Tensor Analysis and the Dynamics of Motor Cortex

Jeffrey S. Seely

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

Abstract

Tensor Analysis and the Dynamics of Motor Cortex

by Jeffrey S. Seely

Neural data often span multiple indices, such as neuron, experimental condition, trial, and time, resulting in a tensor or multidimensional array. Standard approaches to neural data analysis often rely on matrix factorization techniques, such as principal component analysis or nonnegative matrix factorization. Any inherent tensor structure in the data is lost when flattened into a matrix. Here, we analyze datasets from primary motor cortex from the perspective of tensor analysis, and develop a theory for how tensor structure relates to certain computational properties of the underlying system. Applied to the motor cortex datasets, we reveal that neural activity is best described by condition-independent dynamics as opposed to condition-dependent relations to external movement variables. Motivated by this result, we pursue one further tensor-related analysis, and two further dynamical systems-related analyses. First, we show how tensor decompositions can be used to denoise neural signals. Second, we apply system identification to the cortex-to-muscle transformation to reveal the intermediate spinal dynamics. Third, we fit recurrent neural networks to muscle activations and show that the geometric properties observed in motor cortex are naturally recapitulated in the network model. Taken together, these results emphasize (on the data analysis side) the role of tensor structure in data and (on the theoretical side) the role of motor cortex as a dynamical system.

Contents

List of Figures

Acknowledgements

First and foremost, I would like to thank my parents, Scott and Katherine Seely, and my brother, Zachary Seely, for their love and support throughout graduate school.

Thanks to my undergraduate mentors, Patrick Crotty, Dan Saracino, and Vic Mansfield. Thanks to Professor Crotty for introducing me to computational neuroscience. Without him, I would not have taken this path. Professor Saracino deserves several pages worth of acknowledgment, but the following will have to suffice. He set a high bar, and few, if any, I have encountered in the research world have matched his effortless genius and his clarity in mathematical thinking. For every scientific problem I have worked on, my approach has been influenced by his example. A special thanks to Vic Mansfield. He has been a distinct example, a role model, in how head and heart need not be disconnected in a scientific career. I also have him, in part, to thank for deciding to go to graduate school; through him, I met the Dalai Lama, who in a conversation talked nonstop about the importance of scientific pursuits. I figured it was a sign, and Vic agreed. It seems fitting, then, to dedicate this thesis to the memory of professor Mansfield.

Thanks to my mentor Carson Chow, for his willingness to take me as a student, and providing the opportunities to do research before applying to graduate school.

Thanks to all of my friends and colleagues I met through the Churchland lab and the theory center. Yashar Ahmadian, Conor Dempsey, Brian DePasquale, Gamal Elsayed, Sean Escola, Hagai Lalazar, Kyo Iigaya, Patrik Kaifosh, Saul Kato, Antonio Lara, Grace Lindsay, Najja Marshall, Raoul-Martin Memmesheimer, Josh Merel, Andrew Miri, David Pfau, Abby Russo, Merav Stern, and Greg Wayne.

Thanks to the theory center alumni for being part of a network of colleagues to visit, no matter where I am in the world. Anthony Decostanzo, David Sussillo, and Taro Toyoizumi.

Thanks to my thesis committee members. Thanks to Ken Miller for his guitar performance at music night, and for keeping seminar speakers honest and clear through insightful questions. Thanks to John Cunningham for providing clarifying feedback in our collaborations. Thanks to Liam Paninski for creating an army of smart statisticians that I can rely on for advice. Thanks to Jonathan Pillow for being the superior tennis player in our doubles match in Okinawa.

Thanks to my advisors. Thanks to Mark Churchland for years of guidance and support, for giving me the opportunity to pursue interesting projects on some of the most interesting neuroscience datasets currently available. Thanks to Mark for always staying closely involved in this thesis work, in each stage of analysis and writing, to a degree that I appreciate to be exceptionally rare. Thanks to Larry Abbott for being the ultimate feedback control mechanism for finding the right questions to ask, and for finding the right approaches to solving them. Through his feedback, my scientific thinking has become a little less chaotic, and a little more coherent.

To Professor Vic Mansfield

Chapter 1

Introduction and Mathematical Preliminaries

In systems neuroscience, new experimental techniques are driving the collection of increasingly rich, high-dimensional datasets, requiring the development of new data analysis techniques as well as new theoretical paradigms for framing hypotheses.

Making sense of complex neural data will follow from the interaction between both efforts: data analysis techniques driven by theoretical paradigms and vice versa. This work contributes progress on both fronts. On the data analysis side, we borrow from the tensor decomposition literature and show how the tensor structure of certain neural datasets contain important information that is lost in matricization. On the theoretical side, we pursue a dynamical systems perspective for interpreting time-varying neural activity.

Tensor decompositions are utilized in a simple denoising applications in Chapter 3. The dynamical systems perspective is explored in the context of motor control in Chapters 4 and 5. The main contribution of this work resides at the intersection: Chapter 2 shows how the tensor structure of neural data can reveal whether time-varying neural activity is generated by internal dynamics or is driven by external inputs. We apply this theoretical result to data from motor and visual cortex. We show that time-varying activity in the motor cortex data is explained by internal dynamics, while time-varying

activity in the visual cortex data is explained by external inputs.

In this chapter, we lay mathematical foundations. A review of tensor analysis is presented first, followed by a review of linear dynamical systems. Their relationship becomes clear in Chapter 2.

1.1 Tensors

1.1.1 Intuition and definitions

Tensors are typically defined in one of two ways: as n-dimensional arrays or as multilinear functions. In data analysis, we may prefer to view tensors as n-dimensional arrays, but we can nevertheless borrow from the the theory of multilinear functions to build intuition. Therfore, we outline the relation between the two viewpoints.

A **tensor** is an n-dimensional array. Tensors are higher-order generalizations of vectors and matrices. A scalar is a zeroth order tensor, a vector is a first-order tensor, and a matrix is a second-order tensor. The **order** of a tensor is number of indices required to access an element of the array.

Alternativley, a tensor is a multilinear function, which can always be represented by an n-dimensional array. A function $f : V_1 \times \cdots \times V_n \to W$ is **multilinear** if it is linear in each of its arguments separately, where each V_i and W are vector spaces. That is, f is multilinear if and only if

$$f(v_1, \ldots, \alpha(v_i + v_i'), \ldots, v_n) = \alpha f(v_i, \ldots, v_i, \ldots, v_n) + \alpha f(v_1, \ldots, v_i', \ldots, v_n) \tag{1.1}$$

for all i. The corresponding array representation of f depends on a choice of bases in the vector spaces $V_1, \ldots, V_n$ and W. The array is sometimes called a "hypermatrix" to distinguish it from its associated tensor (multilinear function) [23]. To extract the array representation of f, consider the following

examples, where each $\mathbb{R}^i$ is given the canonical basis,

Function		**Array representation**
Linear functional:	$f : \mathbb{R}^n \to \mathbb{R}$	$1 \times n$ row vector
Linear function:	$f : \mathbb{R}^n \to \mathbb{R}^p$	$p \times n$ matrix
Bilinear functional:	$f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$	$m \times n$ matrix
Bilinear function:	$f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^p$	$p \times m \times n$ array
Trilinear functional:	$f : \mathbb{R}^l \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$	$l \times m \times n$ array

From the definition of multilinearity, it follows that (see [23]) a bilinear functional $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ can be written as $f(u, v) = \sum_{i,j} a_{ij} u_i v_j$, i.e. $f(u, v) = u^\top A v$, with $u \in \mathbb{R}^m$, $v \in \mathbb{R}^n$, and $A \in \mathbb{R}^{m \times n}$. The matrix A is thus a collection of coefficients for the terms $u_i v_j$. An extension to multiple outputs, $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^p$, is thought of as a collection of p single-output functions, thus the corresponding 3-dimensional array is a collection of p matrices of size $m \times n$. Examples of bilinear functions include matrix multiplication, the matrix determinant, inner products, and cross products. Bilinear functions are ubiquitious in mathematics, and it is worth stating their importance to help motivate why one might wish to consider multilinear functions more generally (i.e. why stop at order two?).

Trilinear functionals $f : \mathbb{R}^l \times \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ satisfying multilinearity (Eq. 1.1) can be written as $f(u, v, w) = \sum_{i,j,k} a_{ijk} u_i v_j w_k$. Here, the coefficients can be collected into a $l \times m \times n$ array, denoted $\mathcal{A}$. Unlike in the bilinear case, there is no way to write f succinctly using matrix-vector notation, since we need a more general notion of the transpose operation.

Thus, tensors can be viewed as either a multilinear function f, or as the collection of coefficients of the terms of f formatted in an array. From a data analysis standpoint, it is tempting to consider tensors as "just" arrays

of numbers. Yet even data arrays have an underlying corresponding function. For example, an $N \times T$ (neuron by time) matrix might be thought of as T samples of N-dimensional data vectors, but the matrix also represents a linear function $f : \mathbb{R}^T \to \mathbb{R}^N$. This function takes any T-dimensional vector (a neural response over time) and outputs an N-dimensional vector whose components are the dot products of each actual neuron's response and the putative input response. Thus, the $N \times T$ data matrix has an associated function f that computes the similarity of all possible neural responses (vectors in $\mathbb{R}^T$) with each of the observed responses. Similarly, the notion of matrix rank can be interpreted in the "data" viewpoint as the span of the data vectors; equivalently, in the "function" viewpoint, the rank is the dimension of the range of the function f. In all cases, the "data" viewpoint and "function" viewpoint coincide. The idea is that while we may perfer to stick with the data viewpoint, we can borrow from the theory of multilinear functions when discussing n-dimensional arrays of data.

1.1.2 Common occurances of tensors

Tensors arise most frequently in practice whenever data span multiple indices, such as neuron, time, stimulus, subject, and trial [9]. In this work, we will focus on these types of data tensors.

Tensors arise naturally in statistics as well. One can take outer products of a data matrix to form arrays containing higher-order statistics. Such a construction is often useful as the following example illustrates. For random variables x, y, and z, their third-order moment tensor $\mathcal{M}$ is defined as $\mathcal{M}_{ijk} = \mathbb{E}(x_i y_j z_k)$. Cumulant tensors are defined similarly. Random variables are independent if and only if their cumulant tensor is **diagonal** [10]—$\mathcal{A}_{ijk} = 0$ whenever $i \neq j$, $i \neq k$, or $j \neq k$. This observation has immediate

applications for "blind" problems, such as blind source separation or independent component analysis. A natural method for independent component analysis involves a decomposition of cumulant tensors [12]. More generally, tensor decompositions provide a general framework for learning latent variable models [3].

Tensors arise in even more familiar settings. For a multivariable scalar function $f : \mathbb{R}^n \to \mathbb{R}$, we can store the partial derivatives of order d in a d-th order tensor. The Jacobian is a first-order tensor $a_i = \frac{\partial f}{\partial x_i}$ which we denote as a row vector. The Hessian is a second order tensor $A_{ij} = \frac{\partial^2 f}{\partial x_i x_j}$. The third order tensor, $\mathcal{A}_{ijk} = \frac{\partial^3 f}{\partial x_i x_j x_k}$, and so on. The Taylor expansion of a function $f : \mathbb{R}^n \to \mathbb{R}$ is $f(x) = f_0 + f_1(x) + f_2(x, x) + f_3(x, x, x) + \dots$ where each f_i is a ith order multilinear function represented by arrays, $a_0, a_1, A_2, \mathcal{A}_3, \dots$. Thus, tensors naturally arise even in the simplest settings—as terms in the Taylor expansion of a function.

1.1.3 Tensor notation

Scalars and vectors will be denoted by lowercase letters, e.g. a. Matrices will be denoted by uppercase letters, e.g. A. Higher-order tensors will be denoted by script letters, e.g. $\mathcal{A}$. We denote the (i, j, k) entry of the third-order tensor $\mathcal{A}$ by $\mathcal{A}_{ijk}$.

Subtensors are denoted using colon notation. The ith row of matrix A is denoted by $A_{i:}$, while the jth column is denoted by $A_{:j}$.

The vector subtensors of higher-order tensors are sometimes called **fibers**: $\mathcal{A}_{:ij}$, $\mathcal{A}_{i:j}$ and $\mathcal{A}_{ij:}$ refer to mode-1, mode-2, and mode-3 fibers, respectively.

Matrix subtensors are called **slices**, e.g. $\mathcal{A}_{::i}$, $\mathcal{A}_{:i:}$ and $\mathcal{A}_{i::}$.

1.1.4 Tensor operations

Many tensor operations can be understood in terms of **matricization** or flattening. A tensor $\mathcal{X} \in \mathbb{R}^{N \times C \times T}$ has three mode-n matricizations

$$X_{(1)} \in \mathbb{R}^{N \times CT} \tag{1.2}$$

$$X_{(2)} \in \mathbb{R}^{C \times NT} \tag{1.3}$$

$$X_{(3)} \in \mathbb{R}^{T \times NC} \tag{1.4}$$

The mode-1 matricization, $X_{(1)}$, consists of all N-dimensional fibers of $\mathcal{X}$ (all CT of them) arranged side-by-side to create a $N \times CT$ matrix. The order of the vectors is inconsequential, but by convention ordering is lexicographic. Consistent ordering is necessary to define the inverse operation, **tensorization**.

The n**-mode product** of a tensor $\mathcal{X}$ with a matrix U, denoted $\mathcal{X} \times_n U$, defines multipliplication by a matrix "along" the nth mode of a tensor. For instance, the 1-mode product of $\mathcal{X} \in \mathbb{R}^{N \times C \times T}$ with $U \in \mathbb{R}^{K \times N}$ is

$$(\mathcal{X} \times_1 U)_{k,c,t} = \sum_i \mathcal{X}_{i,c,t} U_{k,i} \tag{1.5}$$

with 2-mode and 3-mode products defined similarly. Another approach to defining the n-mode product is to note the following one-to-one correspondence [22]:

$$\mathcal{Y} = \mathcal{X} \times_n U \quad \iff \quad Y_{(n)} = U X_{(n)} \tag{1.6}$$

That is, $\mathcal{X} \times_n U$ amounts to performing the mode-n unfolding of $\mathcal{X}$, multiplying on the left by U, then reshaping the result back into a tensor.

Since matrices are second order tensors, we can recast familar matrix operations using the above notation. For example, we can rewrite the matrix

SVD, $A = USV^\top$ as $A = S \times_1 U \times_2 V$. We can write the bilinear form $x^\top Ax$ as $A \times_1 x \times_2 x$. Note that for a matrix A, $A_{(1)} = A$ and $A_{(2)} = A^\top$.

1.1.5 Neuron by condition by time tensors

Our primary object of study is a tensor indexed by neuron, experimental condition, and time. We let N, C, and T correspond to the total number of neurons, conditions, and time points, respectively.

Throughout the systems neuroscience literature, datasets are often recorded across many neurons, conditions, and times [24, 7, 15]. Recording from multiple neurons is motivated by the idea that information is represented at the population level—a vector in $\mathbb{R}^n$—as opposed to the single-neuron level. Capturing response variation across time, as opposed to looking at static responses or time-averaged firing rates, is necessary for inferring dynamic properties of computation. Recording responses across multiple experimental conditions captures important relationships between stimuli/behaviors and neural responses.

Thus, to capture these three goals simultaneously, one must record data across each of the three indices. Yet in many classic studies, it is not uncommon to conceptually fix at least one of the indices while only studying variation across the remaining indices. For example, classic studies of visual cortex revealed that a neuron's response depends on the angle of contrast orientation within that neuron's ("classical") receptive field [17]. Other studies of visual cortex might focus primarily on response dynamics [32, 31]. Other approaches might focus on responses across neurons and conditions, but not time, as in studies of the topographic organization of preferred directions [38].

Many datasets are simplified by averaging across time, or even averaging across neurons, to create simpler objects to study—e.g. $N \times C$ matrices as

in the analysis of population vectors in motor cortex [15]. In other cases, datasets may include multiple neurons, conditions, and times, but analyses focus on one of the three matrix unfoldings, ignoring relationships among the others indices. For example, it is common to matricize an $N \times C \times T$ tensor into an $N \times CT$ matrix to perform principal component analysis or other dimensionality reduction techniques [11].

By studying variation across one index at a time, there is no doubt that neuroscience has made substantial progress. Yet here we will demonstrate the strength of data analysis techniques that consider the structure across all three indices *simultaneously*. Recent studies that take advantage of variation across all indices include demixed principal component analysis [6, 21] and various tensor decomposition work in neuroimaging [9].

Not all data are naturally formatted as third-order arrays. Datasets where different neurons/conditions correspond to different time-lengths have no natural $N \times C \times T$ format. Each individual neural response across time is in a vector space $\mathbb{R}^{T_{n,c}}$ of different dimension, rendering tensor analyses inapplicable. Often, this situation can be rectified: lock the data to an experimental cue (stimulus onset, behavioral onset, etc.) and consider a fixed number of time points before and after the cue. More sophisticated methods can also be used, such as dynamic time warping [27] to ensure equal time lengths.

For a dataset $\mathcal{X} \in \mathbb{R}^{N \times C \times T}$, we can interpret the meaning of each of the three vector spaces $\mathbb{R}^N$, $\mathbb{R}^C$, and $\mathbb{R}^T$, as well as each of the three spaces corresponding to the matrix slices, $\mathbb{R}^{C \times T}$, $\mathbb{R}^{N \times T}$, and $\mathbb{R}^{C \times T}$.

Vectors in $\mathbb{R}^N$ correspond to population response patterns—the response of an entire neural population at a particular condition and particular time point. Vectors in $\mathbb{R}^C$ are **tuning functions**—how a particular neuron and a particular time depends on experimental condition, which could be labeled

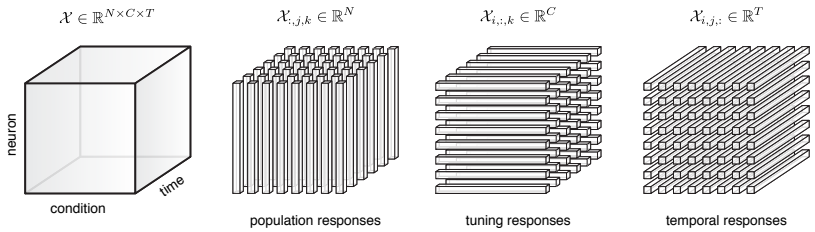

FIGURE 1.1: Neuron by condition by time tensor and its vector fibers.

simply as $c_1, c_2, \ldots$, or by experimental parameters associated with that condition, such as $\theta_1, \theta_2, \ldots$. Vectors in $\mathbb{R}^T$ are response patterns over time, usually called a peri-stimulus time histogram (PSTH). Each of these three vector spaces thus have an natural interpretation familiar to the neuroscience community.

Slices of $\mathcal{X}$ include $\mathcal{X}_{n::}$—the response of one neuron n across all conditions and times—$\mathcal{X}_{:c:}$—the response of all neurons and times for a particular condition c—and $\mathcal{X}_{::t}$—a snapshot of all neurons and conditions for a particular time t.

For a third-order tensor, vector fibers and matrix slices are linked by the row-column "duality" of matrices in the following sense. The columns of $X_{(1)}$ are vectors in $\mathbb{R}^N$, while the rows are vectors in $\mathbb{R}^{CT}$, which can be reshaped to matrices of size $C \times T$—the slices $\mathcal{X}_{n::}$. The span of both the rows and columns of $X_{(1)}$ have the same dimension and are in some sense different views of the same data. Yet, the vector spaces $\mathbb{R}^N$, $\mathbb{R}^C$, and $\mathbb{R}^T$ are *not* linked by this duality. In other words, the matrices $X_{(1)}$, $X_{(2)}$, $X_{(3)}$ can each have different ranks for a given tensor. This is a fundamental difference between matrices and tensors. From the standpoint of analyses based on SVD, matrices (2nd order tensors) have one story to tell, while 3rd order tensors have three.

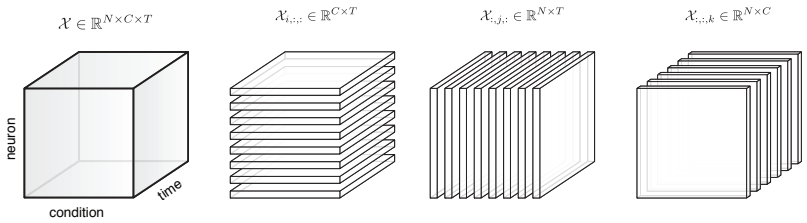

FIGURE 1.2: Matrix slices of $N \times C \times T$ tensors.

1.1.6 Tensor decompositions and tensor rank

The main tools of tensor-based data analysis are tensor decompositions. Like matrix decompositions, tensor decompositions decompose a dataset into simpler, potentially more interpretable parts. Tensor decompositions are also used to reveal the tensor rank of a dataset—a succinct numerical summary of the complexity of the tensor as a whole or across different modes. One only needs to look toward the ubiquity and utility of matrix decompositions to motivate tensor decompositions. However, generalizing decompositions from second order to higher order arrays introduces numerous subtleties. As stated above, there is no single generalization of the matrix SVD—the canonical matrix decomposition—leading to different notions of tensor rank, and introducing more choices to any data analysis procedure. Yet, surprisingly, some higher-order tensor decompositions enjoy uniquness properties not available to second order arrays.

Recall the matrix singular value decomposition:

$$A = USV^\top = \sum_r s_r u_r v_r^\top \tag{1.7}$$

where $UU^\top = I$, $VV^\top = I$, $S = \mathrm{diag}(s)$ for a vector of singular values s. The vectors u_r and v_r are the rth columns of U and V, respectively. The matrix SVD says that any linear transformation can be decomposed into an

orthonormal transformation (V), followed by a scaling of coordinates (S), followed by an orthonormal transformation (U). Equivalently, any linear transformation can be decomposed into the sum of mutually orthogonal rank-1 transformations.

It is straighforward to generalize the 'sum of rank-1 components' notion to higher order tensors:

$$\mathcal{X} = \sum_{r}^{R} u_r \circ v_r \circ w_r \tag{1.8}$$

where $\circ$ denotes the outer product, i.e. $u_r \circ v_r \circ w_r$ forms a third-order array for $u \in \mathbb{R}^N$, $v \in \mathbb{R}^C$, and $w \in \mathbb{R}^T$ (to follow our neuron-by-condition-by-time example). The **rank** of $\mathcal{X}$ is the smallest R for which the above decomposition holds. This decomposition is known as the canonical polyadic (CP) decomposition. Typically, one uses this decomposition when the rank-1 components are expected to be interpretable in some way. For third order tensors (and higher), a CP decomposition cannot be performed in general while simultaneously requiring that $UU^\top = I$, $VV^\top = I$, and $WW^\top = I$. Thus, there is no single generalization of the matrix SVD.

In terms of matricization, we can write the CP decomposition as follows:

$$X_{(1)} = U(W \odot V)^\top \tag{1.9}$$

$$X_{(2)} = V(W \odot U)^\top \tag{1.10}$$

$$X_{(3)} = W(V \odot U)^\top \tag{1.11}$$

where $\odot$ denotes the Khatri-Rao product (see [22]). The matricized versions of tensor decompositions allow for straightforward numerical implementations.

The second generalization of the matrix SVD is known as the Tucker decomposition:

$$\mathcal{X} = \sum_p \sum_q \sum_r \mathcal{S}_{pqr}\ u_p \circ v_q \circ w_r \tag{1.12}$$

$$\mathcal{X} = \mathcal{S} \times_1 U \times_2 V \times_3 W \tag{1.13}$$

In terms of matricization, the Tucker decomposition can be written as:

$$X_{(1)} = US_{(1)}(W \otimes V)^\top \tag{1.14}$$

$$X_{(2)} = VS_{(2)}(W \otimes U)^\top \tag{1.15}$$

$$X_{(3)} = WS_{(3)}(V \otimes U)^\top \tag{1.16}$$

We can further require that U, V, and W are orthonormal.

