Reverse-engineering low-rank recurrent neural networks

Adrian VALENTE

August 2019

Under the supervision of:

Srdjan OSTOJIC,

Laboratoire de neurosciences cognitives et computationnelles,

ENS Paris

Abstract

Some cortical computations have recently been interpreted in the context of dynamical systems, and it has been shown how features like stable points and low-dimensional manifolds can explain how they are implemented in the brain. However, a mechanistic explanation of how these dynamics arise from the connectivity between neurons is still lacking. The low-rank recurrent neural network [MO18] is a recently proposed model where the links between connectivity and dynamics can be understood. In this work, we put this model to application by training low-rank RNNs to do basic cognitive neuroscience task, and by reverse-engineering them to understand how the connectivity found by the training algorithm implements the desired computation. As in [MO18], we find that the geometrical arrangement between left and right eigenvectors of the connectivity matrix is a key component, but more surprisingly we find that the neurons of trained networks seem to spontaneously organise themselves in different cell populations with different connectivity statistics, and that this cell diversity gives rise to a very rich repertoire of dynamical behaviors.

Acknowledgments

I wish to express my deepest gratitude to Srdjan Ostojic for enabling me to do this internship, and to all members of the LNC2 that have welcomed me, especially Alexis Dubreuil and Manuel Beiran. I also wish to thank Wulfram Gerstner for his kind guidance at EPFL.

Contents

Introduction

Patterns of activity in the brain are thought to be the basis for numerous aspects of behavior, including decision making or memory. In the past, a certain number of these brain patterns have been explained by studying the output of individual neurons to environmental variables, leading to the tuning curve approach, which has had some successes. However, limits of this approach have become apparent in the recent years. Notably, it was shown that in many areas, neurons exhibit mixed selectivities (their responses being determined by a high number of factors), and variable selectivities (their response profile varying with time). These observations have lead instead to a focus on *networks* of neurons [Yus15].

Recent progress in measurement methods, notably in multi-electrode arrays and calcium imaging have helped this paradigm shift by enabling simultaneous recordings of hundreds of neurons. Studies of these recordings have shown to be particularly informative about some complex and high level variables, for example in decision making or timing tasks [MSSN13, WNHJ18]. Indeed, although neural responses seem complex and unpredictable when looking at individual neurons, simple descriptions become apparent when looking at the activity of a whole assembly of neurons embedded in a very high-dimensional space (the *state space*) and applying dimensionality reduction algorithms to it [CY14].

These observations are related to the hypothesis that assemblies of neurons can be understood as *dynamical systems*, and that they exhibit features such as fixed points or limit cycles that best explain how a network can perform a particular task [Sus14, BM09]. Although one can uncover these dynamics from recordings, the question of understanding how they emerge from the wiring of neural circuits remains unanswered. Indeed, a mechanistic model linking connectivity to dynamics in neural networks is still out of reach, notably because of the complexity induced by recurrent circuitry [GG15, MO18].

A model that can be useful in understanding this relation is the artificial recurrent neural network (RNN). By taking very simple hypotheses (one single activity variable per neuron, related to firing rate by a non-linear transfer function), this network has been able to reproduce complex activity patterns recorded experimentally. It can be trained with mathematical optimization algorithms to exhibit a particular behavior, and very strikingly, even when trained not to reproduce the exact recorded activity but only the behavior, lots of similarities can be found between the activity patterns of trained RNNs and biological networks [MSSN13, WNHJ18]. This raises the hope that a good understanding of RNNs can be useful in understanding how biological networks perform computations.

However, a theoretical understanding of RNNs is currently lacking. The method used to train them does not give much insight into how they actually accomplish a task, which is why they are often considered as "black boxes" [SB12]. Recently, Mastrogiuseppe and Ostojic have proposed such a theory in the special case of RNNs whose connectivity is *low-rank* [MO18]. This special case is particularly interesting, not only because it is mathematically tractable, but also because it relates to the fact that measured activity during a task tends to be explained by a very low number of dimensions [CY14], and that a low-dimensional activity is one characteristic of low-rank RNNs. Moreover, it appears that low-rank approximations of connectivity matrices can retain some of their computational characteristics. For all these reasons, we believe that an understanding of the emergence of dynamics in low-rank RNNs can shed light on the mechanisms of higher rank RNNs and ultimately of biological networks.

To study how low-rank RNNs are able to perform a task, and what limits rank imposes on computations, we apply the following systematic procedure:

- we train by standard optimization techniques (e.g. gradient descent) an RNN that we force to have a certain rank on a task,
- we reverse-engineer the obtained network by examining its patterns of activity and its dynamical properties, and link them to properties of the connectivity matrix. This leads us to hypotheses concerning the form that the connectivity matrix should take in order for the network to accomplish the task.
- we test our hypotheses by trying to build a network able to accomplish the task "by hand" (ie. without training), by sampling the coefficients from well-chosen distributions.

Along with this "experimental" procedure, we develop the theory linking the connectivity matrix to the network dynamics in some particular cases. This work has led us to understand that a fundamental parameter controlling what a network is able to accomplish, beyond the rank, is the number of populations it contains. Indeed, even a very low rank network can exhibit rather complex dynamics by combining various groups of neurons, each with different connectivity statistics, in a single low-rank structure. Moreover, trained networks seem to naturally exhibit these different neural populations, although it had never been enforced by the training procedure. It is particularly surprising to see a biological feature emerge naturally as a result of training general purpose networks.

This observation raises the idea that cell populations could be an essential computational tool. It bridges the gap between the general purpose RNN model and more classical models of computational neurosciences that combine different populations of neurons.

In this report, we will gradually develop our current understanding of low-rank networks by examining first rank one network and then rank two networks. For each case, we will start by developing the theory and exploring the possible behaviors with simulations. Then we will use the developed tools to reverse-engineer actual low-rank RNNs trained on specific tasks (random dots motion for the rank 1 case, delayed match-to-sample for rank 2).

Chapter 1

General formalism

1.1 Theoretical framework

We will throughout this report follow the formalism that has been developed in [MO18].

We will thus consider the classical model of the rate recurrent neural network, where each neuron receives an input current $x_i(t)$ obeying the rate equation:

$$\tau \dot{x}_i(t) = -x_i(t) + \sum_{j=1}^{N} J_{ij} \phi\left(x_j(t)\right) + I_i \tag{1.1}$$

where $\mathbf{I}$ represents the input to the network, $\mathbf{J}$ is the connectivity matrix (vectors and matrices will be in bold font), and ϕ the transfer function of the units, for which we choose the hyperbolic tangent for the remainder of this report.

We will often write this equation in matrix form

$$\tau \dot{\mathbf{x}}(t) = -\mathbf{x}(t) + \mathbf{J}\phi(\mathbf{x}(t)) + \mathbf{I}$$

where all vectors are in bold font and we take ϕ applied to a vector to mean the elementwise application of the function.

In [MO18], the connectivity matrix is considered to be the sum of a structured low-rank matrix and of a random gaussian matrix:

$$J_{ij} = g\chi_{ij} + P_{ij} \tag{1.2}$$

where $\chi_{ij} \overset{iid}{\sim} \mathcal{N}(0,1)$ and $\mathbf{P}$ is the low-rank component that we can write as a sum of outer products

$$P_{ij} = \frac{1}{N} \sum_{k=1}^{K} m_i^{(k)} n_j^{(k)} \tag{1.3}$$

with $K \ll N$ and the $\mathbf{m}^{(k)}$ and $\mathbf{n}^{(k)}$ are vectors of dimension N.

In [MO18], authors have shown that if the noise component is sufficiently small, then the activity of the network is essentially equivalent to the one induced by the low-rank component only. In particular, this is the case for $g < 1$. Hence, for our purposes, we will ignore the noise component and focus on purely low-rank networks, by taking $\mathbf{J} = \mathbf{P}$.

An important observation is that in this case, for the autonomous system ($\mathbf{I} = \mathbf{0}$), the activity decays exponentially towards the linear subspace formed by the $\mathbf{m}^{(k)}$ vectors, which is stable by the differential equation. Hence, one can study the system by restricted dynamics on this hyperplane and still get all the necessary information about its evolution, since all fixed points and limit cycles are contained in it.

1.2 Experimental setup: trainable RNNs

In this work, we will focus not only on giving possible theoretical solutions but we also wish to train RNNs on actual tasks using standard optimization techniques, in order to see what interesting solutions are found by the algorithm and if we are able to reverse engineer them.

For this, we have implemented two trainable RNNs, a full-rank and a low-rank one, that I will briefly describe.

The full rank RNN implements the equation:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \sigma\boldsymbol{\eta}_t + \alpha(-\mathbf{x}_t + \mathbf{W}_r\phi(\mathbf{x}_t) + \mathbf{W}_i\mathbf{I}_t) \tag{1.4}$$

and gives a readout:

$$z_t = \mathbf{w}_o^T\mathbf{x}_t \tag{1.5}$$

where the matrices $\mathbf{W}_r$, $\mathbf{W}_i$ and the vector $\mathbf{w}_o$ are the trainable parameters, trained by gradient descent (or sophisticated versions like Adam) on the training data. The term $\sigma\boldsymbol{\eta}_t$ in (1.4) indicates white noise that is applied to each neuron to account for internal variability in neuronal processes. Note that here the actual input to the network is represented by $\mathbf{W}_i\mathbf{I}_t$, and $\mathbf{I}_t$ is a vector containing the different parameters that we wish to feed to the network. Finally, the parameter α accounts for both the time constant of the model and the integration timestep: $\alpha = dt/\tau$, where we typically use $dt = 20$ms and $\tau = 100$ms.

In a similar fashion, the low-rank RNN implements the equation:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \sigma\boldsymbol{\eta}_t + \alpha(-\mathbf{x}_t + \mathbf{M}\mathbf{N}^T\phi(\mathbf{x}_t) + \mathbf{W}_i\mathbf{I}_t) \tag{1.6}$$

with the readout (1.5), where $\mathbf{M}$ and $\mathbf{N}$ are both matrices of size $N \times K$ for a network of size N and of rank K. They are trainable parameters, along with the input and output weights.

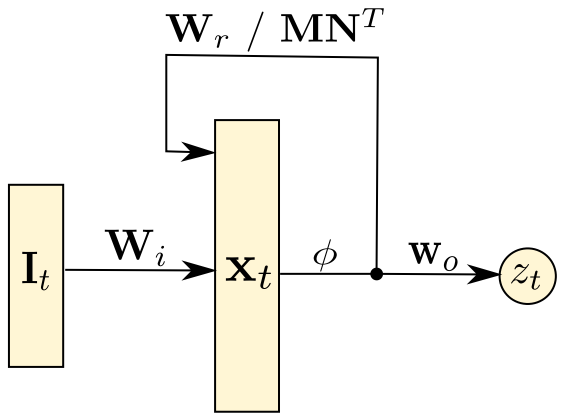


Figure 1.1: Simplified representation of the full and low rank RNNs used

1.2.1 Degeneracy

The decomposition of the recurrence weights in two matrices $\mathbf{W}_r = \mathbf{MN}^T$ introduces a degeneracy, since there are multiple ways to decompose the same $\mathbf{W}_r$ with this form. However, one way to make this decomposition unique is to force $\mathbf{M}$ and $\mathbf{N}$ to be orthogonal matrices, effectively making this a singular value decomposition of $\mathbf{W}_r$. Hence, we will apply after training a normalization step to our low-rank networks, applying first an SVD:

$$\mathbf{USV}^T = SVD(\mathbf{MN}^T)$$

and then setting:

$$M = \mathbf{U}\sqrt{\mathbf{S}}$$
$$N = \mathbf{V}\sqrt{\mathbf{S}}$$

where we apply the square root to each of the elements of the positive diagonal matrix $\mathbf{S}$.

