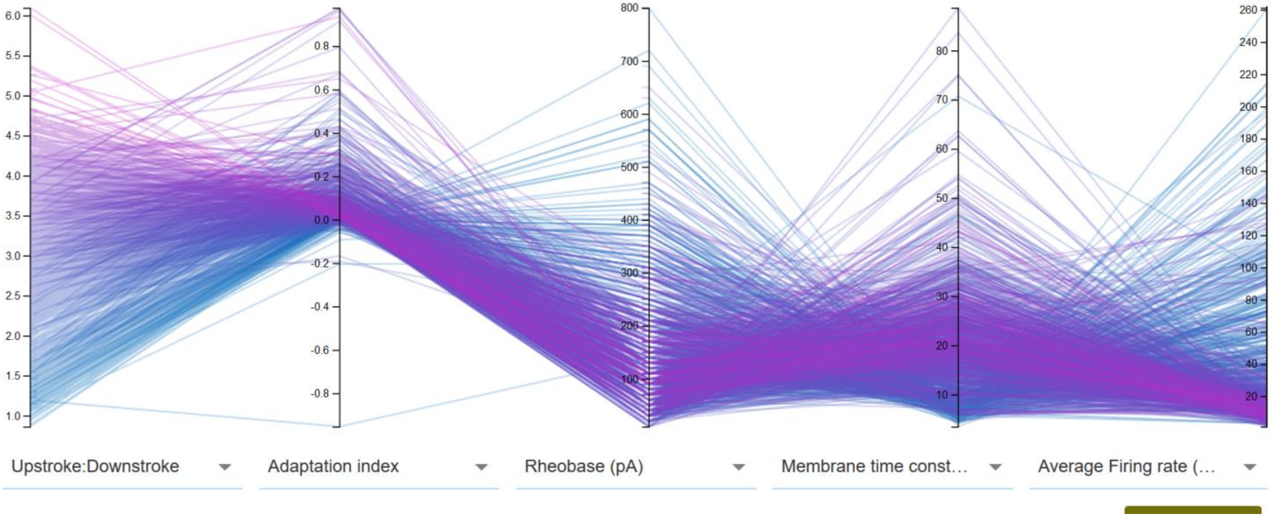
I will now discuss 4 further features of BNNs:

- **diversity of neurons and synapses**
- **generic firing patterns recorded in simultaneous recordings from many neurons**
- **surprising aspects of synaptic plasticity**
- **drifting neural codes**

I will show that these 4 features suggest that BNNs use strategies for computation and learning that are quite different from those which are familiar from ANNs or Computer Science.

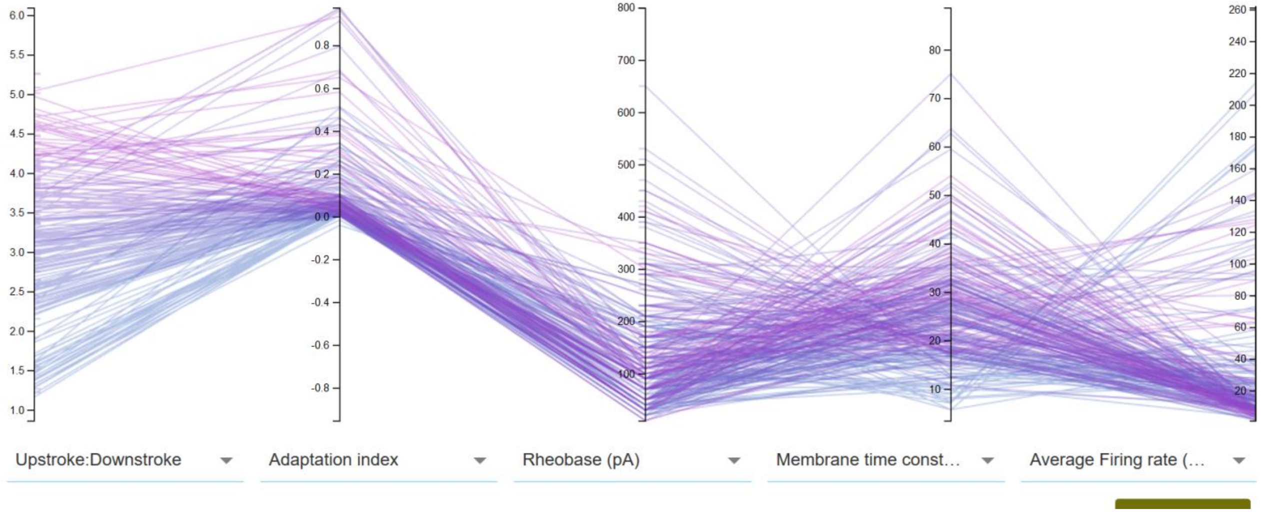
Data from the Allen Brain Atlas suggest a large diversity of parameters of neurons, especially in humans

Mouse



Upstroke:Downstroke describes the form of spikes,
Adaptation index describes descrease in firing rate for fixed input current
Rheobase is the minimial current needed to make the neuron fire

human



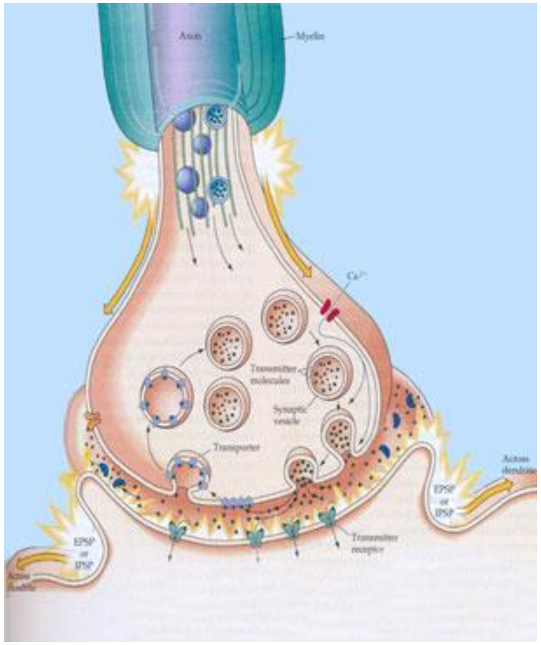
Another source of diversity in BNNs:
Diversity of short-term plasticity of synapses

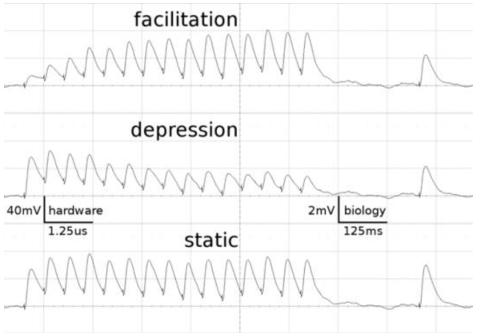
Every synapse has a complex inherent temporal dynamics and can **NOT** be modeled by a single parameter w like in artificial neural networks.

A standard model from
[Markram, Wang, Tsodyks, PNAS 1998]
Models short-term plasticity with 3 parameters
U, D, F:

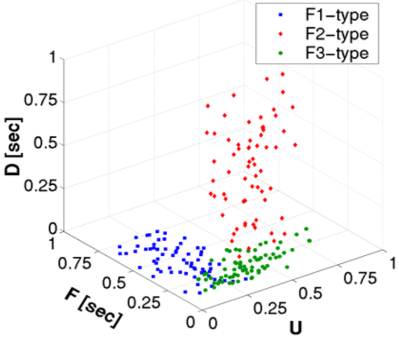
The amplitude A_k of the postsynaptic potential for the k^{th} spike in a spike train with inter-spike intervals $\Delta_1, \Delta_2, \ldots, \Delta_{k-1}$ is modeled by the equations

$$A_k = w \cdot u_k \cdot R_k$$
$$u_k = U + u_{k-1} (1-U) \exp(- \Delta_{k-1} /F)$$
$$R_k = 1 + (R_{k-1} - u_{k-1} R_{k-1} -1) \exp(- \Delta_{k-1} /D)$$



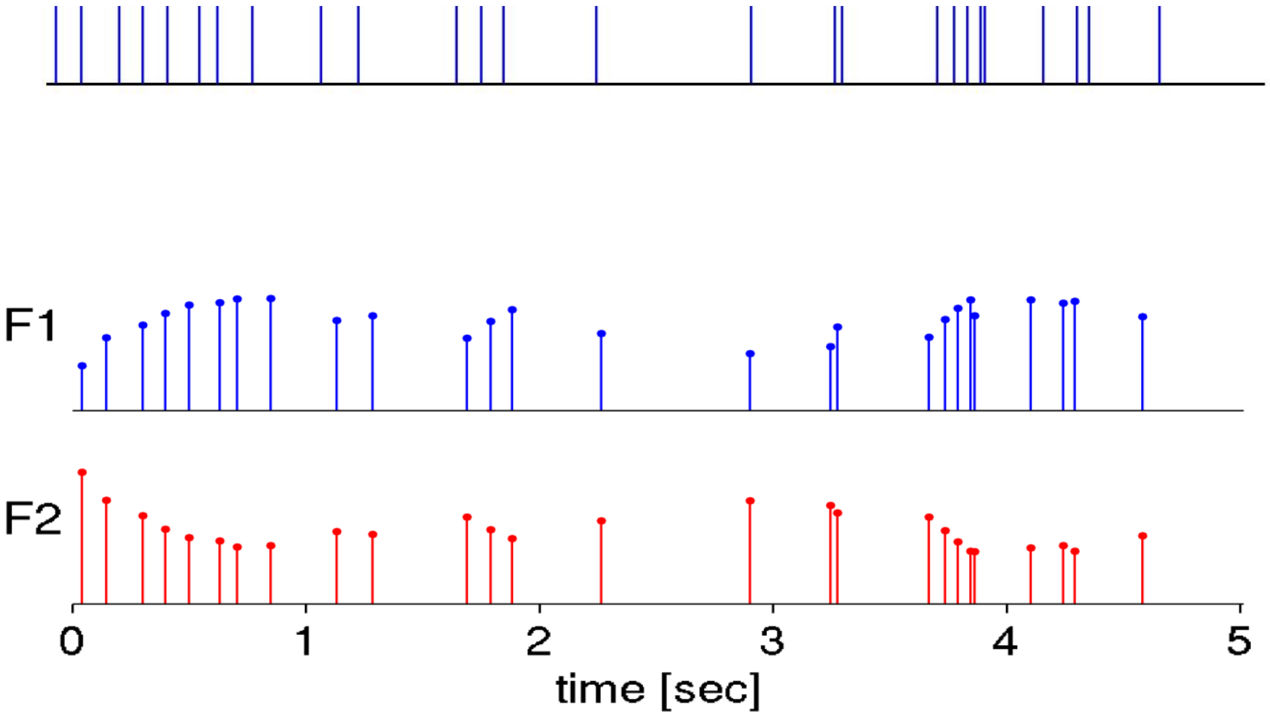


UDF values for different types of synaptic connections
(Gupta, Wang, Markram; Science, 2000):



