Disentangling the roles of dimensionality and cell classes in neural computations

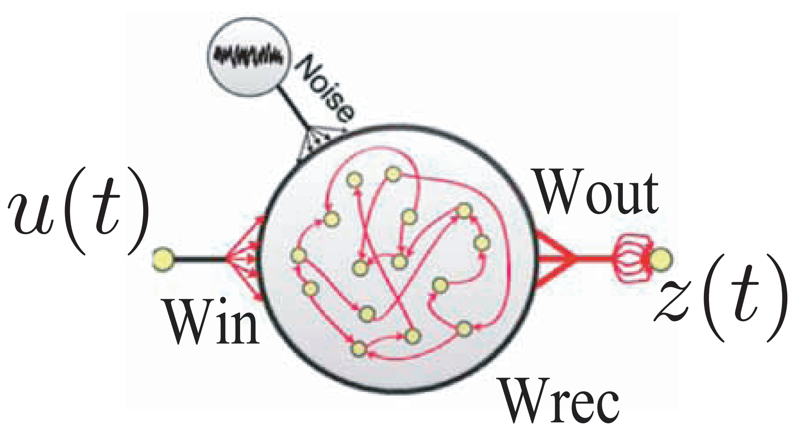
Alexis Dubreuil[1], Francesca Mastrogiuseppe[2], Srdjan Ostojic[1]

1. LNC, Ecole Normale Supérieure, PSL Research University, Paris, France 2. Gatsby Computational Neuroscience Unit, UCL, London

Introduction

- How the two key concepts of cell classes and low-dimensional trajectories interact to shape neural computations ?
- We propose a method which combines artificial neural networks training and a recent theory linking dimensionality and connectivity structure [1] .
- We generate network models of low dimensionality and fixed number of cell classes which implement a series of behavioral tasks and allows us to explore the roles played by dimensionality and cell classes in neural computations.

Low-rank, multi-populations RNNs



Equations for RNN:

$$\tau\frac{d\vec{x}}{dt} = -\vec{x} + W_{rec}\vec{\Phi}(\vec{x}) + u(t)\vec{W}_{in}$$

Network's output $z(t) = \vec{W}_{out}^T\vec{\Phi}(\vec{x}(t))$

with $\Phi_k(\vec{x}) = \tanh(x_k)$

Rank K RNN:

$$W_{rec} = \frac{1}{N}\sum_{k=1}^{K}\vec{m}^k\vec{n}^{kT}$$

K-dimensional activity:

$$[x_i] = \kappa_1 m_i^1 + \kappa_2 m_i^2 + ... + I_i$$

$$\kappa_1 = \frac{1}{N}\sum_{j=1}^{N} n_j^1\Phi_j(\vec{x})$$

Single population network [1]:

For connectivity vectors $\vec{a}, \vec{b}, ...$

$$a_i, b_i, ... \sim \mathcal{N}(0, \{\sigma_{ab}\})$$

P-populations network:

$$a_{i\in p}, b_{i\in p}, ... \sim \mathcal{N}(0, \{\sigma_{ab}^p\})$$

$$p = 1, ..., P$$

Latent recurrent dynamics (e.g. K=2):

$$\dot{\kappa}_1 = -\kappa_1 + \tilde{\sigma}_{n_1m_1}\kappa_1 + \tilde{\sigma}_{n_1m_2}\kappa_2 + \tilde{\sigma}_{n_1W_{in}}u(t)$$

$$\dot{\kappa}_2 = -\kappa_2 + \tilde{\sigma}_{n_2m_1}\kappa_1 + \tilde{\sigma}_{n_2m_2}\kappa_2 + \tilde{\sigma}_{n_2W_{in}}u(t)$$

with functional connectivities:

$$\tilde{\sigma}_{ab} = \sigma_{ab}\langle\phi'\rangle \quad \text{or} \quad \tilde{\sigma}_{ab} = \sum_{p=1}^{P}\sigma_{ab}^p\langle\phi'\rangle_p$$

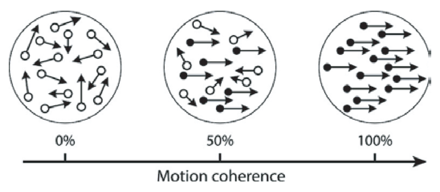
Approach:

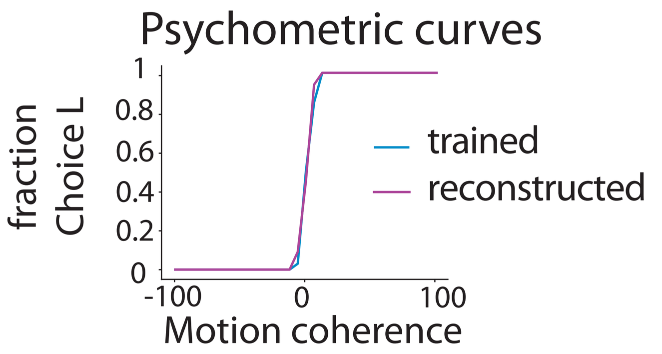
Train minimal-rank RNN → Reverse-engineer RNN → Construct a class of degenerate solutions

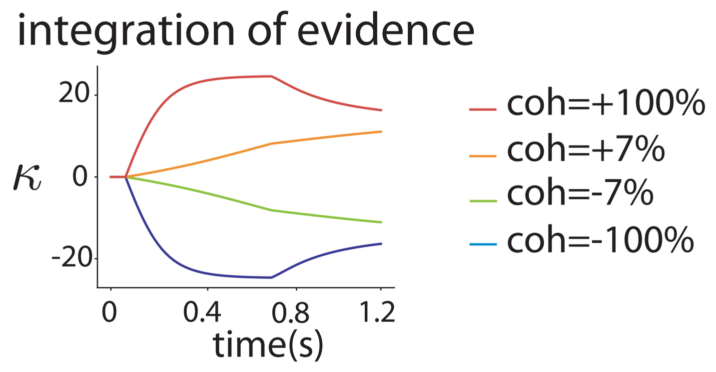
Behavioral task	Cognitive operations	Minimal rank / # of cell classes
Random dot motion task	stimulus integration	K = 1 P = 1
Mante task	stimulus integration contextual gating	K = 1 P = 2
Romo task	parametric working memory comparison	K = 2 P = 1
Delay-Match-to-Sample (two items {A,B})	object working memory comparison	K = 2 P = 2

Increasing rank increases the number of internal latent variables available for computation