1.2.2 Training tricks

Training so-called "vanilla RNNs" (ie. RNNs that do not use gating tricks, like our model) is notoriously hard in the field of machine learning. This is balanced by the fact that we train on tasks that are extremely simple compared to classical ML tasks. However, when focusing on very low-rank networks, training can still be very long to converge. Here are 3 tricks that we have found to be useful to accelerate training:

- Shaping: traditionally in behavioral experiments on animals, trainers start with simple tasks that are gradually complexified. This approach has been also exploited in ML, with so-called curriculum learning. We have exploited it, for example by training a network in the DMS task first with shorter delays and then gradually increasing the delay.

- Truncating a high-rank network: training a full-rank RNN is usually easier than a low-rank one. Interestingly, we noticed than when truncating the connectivity matrix of a full-rank network (by keeping the k largest singular values), the network can often perform equally well down to a rank of around 10, and that even when bringing it to an extremely low rank like 1 or 2, it then needs very little training to be able to accomplish the task again. It can thus be a good strategy to initialize our low-rank network with a structure taken from a high rank RNN. Moreover, this fact points to an interesting property of full rank RNNs that deserves to be studied in further work.
- Modification of the scalings: to ease gradient propagation, we have found useful to replace equation (1.3) by $P_{ij} = \sum_{k=1}^{K} m_i^{(k)} n_j^{(k)}$ (remove the $1/N$ factor) and initialize the coefficients to be of order $1/\sqrt{N}$ instead of 1.

Chapter 2

Rank one networks

To get a first intuition of the interest of studying low-rank networks, we will focus on the simplest case of rank one networks, and see how analytical solutions can shed light on empirical observations.

In the case of rank one networks, the equation (1.1) reduces to:

$$\tau \dot{x}_i(t) = -x_i(t) + \frac{1}{N} \sum_{j=1}^{N} m_i n_j \phi\left(x_j(t)\right) + I_i \tag{2.1}$$

Such a network is entirely specified by 2 vectors $\mathbf{m}$ and $\mathbf{n}$. Moreover, as explained above, for equation (2.1) with zero input, the activity decays exponentially toward the hyperplane spanned by $\mathbf{m}$ (a line). Indeed, the activity can be decomposed in:

$$\mathbf{x}(t) = \kappa(t)\mathbf{m} + \mathbf{x}^{\perp}(t)$$

with $\mathbf{x}^{\perp}(t) = \mathbf{x}_0^{\perp} e^{-t/\tau}$. Hence, the interesting behavior of the system is contained in the evolution of the scalar κ. The system can thus be reduced to a 1D differential equation.

2.1 Fixed points of the autonomous system

The fixed points of (2.1) with $\mathbf{I} = 0$ are aligned with $\mathbf{m}$, thus we can write them as $\mathbf{x}^{\star} = \kappa \mathbf{m}$ where κ is a solution of:

$$\kappa = \frac{1}{N} \mathbf{n}^T \phi(\kappa \mathbf{m}) \tag{2.2}$$

We will focus in this chapter on the case where the coefficients of $\mathbf{m}$ and $\mathbf{n}$ are sampled from gaussian distributions. In this case, the r.h.s. is a sigmoidal function (let us call it F), whose derivative with respect to κ is:

$$F'(\kappa) = \frac{1}{N}\mathbf{n}^T(\phi'(\kappa\mathbf{m}) \odot \mathbf{m}) = \frac{1}{N}\sum n_i\phi'(\kappa m_i)m_i \tag{2.3}$$

where $\odot$ represents the elementwise product.

The number of fixed points depends on this derivative in 0 (see figure 2.1), which is simply the overlap between the vectors $\mathbf{m}$ and $\mathbf{n}$:

$$F'(0) = \frac{1}{N}\mathbf{n}^T\mathbf{m} \tag{2.4}$$

- if this overlap is less than 1, then there is one single fixed point in 0
- if it is greater than one, there are 3 fixed points, one is 0 and the two other are $\pm\mathbf{x}^\star$.

An analysis of the jacobian would furthermore show that when there is only the fixed point in 0, it is stable, whereas when there are 3 fixed points, the 0 fixed point is unstable and the 2 other are stable, hence leading to a bistable system.

We also note that it can be easier to understand rank one networks as dynamics within a potential along the line spanned by $\mathbf{m}$ by writing:

$$\dot{\kappa} = -\kappa + \frac{1}{N}\mathbf{n}^T\phi(\kappa\mathbf{m}) = -\frac{dV}{d\kappa} \tag{2.5}$$

which we can then visualize as in figure 2.1. Here, stable fixed points are indicated by minima and unstable fixed points by maxima of the potential function.

2.2 Effect of a constant input

Let us now add to (2.1) a constant input $\mathbf{I}$. The dynamics can now unfold along the 2D plane spanned by $\mathbf{m}$ and $\mathbf{I}$. However, a further look shows that they actually develop on an affine 1D line. To see this, let us first orthogonalize $\mathbf{I}$ with respect to $\mathbf{m}$:

$$\mathbf{I} = I_\parallel\mathbf{m} + \mathbf{I}_\perp$$

Then take the decomposition:

$$\mathbf{x}(t) = \kappa(t)\mathbf{m} + \lambda(t)\mathbf{I}_\perp$$

By applying (2.1) and decomposing on the basis $\mathbf{m}, \mathbf{I}_\perp$, we obtain:

$$\tau\dot{\kappa} = -\kappa + \frac{1}{N}\mathbf{n}^T\phi(\kappa\mathbf{m} + \lambda\mathbf{I}_\perp) + I_\parallel$$
$$\tau\dot{\lambda} = -\lambda + 1$$

and we see that the second equation decays exponentially towards $\lambda = 1$.

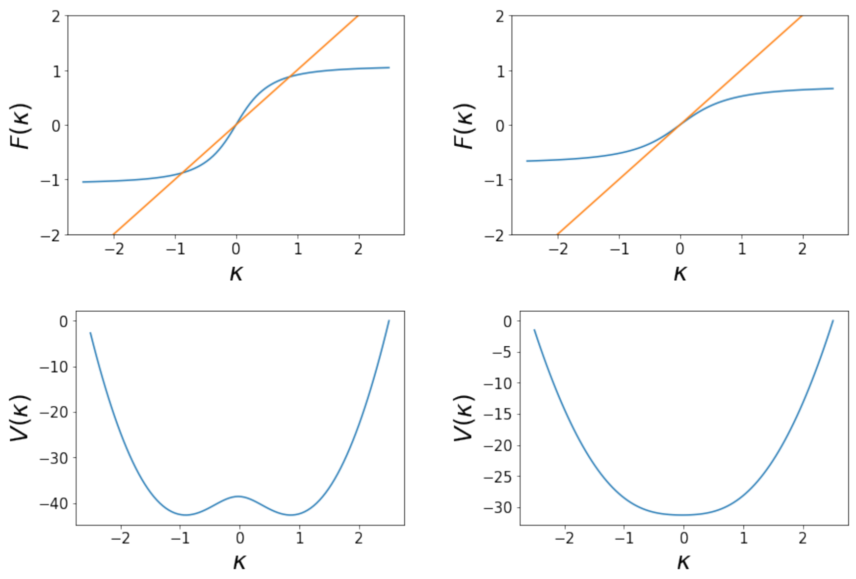


Figure 2.1: Top: function $F(\kappa)$ for 2 different values of overlap $\frac{1}{N}\mathbf{n}^T\mathbf{m}$, 2 on left and 0.8 on right. The intersection with the line $y = x$ in orange give the fixed points of the system. Bottom: corresponding potential plots.

Hence, the interesting aspects of the dynamics are actually contained in the affine line $\mathbf{I}_\perp + Vect(\mathbf{m})$ as can be seen in figure 2.2. This fact extends easily to higher rank networks and we will reuse it along the report.

Thus, we have shown that a constant input has the effect of displacing the line of the dynamics along an orthogonal direction of phase space. Moreover, we will show that it can interestingly modify the dynamics.

Indeed, the fixed points are now specified by κ verifying:

$$\kappa = \frac{1}{N}\mathbf{n}^T\phi(\kappa\mathbf{m} + \mathbf{I}_\perp) + I_\parallel \tag{2.6}$$

Let us first deal with the term $I_\parallel$: it adds a constant term to the flow on κ, hence translating the fixed points towards a certain direction, as one can visualize by imagining the sigmoid moving up in figure 2.1. If there were 3 fixed points, the unstable point and one of the stable points will eventually disappear in a saddle-node bifurcation. However, we will avoid this term in the remainder to facilitate analyses. Let us then simply replace equation (2.6) by:

$$\kappa = \frac{1}{N}\mathbf{n}^T\phi(\kappa\mathbf{m} + \mathbf{I}) \tag{2.7}$$

assuming $\mathbf{I}$ is orthogonal to $\mathbf{m}$. Let us denote the r.h.s. by $F_{\mathbf{I}}(\kappa)$.

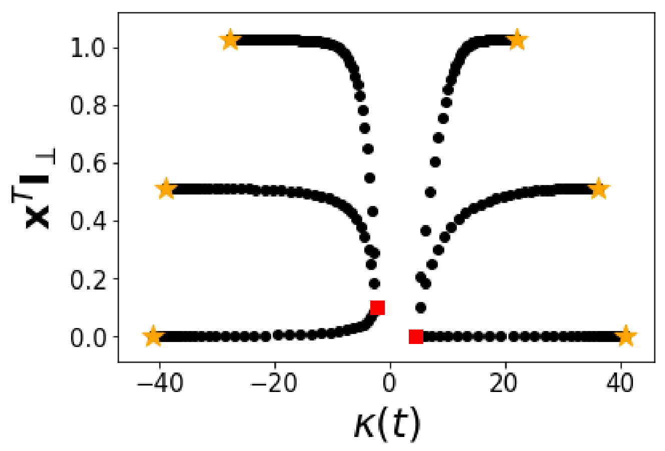


Figure 2.2: 2 sets of trajectories of vector $\mathbf{x}$ with 2 different starting points and 3 different input intensities (same input direction), shown in the plane spanned by $\mathbf{m}$ and $\mathbf{I}$ for a bistable network ($\frac{1}{N}\mathbf{n}^T\mathbf{m} = 2$). Trajectories go from red square to orange star. Notice how they quickly converge towards the stable affine subspace of the dynamics.

There can be 2 effects in this equation:

- if $\mathbf{I}$ has an overlap with $\mathbf{n}$, then the fixed points are shifted. Notably $F_{\mathbf{I}}(0) \approx \mathbf{n}^T\mathbf{I}$, while the input is sufficiently small, showing the sigmoid of figure 2.1 is actually shifted upwards for positive overlaps, downwards for negative ones. This can eventually lead to a saddle-node bifurcation between the unstable and one of the stable fixed points, and the activity going to the only stable fixed point remaining (see figure 2.3).

- if $\mathbf{I}$ is orthogonal to $\mathbf{n}$ we still have a fixed point in $\kappa = 0$ since $F_{\mathbf{I}}(0) = 0$ (assuming $\mathbf{n}$ is orthogonal to $\phi(\mathbf{I})$, which happens except in convoluted cases), and as we did for the autonomous system, it suffices to study the derivative $F'_{\mathbf{I}}(0)$. It turns out that we can write

$$F'_{\mathbf{I}}(0) = \frac{1}{N}(\mathbf{n} \odot \phi'(\mathbf{I}))^T\mathbf{m} = \frac{1}{N}\mathbf{n}_{eff}^T\mathbf{m} \tag{2.8}$$

  with $\odot$ the elementwise product of 2 vectors, and where we thus define $\mathbf{n}_{eff} = \mathbf{n} \odot \phi'(\mathbf{I})$. Thus, the dynamics with an input depend on what we will call the *effective overlap* between $\mathbf{m}$ and $\mathbf{n}$ in presence of an input. See figure 2.3 for an illustration.