Functional consequence of the inherent dynamics of synapses: A single neuron can send different messages to different target neurons

Shown are the amplitudes of synaptic responses of two common types of synapses to the same spike train (F1 is facilitating and F2 is depressing):



one spike train, sent to two synapses

output amplitudes of synapse

output amplitudes of synapse

Implications of these data for modelling and theory of computations in BNNs

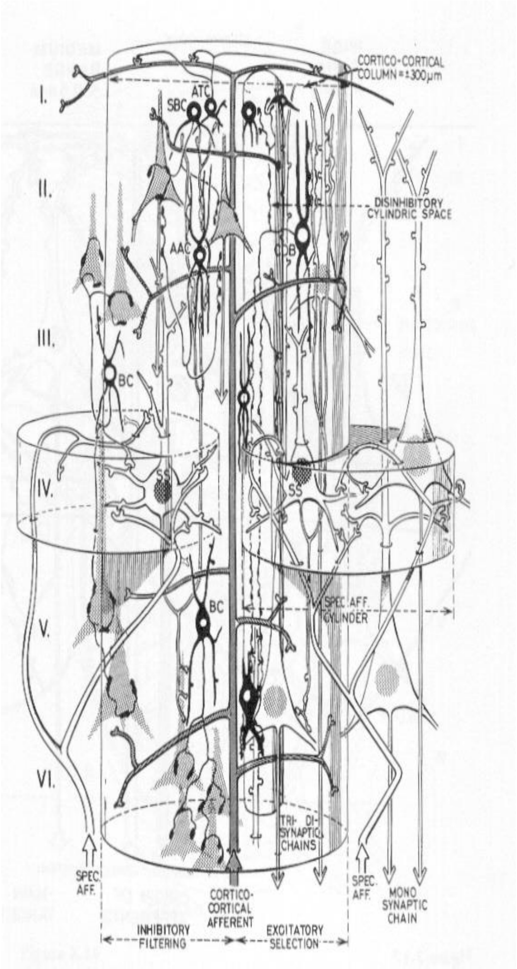
Theoretical models for BNNs usually contain just one or two types of neurons, and short-term plasticity of synapses is ignored.

Does this affect the computational paradigms that are supported by these models in a significant manner?

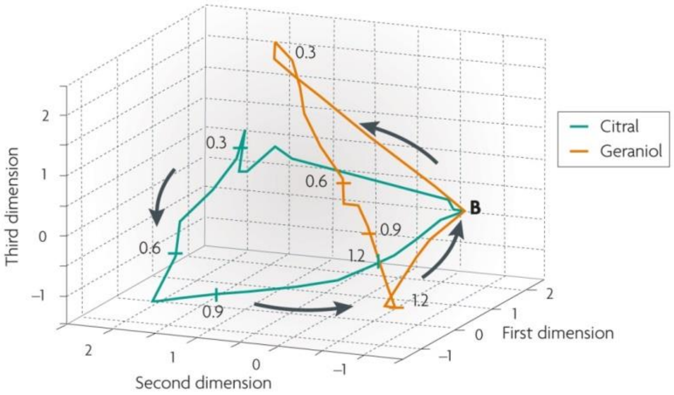
Yes: it becomes **close to impossible to simulate ANNs or other popular network architectures from Computer Science** by a model whose synapses are subject to **short-term plasticity** (because the output of a synapse depends in a complicated way on the history of its spike inputs).

Note that differences in the firing dynamics of excitatory and (various types of) inhibitory neurons also make it difficult, if not impossible, **to implement subtraction by combining outputs of an excitatory and an inhibitory neuron.**

Diversity of neurons and synapses in recurrent BNNs



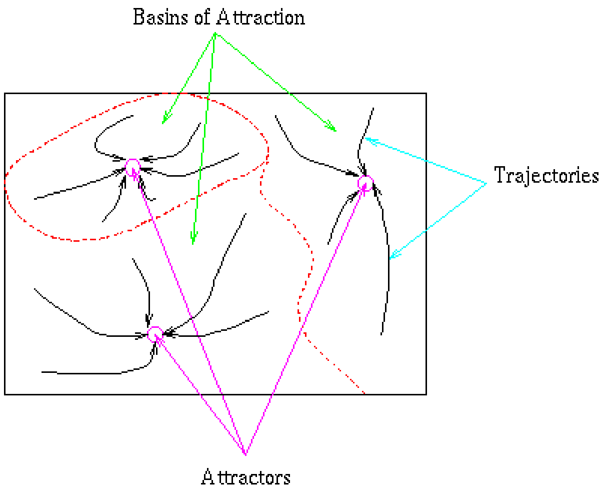
I will discuss models for computations of BNNs with diverse components on *time series*



These are the first 3 principal components of the firing activity of 87 neurons in the locust olfactory lobe for 2 different odors, numbers indicate time in seconds since stimulus onset (Broome, Jayaraman, Laurent, 2006)

Earlier computational models for BNNs as dynamical systems had focused on attractors.

But experimental data suggest that longer lasting attractor states are rare in BNNs, and most **computations** have to take place **on trajectories** of network states.



A nice class of computational operations (filters) on time series:

Filters F that are time-invariant (i.e, input driven) and only require a fading memory.

These are exactly those that can be represented by Volterra series:

$$(F\,u(\cdot))\,(t) \quad = \quad \alpha_1 \int_0^\infty d\,\tau_1\; h_1\,(\tau_1)\,u\,(t-\tau_1)$$

$$+\,\alpha_2 \int_0^\infty\!\!\int_0^\infty d\,\tau_1\; d\,\tau_2\; h_1\,(\tau_1,\tau_2)\,u\,(t-\tau_1)\cdot u\,(t-\tau_2)$$

$$+\ldots$$

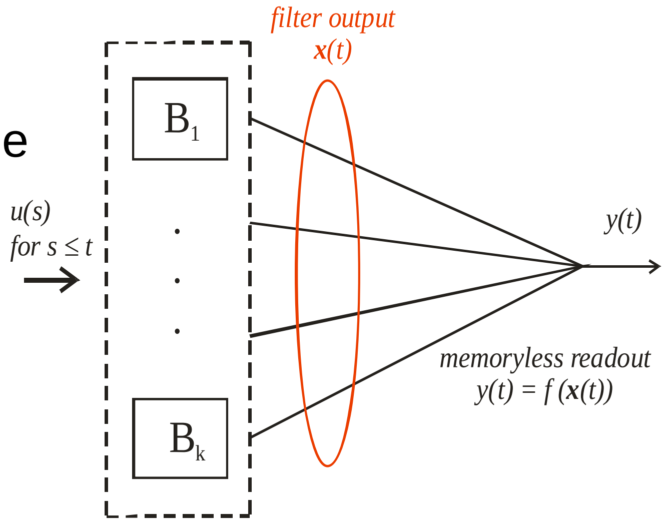
This is one of very few theoretical results which show that diversity of units may increase the computational capability of a network

(for computations on time series u(t))

Theorem (Boyd and Chua, 1985)

Any time-invariant filter with fading memory can be approximated with any degree of precision by this simple computational model, which only requires

- ***that*** there is a rich enough pool **B** of basis filters (time invariant, with fading memory) from which the basis filters $B_1, \ldots, B_k$ in the filterbank can be chosen (**B** needs to have the pointwise separation property) *and*
- ***that*** any continuous bounded function can be approximated by the memoryless readout.



Def: A class **B** of basis filters has the pointwise separation property if there exists for any two input functions $u(\bullet), v(\bullet)$ with $u(s) \neq v(s)$ for some $s \leq t$ a basis filter $B \in$ **B** with $(Bu)(t) \neq (Bv)(t)$.

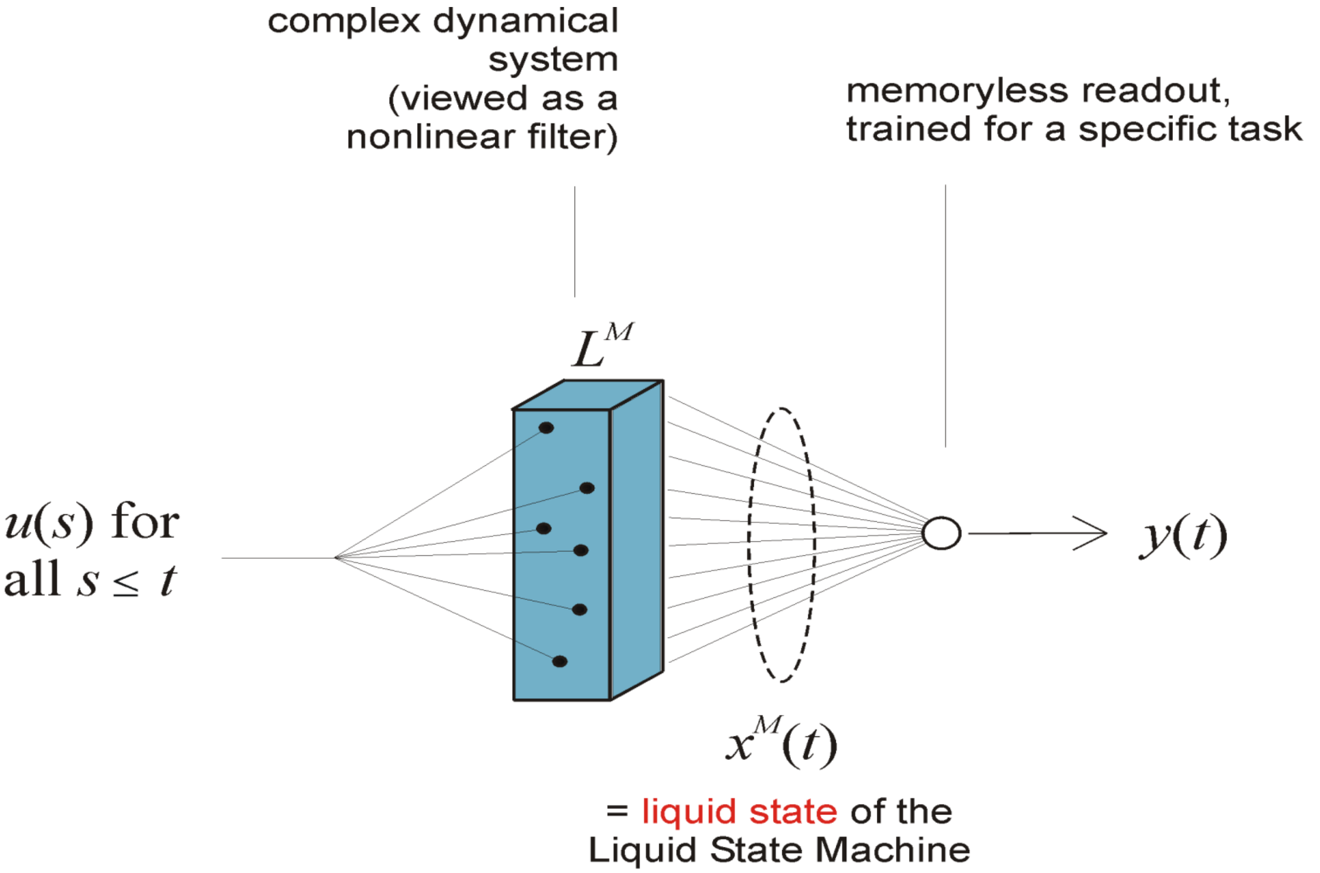
Open problem: Can theory provide further insight into computational benefits of having **diverse** computational units within some network?