A simple procedure for performing a Tucker decomposition is known as **higher-order SVD (HOSVD)**: obtain U, V, and W as the left singular vectors of $X_{(1)}$, $X_{(2)}$, and $X_{(3)}$, respectively. Then, obtain $\mathcal{S}$ as $\mathcal{S} = \mathcal{X} \times_1 U^\top \times_2 V^\top \times_3 W^\top$. The **truncated HOVSD** is obtained as,

$$\mathcal{X} \approx \mathcal{S}_{1:p,1:q,1:r} \times_1 U_{:,1:p} \times_2 V_{:,1:q}, \times_3 W_{:,1:r} \tag{1.17}$$

for a rank-(p, q, r) approximation of $\mathcal{X}$. The tuple (p, q, r) is the **multilinear rank** of $\mathcal{X}$ when the approximation is exact. The multilinear rank is the second generalization of tensor rank after the CP rank (or just rank) defined above. More simply, the multilinear rank of $\mathcal{X}$ is defined as the tuple of ranks of each of its unfoldings.

One distinguishes between HOSVD and Tucker decompositions in the following sense. For a given choices of (p, q, r), the choices of $\mathcal{S}, U, V, W$ in

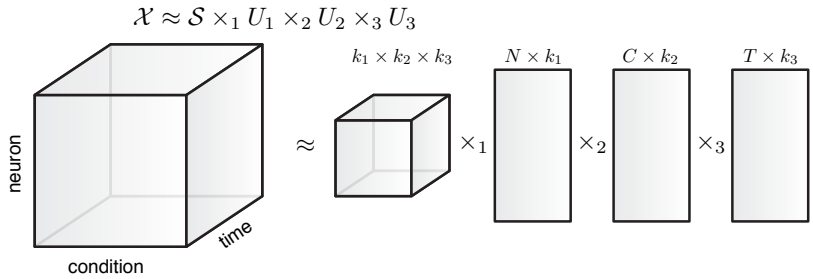

FIGURE 1.3: Truncated higher-order SVD.

the minimization problem

$$\begin{aligned} \underset{\mathcal{S},U,V,W}{\text{minimize}} \quad & \|\mathcal{X} - \mathcal{S} \times_1 U \times_2 V \times_2 W\|_F^2 \\ \text{subject to} \quad & U^\top U = I, \\ & V^\top V = I, \\ & W^\top W = I \end{aligned} \tag{1.18}$$

are not given by HOSVD, unlike in the matrix case. The Tucker decomposition emphasizes that any choices of $\mathcal{S}, U, V, W$ can be made—perhaps one that approximates $\mathcal{X}$ more closely than that obtained by HOSVD.

The literature on tensor decompositions is rich and this overview only scratches the surface. There is significant emphasis placed on the constraints of the factor matrices U, V, W—either in the CP or Tucker setting—leading to, for example, nonnegative tensor decompositions [8]. One can mix and match constraints [33]; e.g. a nonnegative constraint on U, an orthonormal constraint on V, and a sparsity constraint on W. Implementing *structure* such as diagonal, Toeplitz, Hankel, etc. on the factor matrices offer even further flexibility. A block-Hankel structure on the temporal factor matrix, for example, can be used in the context of linear system identification [29]. Factor matrices can also be shared across different datasets/tensors, leading to what is known as data fusion [33]. There is also extensive literature on tensor train decompositions, which allow for efficient decompositions of tensors with very high order [28].

1.2 Linear Dynamical Systems

The subject of dynamical systems is central in the following chapters. In particular, we will study dynamical systems as observed across multiple conditions, resulting in data objects that can be formatted into $N \times C \times T$ tensors. In particular, we will ask whether a linear dynamical system that is primarily input-drive or primarily autonomous best account for data. Further, we will show that this distinction essentially depends on the multilinear rank of the $N \times C \times T$ data. With this motivation, let us review the subject of linear dynamical systems.

1.2.1 Overview

Often, one associates a discrete-time linear dynamical system with the following pair of equations,

$$x(t+1) = Ax(t) + Bu(t) \tag{1.19}$$

$$y(t) = Cx(t) + Du(t) \tag{1.20}$$

where $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^p$ is the output, and $x \in \mathbb{R}^n$ is the **state**, and all matrices are of appropriate dimension.

Since the system is in discrete time, the solution amounts to simple algebra:

$$y(t) = CA^t x(0) + \sum_{\tau=0}^{t-1} CA^{t-(\tau+1)} Bu(t) + Du(t), \quad t > 0 \tag{1.21}$$

The output $y(t)$ includes an autonomous term, a (discrete) convolution term, and a feedthrough term. The above two sets of equations imply that we can think of linear dynamical systems as state-space models or as input-output

models. The latter is more general and provides the proper definition of a linear dynamical system.

We can define a dynamical system as a function that maps input signals, $\mathbb{U} = \{u \mid u : \mathbb{Z} \to \mathbb{R}^m\}$ to output signals, $\mathbb{Y} = \{y \mid y : \mathbb{Z} \to \mathbb{R}^p\}$. Here $\mathbb{U}$ denotes the set of all functions from $\mathbb{Z}$ to $\mathbb{R}^m$. A linear dynamical system thus defined as $\mathcal{D} : \mathbb{U} \to \mathbb{Y}$ where $\mathcal{D}$ is linear. For a given $\mathcal{D}$, we have a corresponding set of matrices $H(i, j)$ such that

$$y(i) = \sum_j H(i, j)u(j), \quad i, j \in \mathbb{Z} \tag{1.22}$$

Familiar properties are defined as follows. The system $\mathcal{D}$ is causal if $H(i, j) = 0$ for $i \leq j$. The system $\mathcal{D}$ is time-invariant if $H(i, j) = H_{i-j}$, where H_{i-j} is just notation for a particular matrix in $\mathbb{R}^{p\times m}$. In other words, time-invariance is the property that $H(i, j) = H(k, l)$ whenever $i - j = k - l$.

Time-invariance implies that we can write $\mathcal{D}$ as a discrete convolution (a sum):

$$y(t) = (H * u)(t) = \sum_{\tau=-\infty}^{\infty} H_{t-\tau}u(\tau) \tag{1.23}$$

Time-invariance and causality imply a way to write the **impulse response** description of a linear dynamical system,

$$y(t) = H_0u(t) + H_1u(t - 1) + H_2u(t - 2) + \cdots \tag{1.24}$$

Suppose we started with the state-space model Eq. 1.19. We can write the corresponding impulse response parameters in terms of state space matrices:

$$H_t = \begin{cases} CA^{t-1}B & t > 0 \\ D & t = 0 \\ 0 & t < 0 \end{cases} \tag{1.25}$$

corresponding to our derivation in Eq. 1.21 under the assumption that $x(0) = 0$. The sequence H_t itself is referred to as the impulse response of the system. The kth column of H_t is the response of $y(t)$ when an impuse was applied at time zero in the kth component of the input, i.e. $u_k(0) = 1$ while $u_i(t) = 0$ for all other i and t.

From the impulse response to transfer function representations of $\mathcal{D}$. A useful property of linear dynamical systems is that they admit a description in the Laplace domain. For discrete-time analog of the Laplace transform is the z-transform. A z-domain description of $\mathcal{D}$ can be obtained by simply taking the z-transform of the impulse response,

$$\hat{H}(z) = D + C(zI - A)^{-1}B \tag{1.26}$$

which is known as the **transfer function**. The corresponding z-domain representation of $\mathcal{D}$ is

$$\hat{y}(z) = \hat{H}(z)\hat{u}(z) \tag{1.27}$$

A transfer function is a $p \times m$ complex matrix that maps the z-transform of an input signal to the z-transform of an output signal.

From state-space to transfer function representations: The transfer function can be derived directly from the state-space description, which depends on a key property of z-transforms: $\mathcal{Z}(f(t+1)) = z\hat{f}(t) - zf(0)$. This property

alone indicates why z-transforms (Laplace transforms) are a powerful way to solve difference (differential) equations: they turn calculus into algebra.

Thus, we can take the z-transform of the state-space equation

$$\mathcal{Z}(x(t+1)) = \mathcal{Z}(Ax(t) + Bu(t)) \tag{1.28}$$

$$z\hat{x}(z) - zx(0) = A\hat{x}(z) + B\hat{u}(z) \tag{1.29}$$

$$\hat{x}(z) = (zI - A)^{-1}zx(0) + (zI - A)^{-1}B\hat{u}(z) \tag{1.30}$$

Thus,

$$\hat{y}(z) = C(zI - A)^{-1}zx(0) + \left[C(zI - A)^{-1}B + D\right]\hat{u}(z) \tag{1.31}$$

When $x(0) = 0$ we recover the above equality: $\hat{y}(z) = \hat{H}(z)\hat{u}(z)$.

From state-space to ARX representations. In signal processing, autoregressive (AR) models are frequently used for modeling time series. Here, we show that ARX (AR with eXogeneous inputs) models are equivalent to transfer function descriptions of linear dynamical systems. To proceed, let's suppose we can replace the complex variable z with the shift operator q: $qf(t) = f(t+1)$. This substitution is feasible whenever $f(0) = 0$. Thus (following [37]), we have

$$qx(t) = Ax(t) + Bu(t) \tag{1.32}$$

$$x(t) = (qI - A)^{-1}Bu(t) \tag{1.33}$$

and

$$y(t) = \left(C(qI - A)^{-1}B + D\right)u(t) = H(q)u(t) \tag{1.34}$$

Using an explicit formula for $(qI - A)^{-1}$, we can write $H(q)$ as

$$H(q) = \frac{C \operatorname{adj}(qI - A)B + D \det(qI - A)}{\det(qI - A)} \tag{1.35}$$

Although not terribly useful in practice, Eq. 1.35 emphasizes that $H(q)$ is just a matrix of rational functions—quotients of two polynomials. This observation follows from the fact that both adj and $\det$ specify polynomials in the shift operator q. Thus, each entry of the matrix $H(q)$ can be written as a rational function as follows,

$$H_{ij}(q) = \frac{R_{ij}(q)}{S(q)}) \tag{1.36}$$

For some polynomials R_{ij} and S. The zeros of S are the **poles** of the system and correspond to the eigenvalues of A in the corresponding state-space description. The zeros of R_{ij} are the **zeros** of $\mathcal{D}$, where each input-output pair u_j, y_i has a distinct set of zeros.

Now, we can write the components of y as a function of u:

$$y_i(t) = \sum_{j}^{m} \frac{R_{ij}(q)}{S(q)} u_j(t) \tag{1.37}$$

To make the following notation concise, suppose we set $m = p = 1$. Multiply y by S and apply the shift operator:

$$\sum_{k}^{n} s_k y(t+k) = \sum_{k}^{n} r_k u(t+k) \tag{1.38}$$

where r_i and s_j are the coefficients of the polynomials R and s.

Finally, set $s_n = 1$ by convention (divide all terms by s_n if is not 1). Then we have the following **autoregressive with exogenous inputs (ARX)** model:

$$y(t) = r_n u(t) + r_{n-1} u(t-1) + \cdots + r_0 u(t-n) \tag{1.39}$$

$$s_{n-1} y(t-1) + s_{n-2} y(t-2) + \cdots + s_0 y(t-n) \tag{1.40}$$

where we shifted $y(t+n)$ to $y(t)$ on the left hand side to state the following more explicity: ARX models define linear dynamical systems by specifying the current output $y(t)$ as a weighted sum of the past n inputs (including the current input) and of the past $n-1$ outputs (not including the currount output).

From ARX to impulse response. To tie things up, note how we can take the Laurent expansion of $H(z)$ (or $H(q)$) to recover the impulse response sequence H_t.

1.2.2 Implications for identification

The **system identification** problem states, given sequences $u(t)$ and $y(t)$, determine $\mathcal{D}$. This is, of course, a frequently encountered problem in practice, and one we explore in Chapter 4. The previous section emphasizes that there are multiple ways to approach identification, given that there are multiple parameterizations (representations) of a dynamical system $\mathcal{D}$.

A generic linear dynamical system $\mathcal{D}$ can be any linear mapping from inputs to outputs, but we often constrain the model to be causal and time-invariant. As developed above, the state-space, impulse-response, and ARX models incorporate these constraints naturally, which are generally desired from an identification standpoint. Nevertheless, these constraints can be easily relaxed. Time-varying dynamical systems can be specified by time-varying parameters in any description. Acausal descriptions can be obtained in the obvious way by allowing (in external descriptions) $H(i,j)$ to be nonzero

for all i, j. For state-space models, it's easy to see that acausality requires feedback: replace u with $\begin{bmatrix} u & Ky \end{bmatrix}^\top$ for some feedback matrix K. The corresponding input-output relationship is acasual. (As an aside, "acasual" processing is ubiquitous in the brain, e.g. as in predictive coding [30], and simply by studying linear dynamical systems we can convince ourselves of the necessary role that feedback must play in such processing.)

A state-space model specifies a multi-input multi-output ($m > 1, p > 1$) system where the poles of every input-output pair are shared, since a single matrix A is responsible for the intervening transformation between u_i and y_j. This is otherwise evident from the denominator of Eq. 1.35. A state-space model is overparameterized—more parameters than degrees of freedom. This can be noted from the fact that any change of basis in the state-space $\mathbb{R}^n$ yields equivalent input-output transformations, though yield different values of A, B and C. Intuitively, the eigenvalues of A determine the dynamics of the system thus A is determined by only n degrees of freedom (when the state-space description is **minimal**—A has full rank).

The impulse-response model is attractive from a model-fitting perspective. Simply regress outputs against past inputs. Formally, however, the impulse response is an infinite set of parameters with finite degrees of freedom. By only regressing against l past inputs we obtain a finite impulse response model.

The ARX representation is also attractive for identification purposes. Whereas the impulse response contains an infinite number of parameters with finite degrees of freedom, the ARX model simply parameterizes those degrees of freedom explicitly. This is accomplished by regressing outputs against past inputs *and* outputs. The corresponding coefficients translate exactly to the coefficients of a transfer function, which have an immediate interpretation as polynomial coefficients for the poles and zeros of the system.

However, neither the impulse-response or ARX models are capable of naturally capturing shared dynamics (shared poles) without explicit constraints. By fitting an ARX model to data where $p > 1, m > 1$, we are essentially fitting a distinct dynamical system (a distinct set of poles/zeros) to each input-output pair. This is true, but less explicitly, for impulse-response fitting as well. From a model-fitting perspective, this may be irrelevant. From a scientific perspective, this distinction bears consequences. In chapter X, we fit muscle activity from motor cortex activity, modeling the transformation as a linear dynamical system. We wish to model the "spinal dynamics" as the state variable x, with the constraint that a shared spinal circuit responsible for all input-output transformations, regardless of the input M1 neuron and output muscle. The state-space description captures this constraint naturally.

The overparameterization of state-space models has been studied in the context of recent machine learning results [16]. The study showed that gradient descent on state-space models converge to global optimizers, despite a nonconvex loss function for state-space parameterizations (this result depends on the state-space overparameterization).

Chapter 2

Tensor Analysis Reveals Distinct Population Structure that Parallels the Different Computational Roles of Areas M1 and V1

Abstract

Cortical firing rates frequently display elaborate and heterogeneous temporal structure. One often wishes to compute quantitative summaries of such structure—a basic example is the frequency spectrum—and compare with model-based predictions. The advent of large-scale population recordings affords the opportunity to do so in new ways, with the hope of distinguishing between potential explanations for why responses vary with time. We introduce a method that assesses a basic but previously unexplored form of population-level structure: when data contain responses across multiple neurons, conditions, and times, they are naturally expressed as a third-order tensor. We examined tensor structure for multiple datasets from primary visual cortex (V1) and primary motor cortex (M1). All V1 datasets were 'simplest' (there were relatively few degrees of freedom) along the neuron mode,

while all M1 datasets were simplest along the condition mode. These differences could not be inferred from surface-level response features. Formal considerations suggest why tensor structure might differ across modes. For idealized linear models, structure is simplest across the neuron mode when responses reflect external variables, and simplest across the condition mode when responses reflect population dynamics. This same pattern was present for existing models that seek to explain motor cortex responses. Critically, only dynamical models displayed tensor structure that agreed with the empirical M1 data. These results illustrate that tensor structure is a basic feature of the data. For M1 the tensor structure was compatible with only a subset of existing models.

Introduction

Cortical neurons often display temporally complex firing rate patterns (*e.g.*, [1,2]). Such temporal structure may have at least two non-exclusive sources. First, temporal structure may reflect external variables that drive or are being encoded by the population; *e.g.*, a time-varying stimulus or a time-varying parameter represented by the population [3,4]. Second, temporal structure may reflect internal population-level dynamics. For example, oscillatory responses are observed in isolated spinal populations [5], and even sensory areas exhibit response transients due to cellular and network dynamics [6]. One often wishes to disentangle the contributions of external variables and internal dynamics. Yet without full knowledge of the relevant external variables, response patterns can in principle originate from either source [7]. For example, a sinusoidal response might reflect a sinusoidal external variable, oscillatory population dynamics, or both.

Motor cortex (M1) presents a paradigmatic example where temporal response complexity [1,8-10] has fed a long-standing debate [11-21]. Guided by one viewpoint, many studies have focused on the possibility that M1 responses reflect specific external behavioral variables, and have sought to determine their identity (reach direction, velocity, joint torques, muscle forces, etc. [21]) and reference frame [22-28]. Guided by another viewpoint, recent studies suggest that the temporal structure of M1 responses may largely reflect the evolution of internal population dynamics [29-33]. This second viewpoint is embodied in recurrent network models of pattern generation [34-36], and is broadly compatible with control-theory models [37-39] where dynamics may involve both internal recurrence and feedback.

While not necessarily opposed, the first and second viewpoints often make different predictions even when starting with shared assumptions. Suppose one began with the assumption that, during reaching, motor cortex controls muscle activity more-or-less directly [14]. The first viewpoint predicts that neural responses will be a function of (will 'encode') the patterns of muscle activity. The first viewpoint does not predict that neural responses should obey dynamics: the future neural state would not be a consistent function of the present neural state. While muscle activity is 'dynamic' in the sense that it is

time-varying, it is not typically true that the set of muscle activations obeys a single dynamical system (*i.e.* a fixed flow field) across different reaches. The second viewpoint, in contrast, predicts that the motor cortex population response should obey consistent dynamics. The second viewpoint, like the first, predicts that muscle activity will be a function of neural responses [40,41]. Yet because that function is presumably non-invertible, neural responses will not be a function of muscle activity, in opposition to the first viewpoint.

The hypothesis that neural responses reflect external variables (*e.g.*, muscle activity itself) and the hypothesis that neural responses reflect internal dynamics (*e.g.*, the dynamics that produce muscle activity) could be readily distinguished were it known that muscle activity was the relevant external variable. However, that assumption is itself the subject of controversy [8,14,15,17,27,40,42-45]. It therefore remains debated whether M1 response structure originates from a representation of external movement variables or the unfolding of internal dynamics. Recent experimental studies [30,46] and reviews [19,32] have advanced both positions.

Motor cortex thus illustrates a general need: the ability to infer the predominant origin of time-varying responses. We report here that a basic but previously unmeasured feature of neural population data is surprisingly informative to this need. We considered the population response as a third-order tensor (a three-dimensional array) indexed by neuron, condition and time. We were motivated by the idea that tuning for external variables constrains structure across neurons; if there are ten relevant external variables, responses are limited to ten degrees of freedom across neurons. We refer to this setting as 'neuron-preferred.' Conversely, internal dynamics constrain structure across conditions; if a population obeys the same dynamics across conditions, responses will have limited degrees of freedom across conditions. We refer to this situation as 'condition-preferred.' Neuron-preferred or condition-preferred structure is hidden at both the single-neuron level and in standard population-level analyses—*i.e.* this structure is hidden if the data is viewed only as a matrix.

Intuitions regarding neuron-preferred versus condition-preferred structure can be gained by considering linear models. For example, the input-driven system

$$x(c,t) = Bu(c,t), \tag{1}$$

and the autonomous dynamical system

$$x(c,t+1) = Ax(c,t), \tag{2}$$

can be viewed as two different generators of a data tensor $\mathcal{X} \in \mathbb{R}^{N \times C \times T}$, with $x(c,t) \in \mathbb{R}^N$ the vector of N neural responses at time t for condition c, $u(c,t) \in \mathbb{R}^M$ the vector of M input variables, $B \in \mathbb{R}^{N \times M}$, and $A \in \mathbb{R}^{N \times N}$. Time-varying structure of $\mathcal{X}$ generated by the first equation is inherited from the time-varying structure of $u(c,t)$, while for the second it is inherited from the time-varying structure of A^t, since equation (2) can be expressed as $x(c,t) = A^t x(c,0)$. As will be formalized later, neuron-preferred tensor structure follows naturally from equation (1): each $C \times T$ 'slice' of the data tensor $\mathcal{X}$ (*i.e.,* the data for a given neuron across all conditions and times) is a linear combination of a bounded number of basis elements, each of size $C \times T$. Condition-preferred structure follows naturally from equation (2): each $N \times T$ 'slice' of the data tensor $\mathcal{X}$ (*i.e.,* the data for a given condition across all neurons and times) is a linear combination of a bounded number of basis elements, each of size $N \times T$. We choose the term 'neuron-preferred' to describe the case where there are fewer degrees of freedom across neurons, and the term 'condition-preferred' to describe the case where there are fewer degrees of freedom across conditions. Thus, the 'preferred mode' is the mode (neuron or condition) from which the data tensor can be most accurately reconstructed using the smallest number of basis elements.

Our investigation of the preferred mode was guided by a three-part hypothesis. First, we hypothesized that empirical population responses may often have a clear preferred mode. Second, we hypothesized that the preferred mode likely differs between brain areas. To address these hypotheses, we assessed the preferred mode for three neural datasets recorded from primary visual cortex (V1) and four neural datasets recorded from M1. V1 datasets were

strongly neuron-preferred, while M1 datasets were strongly condition-preferred. Third, we hypothesized that the preferred mode might be informative regarding the origin of population responses. We concentrated on models of M1, and found that existing models based on tuning for external variables were neuron-preferred, in opposition to the M1 data. However, existing models with strong internal dynamics were condition-preferred, in agreement with the data. Model success or failure depended not on parameter choice or fit quality, but on model class. We conclude that tensor structure is informative regarding the predominant origin of time-varying activity, and can be used to test specific hypotheses. In the present case, the tensor structure of M1 datasets is consistent with only a subset of existing models.

Results

Time-varying response structure

We analyzed nine physiological datasets: three recorded from V1 during presentation of visual stimuli, four recorded from M1 during reaching tasks, and two recorded from muscle populations during the same reaching tasks. Each dataset employed multiple conditions: different stimuli/reaches. Each neuron's response was averaged across trials within a condition and smoothed to produce a firing rate as a function of time. Some recordings were simultaneous and some were sequential, but in all cases the same set of conditions was employed for every neuron. Stimuli were never tailored to individual neurons (*e.g.*, to their preferred direction or receptive field). This allows for analysis of the true population response, indexed by neuron, condition, and time. For the muscle populations, electromyographic (EMG) voltages were converted to a smooth function of intensity versus time via standard rectification and filtering. Muscle populations were then analyzed in the same way as neural populations, but individual elements were muscles rather than neurons. We analyzed ten further datasets simulated using existing models of M1.

We first focus on two datasets: one from V1 (**Fig 1a**) and one from M1 (**Fig 1b**). The V1 dataset was recorded using a 96-electrode array from an anesthetized monkey viewing one-second movies of natural scenes (25 movies, 50 trials each). The M1 dataset was recorded using a pair of implanted 96-electrode arrays, spanning the arm representation of primary motor cortex and the immediately adjacent region of dorsal premotor cortex (all results were similar if primary motor and premotor cortex were treated separately). Neural responses were recorded during a delayed reach task: the monkey touched a central spot on a screen, was presented with a target, then executed a reach following a go cue. We analyzed data for 72 conditions (**Fig 1b**, insets), each involving a different reach distance and curvature (average of 28 trials per condition) [30].