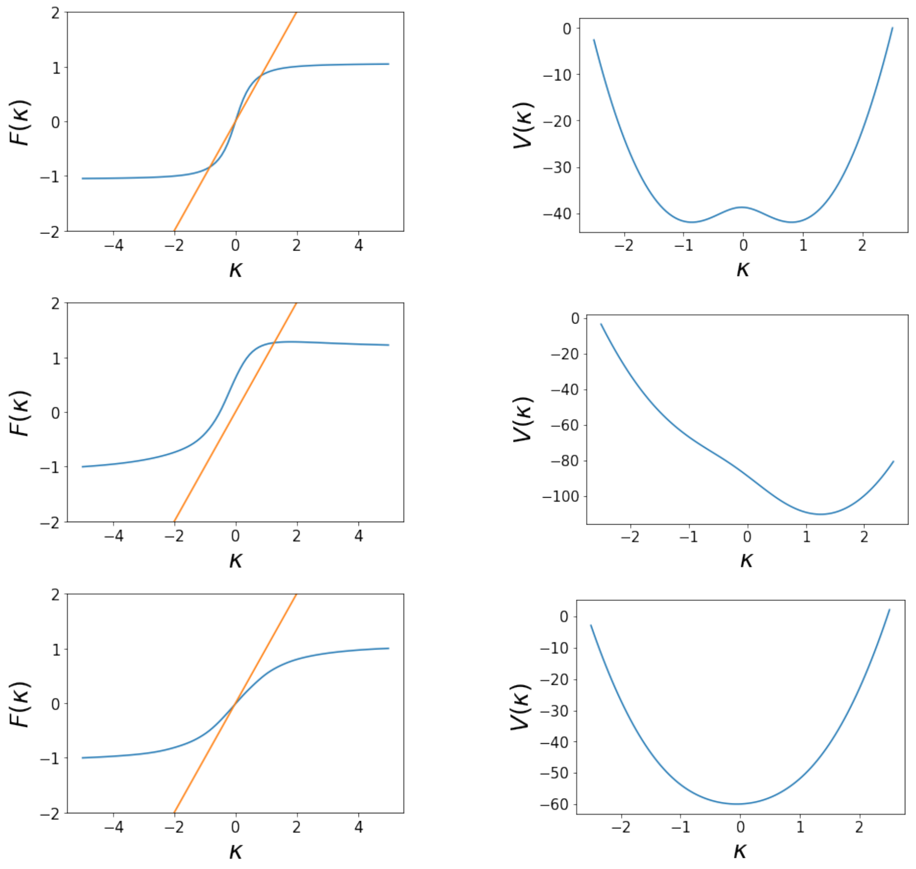


Figure 2.3: Effect of an input on rank one network. Top row: network without input. Middle row: input aligned with $\mathbf{n}$. Bottom row: input orthogonal to $\mathbf{n}$. On the left, plot of F function with $y = x$ line, on the right, potential plots of the flow on the stable affine line of the dynamics.

Chapter 3

Study of the random dots motion task

3.1 Task

We will illustrate how a rank 1 network can be trained and reverse engineered on an actual task, called random dots motion task (RDM). This analysis will be rather simple, but useful as a testbed to understand the more complex analysis done on the delayed match-to-sample task.

In the task as studied in cognitive neuroscience, the subject is in front of a monitor where, after a fixation period, a pattern of dots moving randomly appears, with a certain coherence. Negative coherence values indicate that the dots tend to move left, and positive values indicate that they tend to move right, while the absolute value indicates how strong the movement is (for a coherence value of 0, points move randomly in all directions).

The subject must decide whether the movement is more towards the left or right and indicate its choice by a saccade (either after a fixed period of time or when the subject chooses depending on the setup).

When the experiment is done on animals, standard sigmoidal psychometric curves are obtained, indicating that the task is correctly performed.

3.2 Network task

In our study, we are not interested in the visual processing part but rather in understanding how a neural network might be able to perform the decision making part of the task. Hence, we will assume that our network receives an input signal that indicates the perceived direction of motion at each instant, and that this signal will be a white noise with a mean sligthly positive or negative. It must give an output signal equal to +1 if the input mean was positive or -1 otherwise.

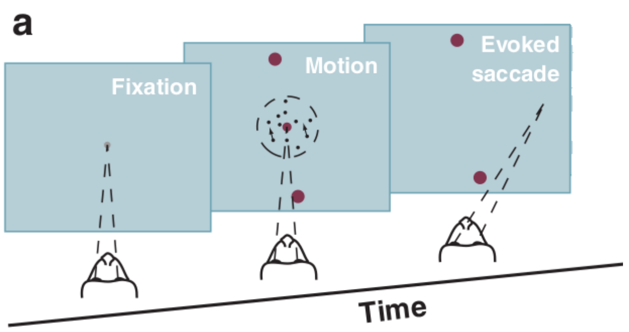


Figure 3.1: Task structure (adapted from [GS07])

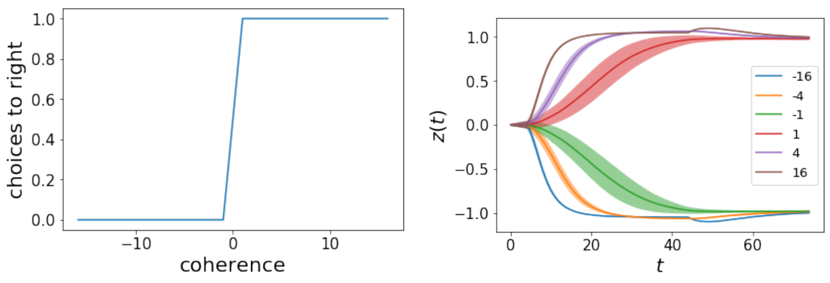


Figure 3.2: Left: psychometric curve for a trained rank one network. Right: readout mean and standard deviation for different values of motion coherence, for a trained rank one network.

3.3 Results

This task is very easy, and training a rank 1 network from scratch already gives perfect result, with a perfect psychometric curve, and outputs that converge towards the correct answer quicker depending on how hard it is to integrate the evidence (see figure 3.2)

A second step after training is to understand how the network effectively does the task. For this, we can look at the only eigenvalue of the network, which is between 1.1 and 1.5 for most trainings and is equal to the overlap between $\mathbf{m}$ and $\mathbf{n}$. This indicates (given our theoretical analyses), that the network is actually bistable, as can be confirmed by looking at the spontaneous activity under 0 input (thanks to neuronal noise, the networks eventually chooses one of the 2 stable states).

Hence, to do the task, the network only has to achieve 3 conditions:

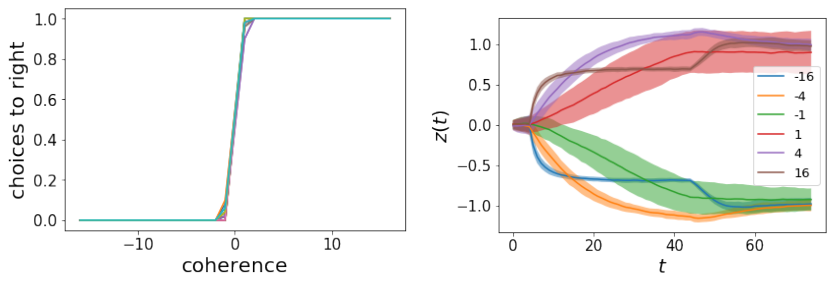


Figure 3.3: Left: set of 20 psychometric curves for networks defined by equations (3.1) - (3.4) with single set of parameters found after training and resampling of the gaussian basis ($N = 500$). Right: readout for one such network for different coherence values.

- the fixed point at 0 must be unstable (to force a choice) but sufficiently slow, so that the network acts as an integrator around 0. This can be done with an overlap $\frac{1}{N}\mathbf{n}^T\mathbf{m}$ slightly bigger than 1.
- the input must be aligned with $\mathbf{n}$ or $\mathbf{m}$ so that it is integrated.
- the readout vector must be aligned with $\mathbf{m}$ since the stable points lie on it, and its norm adjusted to have a value of ± 1 at these points.

Finally, we should test our hypothesis that this is a sufficient mechanism by reproducing "by hand" a network that is able to do the task. For this, we define a network defined by a very small number of parameters, with:

$$\mathbf{n} = \sqrt{\rho}\mathbf{x}_1 + \alpha_n \mathbf{x}_2 \tag{3.1}$$
$$\mathbf{m} = \sqrt{\rho}\mathbf{x}_1 + \alpha_m \mathbf{x}_3 \tag{3.2}$$
$$\mathbf{w}_o = w_o \mathbf{m} \tag{3.3}$$
$$\mathbf{w}_i = \mathbf{x}_4 + \sigma_n \mathbf{x}_2 + \sigma_m \mathbf{x}_3 \tag{3.4}$$

where the vectors $\mathbf{x}_i$ are random gaussian vectors (we assume that sampling coefficients from gaussian distributions is sufficient to retrieve the desired behavior). We simply train the free parameters ρ, α_n, α_m, w_o, σ_n, σ_m, so that they reach exact values giving good integration behavior. We find a network that is perfectly able to do the task, and remains able to do it whenever we resample the $\mathbf{x}_i$ and keep the parameters (see figure 3.3). Thus the computation is done simply because of this simple arrangement between the vectors defining the network.

One can note however some differences in behavior between the reproduced and the trained networks. It appears that the trained network moreover uses a mechanism with different populations that enables a finer control, but we will not detail it as it is not essential to the computation.

Chapter 4

Rank two networks

We will now give an overview of the behaviors that can be observed in a rank two network. For these networks, the equation (1.1) becomes:

$$\tau \dot{\mathbf{x}} = -\mathbf{x} + \frac{1}{N}(\mathbf{m}_1 \mathbf{n}_1^T + \mathbf{m}_2 \mathbf{n}_2^T)\phi(\mathbf{x}) + \mathbf{I} \tag{4.1}$$

Here the activity without input will decay towards the plane formed by $\mathbf{m}_1$ and $\mathbf{m}_2$, so that we can write:

$$\mathbf{x}(t) = \kappa_1(t)\mathbf{m}_1 + \kappa_2(t)\mathbf{m}_2 + \mathbf{x}_\perp(t)$$

with $\mathbf{x}_\perp$ quickly decaying to 0.

Moreover, as explained in section 1.2, the same network can be parametrized with different choices of $\mathbf{m}$ and $\mathbf{n}$ vectors due to a degeneracy, and in particular, one can always choose a parametrization based on a SVD such that $\mathbf{m}_1$ is orthogonal to $\mathbf{m}_2$, and $\mathbf{n}_1$ is orthogonal to $\mathbf{n}_2$. We will assume that it is always this parametrization that is chosen.

4.1 Phase portrait of the autonomous system

To understand the dynamics of the autonomous system (ie. without an input), it is sufficient to look at the dynamics on the plane $\mathbf{m}_1$, $\mathbf{m}_2$, that can be plotted as a flow field in 2D.