E.g., how well can one approximate with a **fixed** set of randomly chosen basis filters all filters that a BNN needs to compute?

Abstract version of this computational model: Liquid State Machine

(“liquid” because the state of the dynamical system is allowed to be “liquid” rather than discrete and static, like in a finite state machine).

This model generalizes finite state machines to continuous input values u(s), continuous output values y(t), and continuous time t.



complex dynamical system (viewed as a nonlinear filter)

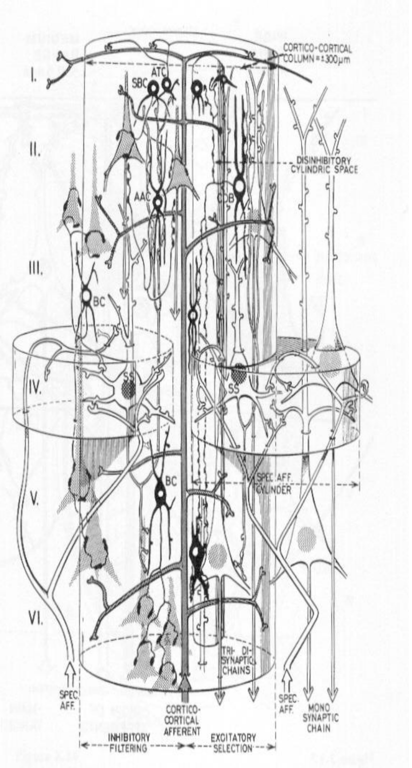
memoryless readout, trained for a specific task

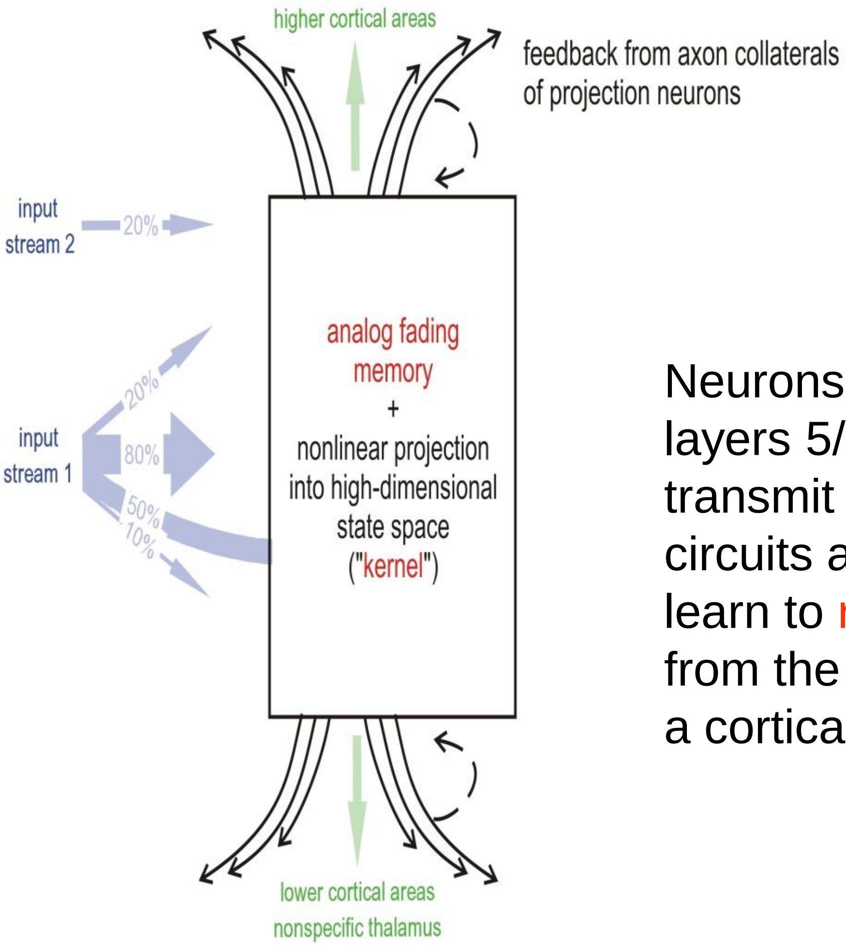
= liquid state of the Liquid State Machine

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

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

Resulting computational model for a cortical column





higher cortical areas

feedback from axon collaterals of projection neurons

input stream 2

20%

input stream 1

20%

80%

50%

10%

analog fading memory

+

nonlinear projection into high-dimensional state space ("kernel")

lower cortical areas

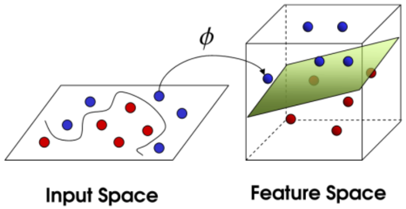
nonspecific thalamus

Neurons on layers 2/3 and layers 5/6 (whose axons transmit information to other circuits and brain areas) learn to read out information from the state trajectory of a cortical column

This model proposes that a cortical column is designed to carry out two generic computational operations on *sequences (filters):* To **integrate information over time (fading memory)**, and to **project information nonlinearly into a high-D state space.**

The computational power of linear readouts gets boosted through a generic nonlinear projection into a high-D space

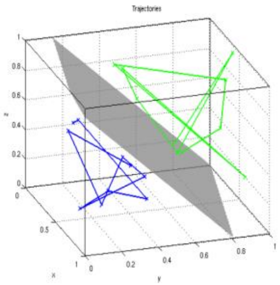
This principle is well-known from Machine Learning: kernels of Support Vector Machines



Example: *If a circuit precomputes all products $x_i \cdot x_j$ of n input variables $x_1,...,x_n$, then a subsequent linear readout can compute any quadratic function of the original input variables $x_1,...,x_n$.*

However this projection into high-D does not have to compute any concrete nonlinear operation: It suffices if saliently **different** inputs are mapped onto li**nearly independent** outputs.

This principle also supports separation of time series by linear readouts after a nonlinear projection into a high-D space:



Echo state networks (G. Jäger) use the same principle for ANNs. But for recurrent ANNs without noise one does not need diverse network units, randomly constructed recurrent connectivity and weights provides there enough diversity.

This models suggests that computations can be multiplexed in BNNs

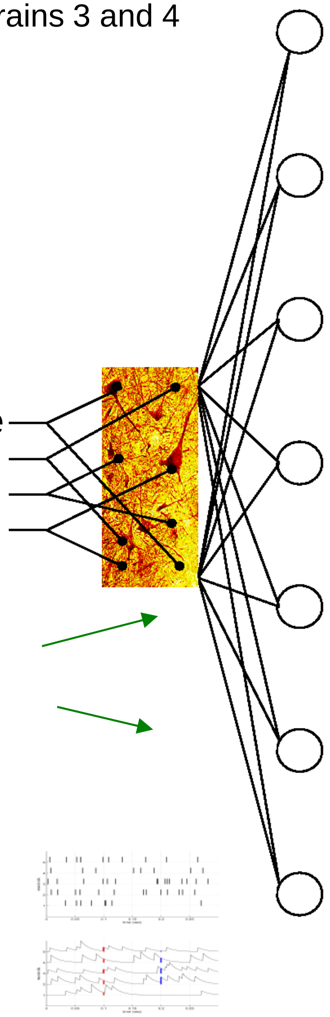
(M., Natschlaeger, Markram, 2002)