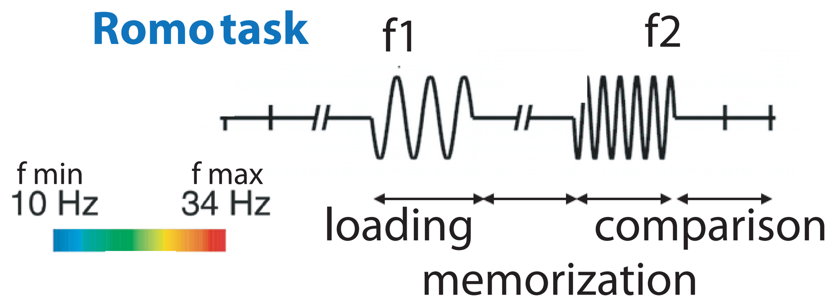
Random dots motion task

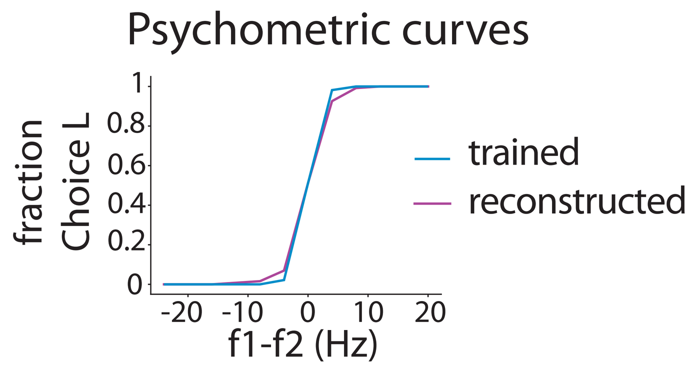


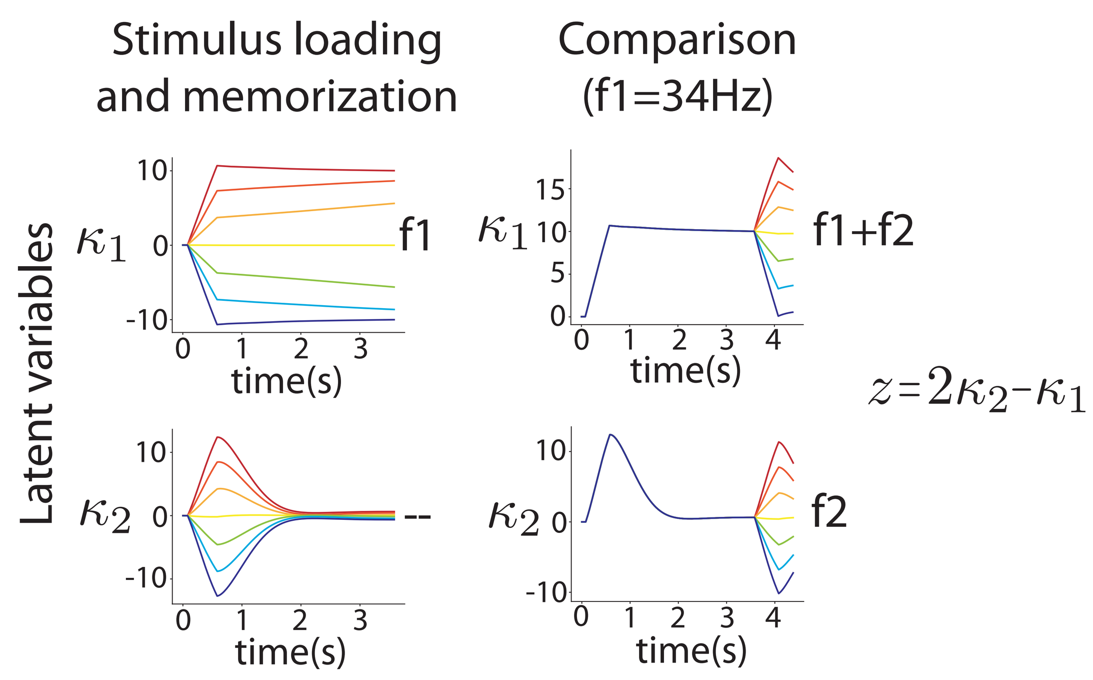




Romo task



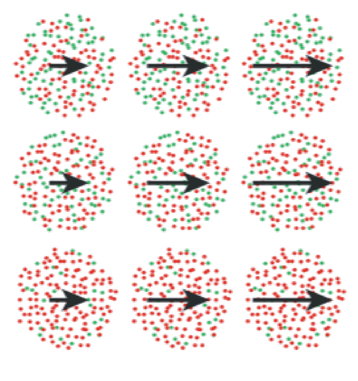


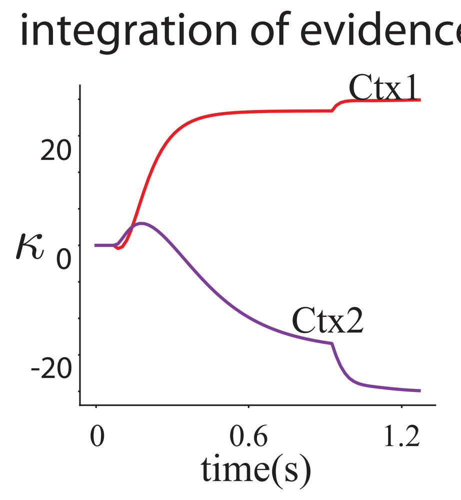


Multiple populations allow multiple operations to be performed on available latent variables

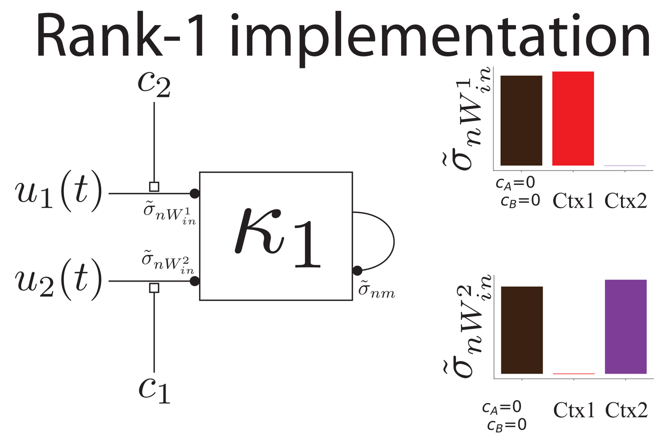
Context-dependent perceptual discrimination task