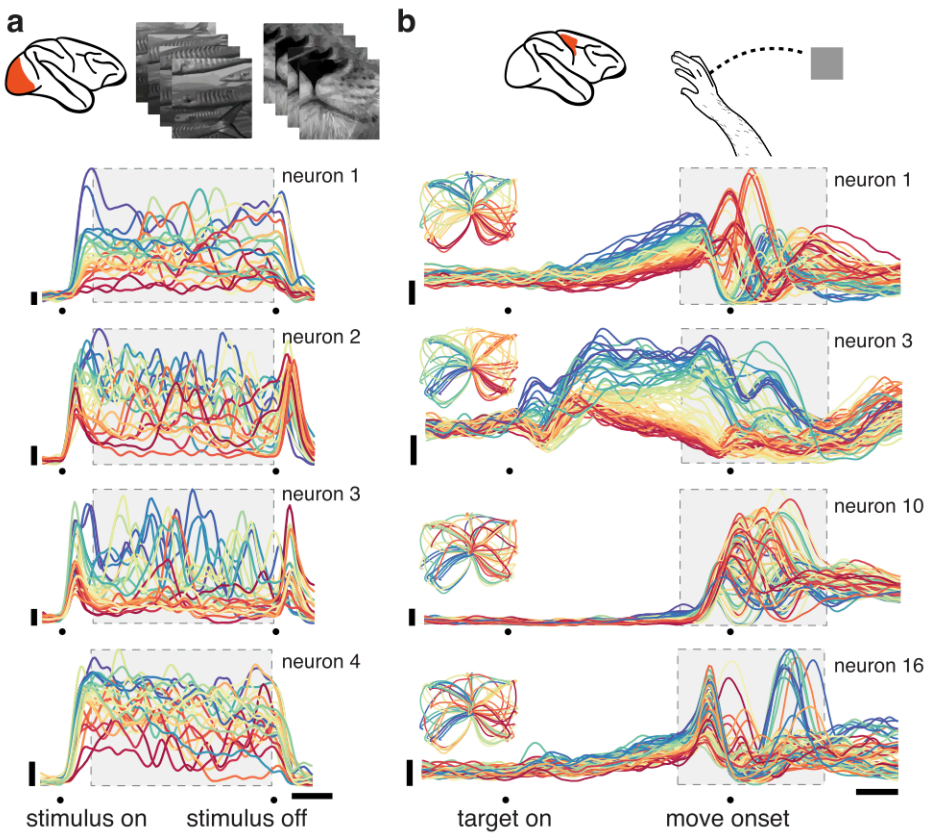

Fig 1. Illustration of the stimuli/task and neural responses for one V1 dataset and one M1 dataset. (**a**) Responses of four example neurons for a V1 dataset recorded via an implanted electrode array during presentation of movies of natural scenes. Each colored trace plots the trial-averaged firing rate for one condition (one of 25 movies). For visualization, traces are colored red to blue based on the firing rate early in the stimulus. (**b**) Responses of four example neurons for an M1 dataset recorded via two implanted electrode arrays during a delayed-reach task (monkey J). Example neurons were chosen to illustrate the variety of observed responses. Each colored trace plots the trial-averaged firing rate for one condition; *i.e.*, one of 72 straight and curved reach trajectories. For visualization, traces are colored based on the firing rate during the delay period between target onset and the go cue. Insets show the reach trajectories (which are the same for each neuron) using the color-coding for that neuron. M1 responses were time-locked separately to the three key events: target onset, the go cue, and reach onset. For presentation, the resulting average traces were spliced together to create a continuous firing rate as a function of time. However, the analysis window included primarily movement-related activity. Gray boxes

indicate the analysis windows (for V1, $T = 91$ time points spanning 910 ms; for M1, $T = 71$ time points spanning 710 ms). Horizontal bars: 200 ms; vertical bars: 20 spikes per second.

Both V1 and M1 neurons displayed temporally complex response patterns (**Fig 1**). Each colored trace plots the trial-averaged firing rate over time for one condition: a particular movie (**Fig 1a**) or reach (**Fig 1b**). V1 neurons exhibited multiphasic responses throughout the stimulus. M1 neurons exhibited multiphasic activity over a ~700 ms period that began shortly after the go cue. Tight standard error bars (not displayed) confirmed that temporal response structure was statistically reliable rather than the result of sampling noise. In M1 it has been debated whether such structure primarily reflects external factors such as reach kinematics or primarily reflects internal dynamics. Both hypotheses can claim support from surface-level features of the data. Responses vary strongly with reach kinematics (insets show reach trajectories color-coded according to the response properties of the neuron in that panel) as proposed by the first hypothesis. On the other hand, responses show some quasi-oscillatory features that could reflect underlying dynamics. Might a comparison with V1—where responses are known to be largely externally driven—be illuminating regarding the source of temporal response structure in M1?

V1 and M1 responses differed in a number of straightforward ways including frequency content and the overall response envelope. Such differences are expected given the different pacing of the task and stimuli. We wondered whether V1 and M1 datasets might also differ in deeper ways that are hidden at the level of the single neuron but clear at the level of the population. In general, a population response can differ across neurons, conditions, and time. While structure across time can be partially appreciated via inspection of single neurons (as in **Fig 1**), the joint structure across neurons and conditions is less patent. Are some datasets more constrained across neurons ('neuron preferred') and others more constrained across conditions ('condition preferred')? If so, might that carry implications?

Preferred-mode analysis of V1 and M1

Neural population data is often analyzed in matrix form, allowing a number of standard analyses. Such analyses include assessing covariance structure and applying principal component analysis to extract the most prevalent response patterns [47]. One can then quantify, for a given number of extracted response patterns, how well they reconstruct the original data. This can provide a rough estimate of the number of degrees of freedom in the data [48].

However, when recordings span multiple neurons, conditions and times, the data are naturally formulated not as a matrix but as a third-order tensor of size $N \times C \times T$, where N is the number of neurons, C is the number of conditions, and T is the number of times. Each of these three indices is referred to as a 'mode' [49]. One can consider an $N \times C \times T$ tensor as a collection of N matrices, each of size $C \times T$ (one per neuron), or as a collection of C matrices, each of size $N \times T$ (one per condition) (**Fig 2a**). One can then reconstruct the population tensor in two ways. First, one can reconstruct the responses of each neuron as a linear combination of a small collection of 'basis-neurons,' each of size $C \times T$ (**Fig 2b**, red matrices). Second, one can reconstruct each condition as a linear combination of a small collection of 'basis-conditions,' each of size $N \times T$ (**Fig 2b**, blue matrices). Unlike in the matrix case, for tensors a 'preferred mode' can exist.

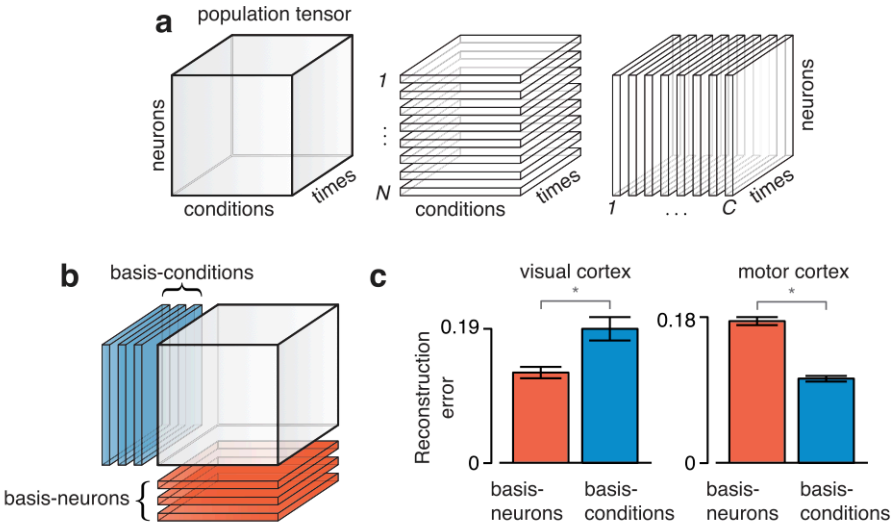

Fig 2. Schematic illustration of population tensor and results of a simplified preferred-mode analysis for two datasets. (**a**) The population

response can be represented as firing rate values arranged in an $N \times C \times T$ array, *i.e.* a third-order tensor indexed by neuron, condition, and time. That population tensor (left) can be thought of as a collection of $C \times T$ matrices (one for each neuron, middle) or a collection of $N \times T$ matrices (one for each condition, right). (**b**) The population tensor may be approximately reconstructed (via linear combinations) from a set of 'basis-neurons' ($C \times T$ matrices, red) or from a set of 'basis-conditions' ($N \times T$ matrices, blue). Depending on the nature of the data, the basis-neurons or the basis-conditions may provide the better reconstruction. (**c**) Normalized reconstruction error of the population tensors for the V1 and M1 datasets shown in **Fig 1** when reconstructed using basis neurons (red) or basis conditions (blue). Error bars show the standard errors across conditions (Methods). The number of basis elements (12 for V1 and 25 for M1) was the same for the neuron and condition modes and was chosen algorithmically (Methods). Robustness of the preferred mode with respect to the number of basis elements is shown in subsequent analyses.

To assess the preferred mode we applied the singular value decomposition (SVD) to the neuron and condition modes of the population tensor (Methods), yielding a set of basis-neurons and a set of basis-conditions. Performing SVD along a mode of a tensor, $\mathcal{X} \in \mathbb{R}^{N \times C \times T}$, equates to performing SVD on one of the tensor's matrix 'unfoldings.' We define the 'mode-1' and 'mode-2' unfolding of $\mathcal{X}$ as

$$\begin{aligned} \boldsymbol{X}_{(1)} &:= [X(1) \quad X(2) \quad \cdots \quad X(T)] \in \mathbb{R}^{N \times CT}, \\ \boldsymbol{X}_{(2)} &:= [X(1)^\top \quad X(2)^\top \quad \cdots \quad X(T)^\top] \in \mathbb{R}^{C \times NT}, \end{aligned} \tag{3}$$

where $X(t) \in \mathbb{R}^{N \times C}$ is the $N \times C$ matrix slice of $\mathcal{X}$ at time t. Each row of $\boldsymbol{X}_{(1)}$ corresponds to one neuron, and each row of $\boldsymbol{X}_{(2)}$ corresponds to one condition. The top k right singular vectors of $\boldsymbol{X}_{(1)}$ are of dimension CT, thus can be reshaped to $C \times T$ matrices, corresponding to k basis-neurons. Similarly, the top k right singular vectors of $\boldsymbol{X}_{(2)}$ are of dimension NT and can be reshaped to $N \times T$ matrices, corresponding to k basis-conditions. In this way each neuron (*i.e.,* each row of $\boldsymbol{X}_{(1)}$ and the corresponding $C \times T$ slice of $\mathcal{X}$) can be approximately reconstructed as a linear combination of k basis-neurons. Similarly, each

condition (*i.e.,* each row of $\boldsymbol{X}_{(2)}$ and the corresponding $N \times T$ slice of $\mathcal{X}$) can be approximately reconstructed as a linear combination of k basis-conditions.

To assess the preferred mode we reconstructed each population tensor twice: once using a fixed number (k) of basis-neurons, and once using the same fixed number (k) of basis-conditions. Reconstruction error was the normalized squared error between the reconstructed tensor and the original data tensor. If basis-neurons provided the better reconstruction, the neuron mode was considered preferred. If basis-conditions provided the better reconstruction, the condition mode was considered preferred. (We explain later the algorithm for choosing the number of basis elements k, and explore robustness with respect to that choice).

The above procedure is related to several tensor decomposition techniques, and the preferred mode is related to the tensor's approximate multilinear rank [49]. Here, instead of decomposing a tensor across all modes we simply perform independent mode-1 and mode-2 decompositions and compare the quality of their corresponding reconstructions.

For the V1 dataset illustrated in **Fig 1** the neuron mode was preferred; it provided the least reconstruction error (**Fig 2c,** left). In contrast, for the M1 dataset illustrated in **Fig 1** the condition mode was preferred (**Fig 2c,** right). This analysis considered all time points in the shaded regions of **Fig 1**. Keeping in mind that reconstruction along either mode is expected to perform reasonably well (data points are rarely uncorrelated along any mode) the disparity between V1 and M1 is large: for V1 the basis-neuron reconstruction performed 33% better than the basis-condition reconstruction, while for M1 it performed 68% worse.

The preferred mode emerges as more times are considered

A preferred mode exists because the population tensor spans multiple neurons, conditions, and times. Consider the population response at a single time, yielding an $N \times C \times 1$ subtensor (a matrix). For this case neither mode is preferred—the row rank (neuron mode) of a matrix equals the column rank

(condition mode). How does the preferred mode emerge as more times are considered? We assessed reconstruction error as a function of timespan (**Fig 3**) beginning with a single time-point, halfway through the response. Using this time we chose bases of k elements such that there was a 5% reconstruction error of the $N \times C \times 1$ matrix (this determined the choice of $k = 12$ and 25 for the V1 and M1 datasets). Keeping k fixed, we increased the tensor size, adding both an earlier and a later time point (we considered time points sampled every 10 ms). Thus, reconstruction error was measured for subtensors of size $N \times C \times T_i$ where $T_i = 1,3,5, \ldots, T$.

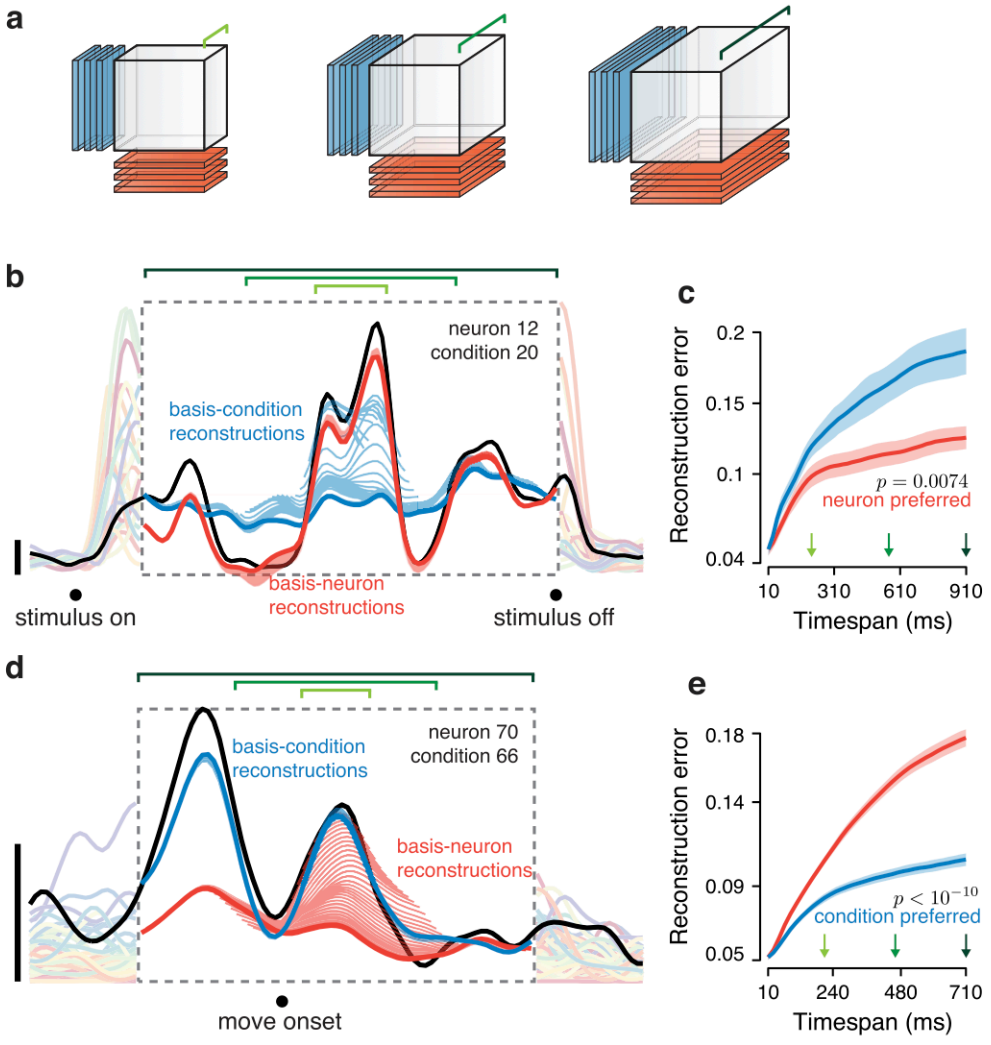

Fig 3. Illustration of the full preferred-mode analysis. Reconstruction error is measured as a function of the number of times included in the population tensor. (**a**) Schematic of the method. A fixed number (three in this

simple illustration) of basis-neurons (red) and basis-conditions (blue) is used to reconstruct the population tensor. This operation is repeated for different subsets of time (*i.e.*, different sizes of the population tensor) three of which are illustrated. Longer green brackets indicate longer timespans. (**b**) The firing rate (black) of one example V1 neuron for one condition, and its reconstruction using basis-neurons (red) and basis-conditions (blue). Short red/blue traces show reconstructions when the population tensor included short timespans. Longer red/blue traces show reconstructions when the population tensor was expanded to include longer timespans. Dark red/blue traces show reconstructions when the population tensor included all times. For illustration, data are shown for one example neuron and condition, after the analysis was applied to a population tensor that included all neurons and conditions (same V1 dataset as in **Fig 1a** and **2c**). The dashed box indicates the longest analyzed timespan. Responses of the example neuron for other conditions are shown in the background for context. Vertical bars: 10 spikes per second. (**c**) Plot of normalized reconstruction error (averaged across all neurons and conditions) for the V1 dataset analyzed in **b**. Red and blue traces respectively show reconstruction error when using 12 basis neurons and 12 basis conditions. The horizontal axis corresponds to the duration of the timespan being analyzed. Green arrows indicate timespans corresponding to the green brackets in **b**. Shaded regions show error bars (Methods). (**d**) As in **b** but illustrating the reconstruction error for one M1 neuron, drawn from the population analyzed in **Fig 1b** and **2c**. (**e**) As in **c** but for the M1 dataset, using 25 basis neurons and 25 basis conditions. The right-most values in **c** and **e** plot the reconstruction error when all times are used, and thus correspond exactly to the bar plots in **Fig 2c**.

The emergence of the preferred mode was often readily apparent even when reconstructing single-neuron responses (note that the entire tensor was always reconstructed, but each neuron can nevertheless be viewed individually). **Fig 3b** shows the response of one V1 neuron for one condition (black trace) with reconstructions provided by the neuron basis (red) and condition basis (blue). Each of the (shortened) light red and light blue traces show reconstructions for a particular timespan (T_i). Dark red and dark blue traces show reconstructions for the full timespan ($T_i = T$). Unsurprisingly, for short timespans (short traces near

the middle of the plot) the two reconstructions performed similarly: blue and red traces both approximated the black trace fairly well. However, for longer timespans the condition-mode reconstruction became inaccurate; the longest blue trace provides a poor approximation of the black trace. In contrast, the neuron-mode reconstruction remained accurate across the full range of times; short and long red traces overlap to the point of being indistinguishable. Thus, the reason why the V1 data were neuron-preferred (**Fig 2c**) is that the neuron basis, but not the condition basis, continued to provide good reconstructions across long timespans.

For the M1 dataset we observed the opposite effect (**Fig 3d**). For very short timespans both the neuron and condition bases provided adequate approximations to the black trace. However, for longer timespans the neuron-mode reconstruction (red) was unable to provide an accurate approximation. In contrast, the condition mode reconstruction remained accurate across all times; short and long blue traces overlap to the point of being indistinguishable.

The disparity in reconstruction error between the preferred and non-preferred mode was often clear at the single-neuron level, and was very clear at the population level. We computed overall reconstruction error for the population tensor as a function of timespan T_i (**Fig 3c,e**). The profile of each trace reflects reconstruction 'stability.' Reconstructions were never perfectly stable; error inevitably grew as more data had to be accounted for. However, stability was considerably better for the preferred mode: the neuron mode for V1 and the condition mode for M1. As can be inferred from the standard errors of the mean (shaded regions) reconstruction error in V1 was significantly lower for the neuron mode for all but the shortest windows ($p = 0.007$ for the longest window). Conversely, reconstruction error in M1 was significantly lower for the condition mode for all but the shortest windows ($p < 10^{-10}$ for the longest window).

When a particular reconstruction fares poorly—*e.g.*, the failure of the condition mode to accurately capture the firing rate of the V1 neuron in **Fig 3b**—it is not trivial to interpret the exact manner in which reconstruction failed. However, the underlying reason for poor reconstruction is simple: the data have

more degrees of freedom along that mode than can be accounted for by the corresponding basis set. For V1, the data have more degrees of freedom across conditions than across neurons, while the opposite was true for M1.

Thus, different datasets can have strongly differing preferred modes, potentially suggesting difference sources of temporal response structure. Before considering this possibility, we ask whether the difference in preferred mode between V1 and M1 is robust, both in the sense of being reliable across datasets and in the sense of not being a trivial consequence of surface-level features of the data, such as frequency content, that differ between V1 and M1 recordings.

Preferred-mode analysis of multiple datasets

To assess robustness we analyzed two additional V1 datasets recorded from cat V1 using 96-electrode arrays during presentation of high-contrast grating sequences[4,50] (**Fig 4b**; top, 50 different sequences; bottom 90 different sequences; panel **a** reproduces the analysis from **Fig 3c** for comparison). For all V1 datasets the neuron mode was preferred: reconstruction error grew less quickly with time when using basis-neurons (red below blue). We analyzed three additional M1 datasets (**Fig 4c-d**; the top of panel **c** reproduces the analysis from **Fig 3e** for comparison), recorded from two monkeys performing variants of the delayed reach task. For all M1 datasets the condition mode was preferred: reconstruction error grew less quickly with time when using basis-conditions (blue below red).

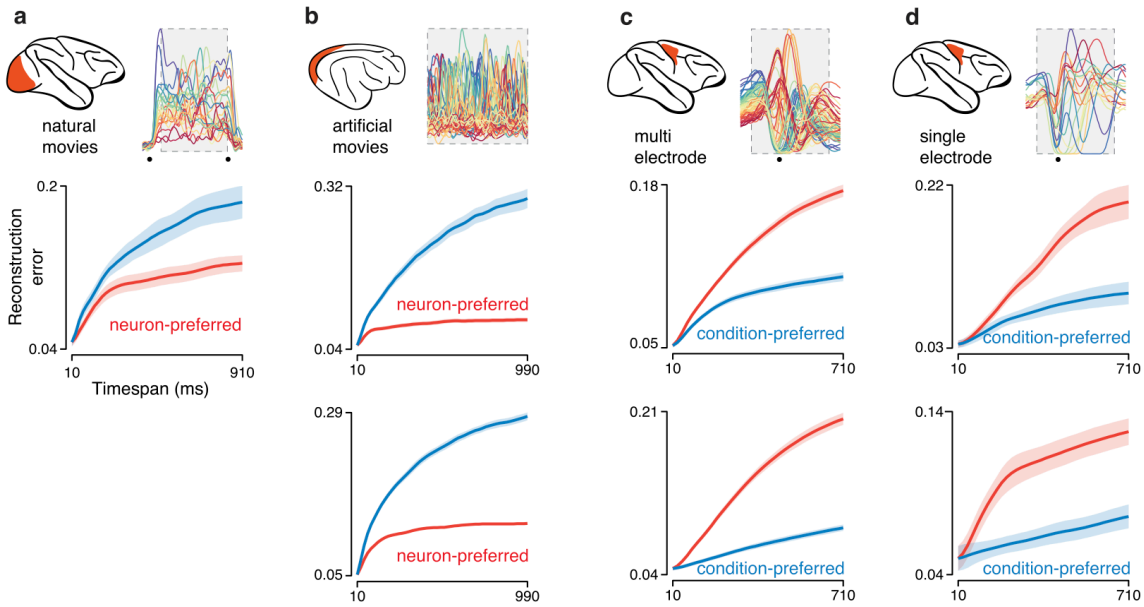

Fig 4. Preferred-mode analysis across neural populations. Each panel corresponds to a dataset type, and plots normalized reconstruction error as a function of timespan (as in **Fig 3c,e**). Excepting panel **a**, two datasets corresponding to two animals were analyzed, yielding two plots per panel. Insets at top indicate the dataset type and show the response of an example neuron. (**a**) Analysis for the V1 population from **Fig 1a**, recorded from a monkey viewing movies of natural scenes. Data are the same as in **Fig 3c** and are reproduced here for comparison with other datasets. (**b**) Analysis of two V1 populations recorded from two cats using grating sequences. (**c**) Analysis of two M1 populations (monkeys J and N) recorded using implanted electrode arrays. The top panel corresponds to the dataset illustrated in **Fig 1b** and reproduces the analysis from **Fig 3e**. (**d**) Analysis of two additional M1 populations from the same two monkeys but for a different set of reaches, with neural populations recorded sequentially using single electrodes.