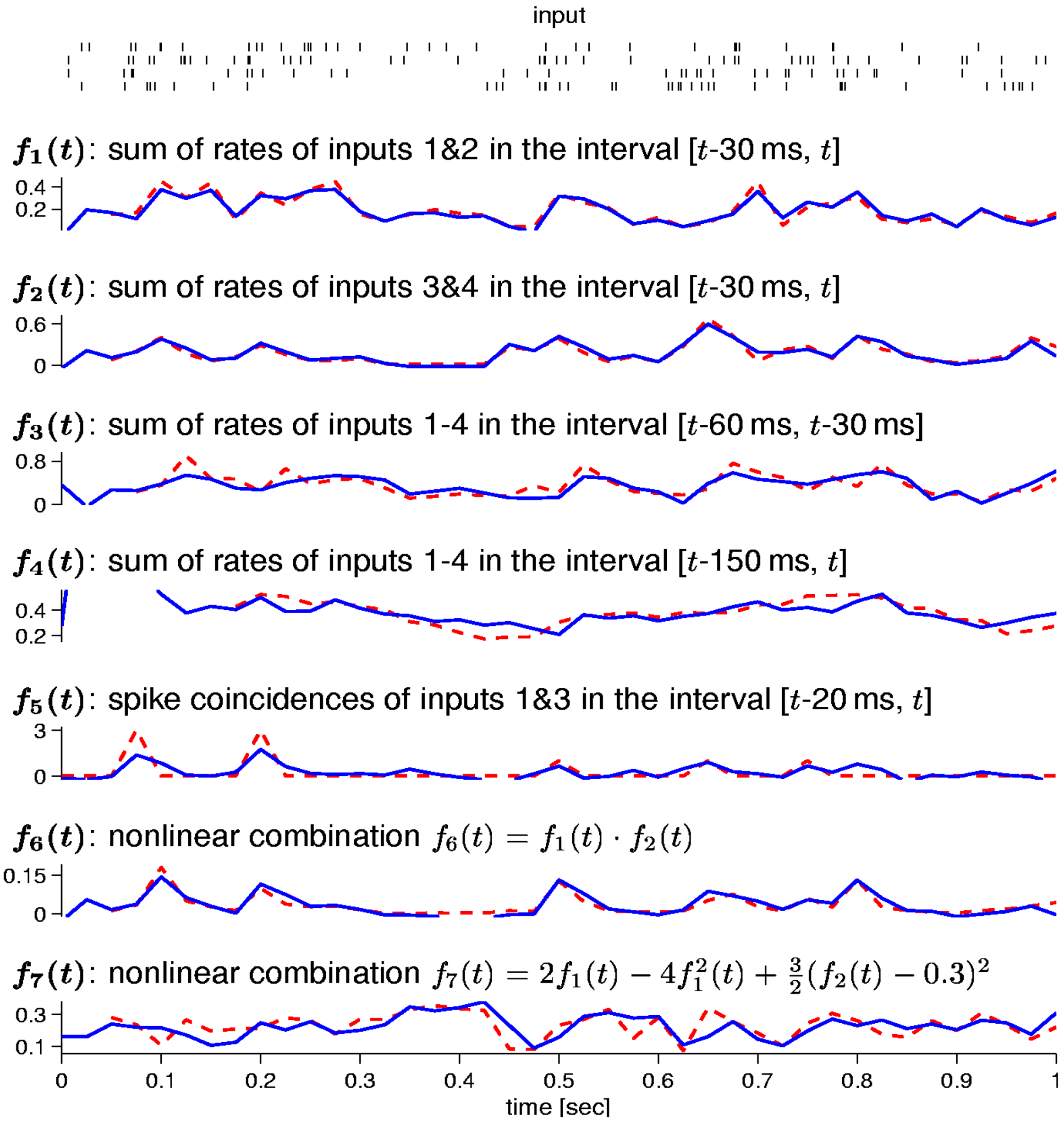
Network input:

4 Poisson spike trains with time-varying firing rates $f_1(t)$ for spike trains 1 and 2 and firing rates $f_2(t)$ for spike trains 3 and 4

Randomly connected network of 135 spiking neurons whose synapses have diverse short-term plasticity:

7 **linear** readouts, trained for 7 different tasks by linear regression (blue traces) receive EPSPs from the 135 network neurons





$f_1(t)$: sum of rates of inputs 1&2 in the interval [t-30 ms, t]

$f_2(t)$: sum of rates of inputs 3&4 in the interval [t-30 ms, t]

$f_3(t)$: sum of rates of inputs 1-4 in the interval [t-60 ms, t-30 ms]

$f_4(t)$: sum of rates of inputs 1-4 in the interval [t-150 ms, t]

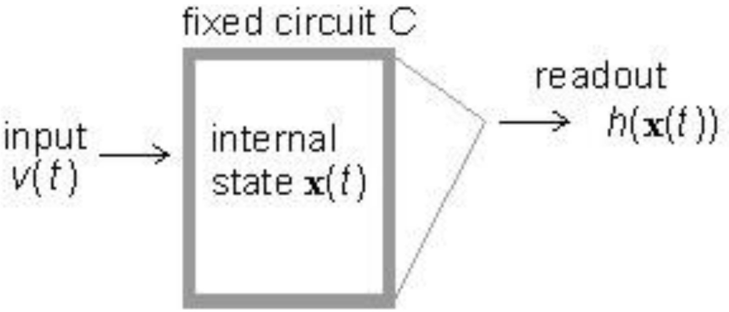
$f_5(t)$: spike coincidences of inputs 1&3 in the interval [t-20 ms, t]

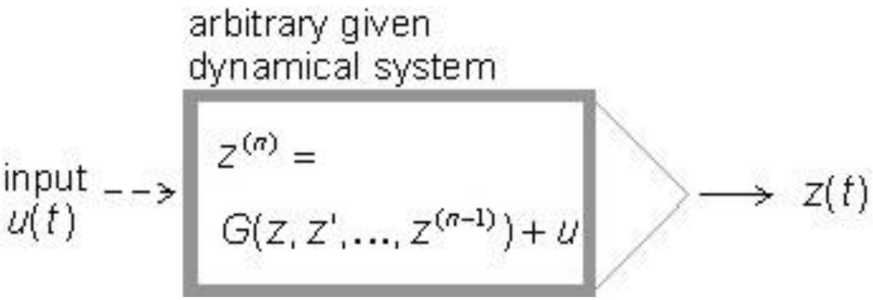
$f_6(t)$: nonlinear combination $f_6(t) = f_1(t) \cdot f_2(t)$

$f_7(t)$: nonlinear combination $f_7(t) = 2f_1(t) - 4f_1^2(t) + \frac{3}{2}(f_2(t) - 0.3)^2$

Theorem: Adding feedback from trained readout neurons drastically increases the computational power of this model (M., Joshi, Sontag, 2007):

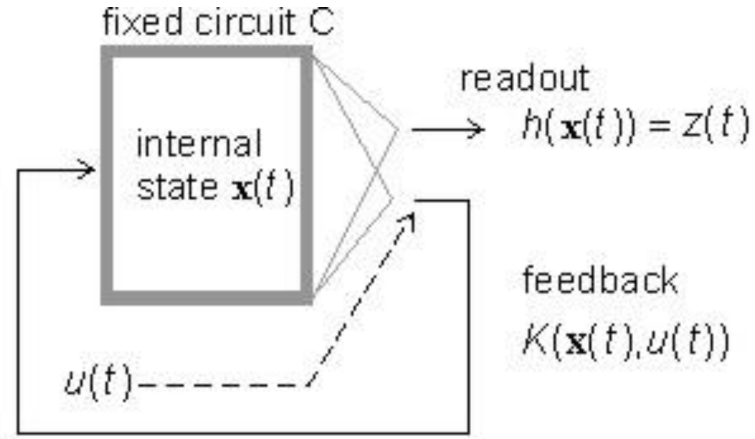
There exists a large class S_n of analog circuits *C* with fading memory (described by systems of *n* first order differential equations) that acquire through feedback universal computational capabilities for analog computing in the following sense:





Any Turing machine can be simulated by such dynamical system (Branicky, 1995).

In particular, the model acquires non-fading memory through feedback.



This holds in particular for continuous functions K, h and neural circuits *C* defined by DEs of the form

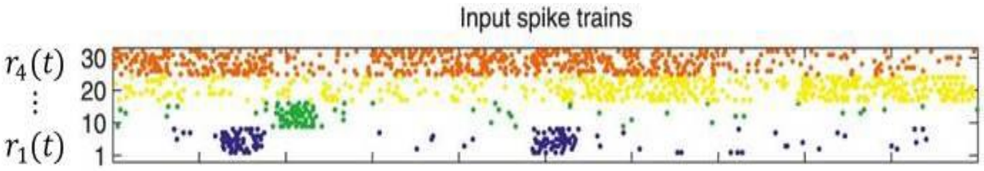
$$x_i'(t) = -\lambda_i x_i(t) + \sigma\left(\sum_{j=1}^{n} a_{ij} x_j(t)\right) + b_i \cdot \sigma(v(t))$$

(under some conditions on the λ_i, a_{ij}, b_i).

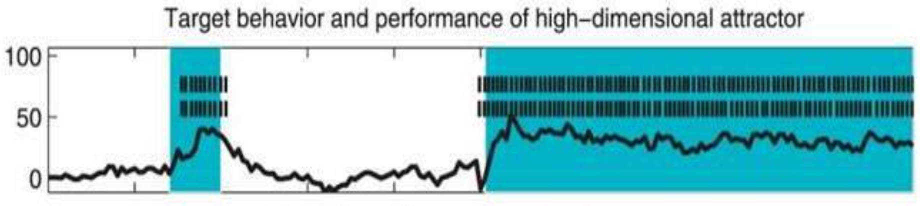
Open problem: Is there a variation of this theoretical result for networks of spiking neurons with noise?

Simulations suggest that feedback from readouts also significantly enhances the computational capability of networks of spiking neurons **with noise** (M. Joshi, Sontag, 2007):

A generic 4-D input trajectory
(rate based):

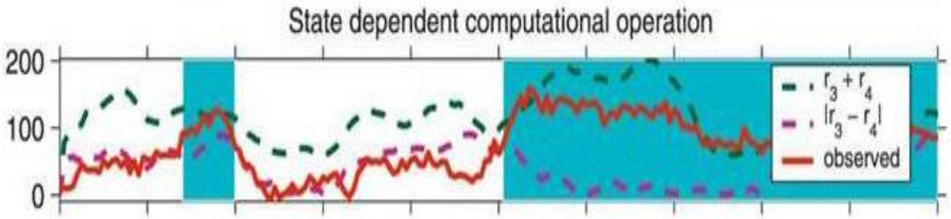


Readout neurons with feedback
were trained (with teacher forcing)
to fire when the last input burst
occurred in channel 4
(this requires a **non-fading** memory):



solid traces show firing rates of linear readouts

Other readouts (with feedback)
were trained to switch the network
computation in dependence of
these 2 non-fading internal states:



Subsequent results on practically successful training methods for readouts with feedback

Sussillo & Abbott (Neuron, 2009) showed that teacher-forcing of the network can be avoided for **ANNs** while training the readouts with feedback (through the FORCE algorithm).

Nicola and Clopath (Nature Comm., 2017) showed that FORCE can also be applied to networks of **spiking** neurons.

Hoerzer, Legenstein, M. (Cerebral Cortex, 2012) showed that the biologically unrealistic FORCE algorithm can be replaced in **ANNs** for the same tasks by reward-modulated Hebbian learning.

Open problem: Can one replace also for **spiking** neural networks the FORCE algorithm by biologically more realistic learning learning methods?