The equations of the dynamics on the 2D system are:

$$\tau \dot{\kappa}_1 = -\kappa_1 + \frac{1}{N}\mathbf{n}_1^T \phi(\kappa_1 \mathbf{m}_1 + \kappa_2 \mathbf{m}_2) \tag{4.2}$$

$$\tau \dot{\kappa}_2 = -\kappa_2 + \frac{1}{N}\mathbf{n}_2^T \phi(\kappa_1 \mathbf{m}_1 + \kappa_2 \mathbf{m}_2) \tag{4.3}$$

that we can write in compact form as $\tau \dot{\mathbf{k}} = F(\mathbf{k})$ where $\mathbf{k} = (\kappa_1, \kappa_2)$ and we define F as the r.h.s. of the equations above.

It is insightful to look at the jacobian of this equivalent system in 0:

$$F'(0) = \begin{pmatrix} \sigma_{11} - 1 & \sigma_{12} \\ \sigma_{21} & \sigma_{22} - 1 \end{pmatrix} \tag{4.4}$$

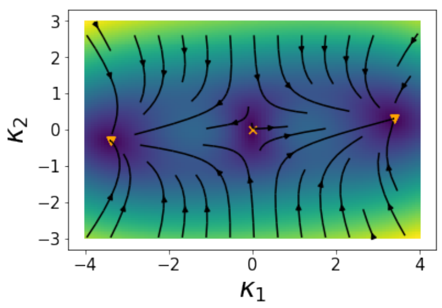
where $\sigma_{ij} = \frac{1}{N}\mathbf{n}_i^T \mathbf{m}_j$. The eigenvalues of $F'(0)$ determine the stability of the fixed point of the system at 0. Here are the possible behaviors of this system when the vectors $\mathbf{m}$ and $\mathbf{n}$ are sampled from Gaussians, illustrated in figure 4.1:

- if both eigenvalues have a negative real part, the 0 fixed point is stable, and is the only fixed point of the system: all trajectories end up at 0.
- if one eigenvalue has a positive real part, the other a negative one, the 0 fixed point is a saddle, and is thus stable in one direction, and unstable in another one (see figure 4.1a). On the manifold of unstability lie 2 stable fixed points, symmetric with respect to 0. Thus, the system is bistable.
- if both eigenvalues are real and positive and one is larger than the other, then 0 is an unstable fixed point, and four fixed points lie on the manifolds of unstability (see figure 4.1b). On the manifold associated with the larger eigenvalue, the fixed points are stable, but on the manifold associated with the smaller eigenvalue, the fixed points are saddles. Thus, the network remains bistable with moreover 2 saddles.
- if both eigenvalues are real and positive and equal, then the whole plane is formed of eigenvectors of the linearized system. Hence, we observe a ring attractor surrounding the origin. (see figure 4.1c). We see on the figure that the ring attractor is not perfect, but tends to exhibit a drift, due to finite size effects.
- finally, if the eigenvalues are non-real but have a positive real part (which corresponds to a strong asymmetry in the matrix $F'(0)$), we will find a limit cycle (see figure 4.1d).

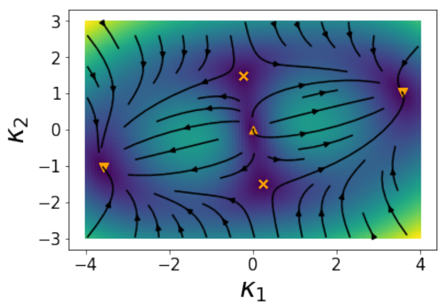
4.2 Effect of multiple populations

One can look at what happens beyond Gaussian distributions for coefficients, and it turns out that interesting behaviors beyond those described above start to be visible when one uses multimodal distributions. Equivalently, one can look at the behavior of a network composed of several populations of neurons, each characterised by Gaussian distributions of coefficients. Those two descriptions are mathematically equivalent.

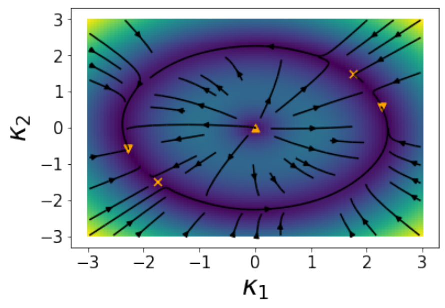
To illustrate this, let us analyze an example where we choose the joint distributions of both $(\mathbf{m}_1, \mathbf{n}_1)$ and $(\mathbf{m}_2, \mathbf{n}_2)$ to be bimodal. More precisely, let us



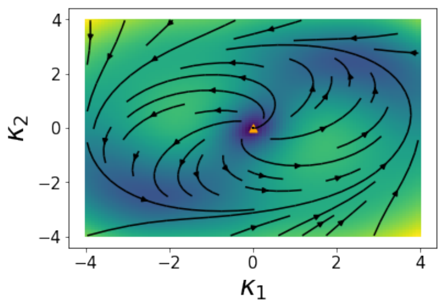
(a)



(b)



(c)



(d)

Figure 4.1: Flow charts for the automous system with Gaussian vectors and for 4 choices of parameters $(\sigma_{11}, \sigma_{12}, \sigma_{21}, \sigma_{22})$ ($N = 500$ neurons): (a) saddle with 2 stable points for values (2.5, 0, 0.5, 0.5); (b) 2 saddles and 2 stable points for values (2.5, 0, 0.5, 1.5); (c) ring attractor for (2, 0, 0, 2); (d) limit cycle for (2.5, -1, 1, 2) (eigenvalues are $2.25 \pm 0.97i$). The lines and arrows represent the flow $(\dot{\kappa_1}, \dot{\kappa_2})$ and the colors indicate the norm of the flow, small for blue and high for yellow.
Fixed points are found with a numerical solver and are shown with the following convention, used in the whole report: upwards triangles indicate sources (both eigenvalues of the jacobian have positive real part), crosses indicate saddles (the jacobian has one eigenvalue with positive real part and another with negative real part), and downward triangles indicate stable fixed points (both eigenvalues with negative real part)

take:

$$\mathbf{m}_1 = \sqrt{\sigma_{11}}\mathbf{z}_1 + \mathbf{x}_1 \tag{4.5}$$
$$\mathbf{n}_1 = \sqrt{\sigma_{11}}\mathbf{z}_1 + \mathbf{x}_2 \tag{4.6}$$
$$\mathbf{m}_2 = \sqrt{\sigma_{22}}\mathbf{z}_2 + \mathbf{x}_3 \tag{4.7}$$
$$\mathbf{n}_2 = \sqrt{\sigma_{22}}\mathbf{z}_2 + \mathbf{x}_4 \tag{4.8}$$

where we sample the $\mathbf{x}_k$ from gaussian distributions and the $\mathbf{z}_k$ from a bimodal distribution (for example a distribution giving 1 or -1 with probability 1/2). Let us for example choose $\sigma_{11} = \sigma_{22} > 1$. With only gaussian vectors, we should obtain a ring attractor, but instead we obtain a flow field as in figure 4.2, with 8 fixed points around the origin, 4 of them stable and 4 of them saddles.

An equivalent description of this model would be with 4 populations of vectors. Indeed, each neuron is characterized by the values $\mathbf{z}_1$ and $\mathbf{z}_2$ take on it. Thus we can define population 1 as the neurons for which $z_{1,i} = 1$ and $z_{2,i} = 1$, population 2 such that $z_{1,i} = 1$ and $z_{2,i} = -1$, population 3 such that $z_{1,i} = -1$ and $z_{2,i} = 1$, and finally population 4 such that $z_{1,i} = -1$ and $z_{2,i} = -1$. Hence, by ordering the neurons by population one gets connectivity vectors defined by population. For example for population 1:

$$\begin{aligned}\mathbf{m}_1^{(1)} &= 1 + \mathbf{x}_1\\ \mathbf{n}_1^{(1)} &= 1 + \mathbf{x}_2\\ \mathbf{m}_2^{(1)} &= 1 + \mathbf{x}_3\\ \mathbf{n}_2^{(1)} &= 1 + \mathbf{x}_4\end{aligned}$$

For population 2:

$$\begin{aligned}\mathbf{m}_1^{(2)} &= 1 + \mathbf{x}_1\\ \mathbf{n}_1^{(2)} &= 1 + \mathbf{x}_2\\ \mathbf{m}_2^{(2)} &= -1 + \mathbf{x}_3\\ \mathbf{n}_2^{(2)} &= -1 + \mathbf{x}_4\end{aligned}$$

and similarly we define vectors for populations 3 and 4. This structure becomes quite apparent if we look at the connectivity matrix after ordering neurons by populations as seen in figure 4.3.

Although this description seems complicated, the behavior of the network becomes trivially simple if one looks at the mean connexions between populations. For example, we have for the mean connexion between neurons of population 1:

$$g_{11} = \mathbb{E}[m_1^{(1)}n_1^{(1)} + m_2^{(1)}n_2^{(1)}] = 2$$

for the mean connexion from neurons of population 1 to population 2:

$$g_{12} = \mathbb{E}[m_1^{(2)}n_1^{(1)} + m_2^{(2)}n_2^{(1)}] = 0$$

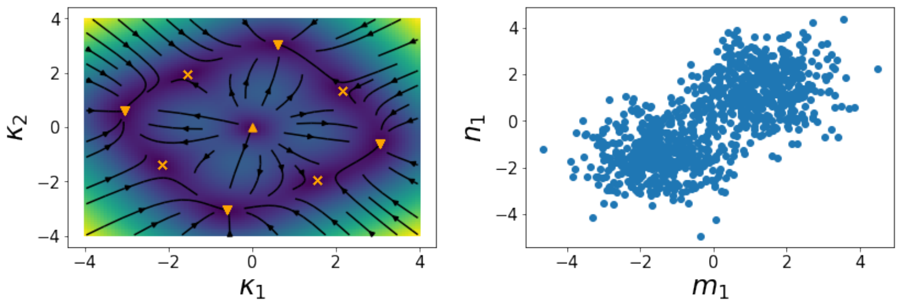


Figure 4.2: Left: low field for a network with bimodal joint distributions, displaying 8 fixed points around the origin. Right: scatter plot showing the bimodality in the joint distribution of m_1 and n_1, or equivalently the fact that there is more than one population of neurons.

and so on, where we compute the expectancies over the gaussian distributions from which the coefficients are sampled. This results in the mean relations between populations illustrated in figure 4.4, consisting of two independent pairs of populations with mutual inhibition, thus implementing winner-takes-all dynamics. Since the 2 pairs are independent and bistable, the whole network can be in four stable states, as seen in figure 4.2. Note however that the populations are not so strongly separated as in figure 4.4. The populations are in reality all connected, but with the mean effects outlined on the figure.

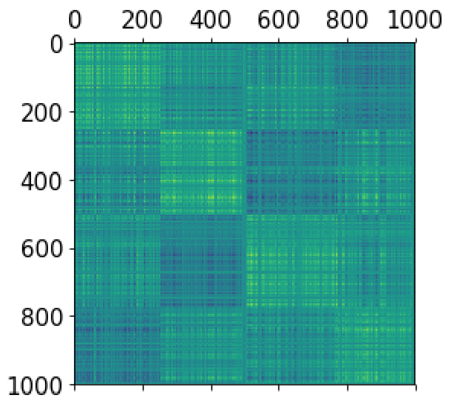


Figure 4.3: Connectivity matrix for the example network, with populations ordered in the same order as in text.