Most datasets involved simultaneous recordings (the three V1 datasets in **Fig 4a,b** and the two M1 datasets in **Fig 4c**). However, the preferred mode could also be readily inferred from populations built from sequential recordings (the two M1 datasets in **Fig 4d**). Critically, we note that sequential recordings employed the same stimuli for every neuron (stimuli were not tailored to

individual neurons) and behavior was stable and repeatable across the time-period over which recordings were made.

To avoid that possibility that the preferred mode might be influenced by the relative number of recorded neurons versus conditions, all analyses were performed after down-selecting the data so that neuron count and condition count were matched (Methods). Typically, there were more neurons than conditions. We thus down-selected the former to match the latter. The preferred mode was, within the sizeable range we explored, invariant with respect to condition count. The three V1 datasets employed a different number of conditions (25, 90, and 50) yet all showed a neuron mode preference. The four M1 datasets employed a similarly broad range (72, 72, 18, and 18 conditions) yet all showed a condition mode preference. We further explored the potential impact of condition count by taking the 72-condition datasets in panel **c** and restricting the number of analyzed conditions. The preferred mode was robust to this manipulation (see Methods) across the range tested (10-72 conditions). We also performed this analysis for all V1 datasets, and again found that the preferred mode was robust (not shown). Thus, even a modest number of conditions is sufficient to produce a clear preferred mode. That preferred mode then remains consistent as more conditions are added.

The preferred mode is not related to surface-level features

Might the differing preferred modes in V1 and M1 be in some way due to differing surface-level features such as frequency content? *A priori* this is unlikely: properties such as frequency content may have an overall impact on the number of basis-set elements required to achieve a given accuracy, but there is no reason they should create a bias towards a particular preferred mode. Such a bias is also unlikely for three empirical reasons. First, as will be shown below, some existing models of M1 yield a condition-mode preference while others yield a neuron-mode preference. This occurs despite the fact that the surface-level structure produced by all such models resembles that of the M1 data. Second, the preferred mode remained unchanged when surface-level features were altered via temporal filtering (see Methods). In particular, V1 datasets

remained neuron-preferred even when filtering yielded responses with lower frequency content than M1 responses. Third, it can be readily shown via construction that data with the surface-level features of V1 (or of M1) can have either preferred mode.

To illustrate this last point we constructed data with the surface-level of features of V1 but with a condition-mode preference. We began with the V1 dataset analyzed in **Fig 4a** and extracted a set of 'basis-conditions' that captured most of the data variance. This was necessarily a large set of basis conditions (24) given the true neuron-mode preference of the data. We artificially reduced that number of basis conditions by summing random sets of the original basis conditions. For example, the new first basis condition might be a sum of the original basis conditions 1, 7, 12 and 23. Thus, the same patterns were present in the data (no basis conditions were removed) but the degrees of freedom were greatly reduced. We then constructed an artificial population response by replacing the original response of each neuron with the linear combination of modified basis conditions that best approximated the original response. This manipulation resulted in a control dataset with responses that are intentionally altered yet retain the surface-level features of the original data (**Fig 5a,** original data; **Fig 5b**, control data). The manipulated V1 data had a strong condition-mode preference, (blue lower than red) in opposition to the true neuron-mode preference of the original data. Using the same procedure (but reducing degrees of freedom within the neuron basis) we constructed control M1 datasets where surface-level features were preserved but where the neuron mode became preferred (**Fig 5d**, red lower than blue) in opposition to the original data (**Fig 5c**, top, blue lower than red). Thus, the preferred mode is not a consequence of surface-level features.

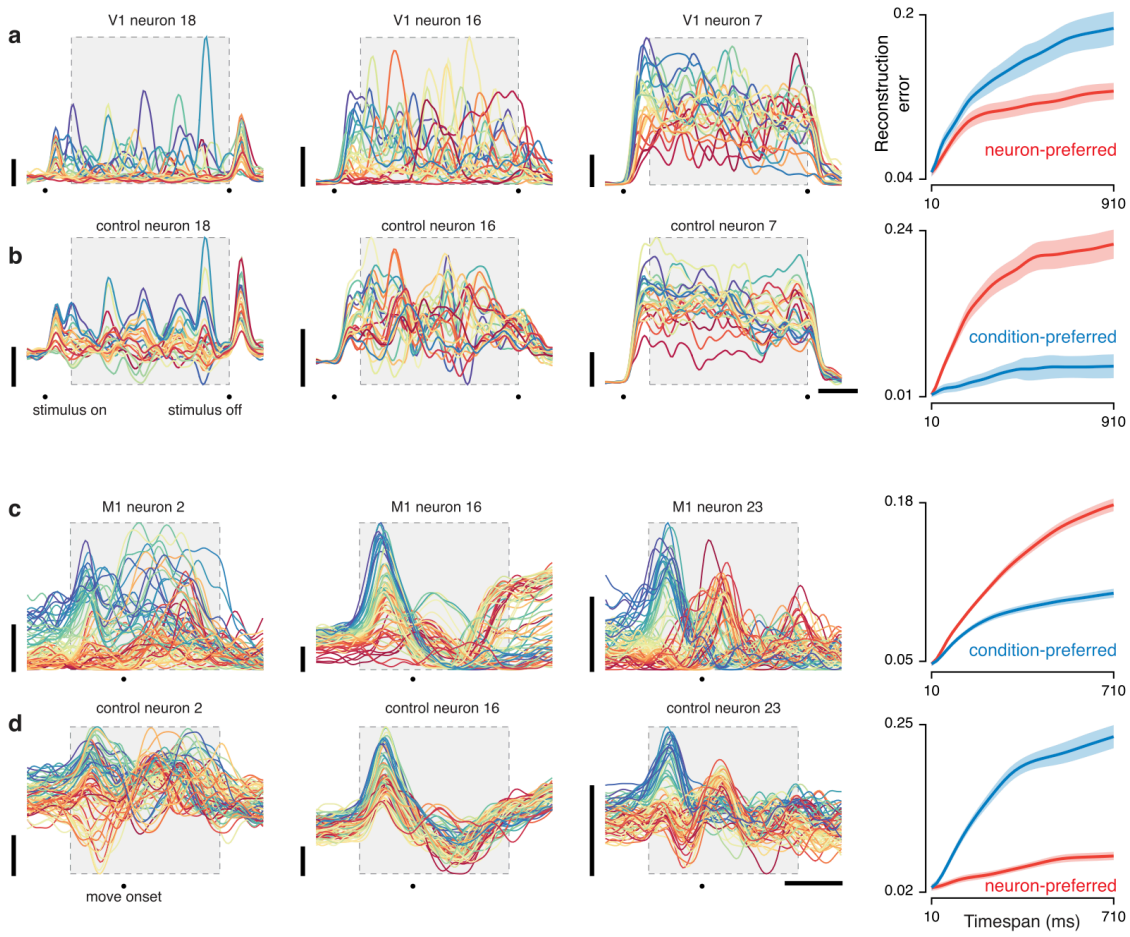

Fig 5. Preferred mode analysis of two control datasets. The preferred mode is not determined by surface-level features. (**a**) Analysis for the empirical V1 dataset from **Fig 3c** and **Fig 4a**. Shown are three example neurons (left panels) and reconstruction error versus timespan (right panel, reproduced from **Fig 3c**). (**b**) Same as in **a** but the V1 dataset was intentionally manipulated to have structure that was simplest across conditions. (**c**) Analysis for the empirical M1 dataset from **Fig 3e**. Shown are three example neurons (left panels) and reconstruction error versus timespan (right panel, reproduced from **Fig 3e**). (**d**) Same as in **c** but the M1 dataset was intentionally manipulated to have structure that was simplest across conditions.

The preferred mode of simulated M1 populations reflects model class

We were interested in the possibility that the origin of temporal structure might influence the preferred mode. Specifically, tuning for external variables

might constrain structure across neurons; if responses reflect a fixed number of external variables then neurons would be limited to that many degrees of freedom. Conversely, internal dynamics might constrain structure across conditions; if each condition evolves according to the same dynamics, conditions could differ along limited degrees of freedom.

The above intuition agrees with the neuron-preferred tensor structure of the V1 datasets, for which the trial-averaged response is expected to be dominated by the stimulus-driven component. Does this intuition extend to, and perhaps help differentiate, models of M1? Many prior studies have modeled M1 responses in terms of tuning for of movement parameters (target direction, reach kinematics, joint torques, etc.). Although the causality is assumed to be reversed relative to V1 (with the M1 representation producing the downstream kinematics), such models formally treat neural responses as functions of time-varying external variables; in particular, responses differ across neurons because different neurons have different tuning for those external variables. M1 'tuning-based models' are thus fundamentally similar to tuning models of V1. On the other hand, some recent studies have modeled M1 responses as the outcome of internal population level dynamics that are similar across conditions. In such models, downstream quantities such as muscle activity are assumed to be a function of cortical activity but cortical activity is not a function of downstream quantities (due to non-invertibility). These M1 'dynamics-based models' are thus fundamentally dissimilar from tuning models of V1.

We analyzed simulated data from five published models of M1, including two models based on tuning for kinematic variables [30] and three models that assumed strong population-level dynamics subserving the production of muscle activity [30,34,36]. All M1 models displayed surface-level features that resembled those of the recorded M1 responses, including a burst of multiphasic responses. Each simulated dataset had neuron and condition counts matched with a corresponding neural population. Each model was simulated twice (top and bottom of the relevant panels in **Fig 6a,b,d,e,f**) with each instance being based on the empirical kinematics or muscle activity for one of the neural datasets.

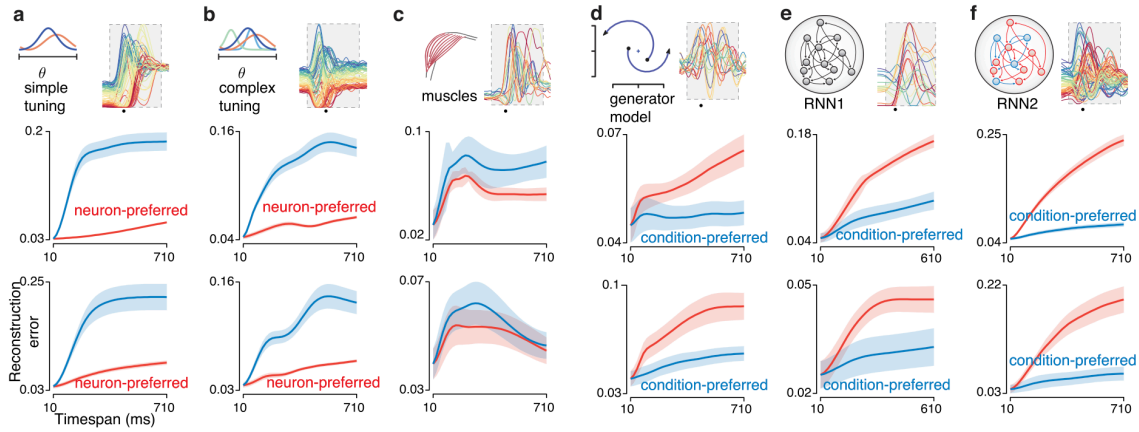

Fig 6. Preferred-mode analysis for non-neural data. Analysis is shown for ten simulated datasets and two muscle populations. Presentation as in **Fig 4**. (**a**) Analysis of simulated M1 populations from the simple tuning model. Two simulated populations (top and bottom) were based on recorded kinematic parameters of two animals (J and N), acquired during the same experimental sessions for which the neural populations are analyzed in **Fig 4c**. (**b**) As in **a**, but M1 populations were simulated based on a more complex tuning model. (**c**) Analysis of populations of muscle responses (monkeys J and N, top and bottom) recorded using the same task/conditions as in **Fig 4d**. (**d**) Analysis of two simulated M1 populations from the dynamical 'generator model' that was trained to reproduce patterns of muscle activity. The model was trained to produce the patterns of deltoid activity from the muscle populations in panel **c**. (**e**) Analysis of two simulated M1 populations from a neural network model trained to produce the patterns of muscle activity shown in panel **c**. (**j**) Analysis of two simulated M1 populations from a 'non-normal' neural network model.

The neuron mode was preferred for the two models that were based on tuning for kinematics (**Fig 6a,b** red below blue). For the first tuning-based model (**Fig 6a**), the relevant kinematic variables were hand velocity and speed (the magnitude of velocity) as in [51]. For the second tuning-based model (**Fig 6b**), the kinematic variables also included hand position and acceleration [52]. Thus, the second tuning-based model reflects the possibility that neural responses are complex due to tuning for multiple movement-related parameters—a position which has recently been argued for based on the ability to decode such parameters [46].

The condition mode was preferred for the three models (**Fig 6d,e,f**) that employed strong population-level dynamics. The model in **Fig 6d** was based on a pair of simple oscillations that followed approximately linear dynamics and provided a basis for fitting empirical patterns of muscle activity [30]. The model in **Fig 6e** was a nonlinear recurrent neural network (RNN) trained to produce the empirical muscle activity patterns [34]. The model in **Fig 6f** was an RNN with 'non-normal' dynamics realized via separate excitatory and inhibitory populations[36]. Critically, these three dynamics-based models were not fit to neural responses; their responses reflect the dynamics necessary to produce the desired outputs. Each has been recently proposed as a possible model of M1 activity during reaches. Despite their substantial architectural differences, all dynamics-based models displayed a condition-mode preference (blue below red).

In a subsequent section we employ a formal approach to explore why different model classes produce different preferred modes. Presently, we simply stress that the preferred mode can be used to test model predictions. In particular, the tuning-based models displayed neuron-preferred tensor structure in opposition to the data. In contrast, the dynamics-based models displayed condition-preferred tensor structure in agreement with the data. Thus, although all models of M1 reproduced (to some reasonable degree) the basic surface-level features of M1 responses, only the dynamics-based models predicted the true condition-mode preference of the M1 population data.

We also analyzed the tensor structure of populations of recorded muscles. Because muscle activity is in some sense an external movement parameter, one might expect the muscle population to be neuron-preferred, in agreement with the tuning-based models above. On the other hand, the dynamics-based models were trained so that a linear projection of the model population response replicated the empirical muscle population response. Given this tight link one might expect the muscle population be condition-preferred. Empirically, the muscle populations had no clear preferred mode: reconstruction error was similar and in some cases overlapping for the neuron and condition modes. There was an overall tendency for the muscle data to be neuron-preferred (the

blue trace tended to be above the red trace at many points) but this was not statistically compelling ($p = 0.37$ and $p = 0.80$).

This analysis of muscle populations again highlights that the preferred mode cannot be inferred from surface-level features. Muscle responses and neural responses share many similar features yet do not show the same tensor structure. The muscle data also highlight that a clear preferred mode need not exist for all datasets. Furthermore, the tensor structure of a system's outputs need not reflect the tensor structure of the system itself. Dynamics-based models built to produce muscle activity showed robust condition-mode preferences (**Fig 6d,e,f**). Yet the muscle populations themselves did not show a condition mode preference (if anything they were weakly neuron-preferred). We return later to the point that the output of a dynamical system need not share the same preferred mode as the system itself.

As a side note, a natural desire is to examine the bases themselves, which might be informative regarding the underlying model. For example, the first basis neuron is essentially the projection of the data onto the first principle component of the $N \times N$ covariance matrix that captures covariance between neurons. The first basis condition is the same, but for a $C \times C$ covariance matrix that captures covariance between conditions. It is indeed possible to make inferences from both such projections [29,30], yet this typically requires specific hypotheses and tailored analysis methods. The fundamental hurdle is that, for any given basis set, there are infinitely many rotations of that basis set that provide equally good reconstruction. Thus, the details of any given projection can be difficult to interpret without bringing additional information to bear. We therefore focus in this study on the quality of the reconstruction, rather than the features of the basis set.

The preferred mode is robust to the number of basis elements

We assessed whether the preferred mode analysis is robust to a key parameter: the number of basis-elements used when quantifying reconstruction error. This is important because it is not possible to directly measure the degrees

of freedom (*i.e.*, the number of basis elements that produces zero reconstruction error) for each mode, given measurement noise and other practical considerations. For this reason, the analyses above compared not degrees of freedom *per se*, but rather the reconstruction error for a fixed number of degrees of freedom. Before concluding that data have fewer degrees of freedom across one mode versus another, one should assess whether the preferred mode is robust with respect to the choice of that fixed number.

To assess robustness we focused on the difference in error between the condition-mode reconstruction and the neuron-mode reconstruction for the longest time window ($T_i = T$). We swept the number of basis elements and plotted the normalized difference in reconstruction errors (**Fig 7**). Positive values indicate a neuron-mode preference and negative values indicate a condition-mode preference. We considered from 1-20 basis elements, stopping earlier if the dataset contained fewer than 20 total degrees of freedom (e.g., the M1 single-electrode data had 18 conditions and the muscle populations contained 8 and 12 recordings respectively). All datasets displayed a preferred mode that was consistent despite differences in the number of basis-elements chosen. In most cases the preferred mode was clearest when a modest number of basis elements was used. Indeed, there was often a peak (for neuron-preferred datasets; data lying in the red shaded area) or trough (for condition-preferred datasets; data lying in the blue shaded area). Unsurprisingly, the difference in reconstruction error trended towards zero as the number of basis elements became large (the difference is necessarily zero if the number of basis elements is equal to the number of neurons / conditions in the data itself).

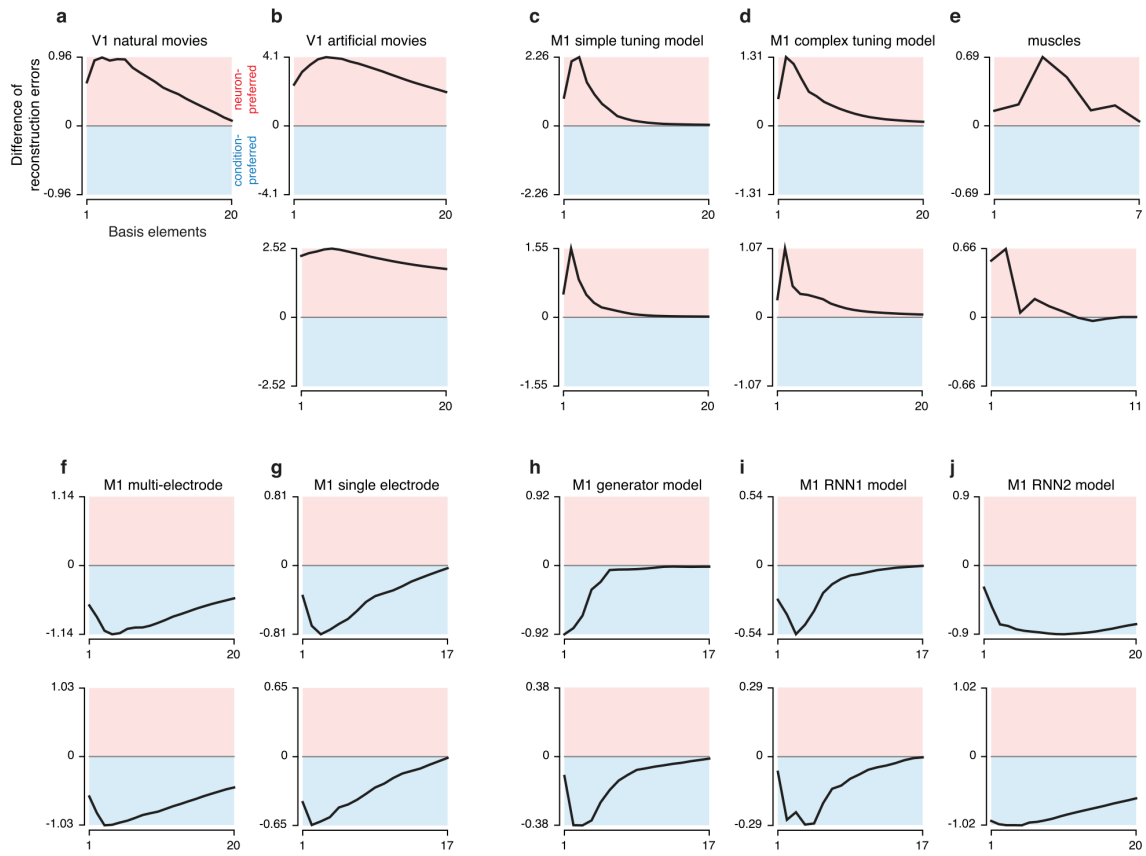

Fig 7. Reconstruction error as a function of the number of basis elements. Each panel plots the difference in reconstruction errors (reconstruction error using k basis-conditions minus reconstruction error using k basis-neurons). The full timespan is considered. Positive values indicate neuron-preferred structure while negative values indicate condition-preferred structure (colored backgrounds for reference). All values in each panel are normalized by a constant, chosen as the smaller of the two reconstruction errors (for the full timespan) plotted in corresponding panels of **Figs 4** and **6**. For most datasets we considered k from 1-20 (mode preference did not flip for higher k in any dataset). For datasets with fewer than 20 neurons (or muscles) values are plotted up to the maximum possible k: the number of neurons (or muscles) in the dataset.

The analysis in **Fig 7** supports the results in **Fig 4**,**6**. All V1 datasets and all M1 tuning-model datasets were consistently neuron-preferred. All M1 datasets and all dynamical M1 models were consistently condition-preferred. The muscle populations, which had trended weakly towards being neuron-preferred in the analysis in **Fig 6**, trended more strongly in that direction when

examined across reconstructions based on different numbers of basis elements (**Fig 7e**). Thus, if a dataset had a clear preference for our original choice of basis elements (the number necessary to provide a reconstruction error <5% when using a single time-point) then that preference was maintained across different choices, and could even become stronger. The analysis in **Fig 7** also underscores the very different tensor structure displayed by different models of M1. Dynamics-based models (panels **h,i,j**) exhibited negative peaks (in agreement with the empirical M1 data) while tuning-based models (panels **c,d**) and muscle activity itself (panel **e**) exhibited positive peaks.

Possible sources of tensor structure

Why did tuning-based models display a neuron-mode preference while dynamics-based models displayed a condition-mode preference? Is there formal justification for the motivating intuition that the origin of temporal response structure influences the preferred mode? This issue is difficult to address in full generality: the space of relevant models is large and includes models that contain mixtures of tuning and dynamic elements. Nevertheless, given reasonable assumptions—in particular that the relevant external variables do not themselves obey a single dynamical system across conditions—we prove that the population response will indeed be neuron-preferred for models of the form:

$$x(t, c) = Bu(t, c), \tag{4}$$

where $x \in \mathbb{R}^N$ is the response of a population of N neurons, $u \in \mathbb{R}^M$ is a vector of M external variables, and $B \in \mathbb{R}^{N \times M}$ defines the mapping from external variables to neural responses. The nth row of B describes the dependence of neuron n on the external variables u. Thus, the rows of B are the tuning functions or receptive fields of each neuron. Both x and u may vary with time t and experimental condition c.

A formal proof, along with sufficient conditions, is given in Methods. Briefly, under equation (4), neurons are different views of the same underlying M external variables. That is, each $u_m(t, c)$ is a pattern of activity (across times and conditions) and each $x_n(t, c)$ is a linear combination of those patterns. The

population tensor generated by equation (4) can thus be built from a linear combination of M basis-neurons. Critically, this fact does not change as time is added to the population tensor. Equation (4) imposes no similar constraints across conditions; *e.g.*, $u(:, c_1)$ need not bear any particular relationship to $u(:, c_2)$. Thus, a large number of basis-conditions may be required to approximate the population tensor. Furthermore, the number of basis-conditions required will typically increase with time; when more times are considered there are more ways in which conditions can differ. A linear tuning model therefore implies a neuron-mode reconstruction that is stable with time and a condition-mode reconstruction that is less accurate and less stable.