Alemi, Machens, Denève, Slotine (arXiv, 2017) showed that methods from **adaptive control theory** imply that an (idealized) **spiking** neural network **can learn** (through suitable local learning rules within the recurrent network) **any „well-behaved“ dynamics.**

Experimental data have confirmed many predictions of the liquid computing model

BNNs apply a fading memory and enable nonlinear computations with linear readouts:

D. Nikolic, S. Haeusler, W. Singer, and W. M.. Distributed fading memory for stimulus properties in the primary visual cortex. PLoS Biology, 2009

S. Klampfl, S. V. David, P. Yin, S. A. Shamma, and W. M.. A quantitative analysis of information about past and present stimuli encoded by spikes of A1 neurons. J. of Neurophys., 2012

Multiplexing of "neural codes" in the network for different tasks:

Rigotti, M., Barak, O., Warden, M. R., Wang, X. J., Daw, N. D., Miller, E. K., & Fusi, S. The importance of mixed selectivity in complex cognitive tasks. *Nature*, *2013*

Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T.. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 2013

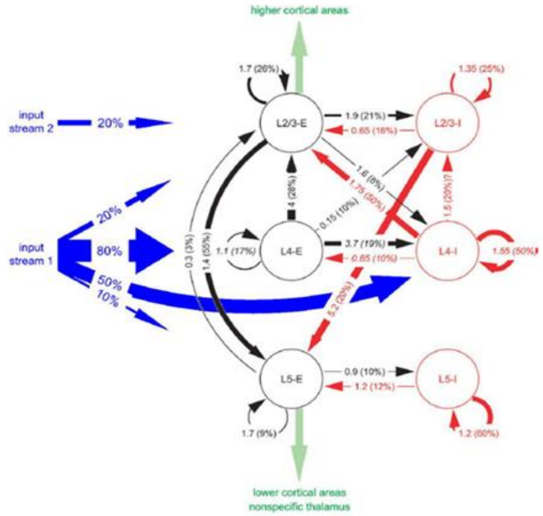
Diversity of neural readouts from the same cortical column:

Chen, J. L., Carta, S., Soldado-Magraner, J., Schneider, B. L., & Helmchen, F.. Behaviour-dependent recruitment of long-range projection neurons in somatosensory cortex. *Nature* 2013.

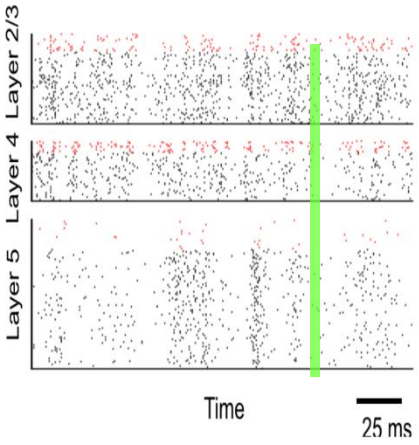
A quick look back at the problem whether models for stochastic computations in BNNs can be made compatible with having a diversity of neurons and synapses

Also SNNs with a diversity of neurons and synapses have a stationary distribution of network states z (Habenschuss, Jonke, M., PLOS CB 2013)

Inputs ***e*** *network states z*

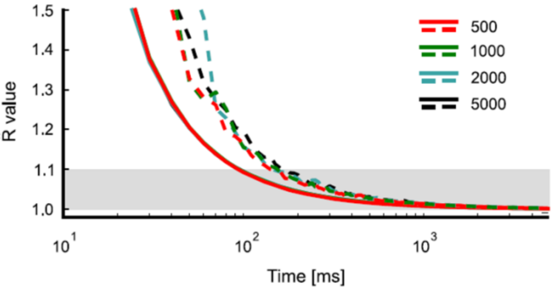


a network state z



Open problem:
Are BNNs designed to support fast convergence to the stationary distribution (from a selected range of initial states)?

Some quick initial simulations seem to support that (shown are curves are from Gelman-Rubin analysis):



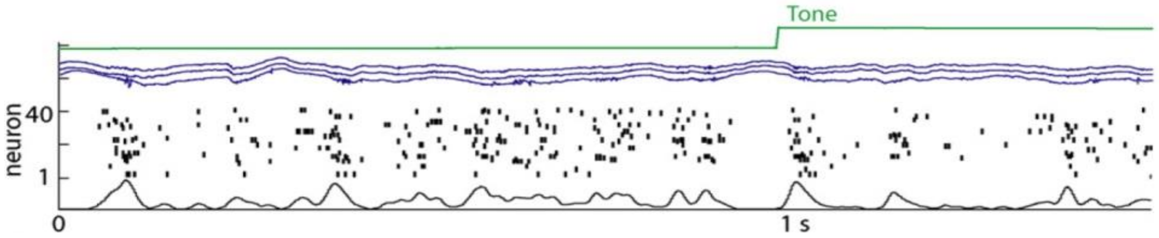
Hence also realistic models for **BNNs can carry out some form of probabilistic Inference**: They can estimate (via MCMC sampling) posterior marginals, conditioned on the external input e which they receive:

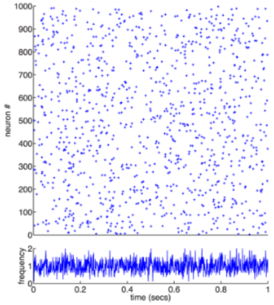
$$p(z_1|\mathbf{e}) = \sum_{v_2,\ldots,v_m} p(z_1, v_2, \ldots, v_m|\mathbf{e})$$

Note: This estimation is in general NP-complete.

Data from simultaneous recordings of many neurons contradict the theoretically predicted asynchronous irregular (AI) firing regime for computations of BNNs

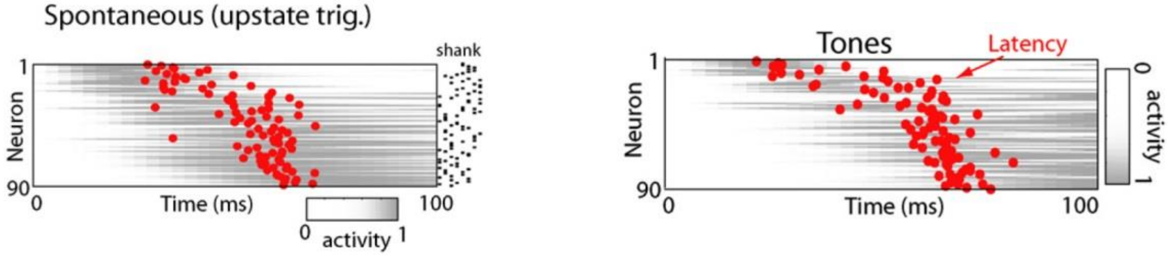
Both spontaneous and stimulus-evoked activity differ strongly from the asynchronous irregular regime that is commonly postulated by theoreticians as the most suitable regime for computations in BNNs:

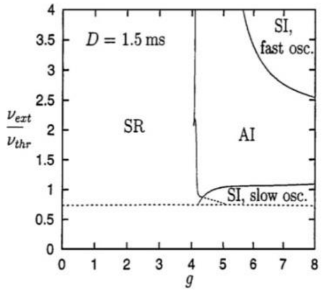




SNN in the AI regime

Experimental data show that firing activity in BNNs is dominated by group-firing patterns, referred to as assemblies or assembly sequences,

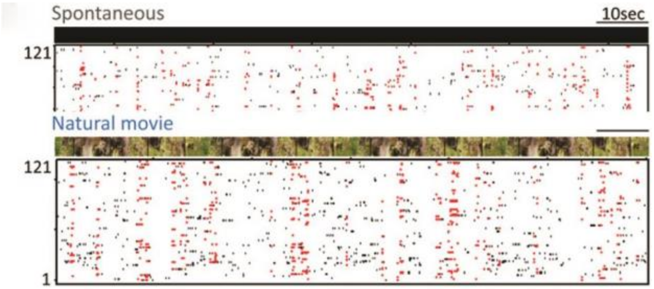




Brunel , J. Of Comp. Neurosc.,, 2000.

Each patch of cortical tissue produces only a rather small repertoire of such stereotypical firing patterns. The shown data are from area A1 in rodents (Luczak, Bartho, Harris, Neuron 2009).

Similar data arise from Ca-imaging in area V1 (Miller, ..., Yuste; PNAS 2014); „core neurons“ of assemblies are colored in red:



Open problem: How do BNNs encode information and compute with assemblies of neurons?

Reviews of hypotheses about the computational role of assemblies, and of related experimental data:

G, Buzsaki: Neural syntax: cell assemblies, synapsembles, and readers. Neuron 2010

Luczak, A., & MacLean, J. N. (2012). Default activity patterns at the neocortical microcircuit level. *Frontiers in integrative neuroscience*, *6*.

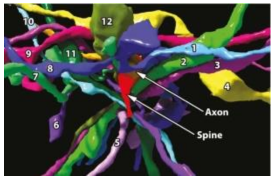
Dasgupta, S., Stevens, C. F., & Navlakha, S. (2017). A neural algorithm for a fundamental computing problem. *Science*, *358*(6364), 793-796.

Ison, M. J., Quiroga, R. Q., & Fried, I. (2015). Rapid encoding of new memories by individual neurons in the human brain. *Neuron*, *87*(1), 220-230.

Quiroga, R. Q. (2016). Neuronal codes for visual perception and memory. *Neuropsychologia*, *83*, 227-241.