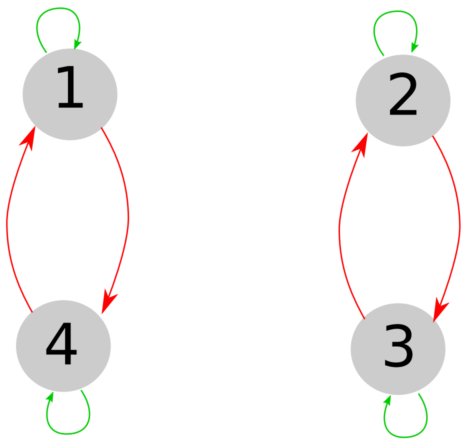


Figure 4.4: Equivalent model to the bimodal joint distributions network, with subpopulations. Green arrows: excitation, red arrows: inhibition.

4.3 Effect of a constant input

As for the rank 1 case, with a constant input we can write the decomposition

$$\mathbf{I} = I_1\mathbf{m}_1 + I_2\mathbf{m}_2 + \mathbf{I}_\perp$$

and write $\mathbf{x}(t) = \kappa_1(t)\mathbf{m}_1 + \kappa_2(t)\mathbf{m}_2 + \lambda\mathbf{I}_\perp$, which gives the equations:

$$\tau\dot{\kappa}_1 = -\kappa_1 + \frac{1}{N}\mathbf{n}_1^T\phi(\kappa_1\mathbf{m}_1 + \kappa_2\mathbf{m}_2 + \lambda\mathbf{I}_\perp) + I_1 \tag{4.9}$$

$$\tau\dot{\kappa}_2 = -\kappa_2 + \frac{1}{N}\mathbf{n}_2^T\phi(\kappa_1\mathbf{m}_1 + \kappa_2\mathbf{m}_2 + \lambda\mathbf{I}_\perp) + I_2 \tag{4.10}$$

$$\tau\dot{\lambda} = -\lambda + 1 \tag{4.11}$$

and shows that the activity quickly decays to an affine plane defined by $\mathbf{I}_\perp + Vect(\mathbf{m}_1, \mathbf{m}_2)$. Thus, in what follows, when we talk of a flow in presence of an input, we will consider the flow in that affine plane.

The effect of an input having overlaps with either the $\mathbf{m}_i$ or the $\mathbf{n}_i$ vectors is quite similar to the rank one case: such an input pushes the activity towards one direction, eventually making some fixed points of the activity disappear in saddle-node bifurcations. This is an important feature in computational tasks since it enables the network to select one of the fixed points when receiving a specific input.

However, another mechanism enables much more flexibility by modulating the entire phase portrait behavior when receiving a certain input, and we will focus on this mechanism since it is at the core of the study of the delayed match-to-sample task.

4.3.1 Effective overlap modulation

For this, we will simply consider inputs orthogonal to both the $\mathbf{m}_i$ or the $\mathbf{n}_i$ vectors. In this case, after having relaxed towards the stable affine plane of the activity, the equations are simply:

$$\tau\dot{\kappa}_1 = -\kappa_1 + \frac{1}{N}\mathbf{n}_1^T\phi(\kappa_1\mathbf{m}_1 + \kappa_2\mathbf{m}_2 + \mathbf{I}) \tag{4.12}$$

$$\tau\dot{\kappa}_2 = -\kappa_2 + \frac{1}{N}\mathbf{n}_2^T\phi(\kappa_1\mathbf{m}_1 + \kappa_2\mathbf{m}_2 + \mathbf{I}) \tag{4.13}$$

that we can write as $\tau\dot{\mathbf{k}} = F(\mathbf{k})$ and similarly to (2.8), the jacobian of the r.h.s. in 0, that we will abusively note $F'(\mathbf{I})$ is:

$$F'(\mathbf{I}) = \begin{pmatrix} \sigma_{11}^{eff} - 1 & \sigma_{12}^{eff} \\ \sigma_{21}^{eff} & \sigma_{22}^{eff} - 1 \end{pmatrix} \tag{4.14}$$

where

$$\sigma_{ij}^{eff} = \frac{1}{N}\mathbf{n}_{i,eff}^T\mathbf{m}_j = \frac{1}{N}(\mathbf{n}_i \odot \phi'(\mathbf{I}_\perp))^T\mathbf{m}_j \tag{4.15}$$

is the effective overlap in presence of the input between $\mathbf{n}_i$ and $\mathbf{m}_j$.

Thus, one way to change the phase portrait in presence of an input is to choose the correlations between the input and the $\mathbf{m}$ and $\mathbf{n}$ vectors carefully to obtain the desired effective overlaps. This is especially possible with different populations, as illustrated in the following example.

Let us study an example of a network with 2 populations of neurons, in which the phase portrait goes from bistable dynamics to a limit cycle in presence of an input. For each population of neurons we define overlaps $\sigma_{ij}^{(k)}$, where k represents the population. Let the $\mathbf{x}_l$ be standard random Gaussian vectors. We can build a population respecting the desired overlaps by defining:

$$\mathbf{n}_1^{(1)} = \sigma_{11}^{(1)}\mathbf{x}_1^{(1)} + \sigma_{12}^{(1)}\mathbf{x}_2^{(1)}$$

$$\mathbf{n}_2^{(1)} = \sigma_{21}^{(1)}\mathbf{x}_3^{(1)} + \sigma_{22}^{(1)}\mathbf{x}_4^{(1)}$$

$$\mathbf{m}_1^{(1)} = \mathbf{x}_1^{(1)} + \mathbf{x}_3^{(1)}$$

$$\mathbf{m}_2^{(1)} = \mathbf{x}_2^{(1)} + \mathbf{x}_4^{(1)}$$

and idem for population 2. The 2 populations can then be merged into a network simply by concatenating the vectors: $\mathbf{m}_1 = (\mathbf{m}_1^{(1)}, \mathbf{m}_1^{(2)})$ and id for others.

Let us choose the following parameters:

$$\sigma_{11}^{(1)} = \sigma_{11}^{(2)} = 3$$

$$\sigma_{22}^{(1)} = \sigma_{22}^{(2)} = 2.5$$

$$\sigma_{12}^{(1)} = \sigma_{12}^{(2)} = 1$$

$$\sigma_{21}^{(1)} = 1$$

$$\sigma_{21}^{(2)} = -1$$

In this network, the only difference between the two populations lies in the overlap between the vectors $\mathbf{m}_1$ and $\mathbf{n}_2$, that is positive in one case and negative in the other. With no input, the overlaps σ_{21} cancel out overall such that $\sigma_{21} = 0$, the jacobian is:

$$F'(0) = \begin{pmatrix} 3 & 1 \\ 0 & 2.5 \end{pmatrix}$$

and we obtain a phase portrait with four fixed points, 2 of them stable, as seen in figure 4.5.

Let us now choose an input $\mathbf{I}$ sampled from a gaussian with a high variance on population 1, and a small variance on population 2. For example:

$$\mathbf{I}^{(1)} = 10\mathbf{x}_5^{(1)}$$

$$\mathbf{I}^{(1)} = 0.1\mathbf{x}_5^{(1)}$$

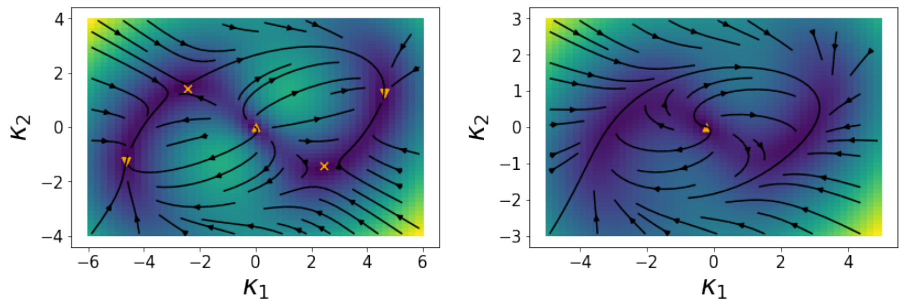


Figure 4.5: Phase portrait for the network given in example, without input on the left and with presence of an input on the right.

that we can interpret as meaning that population 1 receives with a high preference this input. The input has the effect of saturating preferentially the neurons of population 1 while leaving those of population 2 untouched. Thus, the effective overlap σ_{21}^{eff} becomes dominated by the overlap on population 2, since the term $\phi'(\mathbf{I}^{(1)}) \approx 0$ on population 1. Finally, $\sigma_{21}^{eff} \approx \sigma_{21}^{(2)} = -1$, and the jacobian in presence of an input becomes:

$$F'(\mathbf{I}) = \begin{pmatrix} 2 & 1 \\ -1 & 1.5 \end{pmatrix}$$

which is strongly asymmetric indicating the presence of a limit cycle. This is indeed what we find when simulating this model: the phase portrait that is bistable without input becomes a limit cycle in presence of the input (see figure 4.5).

Note that the fact that there are 2 distinct populations might not be readily apparent when looking at simple scatter plots of the coefficients, as one can see in figure 4.6. However, a more careful investigation can show that there is one population of neurons with a positive overlap, that tend to be more saturated in presence of the input, and a population of neurons with a negative overlap that are less saturated with the input. To show this, one can separate the neurons that are more saturated with the input ($\phi'(I_i) < 0.5$) and those that are less saturated ($\phi'(I_i) > 0.5$). It then appears that the saturated neurons form a group with a positive correlation between n_2 and m_1, while non-saturated neurons form a group with a negative overlap, as shown by the linear regressions on figure 4.6.

This example shows how an input can induce a very specific behavior, by transforming bistable dynamics in a limit cycle. Moreover, the method used to build this example network can easily be adapted to modify the jacobian as desired, provided neuronal populations are carefully chosen at the beginning.

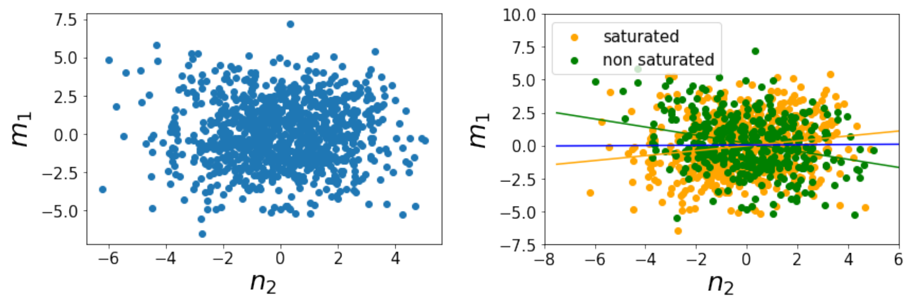


Figure 4.6: Left: coefficients n_2 plotted against m_1 for each neuron of the example model. Right: Same coefficients, separated into saturated and non-saturated neurons (see text), with linear regressions for each case, and linear regression for all neurons in blue.

Chapter 5

Study on the delayed match-to-sample task

We will now show that the mechanisms that we are able to understand analytically are powerful enough to understand how trained RNNs do a task. For this, we study the delayed match-to-sample task (DMS).

5.1 Task and setup

In classical cognitive neuroscience studies, the DMS task consists of presenting a first visual stimulus, that the subject must store in memory for a variable delay period, and then presenting a second stimulus, that must be compared to the first. The subject must give an answer depending on whether the stimuli are identical or not (figure 5.1).