Mante task

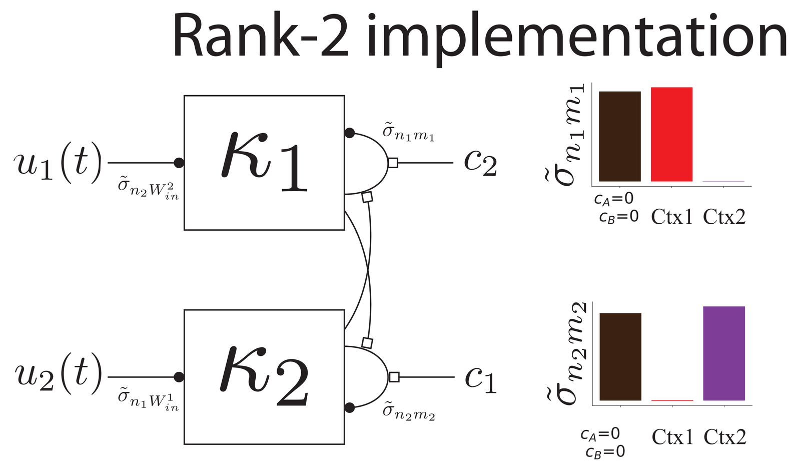


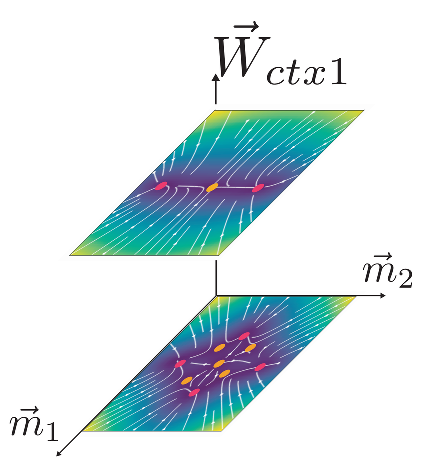


Rank-1 implementation

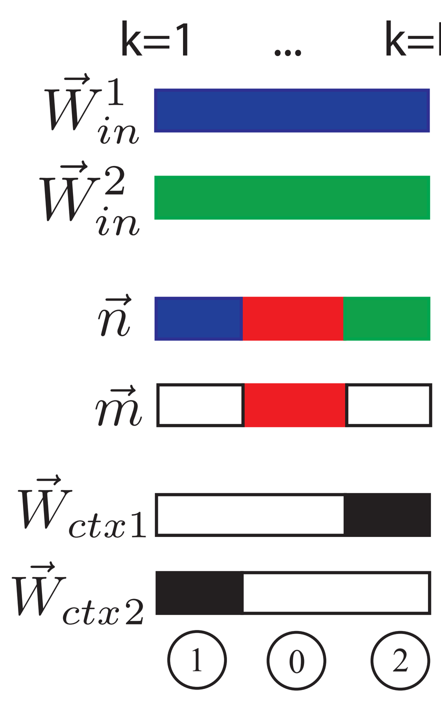


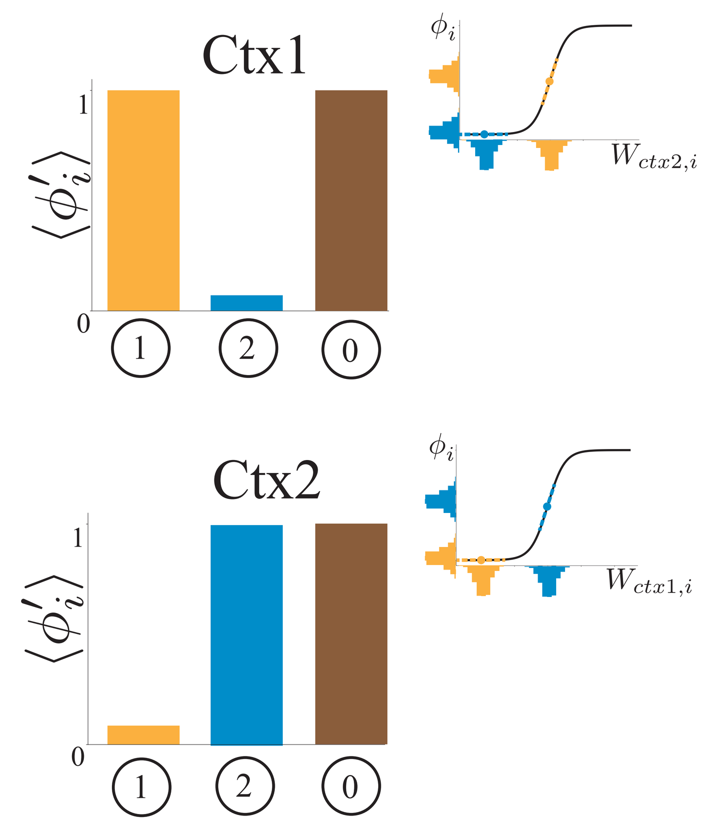
Rank-2 implementation





Control of effective connectivity requires multiple populations

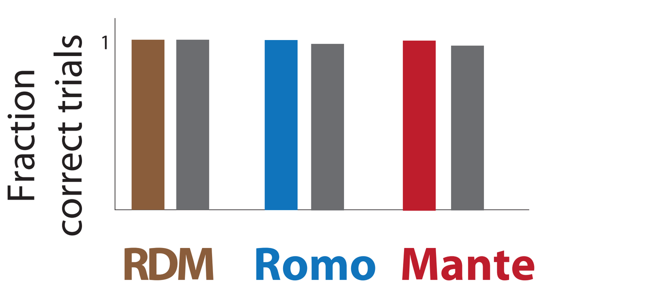


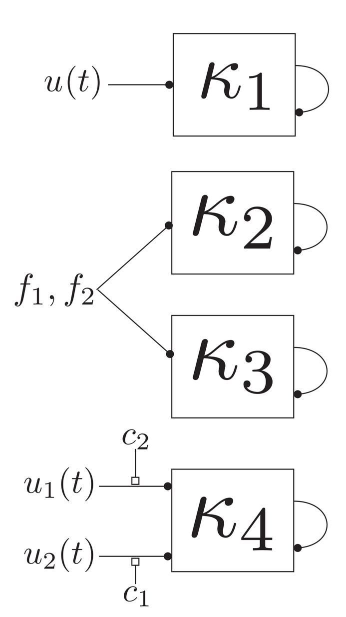


Implementing multiple tasks

Computation in orthogonal spaces

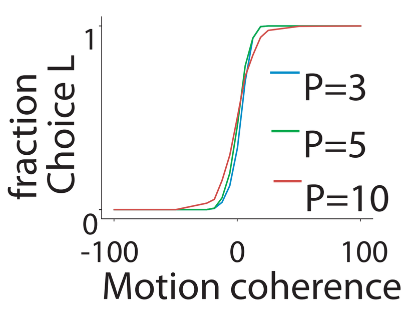
$$W_{rec} = \vec{m}_1^T\vec{n}_1 + \vec{m}_2^T\vec{n}_2 + \vec{m}_3^T\vec{n}_3 + \vec{m}_4^T\vec{n}_4$$

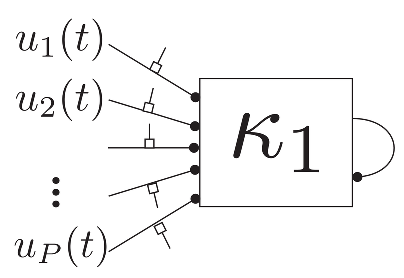




Multiple latent variables perform operations in parrallel: Multiplexing

Computations with P-populations





Multiple possible operations can be performed on a shared latent variable

Summary

- Dimensionality controls the number of latent variables available to implement a computation
- Multiple populations allow network to flexibly switch between different input-output mappings
- Based on these two principles, we have shown how to design networks solving multiple tasks

Reference

[1] F. Mastrogiuseppe, S. Ostojic. Linking connectivity, dynamics and computations in low-rank recurrent neural networks. *Neuron*. 2018