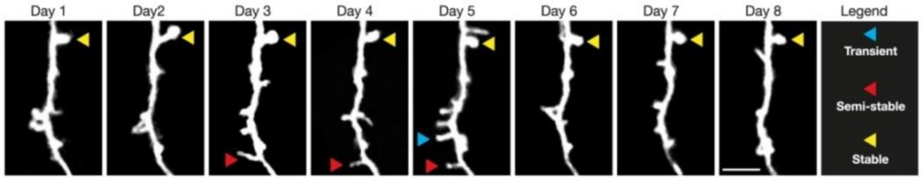
Unexpected properties of synaptic plasticity and neural codes

- Networks in the brain appear to rewire themselves for specific computational tasks:

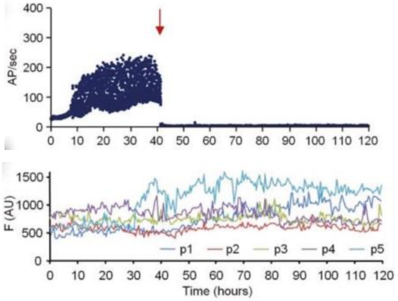


Lichtman Lab

Svoboda Lab



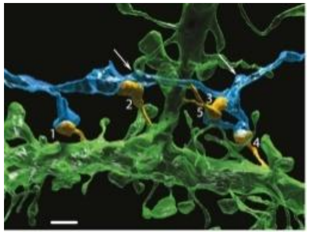
Kasai Lab

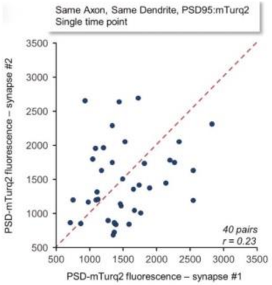


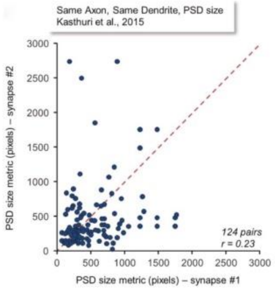
- This rewiring does not require neural activity:

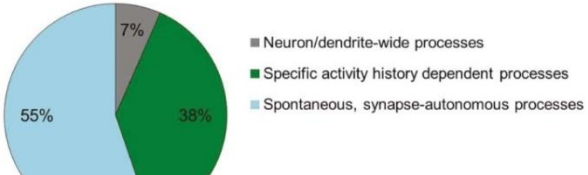
- Other data suggest that deterministic rules for synaptic plasticity that are commonly used in modelling and theory, such as STDP orHebbian learning, capture at most 50% of the actual synaptic plasticity:

estimated correlation of weights of **synapses between the same axon and same dendrite:** r = 0.23





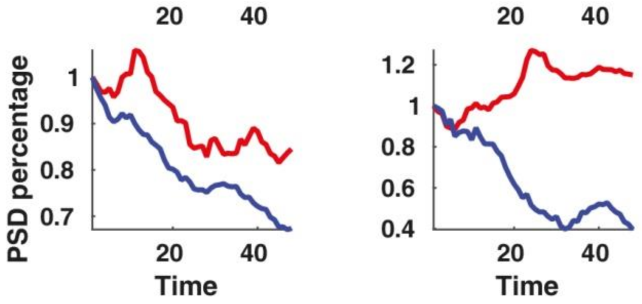




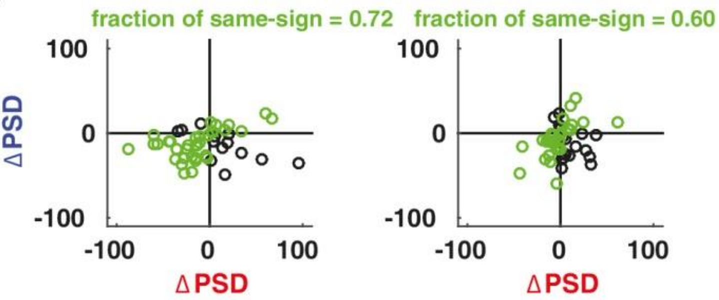
(Dvorkin, Ziv, 2016), using their data from cultures, and ex-in-vivo data from (Kasthuri et al., 2015)

The weak correlation of 0.23 for weights of synapses that have the same axon and the same dendrite could potentially be explained by differences in their initial size, or different learning rates, but...

Normalzation to the size of weights at the beginning of experiment still shows divergent evolution of weight sizes (shown are two sample pairs of traces for synapses with the same axon and the same dendrite; time in h):



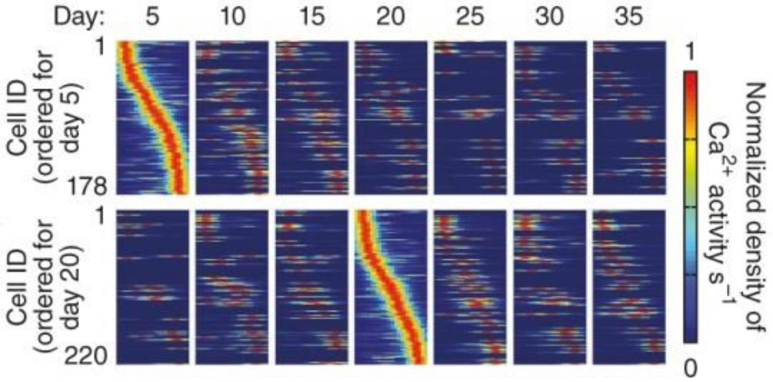
In addition, **weight changes** for synapses with the same pre- and postsynaptic neuron (even the same dendrite) **have frequently different signs:**



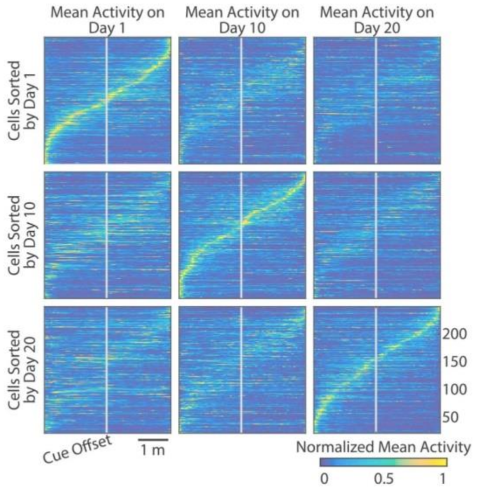
Data shown are from a re-analysis of the data from (Dvorkin and Ziv, 2016), carried out by Jian Liu in Graz.

Consistent with the strong „spontaneous synapse-autonomous component“ of synaptic plasticity, *neural codes are found to drift* on the time-scale of weeks and months

Ziv, Y., Burns, L. D., Cocker, E. D., Hamel, E. O., Ghosh, K. K., Kitch, L. J., ... & Schnitzer, M. J.. Long-term dynamics of CA1 hippocampal place codes. Nature Neuroscience, 2013



Driscoll, L. N., Pettit, N. L., Minderer, M., Chettih, S. N., & Harvey, C. D. (2017). Dynamic reorganization of neuronal activity patterns in parietal cortex. *Cell*, *170*(5), 986-999.



These experimental data motivate a few important open problems for modelling and theory

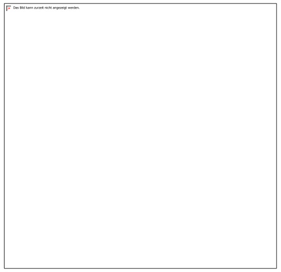
- How do BNNs attain and maintain stable computational function in spite of continuously ongoing synaptic plasticity and drifting neural codes?

- How can BNNs maintain stable computational function even in the presence of changes in the network or task?

- Is there anything that remains invariant in spite of all these ongoing changes in synaptic connections?

Theoretical physics provides a convenient mathematical framework

(Kappel et al., 2015, 2017)



- Formulate synaptic plasticity rules as **stochastic differential equations (SDEs)** for parameters θ_i that control both synaptic connectivity (functioning synaptic connection if $\theta_i > 0$), and the synaptic weight $w_i = exp(\theta_i - \theta_0)$ for a functioning connection:

$$d\theta_i = \underbrace{\left(b\frac{\partial}{\partial\theta_i}\log p^*(\boldsymbol{\theta})\right)}_{\textit{drift}} dt + \underbrace{\sqrt{2Tb}\cdot d\mathcal{W}_i}_{\textit{diffusion}}$$

$d\mathcal{W}_i$ denotes an infinitesimal step of a random walk

b = learning rate, T = temperature

- The Fokker-Planck equation implies that $\frac{\mathbf{1}}{\boldsymbol{Z}}\boldsymbol{p}^*(\boldsymbol{\theta})^{\frac{\mathbf{1}}{\boldsymbol{T}}}$ is a unique ***invariant*** **stationary distribution of the network parameters $\boldsymbol{\theta}$** (this does not imply that any **particular** network configuration $\boldsymbol{\theta}$ is an attractor!).
- The drift term can „program" a desired stationary distribution $\frac{\mathbf{1}}{\boldsymbol{Z}}\boldsymbol{p}^*(\boldsymbol{\theta})^{\frac{\mathbf{1}}{\boldsymbol{T}}}$ into the network.
- Consider the case of **reinforcement learning**, where one sets

$$\boldsymbol{p}^*(\boldsymbol{\theta}) \propto \boldsymbol{p_S}\,(\boldsymbol{\theta}) \cdot \boldsymbol{E}\,[total\ reward|\boldsymbol{\theta}]$$

with a prior $p_S\,(\boldsymbol{\theta})$ that could express for example structural constraints (such as sparse connectivity), but also innate or previously learnt knowledge.

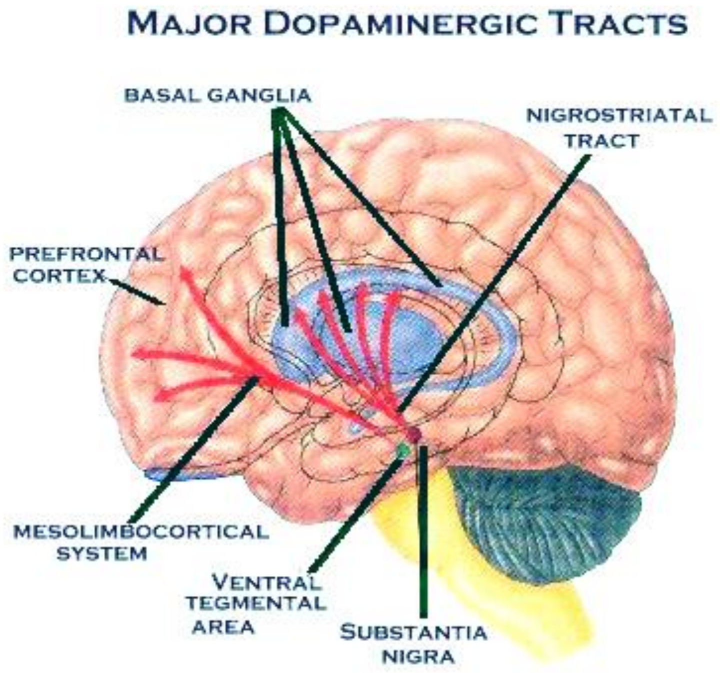
- With a Gaussian prior one reproduces standard models for spontaneous spine dynamics.
- The deterministic drift term $\left(b\frac{\partial}{\partial\theta_i}\log p^*(\boldsymbol{\theta})\right)$ is approximated by a commonly used local rule **for reward-gated STDP,** generalized to include rewiring, with an eligibility trace („tag") e(t) for STSP-events:

$$d\theta_i = b\ \left(\frac{1}{\sigma^2}(\mu - \theta_i)dt + r(t)e_i(t)\right) + \sqrt{2bT}d\mathcal{W}_i$$