Conversely, the population response will not be neuron-preferred (and will typically be condition-preferred) for models of the form:

$$x(t+1, c) = Ax(t, c), \tag{5}$$

Where $A \in \mathbb{R}^{N \times N}$ defines the linear dynamics. This equation admits the solution $x(t, c) = A^{t-1}x(1, c)$. Thus, the matrix A and the initial state $x(1, c)$ fully determine the firing rate of all N neurons for all T times. In particular, the linear dynamics captured by A define a set of $N \times T$ population-level patterns (basis-conditions) from which the response for any condition can be built via linear combination. Critically, this fact does not change as different timespans (T_i) are considered. Although the size of each $N \times T_i$ basis-condition increases as T_i increases, the number of basis-conditions does not. In contrast, the number of necessary basis-neurons may grow with time; neural activity evolves in some subspace of $\mathbb{R}^N$ and as time increases activity may more thoroughly explore this space. Thus, a linear dynamical model implies a condition-mode reconstruction that is stable with time, and a neuron-mode reconstruction that is less accurate and less stable (for proof see Methods).

The above considerations likely explain why we found that tuning-based models were always neuron-preferred and dynamics-based models were always condition-preferred. While none of the tested models were linear and some included noise, their tensor structure was nevertheless shaped by the same factors that shape the tensor structure of more idealized models.

The preferred mode in simple models

Tuning-based models and dynamics-based models are extremes of a continuum: most real neural populations likely contain some contribution from both external variables and internal dynamics. We therefore explored the behavior of the preferred mode in simple linear models where responses were either fully determined by inputs, were fully determined by population dynamics, or were determined by a combination of the two according to:

$$x(t+1,c) = Ax(t,c) + Bu(t,c). \tag{6}$$

The case where responses are fully determined by inputs is formally identical to a tuning model; inputs can be thought of either as sensory, or as higher-level variables that are being represented by the population. When A was set to 0 and responses were fully determined by inputs (**Fig 8a**) the neuron mode was preferred as expected given the formal considerations discussed above. Indeed, because the model is linear, neuron-mode reconstruction error was perfectly stable as times were added (the red trace remains flat). When B was set to zero and responses were fully determined by internal dynamics acting on an initial state, the condition mode was preferred and condition-mode reconstruction error was perfectly stable (**Fig 8d**), consistent with formal considerations.

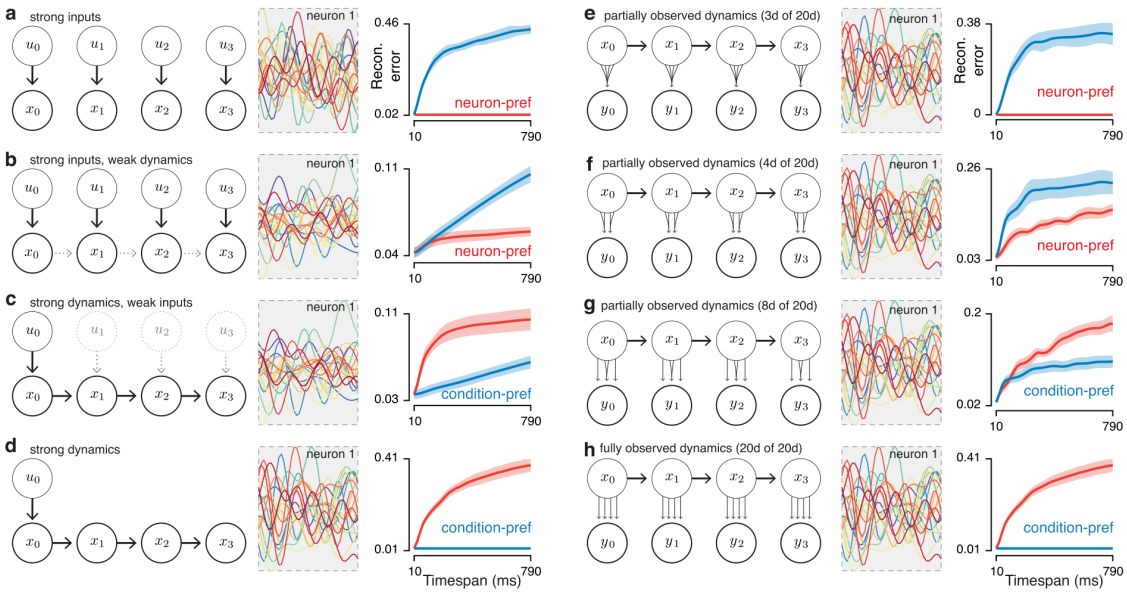

Fig 8. The preferred-mode analysis applied to simulated linear dynamical systems. Left column of each panel: graphical models corresponding to the different systems. Middle column of each panel: response of neuron 1 in each simulated dataset. Colored traces correspond to different conditions. Right column of each panel: preferred-mode analysis applied to simulated data from that system. Analysis is performed on the data x in panels **a-d**, while analysis is performed on the data y in panels **e-h**. (**a**) A system where inputs u are strong and there are no internal dynamics (*i.e.*, there is no influence of x_t on x_{t+1}. (**b**) A system with strong inputs and weak dynamics. (**c**) A system with weak inputs and strong dynamics. (**d**) A system with strong dynamics and no inputs other than an input u_0 at time zero that sets the initial state. (**e**) A system with 20-dimensional linear dynamics at the level of the state x, but where the observed neural responses y reflect only 3 of those dimensions. *I.e.*, the linear function from the state x to the neural recordings y is rank 3. (**f**) A system with 20-dimensional dynamics and 4 observed dimensions. (**g**) A system with 20-dimensional dynamics and 8 observed dimensions. (**h**) A system with 20-dimensional dynamics where all 20 dimensions are observed (formally equivalent to the case in panel **d**).

For models where tuning for inputs was strong relative to dynamics, the neuron mode was preferred (**Fig 8b**). However, because dynamics exerted a modest influence, neuron-mode reconstruction error was not perfectly stable. When dynamics were strong relative to inputs, the condition mode was preferred (**Fig 8c**). However, because inputs exerted a modest influence, condition-mode reconstruction error was not perfectly stable. Thus, simple simulations confirm the expected behavior. A neuron-mode preference is produced when temporal response structure is dominated by tuning for inputs, even if dynamics exert some influence. A condition-mode preference is produced when temporal response structure is dominated by dynamics, even if inputs exert some influence. Thus, the preferred-mode analysis can reveal the dominant source of structure, but does not rule out other contributions.

A potentially confusing point of interpretation is that all neurons necessarily respond to inputs; each neuron is driven by the inputs it receives.

How then can there be a difference in tensor structure between a population that is tuned for inputs versus a population that reflects dynamics? The answer lies in how fully the population reflects dynamics. In the case of tuning for external variables, those variables typically do not fully reflect dynamics. Although the local environment is in some sense 'dynamic,' those dynamics are incompletely observed via the sensory information available to the nervous system. Conversely, if dynamics are produced by the local population they may be fully observed provided that sufficient neurons are recorded.

To illustrate this point we repeated the simulations with the model population either partially (**Fig 8e**) or completely (**Fig 8h**) reflecting an identical set of underlying dynamics. As expected, the case where dynamics are partially observed behaved like the case when the system is input driven: the neuron mode was preferred. As dynamics became more fully reflected, the population switched to being condition-preferred. Thus, condition-preferred structure results from a very particular circumstance: the neural population obeys dynamics that are consistent across conditions and are close to fully reflected in the neural population itself. In contrast, neuron-preferred structure is observed when the temporal structure is inherited from outside the system: from sensory inputs or from dynamics that may be unfolding elsewhere in the nervous system. This explains why there is no paradox in the fact that the muscle populations tended to show neuron-preferred structure (**Fig 6c, Fig 7e**) even though dynamical models that produce muscle activity show condition-preferred structure (**Fig 6d-f, Fig 7h-j**) as does M1 itself. More generally, these simulations illustrate that one may often expect a difference in preferred mode between a system that produces a motor output and a system that 'listens' to that output (*e.g.*, a sensory system that provides feedback during movement).

A key point illustrated by the simulations in **Fig 8a-d** is that the preferred mode is independent of smoothness in the temporal domain. For example, the idealized models in **Fig 8a** and **8d** have responses with closely matched temporal smoothness, yet yield opposing preferred modes. This can be understood via reference to the derivation in the Methods, where assumptions regarding temporal smoothness play no role. For example, a condition-mode

preference will be observed even if dynamics cause rapid fluctuations in the neural state, and indeed even if the dynamics are themselves rapidly time-varying. It is the 'smoothness' across conditions versus neurons that determines the preferred mode, not the smoothness across time. This fact is also illustrated in **Fig 5**, where control manipulations alter the preferred mode while leaving temporal smoothness unchanged.

For the simulations in **Fig 8** and the models in **Fig 6** the preferred mode always reflected the dominant source of temporal structure. Yet with the exception of some idealized models, reconstruction error was rarely perfectly stable even for the preferred mode. The lack of perfectly stability arises from multiple sources including nonlinearities, simulated noise in the firing rate, and contributions by the non-dominant source of structure. We therefore stress that it is difficult, for a given empirical dataset, to ascertain why the preferred mode shows some instability in reconstruction error. For example, in the case of M1 it is likely that the modest rise in condition-mode reconstruction error with timespan (*e.g.*, **Fig 4c,d**) reflects all the above factors.

Discussion

Our analyses were motivated by three hypotheses: first, that population responses will show tensor structure that deviates strongly from random, being simpler across one mode than another; second, that the 'preferred mode' will likely differ across datasets; and third, that the underlying source of temporal response structure influences the preferred mode. The empirical data did indeed deviate strongly from random. V1 datasets were consistently neuron-preferred: the population response was most accurately reconstructed using basis-neurons. M1 datasets were consistently condition-preferred: the population response was most accurately reconstructed using basis-conditions. This difference was invisible at the single-neuron level and could not be inferred from surface-level features of the data. Simulations and formal considerations revealed that neuron-preferred structure arises preferentially in models where responses reflect stimuli or experimental variables. Condition-preferred tensor structure arises preferentially in models where responses reflect population-level dynamics.

Implications for models of motor cortex responses

Given the relationship between model class and preferred mode, the neuron-preferred structure in V1 is entirely expected: all V1 datasets were recorded in the presence of strong visual inputs that are expected to drive the observed response structure [53]. In contrast, the condition-preferred structure of the M1 population response could not be anticipated from first principles because there is little agreement regarding the source of temporal response structure in M1. Several existing M1 models assume that time-varying responses are a function of time-varying movement variables such as reach direction, velocity, and joint torques (for a review see [21]). These variables may be 'dynamic' in the loose sense (they change with time and some may be derivatives of the others) but their values typically do not follow a single dynamical rule that is consistent across conditions. Other recent models are explicitly dynamics-based: the future population state is a function of the present population state, with external inputs serving primarily to set the initial state of the dynamics

[30,34,36]. Tuning-based and dynamics-based models lie on a continuum, but occupy opposing ends and thus make different predictions regarding the tensor structure of the population response. Existing dynamics-based models predict condition-preferred tensor structure, in agreement with the M1 data. Existing tuning-based models predict neuron-preferred structure, in opposition to the M1 data.

Our results thus place strong constraints on models of M1: to be plausible a model must replicate the condition-preferred structure of the empirical population response. Our exploration of current models indicates that this happens naturally for models that include strong dynamics within the recorded population. It does not occur naturally for tuning-based models. We cannot rule out the possibility that future elaborations of tuning-based models might be able to replicate the empirical condition-preferred structure, but the practical possibility of such elaborations remains unclear. There also exist a number of M1 models that we did not examine [35,37,54,55]. It remains an empirical question whether the tensor structure of such models is compatible with the data.

We stress that all current M1 models (including those that successfully predict the empirical preferred mode) are incomplete in key ways and will need to be elaborated or unified in the future. For example, the dynamics-based models we examined do not yet capture the influence of external, sensory-based feedback which is known to be a driver of M1 responses [38,39,56]. Conversely, a recent model of feedback control (not tested here) captures only the dynamics of external feedback loops; the M1 population was modeled as a feedforward network [37]. As future models are developed that incorporate both internal recurrence and sensory feedback, tensor structure provides a simple test regarding whether those models produce realistic population-level responses.

Tensor structure is a basic feature of data, much as the frequency spectrum or the eigenvalue spectrum of the neural covariance matrix are basic features of data. (Indeed, tensor structure is a simple extension to a three-mode array of the standard method of applying principal component analysis to a two-mode array.) Thus, any model that attempts to explain data should succeed in replicating the preferred mode. This requirement is particularly important

because, while models can often be easily modified to produce obvious surface-level features, it is more challenging to also reproduce the underlying tensor structure. Just as importantly, the preferred mode of recorded data can be informative regarding how an appropriate model should be constructed. For every model tested we found that tensor structure is condition-preferred only when the measured population reflects most of the state variables in a dynamical system. In the context of M1, this suggests that successful models will be those where a large percentage of the relevant state variables (sensory feedback, muscle commands and the dynamics that link them) are observable in the M1 population response.

It should be stressed the preferred mode is likely not a feature of a brain area *per se*, but rather of a neural population in the context of the computation being performed by that population. For example, M1 has strong responses to sensory stimuli, especially stretching of the tendons and muscles [56]. In an experiment where responses are driven primarily by externally imposed perturbations of the arm [57,58] it seems likely that M1 would exhibit a neuron-mode structure like that of V1 in the present study. If so, then it would be natural to apply a model in which responses are largely externally driven. If not, then one would be motivated to consider models in which external events set in motion internal dynamics. In either case, knowing the preferred mode would be valuable because it would constrain the set of plausible models.

Interpretational caveats

Interpretation of the preferred mode is most straightforward when there exists one or more models that seek to explain the data. Any model (or model class) that does not replicate the empirical preferred mode must be modified or discarded. Can similarly strong inferences be drawn directly from the preferred mode of the data, without comparison with models? In short they cannot: while a robust preferred mode may suggest a particular class of model, caveats apply. As shown in the derivation (Methods) idealized models produce neuron-preferred structure when responses are driven by unconstrained external variables, and condition-preferred structure when responses are shaped by internal dynamics.

We found that this pattern was robust under less-idealized circumstances: all of the models we examined exhibited a preferred mode consistent with the idealized pattern, even though they departed from idealized assumptions (in particular they were not linear). Such robustness is largely expected. For example, non-linear dynamical systems can often be well approximated by time-varying linear systems, which is all that is required to produce the idealized pattern. Similarly, a non-linear dependency on external variables can often be reconceived as a linear dependency via a change in variables.

That said, there will be limits to the observed robustness. It is possible that a model of one class (*e.g.*, a dynamical systems model) can produce a paradoxical preferred mode (*e.g.*, a neuron-mode preference) under certain circumstances. This might, for example, occur for a neural circuit with strongly nonlinear dynamics that produces long motor sequences. Such a system might be poorly approximated by time-varying linear dynamics, which would result in compromised condition-mode reconstructions. In the case where responses are driven by external variables, an unclear or even paradoxical preferred mode could occur if there is something 'ill-conditioned' about the input. For example, the input could be highly redundant across conditions, resulting in responses that lack enough structure to allow meaningful comparison of reconstruction quality for the neuron mode versus the condition mode. Along similar lines, it would be difficult to interpret the preferred mode in the case where there is little variation in the motor output that can be captured across conditions.

An attractive feature of the preferred mode analysis is that it can be applied without knowledge of the inputs to a system, and provides constraints on potential hypotheses without requiring fully mature models that are ready to be fit directly to data. These advantages are large but, as discussed above, not absolute. First, although potential inputs need not be known, one must have reasonable confidence that the task evokes a range of reasonably rich responses, such that a clear preferred mode can emerge. Second, interpretation of the preferred mode will always be most certain in the case where the preferred mode of the data can be compared with the preferred mode displayed by competing models. In the present case, the preferred mode of the M1 datasets

consistently disagreed with the preferred mode of models where time-varying responses are a function of time-varying movement variables. As this accords with formal expectations, such models are unlikely to provide a good account of the data without major modification.

Future applications

It is likely that neural populations outside of areas V1 and M1 will also display clear preferred modes, which could be diagnostic regarding candidate models. Applicable datasets are those that are sufficiently rich: the experimental task must elicit time-varying responses where PSTHs vary across neurons and conditions. Further, there must be sufficiently many neurons and conditions such that certain low-rank conditions are met (an explanation of these conditions are in Methods under Low-rank assumptions).

As a potential example, some models of decision-making assume that neural responses reflect a small number of task variables (*e.g.*, a 'decision variable' whose value codes the evolving tendency towards a given choice [59]). Other models include internal dynamics that implicitly gate when information is integrated or ignored [60]. None of these decision models sits fully at an extreme—all assume both sensory inputs and some form of integration—but they possess large qualitative differences that may predict different tensor structure. Given the ease with which the preferred mode can be computed for both real and simulated data, the preferred-mode analysis provides a natural way to test whether a given model matches the data at a basic structural level.

Methods

Ethics

All methods were approved in advance by the respective Institutional Animal Care and Use Committees at Albert Einstein College of Medicine (protocol #20150303) and the New York State Psychiatric Institute (protocol #1361). To minimize any potential suffering non-survival surgeries were performed under deep anesthesia with sufentanil citrate, adjusted per the needs of each animal. Survival surgeries were performed under isoflurane anesthesia with carefully monitored post-operative analgesia.

Experimental datasets

We analyzed 9 physiological datasets. Eight have been analyzed previously and one was recorded for the present study. All datasets were based on the spiking activity of a neural population recorded using either multi-electrode arrays (the datasets analyzed in **Fig 4a,b,c**) or sequential individual recordings (the neural dataset analyzed **Fig 4d** and the muscle dataset analyzed in **Fig 6c**). Datasets are available from the Dryad repository (http://dx.doi.org/10.5061/dryad.92h5d).

One V1 dataset (analyzed in **Figs 1, 2, 3, 4a,** and **7a**) was collected using natural-movie stimuli from an anaesthetized adult monkey (*Macaca fascicularis*) implanted with a 96-electrode silicon 'Utah' array (Blackrock Microsystems, Salt Lake City, UT) in left-hemisphere V1. These data were recorded in the laboratory of Adam Kohn (Albert Einstein College of Medicine) specifically for the present study. The left eye was covered. Receptive field centers (2-4 degrees eccentric) were determined via brief presentations of small drifting gratings. Stimuli, which spanned the receptive fields, were 48 natural movie clips (selected from YouTube) with 50 repeats each. The frame rate was 95 Hz. Each stimulus lasted 2.63 s (100 movie frames followed by 150 blank frames). Spikes from the array were sorted offline using MKsort (available at https://github.com/ripple-neuro/mksort/). Single units and stable multi-unit isolations were included. Some neurons showed weak responses and were not analyzed further. Similarly, some stimuli (*e.g.*, those where the region within the receptive fields was blank

or relatively unchanging) evoked weak responses overall. Again, these were not analyzed further. Finally, to ensure we were analyzing a neural population that responds to a shared set of stimulus features, all analyses focused on the subset of units with strongly overlapping receptive fields, defined as the 25 units with receptive fields closest to the center of the stimulus. We insisted upon this criterion because our central analyses would not be as readily interpretable if applied to a set of neurons with distant receptive fields, as they would effectively be responding to different stimuli.

We analyzed two further V1 datasets (**Fig 4b**) recorded from cat V1 as described in [4,50] using Utah arrays implanted so as to overlap areas 17 and 18 (collectively, cat area V1). Stimuli were large stationary gratings, ~30 deg in diameter, and thus spanned the receptive fields of all neurons. Gratings were presented in a rapid sequence—one every 32 ms—each with one of 4 spatial phases and one of 12 orientations. One dataset had five sequences of ~12 s in length. The other dataset had nine such sequences. We wished to segment these long-duration stimuli into 'conditions' with a timescale comparable to that of the other V1 and M1 datasets analyzed here. To do so, we divided the first 10 s of each sequence into 10 one-second segments, which we treated as separate conditions (the stimuli in each second were unrelated to the stimuli in the last second, and are thus effectively different conditions). The two datasets (**Fig 4b**, top, bottom) thus yielded a total of 50 and 90 conditions, respectively. Each condition was observed across multiple (~10) trials. Each dataset consisted of 96 well-tuned multiunit recordings (see [4,50] for details), which were down-selected to match condition counts (50 and 90) of the datasets.

Four M1 datasets were recorded from two male macaque monkeys (*Macaca mulatta*) trained to perform a delayed reach task. These datasets have been described and analyzed previously [29,30]. Briefly, reaches were performed on a fronto-parallel screen for juice reward. To begin each trial the monkey touched a central spot. After a >400 ms hold period, a reach target and up to nine 'barriers' appeared (see **Fig 1** of [29]). The monkey was required to hold its position for a 0-1000 ms delay until a 'go cue', and to then briskly reach to the target while avoiding the barriers. A juice reward was delivered after a

450 ms hold period. This task evoked a large variety of conditions: each corresponding to a particular target and arrangement of barriers. For a given condition, reach trajectories were highly stereotyped across trials (there was only one allowable route through the barriers) allowing a meaningful computation of the average across-trial firing rate. Only trials with delays >450 ms were analyzed (5-40 trials per condition, depending on the dataset); shorter delays simply provided incentive to prepare their movement during the delay. For present purposes, the primary value of the barriers was that they increased the variety of reach conditions, thus increasing the size of the tensor that could be analyzed. In the original dataset some conditions included 'distractor' targets that the monkey had to ignore while preparing the reach. The purpose of those conditions was incidental to the present study and they were not included in the analysis (results were virtually identical if they were included). Neural responses were recorded from M1 and the adjacent region of caudal PMd. Single-electrode and array datasets employed 18 and 72 conditions respectively. Single-electrode datasets consisted of ideally isolated single neurons. Array datasets included both ideal isolations and good multi-unit isolations (*e.g.*, two clear units that could not be separated from one another). Unit counts for the four datasets were 170, 218, 55, and 118 (corresponding, respectively, to panels **c-d** in **Fig 4**), which were down-selected to 72, 72, 18, and 18 to match condition counts.

Two datasets of the responses of muscle populations (analyzed in **Fig 6c**) were recorded using the same monkeys and task as for the M1 datasets. Muscle datasets used the same 18 conditions as the single-electrode datasets. EMG responses were recorded percutaneously using electrodes inserted for the duration of the recording session. Recordings were made from six muscle groups: deltoid, biceps brachii, triceps brachii, trapezius, latissimus dorsi and pectoralis. Multiple recordings were often made from a given muscle (*e.g.*, from the anterior, lateral and posterior deltoid). For monkey J the triceps was minimally active and was not recorded. Muscles were recorded sequentially and then analyzed as a population (just as were the single-electrode datasets). For the two monkeys the resulting populations consisted of 8 and 12 recordings.

Model datasets

We analyzed multiple datasets produced via simulation of published models. The velocity model from [30] was analyzed in **Fig 6a** (here, referred to as the simple tuning model). The complex-kinematic model from [30] was analyzed in **Fig 6b** (here referred to as the complex tuning model). The generator model from [30] is analyzed in **Fig 6d**. The network model of Sussillo et al. [34] is analyzed in **Fig 6e**. The network model of Hennequin et al. [36] is analyzed in **Fig 6f**. Both network models are instantiations of a recurrent neural network (RNN):

$$\begin{aligned} \frac{dx(t,c)}{dt} &= -x(t,c) + Ar(t,c) + Bu(t,c) \\ r(t,c) &= \phi\big(x(t,c)\big) \\ y(t,c) &= Wr(t,c), \end{aligned} \tag{7}$$

where $x \in \mathbb{R}^N$ is the network state, $u \in \mathbb{R}^M$ is the vector of inputs, $y \in \mathbb{R}^P$ is the vector of outputs. The function ϕ is an element-wise nonlinear function, $r \in \mathbb{R}^N$ is interpreted as a firing rate, and the matrices A, B, and W are of appropriate dimensions. The output y is interpreted as muscle activity.