This task combines a working memory aspect with a non-trivial computation (equivalent to doing a XOR). Although working memory has been rather well studied, notably in the context of attractor networks [AB97], there is little

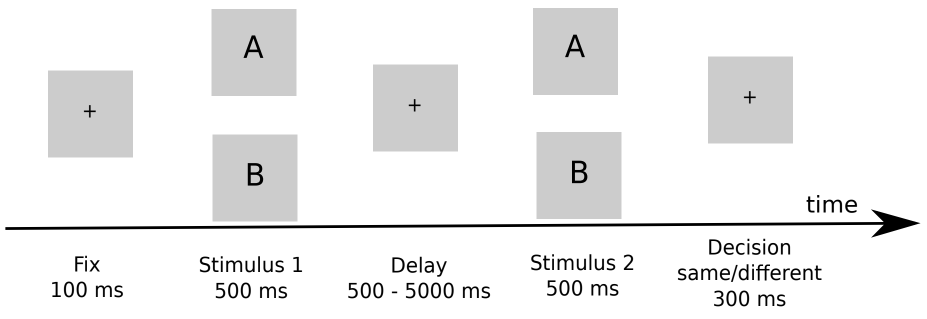


Figure 5.1: Task structure

litterature on the implementation of the computation [EW11].

As for the RDM task, we adapt it to a simplified setup: we assume that we only have 2 stimuli, A and B. The network has 2 input channels, u_A and u_B, that are set to 1 if the corresponding stimulus is being presented, and 0 otherwise. At the end of the trial, *ie* after the second stimulus has been presented, the readout must be +1 for a match, and -1 for a non-match.

Note that although the delay period is variable, forcing the network to implement persistent activity between the stimuli, the presentation periods of the stimuli are fixed, which enables our solutions to take advantage of the precise timing of this presentation. This fact will be discussed later.

We have attempted to find a rank 1 network able to do the task, but trainings have never converged to satisfying solutions. Thus, we hypothesize that the minimal dimensionality needed to do this task is 2.

When training rank 2 RNNs on this task, we have consistently found 2 solutions that we present here. Although for each solution only one example will be shown, actually multiple trainings have resulted in networks following the same mechanism.

5.2 A first solution

We will present a first solution that is easy to understand in terms of effective overlap modulation, and that has initially led us to postulate this mechanism. All figures are made from one particular trained network ($N = 500$ neurons).

The network performs correctly the task, and as can be seen from figure 5.2, its neurons present mixed and variable selectivities. A clearer picture of how the network performs the task appears when looking at the activity projected on the plane $(\mathbf{m}_1, \mathbf{m}_2)$, as one can see in the four trajectories in figures 5.3 and 5.4.

We see that in this solution, the network is bistable when it does not receive an input: notice the 2 stable fixed points in the first row of the panels. Receiving input A initially sends the network state towards its bottom stable point, and receiving input B sends it towards the top stable point. The interesting dynamics appear when the second stimulus is received: if it is input A, the bistability is maintained, thus leaving the network in the same stable state, whereas input B replaces the stable points by a limit cycle, thus sending the network state to the opposite fixed point. Thanks to this mechanism, the top stable point can correspond to the non-match situation, and the bottom one to the match situation.

To explain how this mechanism emerges from the weights of the network, one can first look at the effective jacobian of the system in 0, with and without input, by using equations (4.4) and (4.14), and by computing the overlaps on the trained network. We find:

$$F'(0) = \begin{pmatrix} 3.2 - 1 & -0.4 \\ 1.4 & 1.8 - 1 \end{pmatrix}$$

$$F'(\mathbf{I}_A) = \begin{pmatrix} 2.1 - 1 & 0 \\ 0.8 & 1.1 - 1 \end{pmatrix}$$

$$F'(\mathbf{I}_B) = \begin{pmatrix} 2.1 - 1 & -0.6 \\ 0.9 & 1.1 - 1 \end{pmatrix}$$

We thus find a mechanism similar to the one outlined in section 4.3.1, where the effective overlaps are modulated such that the jacobian becomes sufficiently asymmetric in presence of input B to induce a limit cycle. However, in presence of input A, the jacobian is shifted towards a more symmetric state, maintaining the bistability.

Input B seems to act very specifically on the term σ_{12}^{eff}. We hypothesize that input B acts similarly as the input in the example given in section 4.3.1: it tends to saturate more selectively the neurons that create a positive overlap between $\mathbf{n}_1$ and $\mathbf{m}_2$, ie neurons such that $n_{1,i}m_{2,i} > 0$, than those that create a negative overlap. To show this, let us plot the coefficients $n_{1,i}$ against $m_{2,i}$, and separate those that are more saturated in presence of input B ($\phi'(I_{B,i}) < 0.5$) and those that are less saturated ($\phi'(I_{B,i}) > 0.5$). As seen in figure 5.5, the saturated neurons tend indeed to be those inducing a positive overlap between these 2 vectors, leaving a negative overlap among the non-saturated neurons. One can also notice on figure 5.6 that the neurons that contribute to a positive overlap tend to be more saturated than the ones contributing to a negative overlap.

This mechanism can be understood if we postulate that the network is composed of 2 populations of neurons, one with $\sigma_{12} > 0$ that saturates in presence of input B, and one with $\sigma_{12} < 0$ that is insensitive to input B, as has been shown in the example of section 4.3.1.

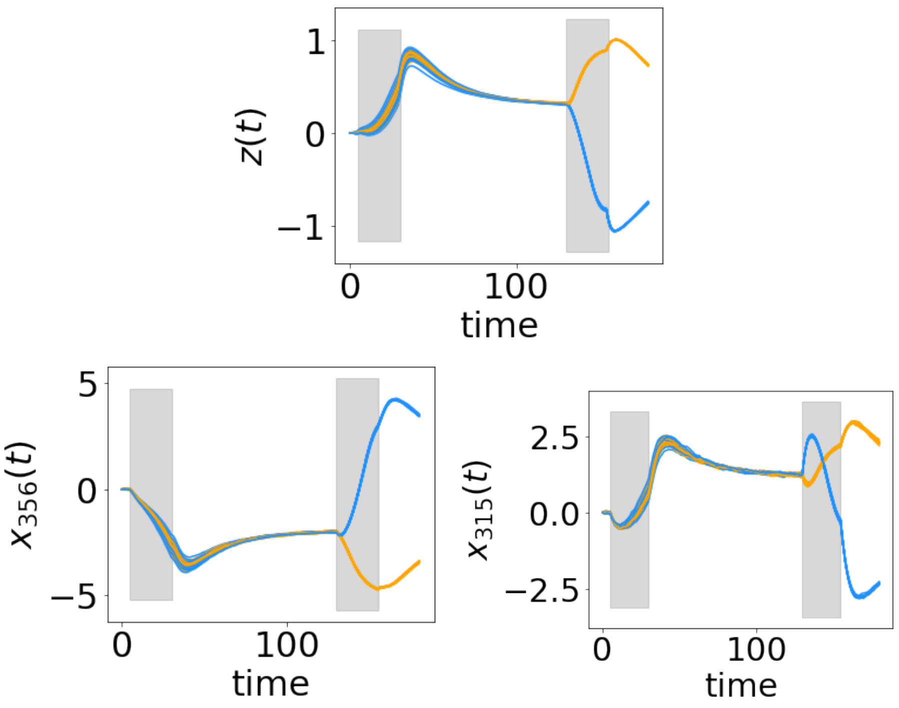


Figure 5.2: Network readout (top) and activity of 2 randomly sampled neurons (below) for multiple trials when shown stimuli A-A (orange traces) and A-B (blue traces). Stimulus presentation periods are indicated in gray

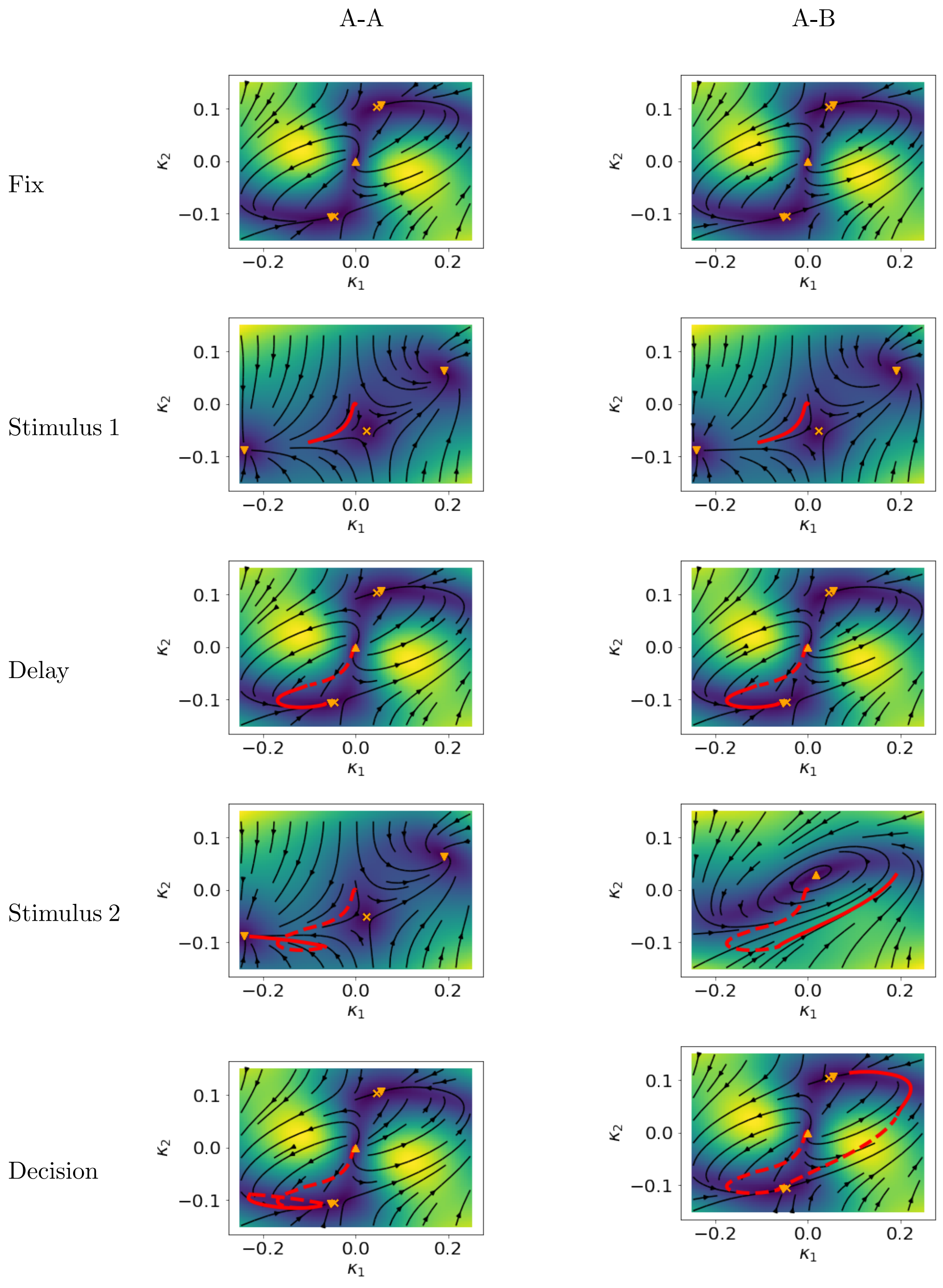


Figure 5.3: Trajectories on first DMS network, for presentations of stimuli A-A on left, and stimuli A-B on left.