Dopamine is the most frequently considered internal brain signal for reward, and it is known to modulate synaptic plasticity

Dopamine (DA) neurons can be found in the

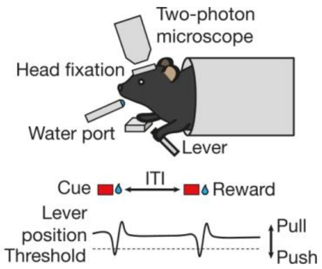
- Substantia Nigra pars compacta (SNc): projects to basal ganglia
- Ventral Tegmental Area (VTA): projects to PFC (prefrontal cortex)

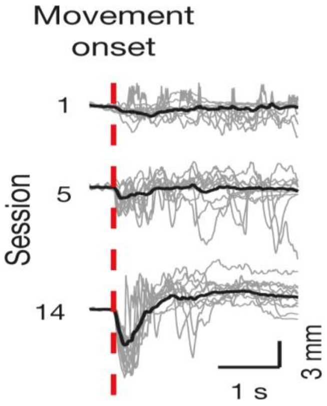


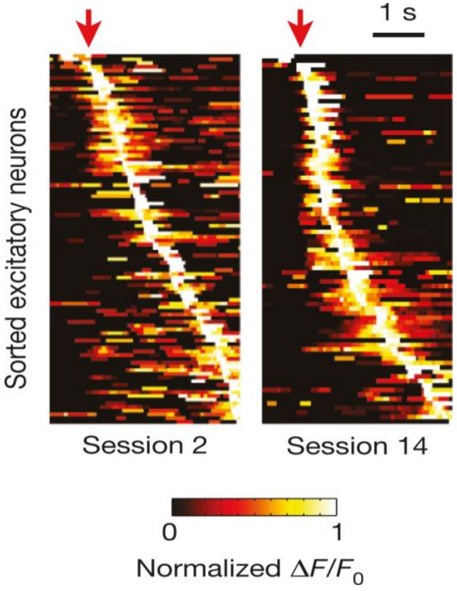
When rats are given the opportunity
to press a lever that causes stimulation
of neurons that emit DA, they will do nothing else,
until they starve.

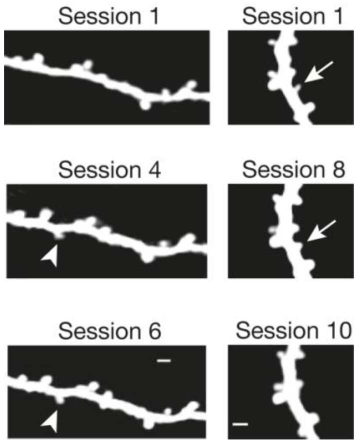
The simple account of DA as a **global reward signal** in the brain **has been questioned** in recent years on the basis of more refined expermental data, e.g. from the work of Okihide Hikosaka (NIH) and Ilana Witten (Princeton). It is not clear how theory can cope with these newer data.

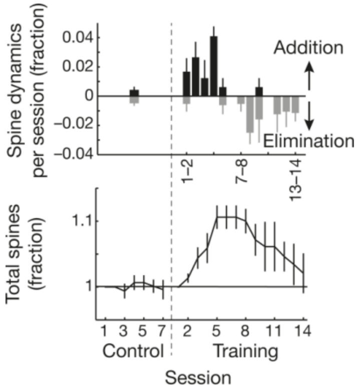
Experimental data on DA-gated self-configuration of recurrent neural networks in the motor cortex





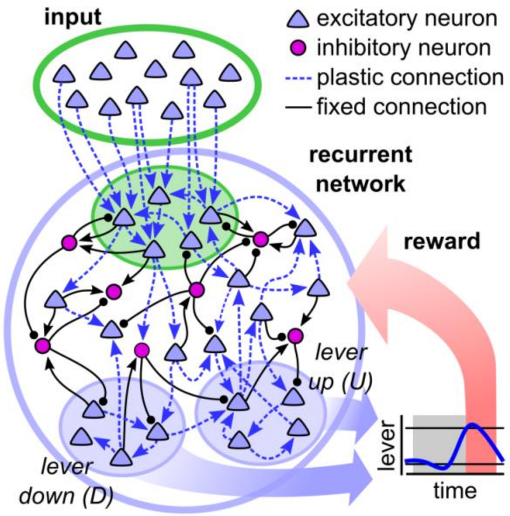


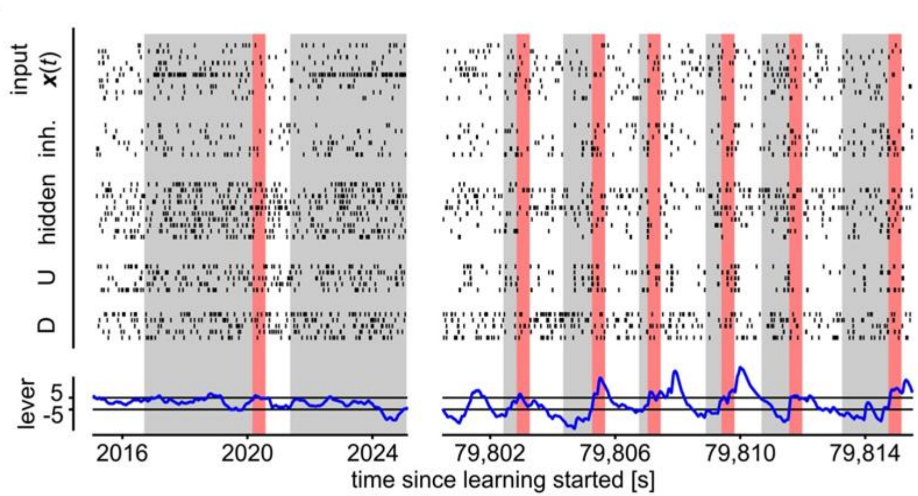


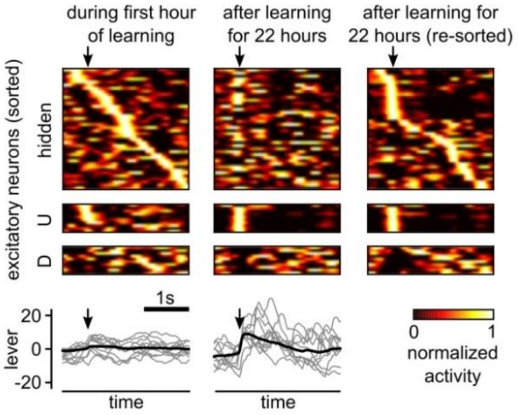


Emergence of reproducible spatiotemporal activity during motor learning;
Andrew J. Peters, Simon X. Chen & Takaki Komiyama; Nature(510) 2014

One can reproduce such reward-gated self-configuration of BNNs in the previously sketched model





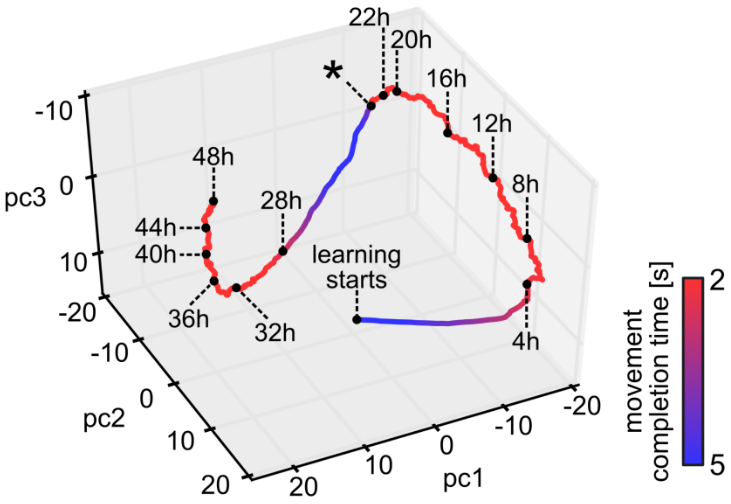


Stereotypical assembly sequences ermerge in the SNN during learning

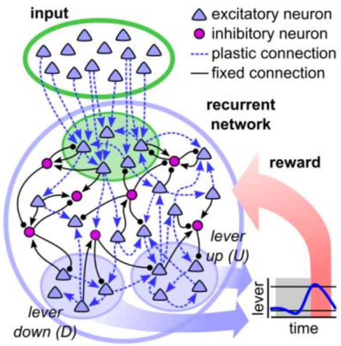
D. Kappel, R. Legenstein, S. Habenschuss, M. Hsieh and W. M. Reward-based self-conguration of neural circuits. Arxiv 2017.

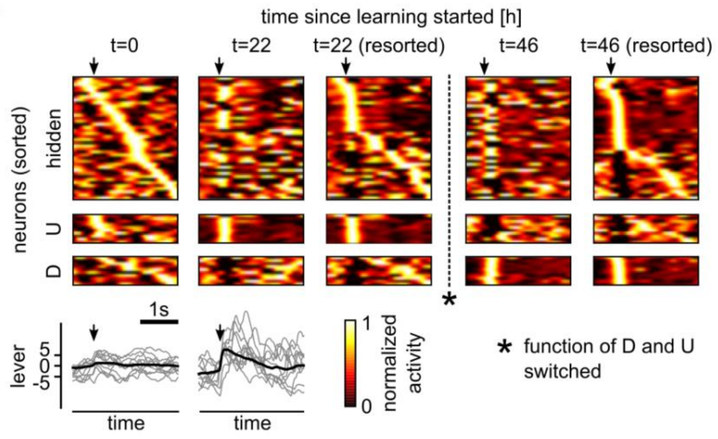
Resulting dynamics of the parameter vector θ

The parameter vector keeps moving even after good performance has been reached (after 4h of learning), causing a drift of the neural code for movement control.