All datasets were from the original simulations analyzed in those publications, with the exception of the RNN model of [36]. We re-simulated that model based on similar procedures described in [36]. After stabilizing the network using their procedure, we needed to specify each of the 72 initial states $x(1,c)$ (one for each condition). We first computed the controllability Gramian of the linearized network (the matrix Q in [36]). The orthonormal columns of Q correspond to potential choices of initial states; the first column is an initial state that evokes the 'strongest' response (in terms of the total energy of the corresponding signals r); the second column gives the next strongest, and so forth. We selected the initial state for each condition to roughly match the temporal pattern of total energy (summed across all neurons) of the empirical neural data. Namely, we first considered the instantaneous power $P(t) \coloneqq r(t)^\top r(t)$. Next, for a given column of Q (a possible choice of initial state), we simulated the network and measured the correlation across times between $P(t)$ of the simulated data and $P(t)$ of the empirical data for a given condition. After

determining the 5 columns of Q that yielded the highest correlations, we chose each $x(1, c)$ to be the weighted sum of those 5 columns that best matched $P(t)$ for that condition. The net effect of this procedure was to produce a rich set of dynamics, flowing from 72 initial states, that provided a possible basis set for producing patterns of EMG for the 72 conditions. We confirmed the network did indeed provide such a basis set (*e.g.*, that the EMG could be fit as a weighted sum of the responses in the network).

Data preprocessing

For all experimental neural data, spike trains were smoothed with a Gaussian kernel (20 ms standard deviation) and sampled every 10 ms. Firing rate values were averaged across trials resulting in a population tensor of size $N \times C \times T$. Each element of this tensor is simply the firing rate for the corresponding neuron, condition and time. To ensure that analysis was not dominated by a few high-rate neurons, we normalized firing rates. Because normalization can occasionally lead to an undesirable expansion of sampling noise for low-rate neurons, we employed a 'soft-normalization' procedure (this same normalization is used in [30]). Each neuron was normalized according to:

$$x_n(c,t) \leftarrow \frac{x_n(c,t)}{5 \; + \; \text{range}_{c,t}\big(x_n(c,t)\big)}, \tag{8}$$

where $i = 1, \ldots, N$. The function $\text{range}_{c,t}(\cdot)$ returns the difference between the maximum and minimum firing rates across all conditions and times for a given neuron. The soft normalization constant 5 mapped high firing rate neurons (*e.g.*, 100 Hz) to a new range close to one. Low firing rate neurons were mapped to a range somewhat less than one (*e.g.*, a neuron with a range of 5 spikes/s would be mapped to a new range of 0.5). This preprocessing allows neurons to contribute roughly equally regardless of their firing rate range. This is especially desirable when analyses involve the mean squared error. For example, without normalization the same relative error will be 25 times greater for a neuron with a 0-100 Hz firing rate range relative to a neuron with a 0-20 Hz firing rate range.

That said, we emphasize that our results (*e.g.*, the preferred mode of a given dataset) did not depend on the choice of soft normalization constant.

We wished to analyze temporal response structure that was different across conditions. We therefore removed the 'cross-condition mean' from the entire population tensor. We averaged the tensor across conditions resulting in an $N \times T$ matrix that we subtracted from every $N \times T$ matrix of data. This is related to the standard PCA step of first removing the mean value of each variable, and ensured that the analysis did not consider response structure that was identical across conditions, such as an elevation of firing rates for all visual stimuli or all reach directions.

All datasets naturally had an unequal number of neurons (N) and conditions (C). To ensure that basis-neuron and basis-condition reconstructions were compared on similar footing, we removed excess neurons or conditions in each dataset so that $N = C$. In most datasets there were more neurons than conditions. In such cases we kept the $N = C$ neurons with the highest ratio of signal to noise. In the V1 dataset of **Fig 1a** there were more conditions than neurons. In this case we retained the $N = C$ conditions that elicited the most temporal complexity in the population response (assessed via the standard deviation of the firing rate across all neurons and times). The specific preprocessing choices (filter length, normalization, equalizing N and C) were made to minimize any potential bias toward basis-neurons or basis-conditions. Still, none of these choices were found the affect the outcome of the analyses.

Preferred-mode analysis

For each population tensor $\mathcal{X} \in \mathbb{R}^{N \times C \times T}$ we quantified how well it could be reconstructed from a small set of k basis-neurons or k basis-conditions (the method for choosing k is described later). To illustrate, we first consider the case of basis-neurons (the case of basis-conditions is entirely parallel). Each of the recorded neurons is a set of T datapoints (one per time) for C conditions and thus forms a $C \times T$ matrix. Each basis neuron is also a $C \times T$ matrix. The data for each of the N neurons (each $C \times T$ matrix within the full population tensor) was

approximated as a weighted sum of k basis-neuron matrices. Weights and basis neurons were chosen to provide the reconstruction with the lowest error.

To find those weights and basis neurons we applied SVD along the neuron mode of the population tensor. This procedure amounts to 'unfolding' (or reshaping) the tensor into a matrix, $\boldsymbol{X}_{(1)} \in \mathbb{R}^{N \times CT}$, where the subscript in parentheses indicates which mode appears as the row index in the matrix (see [49]). The order in which the columns appear in the matrix does not affect our analysis. We applied the SVD to $\boldsymbol{X}_{(1)}$. The right singular vectors of $\boldsymbol{X}_{(1)}$ correspond to vectors of dimension CT, which can be reshaped into $C \times T$ matrices corresponding to 'basis-neurons.' The singular values (squared) of $\boldsymbol{X}_{(1)}$ indicate how much variance is explained by each basis-neuron. The approach to finding basis-conditions is parallel to the above and involves the SVD of $\boldsymbol{X}_{(2)} \in \mathbb{R}^{C \times NT}$. For both reconstructions we assessed the mean squared error between the elements of the original tensor and those of the reconstructed tensor. The reconstructed tensor was produced by multiplying the matrices produced by the SVD after appropriately limiting the inner dimensions based on the number of basis elements k. For example, if $\boldsymbol{X}_{(1)} = USV^{\top}$, then $\boldsymbol{X}_{(1)}^{\text{rec}} = U_{:,1:k} S_{1:k,1:k} V_{1:k,:}^{\top}$. We note that for practical convenience reconstruction error can also be readily computed from the first k singular values. For visualization we express reconstruction error in normalized form, relative to the total variance of the data.

We extended the above analysis to quantify reconstruction error as a function of the number of time-points included in the tensor (**Figs. 3,4,6**). We began by considering a single time-point halfway through the response: $t_{\text{half}} = \text{round}(T/2)$. We used this time to ask how many basis elements (basis-neurons and basis-conditions) were necessary to achieve low reconstruction error. As above we applied the SVD, in this case to the matrix $\mathcal{X}_{:,:,t_{\text{half}}} \in \mathbb{R}^{N \times C \times 1}$. We chose the smallest number k such that normalized reconstruction error using the first k basis elements was less than 5%. Because $\mathcal{X}_{:,:,t_{\text{half}}}$is a matrix, the value of k is the same for basis-neurons and basis-conditions. We then analyzed $\mathcal{X}_{:,:,t_{\text{half}}-1:t_{\text{half}}+1} \in \mathbb{R}^{N \times C \times 3}$ and quantified reconstruction error when using k basis-neurons versus k basis-conditions (*i.e.*, the standard procedure described above

was applied, but to a tensor that contained three times rather than all times). We repeated this for $\mathcal{X}_{:,:,t_{\text{half}}-2:t_{\text{half}}+2} \in \mathbb{R}^{N\times C\times 5}$ and so forth until the full $N\times C\times T$ tensor was analyzed.

To assess statistical reliability, we computed reconstruction error independently for each condition. This yielded a distribution of errors with a given mean and standard error. It is that mean and standard error that are plotted in **Figs 2c**, **3c,e**, **4, 6**, and the right columns of **Fig 8**. We chose to compute the standard error across conditions rather than across both neurons and conditions to be conservative (the latter would have yielded even smaller error bars).

Control datasets and analyses

We performed a three control analyses to assess the robustness of the central method. The outcome of the first of these is shown in the Results; the outcome of the other two are shown here. First, we analyzed two control datasets intentionally constructed to have surface-level features similar to the original empirical datasets. To generate the manipulated V1 dataset, we first extracted the top 24 basis-conditions (out of 25) from the original dataset using SVD. We randomly partitioned the basis set into 6 partitions (4 elements each), and summed the elements within a partition to create a single basis-condition, resulting in 6 total basis-conditions. We then reconstructed the manipulated dataset neuron-by-neuron: each new neuron was a least-squares fit to the original neuron, but using the 6 basis-conditions derived above. This ensured that the manipulated V1 data had relatively few degrees of freedom across conditions, yet resembled the original V1 neurons in terms of basic response properties. The manipulated M1 dataset was constructed analogously, but using 6 basis-neurons derived from the original 72. The outcome of this analysis is shown in **Fig 5**.

Second, to assess robustness of the central method with respect to the number of recorded conditions, we repeated the analysis for one M1 dataset (the dataset from **Fig 3e**) that originally contained 72 conditions. We down-sampled

the data by selecting 10, 20, and 30 conditions. Conditions were selected randomly, but via a procedure that also ensured that the selected conditions were sufficiently different (*e.g.*, that they were not all rightwards reaches). The preferred mode was indeed robust even when the number of conditions was reduced (**Fig 9**).

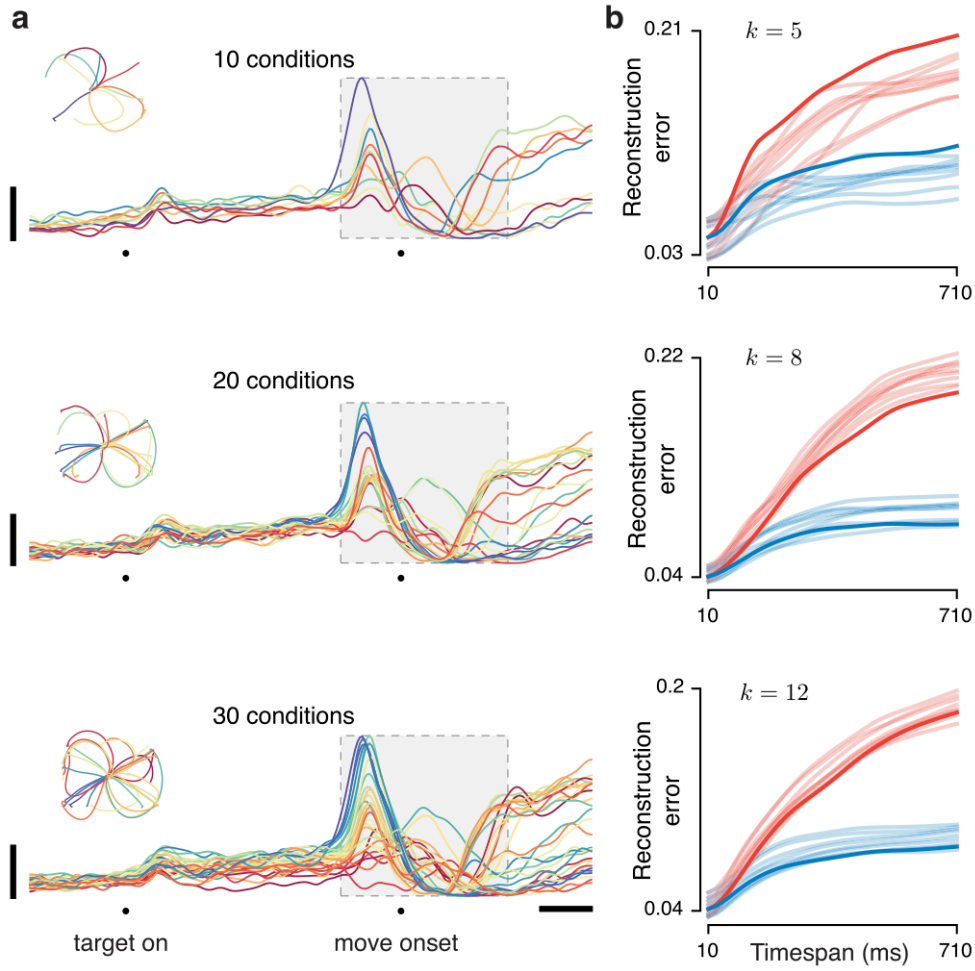

Fig 9. Preferred-mode analysis using a variable number of conditions. (**a**) Responses of one example neuron illustrating an instance of randomly selected sets of 10 (top), 20 (middle), and 30 (bottom) conditions. Horizontal and vertical calibration bars correspond to 200 ms and 20 spikes/s. (**b**) Reconstruction error as a function of timespan for sets of 10 (top), 20 (middle), and 30 (bottom) conditions. Multiple traces are shown: one each for 10 draws of random conditions. Dark traces show the neuron-mode (red) and condition-mode (blue) reconstruction error for the particular sets of conditions

illustrated in **a**. Even for small numbers of conditions (as few as 10) there was a consistent preferred mode. In fact, the preferred mode was even more consistent than it appears, as the comparisons are naturally paired: every red trace has a corresponding blue trace. These tended to move upwards and downwards together (as in the example illustrated with the dark traces) with a reasonably consistent difference between them.

Finally, we analyzed the effect of spike filter widths on the preferred mode for the V1 and M1 datasets (**Fig 10**). This analysis served two purposes. First, spike filtering is a standard preprocessing step and we wanted to ensure that results were not dependent on the particular choice of filter width. Second, the analysis reveals that the preferred mode is not in some way to due to the smoothness or frequency content of neural signals—a potential concern when comparing brain areas whose neurons have fundamentally different response properties, as is the case with V1 and M1.

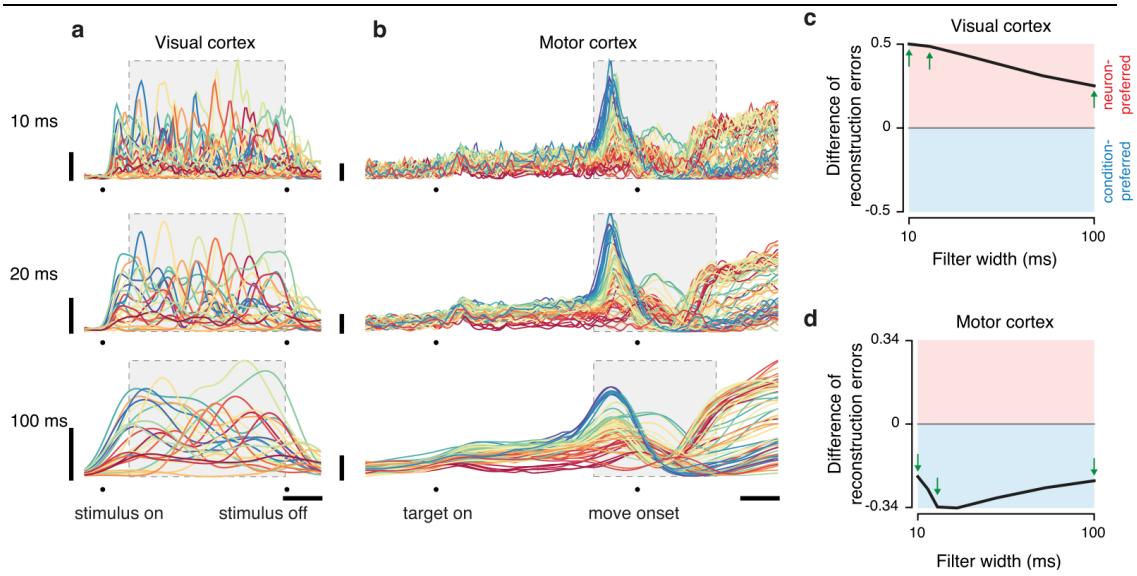

Fig 10. Effect of spike filtering width on the preferred mode. Spike trains from V1 and M1 datasets were filtered with a Gaussian kernel of varying widths (width corresponds to the standard deviation of the Gaussian). (**a**) Response of one example V1 neuron for filter widths of 10 ms, 20 ms (the default value used for all other analyses in this study), and 100 ms. (**b**) Response of one example M1 neuron for the same three filter widths. Horizontal and vertical calibration bars correspond to 200 ms and 20 spikes/s. (**c**) Difference in reconstruction error between the condition mode and the neuron mode

(computed as in **Fig 7**) as a function of filter width, for the V1 dataset from panel **a**. Differences are positive, indicating that the neuron mode incurred less error and is preferred. Green arrows indicate filter widths of 10, 20, and 100, corresponding to the examples shown in **a**. (**d**) Difference in reconstruction error for the M1 dataset from panel **b**. Differences are negative, indicating that the condition mode incurred less error and is preferred. Thus, the preferred mode is robust to filter width, despite the wide range of frequencies highlighted or suppressed by filter width choices.

Linear Models

In **Fig 8** we illustrated some basic properties of the preferred mode using simulations of linear dynamical systems (equation (6)). These simple simulations were separate from the simulations of published models described above. For these simple simulations we chose $N = C = 20$, and $T = 300$. We set $M = 10$ (*i.e.* the input u was ten-dimensional). We first generated the matrices A and B with orthonormal columns; for A, eigenvalues were random but were clustered near 1 to ensure smooth trajectories for our choice of T (this was not a necessary step, but yielded roughly comparable oscillation frequencies to those observed in the datasets of **Fig 4**). Each input u_m was composed of a randomly weighted sum of 20 sinusoids. Sinusoid frequency was determined by the same procedure that generated the eigenvalues of A. Thus, inputs had the same frequency components as the dynamics, ensuring similar single-neuron response properties across simulations. Initial states across conditions were chosen randomly and were constrained to span 10 dimensions. With these parameters fixed, we simulated the system $x(t+1, c) = aAx(t, c) + bBu(t, c)$, where $a \in [0,1]$ and $b \in [0,1]$ determined the strength of dynamics and inputs, respectively. In **Fig 8a-d**, values of a were 0, 0.98, 0.99, and 1 (Note that values of a even slightly lower than unity lead to rapidly decaying 'weak' dynamics). Values of b were 1, 0.05, 0.03, and 0 (note that inputs need to be quite weak before they cease to have a strong effect on a system with persistent dynamics). Each panel in **Fig 8** involved the same choices of A and B, and the same initial states.

Data in **Fig 8e-h** were simulated as above, with $a = 1$ and $b = 0$.

However, the 'data' for which the preferred mode was computed consisted not of the values of the dynamic variable x, but rather of the values of an observation variable y. We treated y as the neural population being driven by 'observing' the dynamic state variable x, with $y(c,t) = Cx(c,t)$. The observation matrix C had different ranks depending on how fully y reflected x. Specifically, C was diagonal with 1s on the first 3, 4, 8, and 20 diagonal entries for **Fig 8** panels **e,f,g,h**, respectively (and 0s elsewhere).

Derivation of the preferred mode for idealized models

Here we show that neuron-preferred structure is expected when responses are driven by unconstrained external variables, while condition-preferred structure is expected when neural responses are shaped by internal dynamics. We consider a dataset $\mathcal{X} \in \mathbb{R}^{N\times C\times T}$, where N, C and T are the number of recorded neurons, experimental conditions, and times. We also consider a set of external signals, or inputs, $\mathcal{U} \in \mathbb{R}^{M\times C\times T}$, where M is the number of external variables. The column vector $x(t,c) \in \mathbb{R}^N$ is the firing rate of every neuron at time $t \in \{1, \dots, T\}$ for condition $c \in \{1, \dots, C\}$. An $N\times C$ matrix 'slice' of $\mathcal{X}$ is denoted $X(t) \in \mathbb{R}^{N\times C}$, and is the population state across all conditions for time t. We define the 'mode-1' and 'mode-2' matrix unfoldings of $\mathcal{X}$:

$$\begin{aligned} \boldsymbol{X}_{(1)} &:= [X(1) \quad X(2) \quad \cdots \quad X(T)] \in \mathbb{R}^{N\times CT}, \\ \boldsymbol{X}_{(2)} &:= [X(1)^\top \quad X(2)^\top \quad \cdots \quad X(T)^\top] \in \mathbb{R}^{C\times NT}. \end{aligned} \tag{9}$$

Each row of $\boldsymbol{X}_{(1)}$ corresponds to one neuron, and each row of $\boldsymbol{X}_{(2)}$ corresponds to one condition. Importantly, rank($\boldsymbol{X}_{(1)}$) is the number of basis-neurons needed to reconstruct $\mathcal{X}$. Similarly, rank($\boldsymbol{X}_{(2)}$) is the number of basis-conditions needed to reconstruct $\mathcal{X}$.

Definition: A dataset $\mathcal{X} \in \mathbb{R}^{N\times C\times T}$ is called neuron-preferred (condition-preferred) when the rank of the matrix unfolding $\boldsymbol{X}_{(1)}$ ($\boldsymbol{X}_{(2)}$) of its sub-tensors $\mathcal{X}_{T_i} \in \mathbb{R}^{N\times C\times T_i}$ does not increase with T_i, while the rank of $\boldsymbol{X}_{(2)}$ ($\boldsymbol{X}_{(1)}$) does increase with T_i.

We evaluate the rank of each unfolding in datasets $\mathcal{X}$ generated by the following model classes:

$$x(t,c) = Bu(t,c), \tag{10}$$

and

$$x(t+1,c) = Ax(t,c). \tag{11}$$

We term equation (10) the tuning model class ($B \in \mathbb{R}^{N\times M}$ defines each neuron's tuning for external variables), and equation (11) the dynamical model class ($A \in \mathbb{R}^{N\times N}$ specifies linear dynamics).

Claim: Models of the form equation (10) (equation (11)) generate datasets having neuron-preferred (condition-preferred) structure.

Part 1: The tuning model class implies neuron-preferred structure. To begin, note that equation (10) can be written as a matrix equation,

$$X(t) = BU(t). \tag{12}$$

For any $T_i \in \{1, \dots, T\}$, equation (12) implies,

$$[X(1) \quad X(2) \quad \cdots \quad X(T_i)] = B[U(1) \quad U(2) \quad \cdots \quad U(T_i)], \tag{13}$$

or, more compactly, $\boldsymbol{X}_{(1)} = B\boldsymbol{U}_{(1)}$. For the mode-2 unfolding, given equation (12) we can also write,

$$\begin{bmatrix} X(1) \\ X(2) \\ \vdots \\ X(T_i) \end{bmatrix} = \begin{bmatrix} B & 0 & \cdots & 0 \\ 0 & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B \end{bmatrix} \begin{bmatrix} U(1) \\ U(2) \\ \vdots \\ U(T_i) \end{bmatrix}, \tag{14}$$

i.e., $\boldsymbol{X}_{(2)} = \boldsymbol{U}_{(2)}(I_{T_i} \otimes B^{\top})$ where I_{T_i} is the $T_i \times T_i$ identity matrix and $\otimes$ denotes the Kronecker product. Thus,

$$\begin{aligned} x(t,c) = Bu(t,c) &\Leftrightarrow \boldsymbol{X}_{(1)} = B\boldsymbol{U}_{(1)} \\ &\Leftrightarrow \boldsymbol{X}_{(2)} = \boldsymbol{U}_{(2)}\left(I_{T_i} \otimes B^{\top}\right). \end{aligned} \tag{15}$$

We can take without loss of generality $\text{rank}(B) = M$. Thus, $\text{rank}\left(\boldsymbol{X}_{(1)}\right) = \text{rank}\left(B\boldsymbol{U}_{(1)}\right) = \min\left(M, \text{rank}\left(\boldsymbol{U}_{(1)}\right)\right) \leq M$. On the other hand $\text{rank}\left(\boldsymbol{X}_{(2)}\right) = \text{rank}(\boldsymbol{U}_{(2)}) \leq \min(C, MT_i)$. (To see this note that $\boldsymbol{U}_{(2)}$ is size $C \times MT_i$ and $\left(I_{T_i} \otimes B^{\top}\right)$ is size $MT_i \times NT_i$ and full rank). Thus, the rank of the mode-1

unfolding is strictly bounded by M (which is fixed by the model) while the rank of the mode-2 unfolding can grow arbitrarily with C and T_i (which can be increased by the experimenter). Thus, datasets generated by the tuning model class are neuron-preferred when the inputs are unconstrained, *i.e.* when $\text{rank}(\boldsymbol{U}_{(2)})$ grows beyond M with increasing T_i. This shows part 1 of the claim.