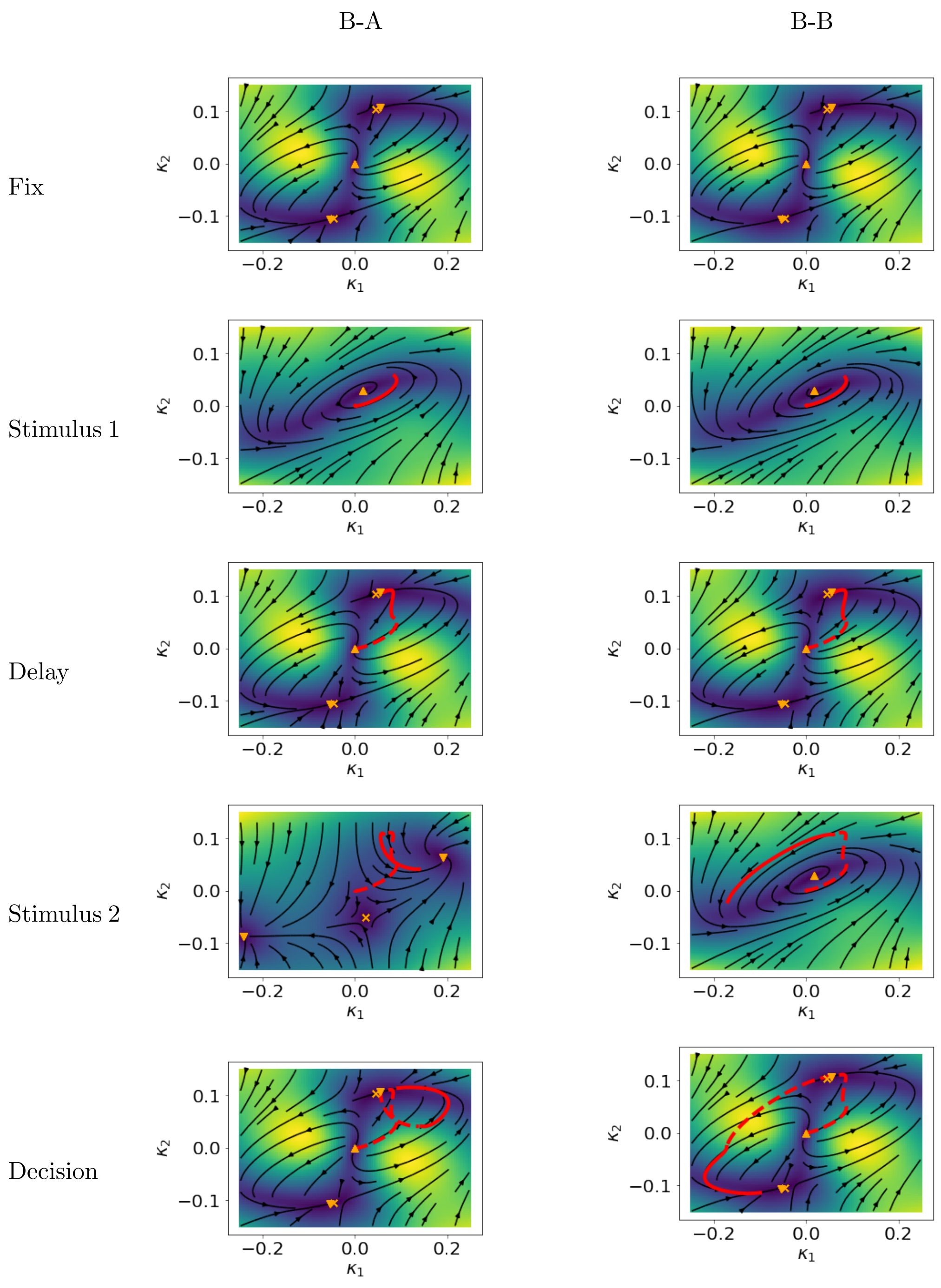


Figure 5.4: Trajectories on the first DMS network, for presentations of stimuli B-A on the left and B-B on the right.

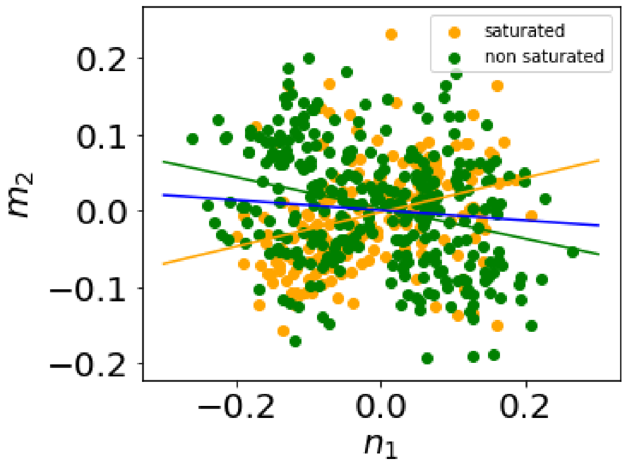


Figure 5.5: Coefficients $n_{1,i}$ against $m_{2,i}$ for each neuron, separated into saturated and non-saturated neurons (see text), with linear regressions for each group. In blue, the linear regression for the whole group.

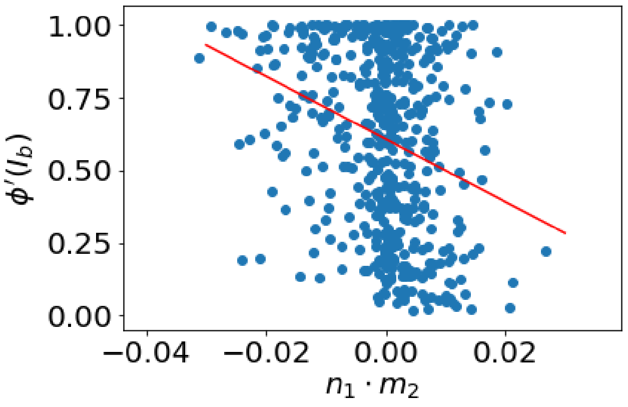


Figure 5.6: Product of coefficients $n_{1,i}m_{2,i}$ against saturation $\phi'(I_{B,i})$ for each neuron, with linear regression.

5.3 A second solution

A second solution has been found by training, albeit less often than the first one that has been presented. It displays other interesting mechanisms that are worth dissecting. As above, all plots are made for a specific trained network with $N = 500$ neurons.

This solution exhibits 4 stable fixed points in its autonomous state, separated by 4 saddles, as seen in figures 5.7 and 5.8. Two of the stable fixed points serve to encode the memory of the first stimulus, the top fixed point corresponding to stimulus B and the bottom one to stimulus A. Each stimulus has the effect of keeping only 2 stable fixed points, the ones at the left and right, and they each have the effect of separating the flow along a different diagonal of the phase portrait. Thus, if the network was at the fixed point encoding the memory of stimulus B for example, it will go in opposite directions if the second stimulus is A or B. Hence, the left fixed point can encode the match situation, and the right fixed point encode the non-match situation.

This network displays the kind of structure that has been described in section 4.2, hence it must some form of bimodality in its connectivity vectors. And indeed, the existence of this bimodality in the trained network is quite apparent when we look at the joint distribution of the coefficients of $\mathbf{m}_1$ and $\mathbf{n}_1$, as can be seen in figure 5.9.

When adding the inputs as we have said, the 2 vertical stable fixed points will merge with 2 of the saddles to create a bistable phase portrait. However depending on the input, the points will merge with the saddle points on one of the 2 diagonals. This can be done by modulating the rotational component of the phase portrait, that is always present in the asymmetry of the jacobian. In this example we find the following jacobians:

$$F'(0) = \begin{pmatrix} 2.1 & -0.5 \\ -0.1 & 1.5 \end{pmatrix}$$

$$F'(\mathbf{I}_A) = \begin{pmatrix} 1.15 & 0 \\ -0.3 & 0.5 \end{pmatrix}$$

$$F'(\mathbf{I}_B) = \begin{pmatrix} 0.9 & -0.4 \\ 0. & 0.5 \end{pmatrix}$$

We observe that with input A $\sigma_{12}^{eff} > \sigma_{21}^{eff}$ while with input B $\sigma_{12}^{eff} < \sigma_{21}^{eff}$. Hence, there is always an asymmetry in the network, but with opposite signs, which induces rotations in opposite directions with input A and B and explains why the stable points merge with different saddles in both cases. Although we assume that the two outlined phenomena explain the behavior of the network, we have not yet attempted to build it manually by lack of time.

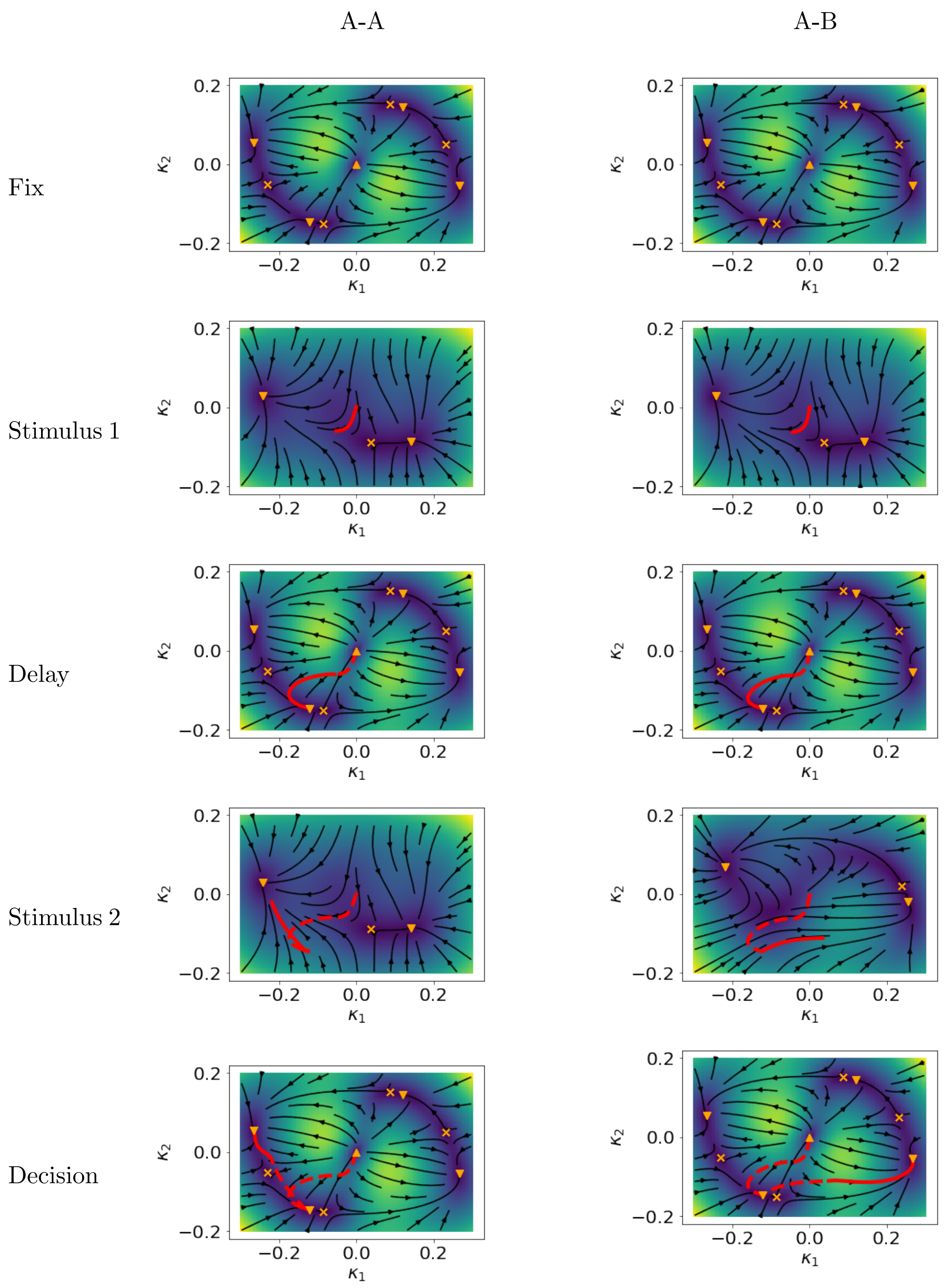


Figure 5.7: Trajectories on second DMS network, for presentations of stimuli A-A on left, and stimuli A-B on left. We can observe the 4 stable fixed points, the left one corresponding the match, the right one to non-match, and the bottom one to stimulus A.