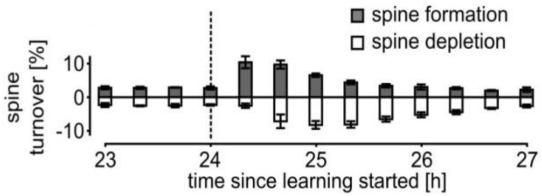


Functional benefit of ongoing parameter dynamics:
Immediate and automatic compensation for a drastic network perturbation: Switch of Function of the populations U and D after 24h





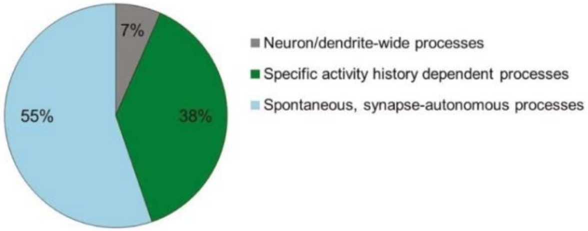
This switch gives rise to a reorganization of network connections, and of the assembly dynamics



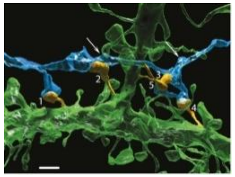
Remarks on this simulated emergence of computational function

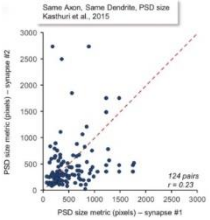
- The simulated model is compatible with experimental data on a strong synapse-autonomous component of synaptic plasticity.

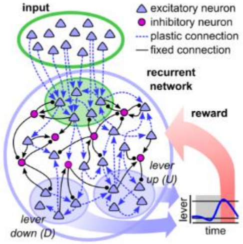
- In fact, this component appears to be functionally important for this model: it worked best with a temperature T that reproduced a similar amount of synapse-autonomous dynamics as found in the data from the Lichtman Lab and Ziv Lab.



- The temperature T that worked best, also reproduced a weak correlation of synaptic weights for multiple connections between neurons, as found in the data from these Labs



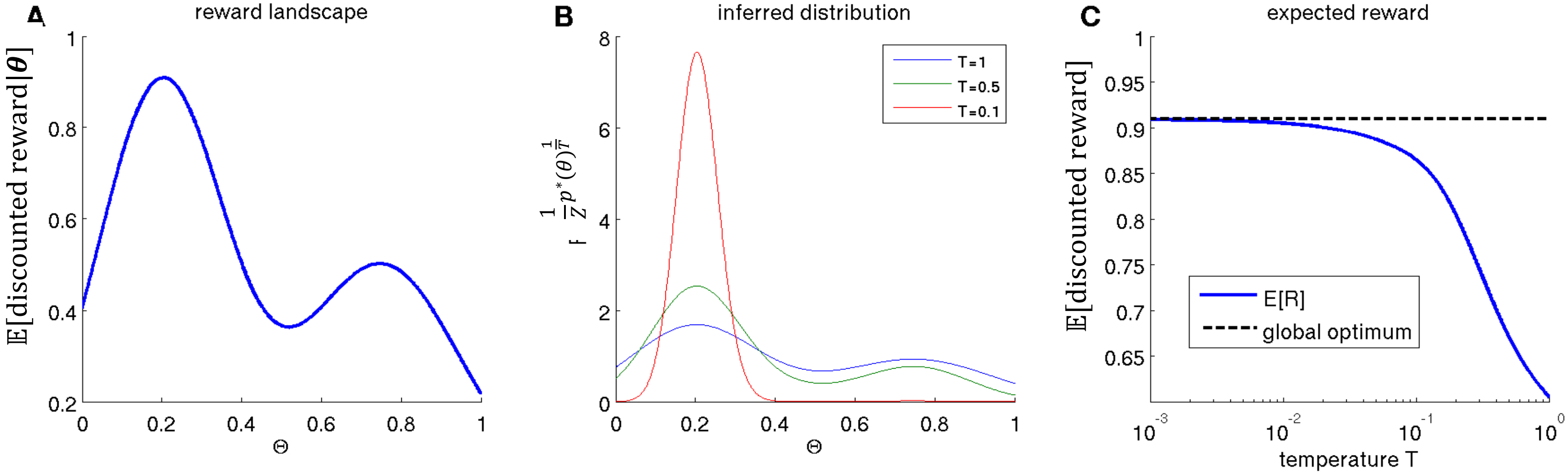




- The learning time in the model was quite long. To scale it up to larger networks, one may need a given „connectome“ that decreases the time required to find a good network configuration.

Is there a deeper link between stochastic network plasticity of BNNs and optimization theory?

- Common optimization methods, such as Simulated Annealing, modify the temperature of a stochastic process
- Theory promises (in principle) convergence to a **globally optimal** solution for a suitable cooling schedule.
- If one decreases the temperature T in the model, $\frac{1}{Z}p^*(\boldsymbol{\theta})^{\frac{1}{T}}$ is more focused on high-reward regions. This increases the search time, but also in the long run the expected reward:



General remarks on the role of reinforcement learning (RL) on our understanding of computations in BNNs

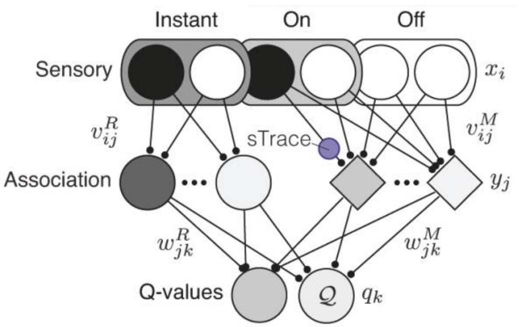
- RL provides one of the most attractive frameworks for modelling the emergence of concrete computational capabilities in BNNs

- STDP with eligibility traces provides on the network level an approximation of one of the most powerful known RL-methods: Policy gradient

- Stochasticity is needed in RL in order to allow exploration

- Several recent successes of Google DeepMind are based on RL for ANNs. Hence RL provides an inspiring link between state-of-the-art Machine Learning and research on BNNs.

Work from the Roelfsema-Lab suggests another possible link between RL-theory and learning in BNNs:
A possible implementation of Q-learning in BNNs



Diamond-shaped units: memory units.

One assumes that some mechanism tags synapses that were active when the current action was chosen (WTA on output level). Weights of tagged synapses are modified according to a TD-signal (implemented through a neuromodulator).

They model learning of the same tasks for which they had previously trained animals in reward-based learning experiments.

Rombouts, J. O., Bohte, S. M., & Roelfsema, P. R. (2015). How attention can create synaptic tags for the learning of working memories in sequential tasks. *PLoS computational biology*, *11*(3), e1004060.

Brosch, T., Neumann, H., & Roelfsema, P. R. (2015). Reinforcement learning of linking and tracing contours in recurrent neural networks. *PLoS computational biology*, *11*(10), e1004489.

Recent review papers that discuss links between Machine Learning and research on BNNs

- Marblestone, A. H., Wayne, G., & Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, *10*.

- Hassabis, D., Kumaran, D., Summerfield, C., & Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, *95*(2), 245-258.

- Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*.

A key issue in current discussions between Machine Learning and Neuroscience

- Many successes in Deep Learning rely on backprop (BP) for feedforward ANNs.

Is BP a learning algorithm that is available to BNNs?

- See the work of Geoff Hinton, Yoshua Bengio, Timothy Lillicrap for attempts to provide a positive answer.

If the answer is negative, what is used by BNNs instead of BP?

- A closely related issue: Applications of Deep Learning for language processing typically use backprop through time (BPTT) for training recurrent ANNs, that are often enhanced through LSTM (long short term memory) units. These results raise the questions:

Is BPTT also available to BNNs?

Are some sort of LSTM units available to BNNs?

3. Methodological issues: How should models for computations in BNNs be chosen?

In order to show for example that BNNs can integrate information over time, should one use the simplest possible model, or integrate data on BNNs that are likely to be relevant for this computation?

Occam‘s razor suggests that some form of model regularization should be used

William of Ockham (circa 1287–1347)

Numquam ponenda est pluralitas sine necessitate
[Plurality must never be posited without necessity]

From the theoretical perspective, the only relevant function of regularization is a **reduction in the number of possible models**, in order to avoid overfitting.

There are many options for regularizing models of BNNs:

- Use the mathematically simplest neuron- and synapse models
- Use models that do not require super-fine tuning of parameters
- Use models that are consistent with more experimental data on BNNs
- Use models that produce network activity similar as observed in BNNs
- Use models that exhibit a similar robustness to network perturbations as BNNs
- Use models that can learn from few examples (like brains can).

What do the 3 levels of Marr imply for modelling computations in BNNs?

(Marr and Poggio, 1976) suggested to model computation in BNNs on 3 different levels:

- ***computational (behavioural) level:*** What needs to be computed?
- ***algorithmic level :*** By what computational strategy?
- ***implementation level:*** How can BNNs implement that strategy?

Theoreticians often use these Marr-levels as justification for ignoring the biological implementation level in their analysis.

But we have seen that experimental data on the biological implementation

level often make us rethink how computations are organized in BNNs, and what their computational goal is.

Further references to these methodological questions, and also to more open problems

- W. M., C. H. Papadimitriou, S. Vempala, and R. Legenstein. Brain computation: A computer science perspective. *Springer Lecture Notes in Computer Science*, vol. 10000, 2018

- W. M.. Searching for principles of brain computation. *Current Opinion in Behavioral Sciences (Special Issue on Computational Modelling)*, 11:81-92, 2016.

The slides of these lectures will be posted by tomorrow on my homepage
http://www.igi.tugraz.at/maass/publications.html

4. Summary

- The organization of computations in BNNs is likely to be very different from schemes that are familiar from Computer Science and ANNs

- I discussed the impact of a few specific types of experimental data on model selection:

--large trial-to-trial variability (suggest stochastic computations, MCMC sampling)

--diversity of neurons and synapses (suggests computational models that benefit from having diverse units)

--characteristic firing patterns of simultaneously recorded neuron (suggests that we need to understand models for stochastic computation with assemblies and assembly sequences)

--unexpected properties of synaptic plasticity and neural codes (suggests to consider stochastic rules for synaptic plasticity and rewiring, and analyze the stationary distribution of network configurations)

- Other candidates for experimental data that provide important clues/constraints for understanding computation and learning in BNNs

--clustering of synapses on selected dendrites

--dendritic spikes that impact neural computation and synaptic plasticity

--inhibition and disinhibition through networks of specific types of inhibitory neurons

- We should probably spend more effort on the selection and justification of our models for computation and learning in BNNs.