Part 2: The dynamical model class implies condition-preferred structure. Equation (11) can be written $X(t+1) = AX(t)$, which admits the solution

$$X(t) = A^{t-1}X(1), \tag{16}$$

where the matrix A^{t-1} maps initial states to the state at time t. We define the tensor $\mathcal{A} \in \mathbb{R}^{N\times N\times T}$ to be the collection of all matrices A^{t-1} for $t = 1, \dots, T$ (from here, the definitions of $\boldsymbol{A}_{(1)}$ and $\boldsymbol{A}_{(2)}$ follow). We can now write

$$\begin{bmatrix} X(1) \\ X(2) \\ \vdots \\ X(T_i) \end{bmatrix} = \begin{bmatrix} I_N \\ A \\ \vdots \\ A^{T_i-1} \end{bmatrix} X(1). \tag{17}$$

More compactly: $\boldsymbol{X}_{(2)} = X(1)^\top \boldsymbol{A}_{(2)}$. To find $\boldsymbol{X}_{(1)}$, given equation (16) we can write

$$\begin{aligned} &[X(1) \quad X(2) \quad \cdots \quad X(T_i)] = \\ &[I_N \quad A \quad \cdots \quad A^{T_i-1}] \begin{bmatrix} X(1) & 0 & \cdots & 0 \\ 0 & X(1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & X(1) \end{bmatrix}. \end{aligned} \tag{18}$$

More compactly: $\boldsymbol{X}_{(1)} = \boldsymbol{A}_{(1)}(I_{T_i} \otimes X(1))$. Thus,

$$\begin{aligned} x(t+1,c) = Ax(t,c) \Leftrightarrow \boldsymbol{X}_{(1)} &= \boldsymbol{A}_{(1)}\left(I_{T_i} \otimes X(1)\right) \\ &\Leftrightarrow \boldsymbol{X}_{(2)} = X(1)^\top \boldsymbol{A}_{(2)}. \end{aligned} \tag{19}$$

We note that the rank of the mode-1 unfolding can grow with T_i,

$$\begin{aligned} \text{rank}\big(X(1)\big) \le \text{rank}([X(1)\ AX(1)]) \le \\ \text{rank}([X(1)\ AX(1)\ A^2X(1)]) \le \cdots, \end{aligned} \tag{20}$$

and can eventually reach the maximum of $\text{rank}(A)$ (due to the Cayley-Hamilton theorem). On the other hand, $\text{rank}\big(\boldsymbol{X}_{(2)}\big) = \text{rank}(X(1))$, where equality follows

because $X(1)^{\top}$ is a submatrix of $\boldsymbol{X}_{(2)}$. The rank of the mode-2 unfolding thus does not grow with T_i. Therefore, datasets generated by the dynamical model class are condition-preferred when $\text{rank}([X(1)\ AX(1)]) > \text{rank}(X(1))$, *i.e.* whenever the matrix A maps the initial states into a subspace not spanned by the columns of $X(1)$. This completes part 2 of the claim.

Low-rank assumptions pertaining to the above derivation

Given the above, a natural expectation is that $X(t) = BU(t) \Rightarrow \text{rank}(\boldsymbol{X}_{(1)}) \leq \text{rank}(\boldsymbol{X}_{(2)})$ with $\text{rank}(\boldsymbol{X}_{(2)})$ growing as more times are considered. Similarly one expects $X(t+1) = AX(t) \Rightarrow \text{rank}(\boldsymbol{X}_{(2)}) \leq \text{rank}(\boldsymbol{X}_{(1)})$ with $\text{rank}(\boldsymbol{X}_{(1)})$ growing as more times are considered. These expectations will indeed hold given reasonable low-rank assumptions. The first inference (that tuning models imply a neuron-mode preference) depends upon recording more neurons and conditions than the presumed number of represented variables, *i.e.*, we need $N > M$ and $C > M$. Otherwise it is possible for $\min(C, MT_i)$ (the limit on $\text{rank}(\boldsymbol{X}_{(2)})$) to be smaller than M (the limit on $\text{rank}(\boldsymbol{X}_{(1)})$). In practice, the adequacy of the data can be evaluated by testing whether results change when more neurons/conditions are added. Importantly, the present results did not depend upon neuron/condition count. For example, effects are equally strong in **Fig 4f** and **Fig 4g** despite a threefold difference in the number of analyzed neurons and conditions. Still, the possibility of data being neuron- or condition-limited is a real one, and provides strong motivation to analyze datasets with many neurons and many diverse conditions.

The second inference (dynamical models imply a condition-mode preference) depends upon the assumption $\text{rank}(X(1)) < \text{rank}(A)$. In other words, the set of initial states (one per condition) must occupy a proper subspace of all states visited as the dynamics governed by A unfold. Otherwise $\text{rank}(\boldsymbol{X}_{(1)}) = \text{rank}(\boldsymbol{X}_{(2)})$ regardless of how many times are considered (*i.e.*, the red and blue traces in **Fig 4** would be equal and would not rise with time). In practice the assumption $\text{rank}(X(1)) < \text{rank}(A)$ is reasonable, both because we never observed the above signature for any dataset and because we have

recently shown that M1/PMd preparatory states do not occupy all dimensions subsequently explored during movement [61].

In summary, the key low-rank assumptions are likely to be valid when considering many neurons and diverse conditions. Models of the form $X(t) = BU(t)$ will thus have a stable rank$\left(\boldsymbol{X}_{(1)}\right)$ and an unstable rank$\left(\boldsymbol{X}_{(2)}\right)$. Models of the form $X(t+1) = AX(t)$ will have a stable rank$\left(\boldsymbol{X}_{(2)}\right)$ and an unstable rank$\left(\boldsymbol{X}_{(1)}\right)$. The converse inferences will also hold. If rank$\left(\boldsymbol{X}_{(1)}\right)$ is stable as times are added then the data can be factored as in equation (13) and thus modeled as $X(t) = BU(t)$. If rank$\left(\boldsymbol{X}_{(2)}\right)$ is stable then the data can be factored as in equation (17) (possibly requiring a time-varying A) and thus modeled as $X(t+1) = AX(t)$.

Time-varying dynamics

Part 2 of the above claim extends naturally to the equation $X(t+1) = A(t)X(t)$, a time-varying linear dynamical system. As long as the dynamics—the (potentially time-varying) vector fields—are the same across conditions then the above arguments hold. Thus, while the appearance of condition-preferred structure depends on the constraints imposed by dynamics, such structure does not depend on time-invariant dynamics. Because dynamical systems can often be approximated as time-varying linear systems (especially over short timescales), condition-preferred structure is likely to be common whenever population structure is shaped by strong dynamics.

Measuring rank

Empirical neural data inevitably include sampling noise in the estimated firing rates, due to finite trial-counts from spiking neurons. Similarly, some degree of nonlinearity is always present in the form of spiking thresholds or deeper nonlinearities in the underling representations or dynamics. Thus, the measured $\boldsymbol{X}_{(1)}$ and $\boldsymbol{X}_{(2)}$ will always be full rank. In practice, we therefore evaluated not the ranks of $\boldsymbol{X}_{(1)}$ and $\boldsymbol{X}_{(2)}$ *per se* but the success of rank-k

reconstructions of $\boldsymbol{X}_{(1)}$ and $\boldsymbol{X}_{(2)}$. In simulations we found that this approach works well. Reconstruction error is increased by the addition of noise or nonlinearities, but this occurs approximately equally for both $\boldsymbol{X}_{(1)}$ and $\boldsymbol{X}_{(2)}$. Thus, the preferred-mode analysis is still able to successfully differentiate datasets generated by static nonlinear tuning models from autonomous nonlinear dynamical models (*e.g.*, **Fig 4**).

Acknowledgements

We thank A. Kohn, M. Carandini, and A. Benucci for providing the visual cortex data analyzed in this paper. We thank D. Sussillo, G. Hennequin, T. Vogels, W. Gerstner, and G. Elsayed for providing simulated data. We thank J.A. Movshon, N. Priebe, S. Lisberger, M. Smith, S. Chang, and L. Snyder for providing data in early development of this work. We thank L. Abbott for advice.

References

1. Churchland MM, Shenoy KV. Temporal complexity and heterogeneity of single-neuron activity in premotor and motor cortex. J Neurophysiol. 2007;97: 4235–4257. doi:10.1152/jn.00095.2007

2. Raposo D, Kaufman MT, Churchland AK. A category-free neural population supports evolving demands during decision-making. Nat Neurosci. Nature Publishing Group; 2014;17: 1784–1792. doi:doi:10.1038/nn.3865

3. Bair W, Koch C. Temporal precision of spike trains in extrastriate cortex of the behaving macaque monkey. Neural Comput. 1996.

4. Benucci A, Ringach DL, Carandini M. Coding of stimulus sequences by population responses in visual cortex. Nat Neurosci. 2009;12: 1317–1324. doi:10.1038/nn.2398

5. Grillner S. Biological pattern generation: the cellular and computational logic of networks in motion. Neuron. 2006;52: 751–766. doi:10.1016/j.neuron.2006.11.008

6. Priebe NJ, Lisberger SG. Constraints on the source of short-term motion adaptation in macaque area MT. II. tuning of neural circuit mechanisms. J Neurophysiol. 2002;88: 370–382.

7. Tong L, Liu RW, Soon VC, Huang YF. Indeterminacy and identifiability of blind identification. IEEE Trans Circuits Syst. 1991;38: 499–509. doi:10.1109/31.76486

8. Wu W, Hatsopoulos N. Evidence against a single coordinate system representation in the motor cortex. Exp Brain Res. Springer-Verlag; 2006;175: 197–210. doi:10.1007/s00221-006-0556-x

9. Hatsopoulos NG, Xu Q, Amit Y. Encoding of movement fragments in the motor cortex. J Neurosci. 2007;27: 5105–5114. doi:10.1523/JNEUROSCI.3570-06.2007

10. Fu QG, Flament D, Coltz JD, Ebner TJ. Temporal encoding of movement kinematics in the discharge of primate primary motor and premotor neurons. J Neurophysiol. 1995;73: 836–854.

11. Mussa-Ivaldi FA. Do neurons in the motor cortex encode movement direction? An alternative hypothesis. Neurosci Lett. 1988;91: 106–111.

12. Fetz EE. Are movement parameters recognizably coded in the activity of single neurons. Behavioral and Brain Sciences. 1992;15: 679–690.

13. Sanger TD. Theoretical Considerations for the Analysis of Population Coding in Motor Cortex. Neural Comput. 1994;6: 29–37. doi:10.1126/science.247.4945.978

14. Todorov E. Direct cortical control of muscle activation in voluntary arm movements: a model. Nat Neurosci. 2000;3: 391–398. doi:10.1038/73964

15. Reimer J, Hatsopoulos NG. The problem of parametric neural coding in the motor system. Adv Exp Med Biol. Boston, MA: Springer US; 2009;629: 243–259. doi:10.1007/978-0-387-77064-2_12

16. Scott SH. Inconvenient truths about neural processing in primary motor cortex. J Physiol. 2008;586: 1217–1224. doi:10.1113/jphysiol.2007.146068

17. Scott SH. Population vectors and motor cortex: neural coding or epiphenomenon? Nature Publishing Group. 2000;3: 307–308. doi:10.1038/73859

18. Graziano MSA, Aflalo TN. Mapping behavioral repertoire onto the cortex. Neuron. 2007;56: 239–251. doi:10.1016/j.neuron.2007.09.013

19. Georgopoulos AP, Carpenter AF. Coding of movements in the motor cortex. Curr Opin Neurobiol. 2015;33C: 34–39. doi:10.1016/j.conb.2015.01.012

20. Chase SM, Schwartz AB. Inference from populations: going beyond models. Prog Brain Res. 2011;192: 103–112. doi:10.1016/B978-0-444-53355-5.00007-5

21. Kalaska JF. From intention to action: motor cortex and the control of reaching movements. Adv Exp Med Biol. 2009;629: 139–178. doi:10.1007/978-0-387-77064-2_8

22. Scott SH, Kalaska JF. Reaching movements with similar hand paths but different arm orientations. I. Activity of individual cells in motor cortex. J Neurophysiol. 1997;77: 826–852.

23. Kakei S, Hoffman DS, Strick PL. Muscle and movement representations in the primary motor cortex. Science. 1999;285: 2136–2139. doi:10.1126/science.285.5436.2136

24. Caminiti R, Johnson PB, Burnod Y, Galli C. Shift of preferred directions of premotor cortical cells with arm movements performed across the workspace. Experimental brain …. 1990.

25. Ashe J, Georgopoulos AP. Movement parameters and neural activity in motor cortex and area 5. Cereb Cortex. 1994;4: 590–600.

26. Ajemian R, Bullock D, Grossberg S. Kinematic coordinates in which motor cortical cells encode movement direction. J Neurophysiol. 2000;84: 2191–2203.

27. Ajemian R, Green A, Bullock D, Sergio L, Kalaska J, Grossberg S. Assessing the Function of Motor Cortex: Single-Neuron Models of How Neural Response Is Modulated by Limb Biomechanics. Neuron. 2008;58: 414–

428. doi:10.1016/j.neuron.2008.02.033

28. Sergio LE, Hamel-Pâquet C, Kalaska JF. Motor cortex neural correlates of output kinematics and kinetics during isometric-force and arm-reaching tasks. J Neurophysiol. 2005;94: 2353–2378. doi:10.1152/jn.00989.2004

29. Churchland MM, Cunningham JP, Kaufman MT, Ryu SI, Shenoy KV. Cortical preparatory activity: representation of movement or first cog in a dynamical machine? Neuron. 2010;68: 387–400. doi:10.1016/j.neuron.2010.09.015

30. Churchland MM, Cunningham JP, Kaufman MT, Foster JD, Nuyujukian P, Ryu SI, et al. Neural population dynamics during reaching. Nature. 2012;487: 51–56. doi:10.1038/nature11129

31. Shenoy KV, Sahani M, Churchland MM. Cortical control of arm movements: a dynamical systems perspective. Annu Rev Neurosci. 2013;36: 337–359. doi:10.1146/annurev-neuro-062111-150509

32. Churchland MM, Cunningham JP. A Dynamical Basis Set for Generating Reaches. Cold Spring Harb Symp Quant Biol. 2014;79: 67–80. doi:10.1101/sqb.2014.79.024703

33. Vaidya M, Kording K, Saleh M, Takahashi K, Hatsopoulos NG. Neural coordination during reach-to-grasp. J Neurophysiol. American Physiological Society; 2015;114: 1827–1836. doi:10.1152/jn.00349.2015

34. Sussillo D, Churchland MM, Kaufman MT, Shenoy KV. A neural network that finds a naturalistic solution for the production of muscle activity. Nat Neurosci. 2015;18: 1025–1033. doi:10.1038/nn.4042

35. Maier MA, Shupe LE, Fetz EE. Dynamic Neural Network Models of the Premotoneuronal Circuitry Controlling Wrist Movements in Primates. J Comput Neurosci. Kluwer Academic Publishers; 2005;19: 125–146. doi:10.1007/s10827-005-0899-5

36. Hennequin G, Vogels TP, Gerstner W. Optimal control of transient dynamics in balanced networks supports generation of complex movements. Neuron. 2014;82: 1394–1406. doi:10.1016/j.neuron.2014.04.045

37. Lillicrap TP, Scott SH. Preference distributions of primary motor cortex neurons reflect control solutions optimized for limb biomechanics. Neuron. 2013;77: 168–179. doi:10.1016/j.neuron.2012.10.041

38. Scott SH. Optimal feedback control and the neural basis of volitional motor control. Nat Rev Neurosci. 2004;5: 532–546. doi:10.1038/nrn1427

39. Todorov E, Jordan MI. Optimal feedback control as a theory of motor coordination. Nature Publishing Group. 2002;5: 1226–1235. doi:10.1038/nn963

40. Schieber MH, Rivlis G. Partial reconstruction of muscle activity from a pruned network of diverse motor cortex neurons. J Neurophysiol. 2007;97: 70–82. doi:10.1152/jn.00544.2006

41. Pohlmeyer EA, Solla SA, Perreault EJ, Miller LE. Prediction of upper limb muscle activity from motor cortical discharge during reaching. Journal of Neural Engineering. 2007;4: 369–379. doi:10.1088/1741-2560/4/4/003

42. Hatsopoulos NG. Encoding in the motor cortex: was evarts right after all? Focus on "motor cortex neural correlates of output kinematics and kinetics during isometric-force and arm-reaching tasks". J Neurophysiol. 2005;94: 2261–2262. doi:10.1152/jn.00533.2005

43. Morrow MM, Pohlmeyer EA, Miller LE. Control of muscle synergies by cortical ensembles. Adv Exp Med Biol. Boston, MA: Springer US; 2009;629: 179–199. doi:10.1007/978-0-387-77064-2_9

44. Georgopoulos AP, Naselaris T, Merchant H, Amirikian B. Reply to Kurtzer and Herter. J Neurophysiol. American Physiological Society; 2007;97: 4391–4392. doi:10.1152/jn.00140.2007

45. Moran DW, Schwartz AB. One motor cortex, two different views. Nature Publishing Group. 2000;3: 963–author reply 963–5. doi:10.1038/79880

46. Pearce TM, Moran DW. Strategy-dependent encoding of planned arm movements in the dorsal premotor cortex. Science. 2012;337: 984–988. doi:10.1126/science.1220642

47. Cunningham JP, Yu BM. Dimensionality reduction for large-scale neural recordings. Nat Neurosci. Nature Publishing Group; 2014;17: 1500–1509. doi:doi:10.1038/nn.3776

48. Sadtler PT, Quick KM, Golub MD, Chase SM, Ryu SI, Tyler-Kabara EC, et al. Neural constraints on learning. Nature. Nature Publishing Group; 2014;512: 423–426. doi:doi:10.1038/nature13665

49. Kolda TG, Bader BW. Tensor Decompositions and Applications. SIAM Rev. 2009;51: 455. doi:10.1137/07070111X

50. Benucci A, Saleem AB, Carandini M. Adaptation maintains population homeostasis in primary visual cortex. Nat Neurosci. Nature Publishing Group; 2013;16: 724–729. doi:doi:10.1038/nn.3382

51. Moran DW, Schwartz AB. Motor cortical representation of speed and direction during reaching. J Neurophysiol. 1999;82: 2676–2692.

52. Georgopoulos AP, Ashe J. One motor cortex, two different views. Nature Publishing Group. 2000;3: 963–author reply 964–5. doi:10.1038/79882

53. Hubel DH, Wiesel TN. Receptive fields of single neurones in the cat's striate cortex. J Physiol. 1959;587: 2721–2732.

doi:10.1113/jphysiol.2009.174151

54. Rokni U, Sompolinsky H. How the brain generates movement. Neural Comput. MIT Press 55 Hayward Street, Cambridge, MA 02142-1315 email: journals-info@mit.edu; 2012;24: 289–331. doi:10.1162/NECO_a_00223

55. Tanaka H, Sejnowski TJ. Computing reaching dynamics in motor cortex with Cartesian spatial coordinates. J Neurophysiol. 2013;109: 1182–1201. doi:10.1152/jn.00279.2012

56. Pruszynski JA, Scott SH. Optimal feedback control and the long-latency stretch response. Exp Brain Res. 2012;218: 341–359. doi:10.1007/s00221-012-3041-8

57. Suminski AJ, Tkach DC, Fagg AH, Hatsopoulos NG. Incorporating feedback from multiple sensory modalities enhances brain-machine interface control. J Neurosci. Society for Neuroscience; 2010;30: 16777–16787. doi:10.1523/JNEUROSCI.3967-10.2010

58. London BM, Miller LE. Responses of somatosensory area 2 neurons to actively and passively generated limb movements. J Neurophysiol. 2013;109: 1505–1513. doi:10.1152/jn.00372.2012

59. Gold JI, Shadlen MN. The neural basis of decision making. Annu Rev Neurosci. 2007;30: 535–574. doi:10.1146/annurev.neuro.29.051605.113038

60. Mante V, Sussillo D, Shenoy KV, Newsome WT. Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature. 2013;503: 78–84. doi:10.1038/nature12742

61. Kaufman MT, Churchland MM, Ryu SI, Shenoy KV. Cortical activity in the null space: permitting preparation without movement. Nat Neurosci. 2014;17: 440–448. doi:10.1038/nn.3643

Chapter 3

Denoising Neural Signals with Tensor Decompositions

3.1 Introduction

An immediate application of tensor decompositions is denoising. For a matrix, truncated SVD can be performed to approximate the matrix with one of lower rank, discarding small singular values that often correspond to noise. When data are formatted as a tensor, even greater denoising performance can be obtained by performing low multilinear rank approximations of the data.

Here, we will show how the Tucker decomposition, or higher-order SVD, can be used to denoise neural data. Neural data is notoriously noisy, predominately due to the fact that datasets offer a very limited view of the high-dimensional state of the brain. If all relevant variables were recorded—all neurons, all synaptic potentials, all adaptation currents—then the appropriate latent factors could be identified, and there is no reason these factors themselves might be particularly 'noisy'. Yet, with limited recording technology, we can only sample an extremely small subset of the relevant variables, leading to this deceptively noisy perspective of the brain.

Thus, experimentalists are stuck with the requirement of recording a large number of repeats or trials for a given condition (for simultaneously recorded

neurons) or a given neuron-condition pair (for sequentially recorded neurons). A large trial count comes at the cost of fewer conditions in the experiment, or fewer recorded neurons.

Here, we will show how tensor decompositions can alleviate this trade-off. Denoising spike train data with relatively few trial counts can reveal 'trial-averaged' neural responses that are on par with high trial count responses. This suggests two things. First, it suggests that neural spike trains are not as noisy as they appear—the underlying signals are indeed present in the data, even when just a few trials are recorded. Second, it suggests that experimentalists can perform a much richer set of experiments, with significantly more neurons and conditions than previously thought possible.

This is a timely result: neural data is growing rapidly in its richness and complexity [11]. Recent datasets have included several thousands of neurons imaged at once, often from the entire brain of the animal [18, 2]. Yet, this increase in the number of neurons recorded does not necessarily coincide with richer data: a neuron by time matrix can be heavily rank deficient if neural responses are too simple across time, i.e. if the conditions of the experiment are not sufficiently rich to evoke complex neural responses. A goal in experimental design is not just to record more neurons, but also to record those neurons across rich, diverse stimuli or behaviors [14].

Typically, there is a trade-off between condition count and trial count. We might require n trails before a given PSTH is 'usable' or sufficiently denoised. For a given amount of time available to perform an experiment in a given day, we are thus limited in the number of different conditions that can be performed by the subject. More conditions can only be obtained at the cost of noisier PSTHs.

The key idea behind the matrix or tensor denoising techniques is as follows: We can 'borrow' trials from other neurons with similar response properties. Suppose we are recording neurons sequentially (not simultaneously).

Suppose we knew that neuron 2 always fired at twice the rate of neuron 1. Instead of recording neuron 1 for n trials and neuron 2 for n trials, we could get away with $n/2$ trials for each. To construct the trial averaged response of neuron 1, we would average its $n/2$ trials, and combine that with the average of neuron 2's trial-averaged response multiplied by 0.5. In this example, we have two neurons but one degree of freedom between them. The concept illustrated by this example extends to more neurons. As long as the degrees of freedom are less than the total number of neurons, we can find the right linear combination of responses to effectively borrow statistical strength from other neurons in the population.

Fortunately, one does not need to know the linear relationships between neurons. We can use the linear combinations obtained by SVD on the neural data matrix.

When a neural population is observed over several conditions, each of equal time length, we can format the data in a $N \times C \times T$ tensor. This gives us three choices for how to denoise using truncated SVD. We can perform truncated SVD on the mode-1, mode-2, or mode-3 unfolding. Some of these decompositions perform better than others, and it is not a priori clear which will perform best for a given dataset.