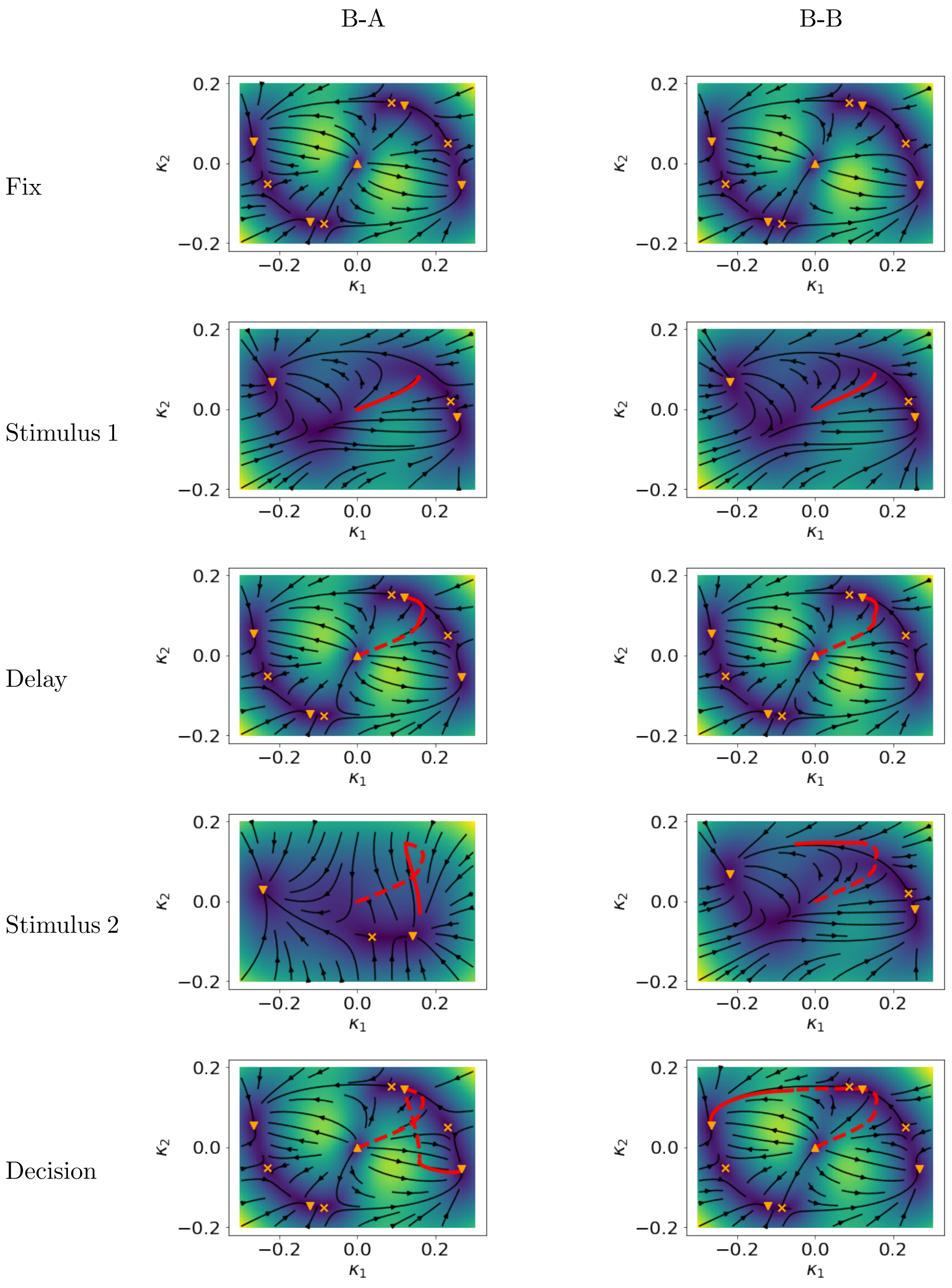


Figure 5.8: Trajectories on the second DMS network, for presentations of stimuli B-A on the left and B-B on the right. We see that the top fixed point serves as memory for stimulus B.

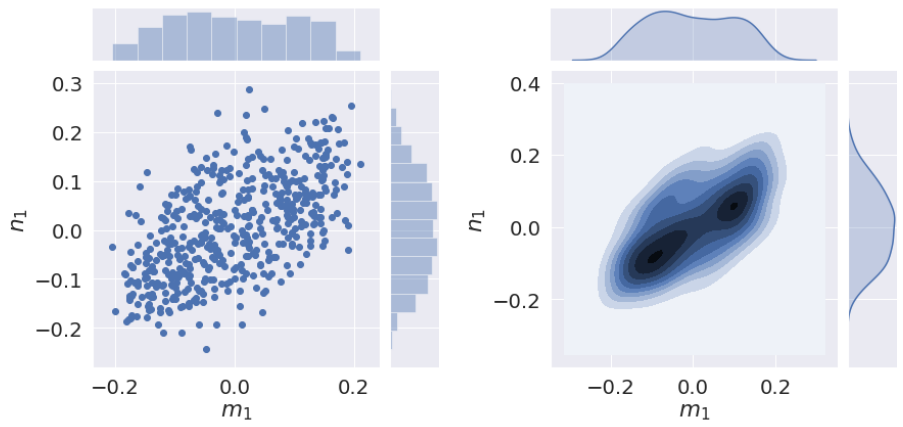


Figure 5.9: Joint distribution of the coefficients $m_{1,i}$ and $n_{1,i}$ for each neuron, scatter plot on the left, kernel density estimator on the right, displaying the bimodality that enables the 4 stable fixed points to appear.

5.4 Further work

As we have noted before, our task is designed such that the stimuli are always presented during a fixed interval of time, and our solutions heavily rely on this fact: in solution 1, if the stimulus presentation was longer, the activity would follow the limit cycle and end in the wrong fixed point. In solution 2, the first stimulus presentation has to be short enough so that the activity does not converge towards one of the fixed points with input, but it must end up near one of the stable points that disappear with input.

It would be interesting to investigate further if a solution independent of the precise timing of the presentation of the stimuli can be found in rank 2. Some attempts at training such a network have not been successful, so we hypothesize that a rank 3 network would be necessary. However such a solution could be not only insightful, but also more robust to perturbations of the task and easier to reproduce by hand since it does not rely on precise timing mechanisms.

Finally, it would be interesting to show how to extend these mechanisms to more than 2 inputs, and understand how the rank or the number of populations has to grow to deal with more inputs.

Conclusion

We have shown that low-rank recurrent networks offer an interpretable model while keeping powerful computational abilities. Notably, the dynamics of a rank k network quickly converge towards a k dimensional subspace of the state space, and can be understood via an equivalent k dimensional dynamical system. Moreover, even in presence of a constant input, the dynamics are simply translated towards a k dimensional affine subspace of the state space. Hence, rank 1 and 2 networks are particularly interesting since their phase portraits can be fully visualized, leading to interpretability of the observed activity.

Our strategy in this work has been to train low-rank RNNs on typical cognitive neuroscience tasks, reverse-engineer the obtained networks, and try to reproduce them. We have observed that for simple tasks like random dots motion, simple geometrical arrangements between the right and left eigenvectors of the connectivity matrix are sufficient to realise the task. However, for more complex tasks like delayed match-to-sample (DMS), trained networks exhibit multimodal distributions of connectivity coefficients, that can alternatively be interpreted as the presence of different populations in the network, and seem to be essential to the realisation of the task.

The interpretation in terms of populations has enabled us to reproduce complex dynamical features. For example, we have observed in one of the solutions to DMS that a bistable network became oscillatory in presence of an input, a phenomenon that could be reproduced by using 2 populations with different connectivity statistics. The fact that this biological feature emerges without being enforced during training suggests that it plays an essential computational role, enriching the role played by dimensionality.

The models that have been found with our procedure are particularily appealing because they do not involve assigning groups of neurons to task-specific roles as is done in many modelling studies [EW11]. Hence, the neurons of trained RNNs naturally exhibit the mixed and variable selectivities that are observed in neural recordings. To the question of the biological plausibility of low-rank networks, and to the issues raised by their non-sparse connectivity matrices, one could argue that they are a useful approximation of more complete RNN models. Indeed, observations tend to suggest that many features of full-rank RNNs can be retained by lower rank ones while making huge gains in interpretability.

Finally, we have shown that although seemingly intractable, the behavior of black-box networks realising complex tasks can be methodically dissected to reveal the whole mechanism leading from the connectivity matrix to the computation. We hope that this can form a basis for studying not only full-rank RNNs but also biological networks. Indeed, current models attempting to extract information from high-dimensional neural recordings are often descriptive (as dimensionality reduction algorithms, [CY14]), or difficult to interpret (like LFADS [POC$^+$18]). By understanding the dynamics of low-rank RNNs, we hope that it will be possible to build models that combine the interpretability of dimensionality reduction algorithms, with the predictive capacities of algorithms like LFADS.

To attain this goal, much work remains to be done, notably in continuing to characterize the different behaviors that can be found in low-rank networks. It will be particularly interesting to look at the solutions that can be found in rank 3 or 4 networks, that should be more difficult to interpret. It will also be interesting to disentangle the computational roles that are played by the dimensionality and by the cell populations, a work for which more task studies should help.

Bibliography

[AB97] D. J. Amit and N. Brunel. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cerebral Cortex*, 7(3):237–252, 1997.

[BM09] Dean V. Buonomano and Wolfgang Maass. State-dependent computations: spatiotemporal processing in cortical networks. *Nature Reviews Neuroscience*, 10(2):113–125, 2009.

[CY14] John P. Cunningham and Byron M. Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(11):1500–1509, 2014.

[EW11] Tatiana A. Engel and Xiao-Jing Wang. Same or different? a neural circuit mechanism of similarity-based pattern match decision making. *Journal of Neuroscience*, 31(19):6982–6996, 2011.

[GG15] Peiran Gao and Surya Ganguli. On simplicity and complexity in the brave new world of large-scale neuroscience. *Current Opinion in Neurobiology*, 32:148–155, 2015.

[GS07] Joshua I. Gold and Michael N. Shadlen. The neural basis of decision making. *Annual Review of Neuroscience*, 30(1):535–574, 2007.

[MO18] Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623.e29, 2018.

[MSSN13] Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013.

[POC$^+$18] Chethan Pandarinath, Daniel J. OShea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, Jaimie M. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805, 2018.

[SB12] David Sussillo and Omri Barak. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2012.

[Sus14] David Sussillo. Neural circuits as computational dynamical systems. *Current Opinion in Neurobiology*, 25:156–163, 2014.

[WNHJ18] Jing Wang, Devika Narain, Eghbal A. Hosseini, and Mehrdad Jazayeri. Flexible timing by temporal scaling of cortical responses. *Nature Neuroscience*, 21(1):102, 2018.

[Yus15] Rafael Yuste. From the neuron doctrine to neural networks. *Nature Reviews Neuroscience*, 16(8):487–497, 2015.