More generally, we can apply low-rank tensor decompositions to denoise across all three modes simultaneously, alleviating the choice of which mode the data is most strongly correlated.

3.2 Tensor denoising method

Here we outline the basic approach in tensor denoising. In general one can assume either a CP or Tucker model for the data. Here, we assume the latter.

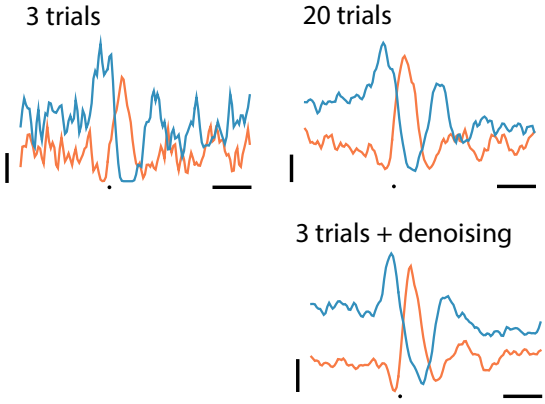

FIGURE 3.1: Top left: lightly filtered (10 ms) PSTH from one neuron across two conditions (blue, orange) averaged across 3 trials. Top right: The same PSTH when averaged across 20 trials. Bottom right: the PSTH after applying tensor denoising on the 3-trial data. Vertical bars: 10 Hz. Horizontal bars, 10ms.

Denoising thus amounts to solving the minimization problem,

$$\begin{aligned} \underset{\mathcal{S},U,V,W}{\text{minimize}} \quad & \|\mathcal{X} - \mathcal{S} \times_1 U \times_2 V \times_2 W\|_F^2 \\ \text{subject to} \quad & U^\top U = I, \\ & V^\top V = I, \\ & W^\top W = I \end{aligned} \tag{3.1}$$

where $\mathcal{X} \in \mathbb{R}^{N \times C \times T}$, $\mathcal{S} \in \mathbb{R}^{P \times Q \times R}$, $U \in \mathbb{R}^{N \times P}$, $V \in \mathbb{R}^{C \times Q}$, and $W \in \mathbb{R}^{T \times R}$. Here, we say that $\mathcal{X}$ is approximated by a rank-(P, Q, R) tensor, where $P \leq N$, $Q \leq C$, and $R \leq T$.

Like in SVD, the factor matrices are usually taken to be orthonormal. When the factors are orthonormal, $\mathcal{S}$ is uniquely determined by the factors U, V, W and does not require optimization:

$$\mathcal{S} = \mathcal{X} \times_1 U^\top \times_2 V^\top \times_3 W^\top \tag{3.2}$$

3.2.1 Higher-Order SVD

The matrices U, V, and W can be obtained by higher-order SVD. Namely, they are the left singular vectors of $X_{(1)}$, $X_{(2)}$, and $X_{(3)}$, respectively. It is well known that this solution is not optimal for the optimization problem 3.1, [22]—one of the key differences between matrices and tensors. The HOSVD solution can be used as an initial estimate for other methods. In practice, however, we found the HOSVD solution to be entirely satisfactory.

3.2.2 Alternating least squares

Nevertheless, if one wishes to obtain a better solution, there are a number of methods to do so [22]. The simplest extension to HOSVD involves performing alternating least-squares: i.e. minimize with respect to U, V, then W sequentially, and repeating until convergence. This is, in spirit, just "coordinate-descent" in the factor matrices. Due to the multilinearity of the model, $\mathcal{S} \times_1 U \times_2 V \times_2 W$, each step of alternating least-squares has a unique, global minima and itself can be obtained by SVD.

3.2.3 Cross-validation

The model has three hyper-parameters: P, Q, and R. Matrix rank minimization is known to be a difficult, discrete optimization problem [36], and the tensor case is no different [22]. Nevertheless, with the relatively small datasets involved in this study, it is not computationally prohibitive to do a (possibly coarse) grid search over the tuple (P, Q, R).

For datasets of simultaneously recorded neurons, each condition observes r_j trials, with $j = 1, \ldots, C$. For datasets of sequentially recorded neurons, each neuron-condition pair observes $r_{i,j}$ trials, with $i = 1, \ldots, N$ and $j = 1, \ldots, C$. We can use the sets of trials to perform cross-validation and select the tensor with the rank that minimizes error on left out trials.

Here, we will proceed with leave-one-out cross-validation (LOOCV). This choice enables us to consider the effect of increasing total trial count. We can start with just $r_{i,j} = 2$ for all i, j and note the ability of tensor denoising to reconstruct the underlying PSTH. We wish to compare this effect to the case when $r_{i,j}$ is larger, giving a sense of how many trials are truly needed in a given experiment. LOOCV is a procedure that can apply to all choices of trial count, even though for a particular choice of trial count, LOOCV might be less preferable to K-fold cross validation for some value of K.

First, let $r_{\text{max}} = \max(\{r_{i,j} \mid i = 1, \ldots, N, j = 1, \ldots, C\})$. Let us select a choice of r_{total}. We construct a dataset of size $N \times C \times T \times r_{\text{total}}$ by randomly resampling each entry of $\mathcal{X}_{n,c,:,r}$ from the actual set of trials recorded in the experiment. This ensures that all data can be formatted into a tensor, and ensures that each neuron-condition pair saw the same number of trials. We sample with replacement. It is natural to consider $r_{\text{total}} \geq r_{\text{max}}$, but this is not necessary: in the present analysis we consider all possible choices of r_{total}, starting with $r_{\text{total}} = 2$.

We start with a chocie of r_{total} and a choice of (P, Q, R). We then iterate $r_{\text{iter}} = 1, ..., r_{\text{total}}$. At each iteration, we consider the data tensor $\mathcal{X} \in \mathbb{R}^{N \times C \times T \times r_{\text{total}}-1}$, and the left-out trial tensor of size $\mathcal{X} \in \mathbb{R}^{N \times C \times T \times 1}$, where r_{iter} corresponds to the trial that was removed. Then, for a given iteration, we average the data tensor across trials, then perform a rank-(P, Q, R) tensor reconstruction using HOSVD. Finally, we measure the mean-squared error between the reconstructed tensor and the left out trial. Averaging the errors across all iterations of r_{iter} yields the error associated with the values r_{total}, P, Q, and R.

Thus, for a given choice of r_{total}, we can select the hyperparameter tuple (P, Q, R) that minimizes cross-validation error.

In practice, one may wish to set one choice of $r_{\text{total}} \geq r_{\text{max}}$, making use of all trials available. In our analysis, we only iterate through different values

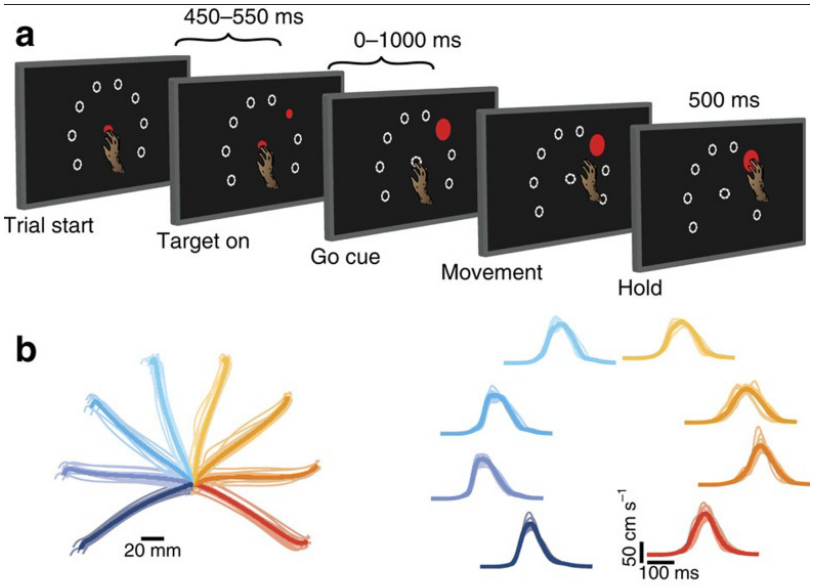

FIGURE 3.2: A: Task design. B: Hand trajectories (left) and speed profiles (right). Figure from [13].

of r_{total} to highlight our main point: a low trial count can recover underlying PSTHs with tensor denoising, while a high trial count is required if one applies standard trial-averaging.

Finally, one may also wish to iterate the above procedure K times, corresponding to K different choices of trial resampling. This adds to the computational burden, and we found it unnecessary.

3.2.4 Experimental data

We used the tensor denoising technique on datasets of sequentially recorded neurons from primary motor cortex (M1). Data were collected from primates (*Macaca mulatta*) during a variant of the delayed reach task [13]. The two datasets (monkey A and B) included $N = 80$ neurons, $C = 24$ conditions, and $T = 130$.

3.2.5 Simulated data

For comparison, we also constructed simulated data. Simulated data included a $N \times C \times T$ tensor of PSTHs that were truly low-rank: rank-$(5, 15, 15)$. Spikes were simulated via a Poisson process, using the PSTH tensor as a rate parameter for the Poisson model. We simulated up to 100 trials in this way. We could have chosen the PSTH patterns in any number of ways. For simplicity, we had them match the real neural datasets. To construct the PSTH tensor, we performed truncated HOSVD on the trial-averaged firing rates of the real data, and using a core tensor of size $5 \times 15 \times 15$. We reconstructed the full tensor from its HOSVD, yielding a low-rank version of the real data.

3.2.6 Pre-processing

We considered data spanning 500ms before movement onset until 800ms after movement onset. Spike trains were filtered with a Gaussian filter with a 10ms standard deviation. Data were then sampled every 10ms, corresponding to a total of $T = 130$ time points.

Sampling the signals before performing a tensor decomposition is desirable. Otherwise our data tensor would be of size $80 \times 24 \times 1300$, adding significantly to the computational cost of the decomposition, which can be prohibitive when this occurs within a grid search loop. Furthermore, it is known that tensor rank can be notoriously difficult to estimate when the tensor is highly lopsided [26]—i.e. when the dimensions of the modes differ by a one or more orders of magnitude.

Filtering with a 10ms Gaussian filter is noteworthy. Filtering is itself a denoising operation, and one might object that this conflicts with the claims of our method. However, at least some filtering is necessary when subsampling the time points. When the filter width is approximately the same as the sampling period, then this procedure can be viewed as a better way to

bin spike counts when compared to the standard histogram spike binning procedure. Second, filtering and tensor denoising are not in conflict at all: filtering represents our belief of "local" smoothness, and tensor densoising will perform global denoising (by finding the best subspace of $\mathbb{R}^T$ that represents the set of signals in the data). There is no reason to not include both in our denoising procedure. Indeed, one could include filter width as an extra hyperparameter in the grid search. It is now worth noting that 10ms is highly conservative: in practice a filter width of at minimum 20ms is used (e.g. [7]), while many studies employ upwards 50 ms.

Finally, we sorted neurons by their variance (across all conditions, times, and trials), and selected the first 80 neurons, resulting in a $80 \times 24 \times 130$ tensor. Subselecting neurons was not entirely necessary, but ensured that both datasets were represented by tensors of the same size, and slightly sped up computation.

3.3 Results

We compared six denoising techniques. Tensor denoising is described above. We also analyzed the performance of matrix denoising on the corresponding mode-1, mode-2, and mode-3 unfoldings of the data. In the matrix denoising case, the rank was determined by the same cross-validation procedure across a brute search of ranks.

We also considered a heuristic technique that searched instead over a range of threshold values. We determined the multilinear rank by the analyzing the mode-1, mode-2, and mode-3 matrices individually. The rank (P, Q, R) was determined by how many bases in the corresponding unfoldings are needed to reconstruct the data with a specified variance. This variance, or threshold, was varied and chosen by the same cross-validation procedure above. In practice, one may wish to use a threshold of 90 or 95 percent

variance accounted for, but this is certainly data-dependent.

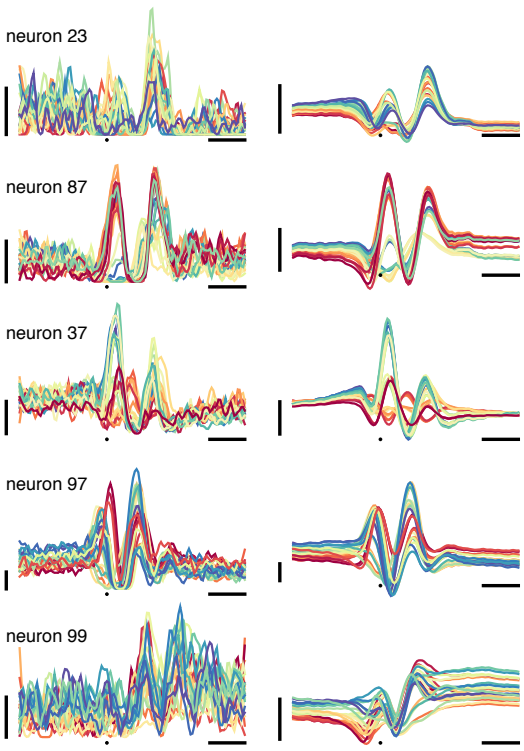

FIGURE 3.3: Example PSTHs (left) and the corresponding denoised PSTHs (right).

We compared these techniques to the baseline technique of simply averaging across trials. To assess performance of each technique, we specified a value for r_{total}, and compared the relative error of the denoised data with respect to some "ground truth" data. In the analysis of experimental datasets, the ground truth data was the trial-averaged tensor when considering *all* trials recorded in the experiment. Here, our r_{total} thus corresponds not to the total number of trials in the raw data, but to the total number of trials in the resampled data. Again, the purpose here is to show the efficacy of tensor denoising when r_{total} is small, such as 2 or 3. In the simulated datasets, the ground truth data was the underlying Poisson rate tensor. Figure 3.4 summarizes the results.

In all cases, the tensor denoising technique performed best. Notably, the heuristic technique for estimating multilinear rank performed just as well, suggesting that in practice, this heuristic can be used in place of a full grid search. All three matrix denoising techniques outperformed simple trial-averaging on the experimental datasets. In particular, denoising the mode-2 and mode-3 unfoldings of the $N \times C \times T$ tensor performed almost as well as the full tensor denoising approach, but were both generally better than denoising the mode-1 unfolding. This emphasizes that, while matrix denoising can perform almost just as well as the more computationally prohibitive

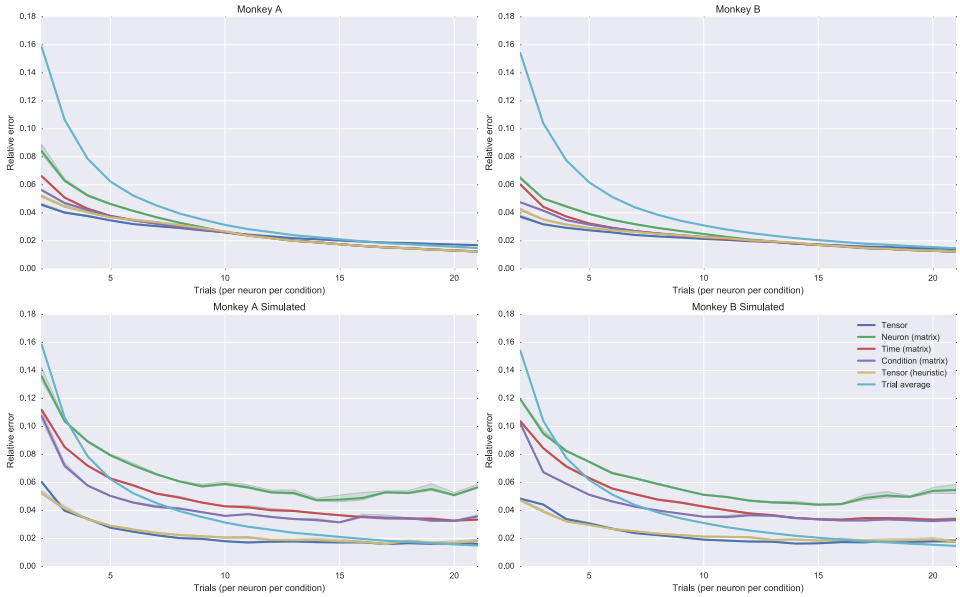

FIGURE 3.4: Error on PSTH reconstructions using a limited number of trials. Each panel corresponds to a different dataset. Top rows: Experimental datasets. Bottom rows: Simulated datasets. Simulated datasets were based on the experimental datasets, but had true multilinear ranks of $(30, 15, 10)$, with Poisson noise. Each line corresponds to one denoising technique, described in the main text. The horizontal axes correspond to the value of r_{total}, i.e. the total number of trials in the resampled data. Error bars are the SEM across different iterations (each iteration corresponds to a different random resample of the ground truth data). Error in reconstruction was computed relative to the corresponding ground truth data, as described in the text.

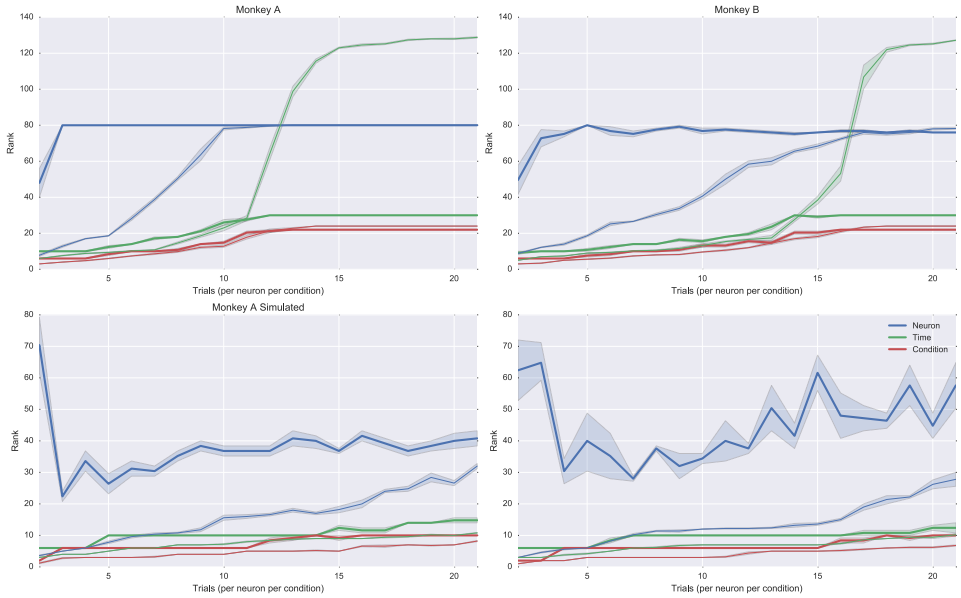

FIGURE 3.5: Average estimated ranks. Thick lines correspond to the estimation of (P, Q, R), in blue, green, and red, respectively. Thin lines correspond to the estimates of matrix ranks of the mode-1, mode-2, and mode-3 unfoldings, respectively.

tensor denoising approach, it is not a priori clear which unfolding to use. Furthermore, the computational cost of matrix denoising is no different than the tensor heuristic approach, which we showed worked better than all three matrix approaches.

The poor performance of the matrix denoising techniques on simulated date for high trial counts, relative to the performance of simple trial-averaging, is suggestive that leave one out cross-validation is overestimating the rank of the data. In practice, when estimating rank from data, one should use a cross-validation procedure that is compatible with the nature of the data, e.g. the trial count. Here, we used LOOCV for consistency—we wished to compare performance across a range of choices for r_{total}, including $r_{\text{total}} = 2, 3$, where LOOCV is appropriate.

Finally, we show the average estimated ranks of the (full) tensor denoising approach and the three matrix approaches (Figure 3.5).

3.4 Tensor denoising on spike train data

In the previous sections, we minimized mean-squared error over the tuple (P, Q, R) via a grid search, implicitly assuming a Gaussian noise model. Ideally, we wish to use a noise model more compatible with neural data, such as Poisson. Extending to the Poisson case is not a trivial matter of just minimizing the negative log-likelihood, since this requires data in $\mathbb{Z}_+$, and in the above preprocessing we were smoothing the spike trains with Gaussian kernels. Thus, to extend to a Poisson noise model we must include a local smoothness term in the cost function, meaning the HOSVD is no longer a suitable minimization algorithm. In this section, we address the Poisson case by turning to the alternating direction method of multipliers (ADMM) [5] as a method for minimizing the cost function. In this approach, we will not minimize over the tuple (P, Q, R), but instead apply a nuclear norm penalty to each of the three unfoldings. In the tensor literature, this is known as the sum of nuclear norms [35]. Since the nuclear norm is the tightest convex approximation of matrix rank, it is natural to suppose that the sum of nuclear norms (of each of the tensor unfoldings) is a good approximation of tensor multilinear rank.

Here, we outline an approach to solve the low multilinear rank denoising problem with Poisson noise. We omit results, since this method achieved similar results to the simpler method presented above. We present the method nevertheless, as it may be applicable to datasets where firing rates are much lower. We start with spike count data: $\mathcal{Y} \in \mathbb{Z}_+^{N \times T \times C \times R}$, with N neurons, T time bins, C conditions, and R trials. The rate parameters are denoted $\mathcal{X} \in \mathbb{R}^{N \times T \times C}$ with bias terms $b \in \mathbb{R}^N$. We assume that entries of $\mathcal{Y}$ are Poisson distributed with parameter $f(\mathcal{X}, b)$. We consider the element-wise firing rate function $f(\mathcal{X}_{n,t,c}, b_n) := s \log(1 + \exp(\frac{\mathcal{X}_{n,c,t} + b_n}{s}))$. The parameter s scales the smoothness of the nonlinearity.

3.4.1 Tensor rank minimization via ADMM

We want an $\mathcal{X}$ of low multilinear rank. We use the sum of nuclear norms as a convex approximation of multilinear rank. Thus, we have

$$\underset{\mathcal{X},b}{\text{minimize}} \quad -\sum_{n,t,c,r} \log p(\mathcal{Y}|f(\mathcal{X},b)) + \beta \sum_{n,t,c} \|\mathcal{X}_{t+1} - \mathcal{X}_t\|_F^2 + \lambda \sum_k \|\mathbf{X}_{(k)}\|_*, \tag{3.3}$$

where $k \in \{1,2,3\}$, corresponding to the different modes of $\mathcal{X}$. The nuclear norm $\|M\|_*$ is the sum of singular values for a matrix M. The second term ensures smooth signals.

Taking the ADMM approach, we obtain,

$$\begin{aligned} &\underset{\mathcal{X},b,\mathcal{Z}}{\text{minimize}} \quad -\sum_{n,t,c,r} \log p(\mathcal{Y}|f(\mathcal{X},b)) + \beta \sum_{n,t,c} \|\mathcal{X}_{t+1} - \mathcal{X}_t\|_F^2 + \lambda \sum_k \|\mathbf{Z}_{(k)}\|_* \\ &\text{subject to} \quad \mathcal{X} = \mathcal{Z}, \end{aligned} \tag{3.4}$$

with $\mathcal{Z} \in \mathbb{R}^{N \times T \times C}$.

The augmented Lagrangian is

$$\begin{aligned} \mathcal{L}(\mathcal{X}, b, \{\mathbf{Z}_{(k)}\}_{k=1}^3, \{\mathbf{A}_{(k)}\}_{k=1}^3) = &-\sum_{n,t,c,r} \log p(\mathcal{Y}|f(\mathcal{X},b)) + \beta \sum_{n,t,c} \|\mathcal{X}_{t+1} - \mathcal{X}_t\|_F^2 \\ &+\lambda \sum_k \|\mathbf{Z}_{(k)}\|_* + \eta \sum_k \langle \mathbf{A}_{(k)}, \mathbf{X}_{(k)} - \mathbf{Z}_{(k)} \rangle + \sum_k \frac{\eta}{2} \|\mathbf{X}_{(k)} - \mathbf{Z}_{(k)}\|_F^2 \end{aligned}$$

with Lagrange multiplier $\mathcal{A} \in \mathbb{R}^{N \times T \times C}$. The ADMM algorithm is,

$$\begin{aligned} \{\mathcal{X}^{(t+1)}, b^{(t+1)}\} &= \underset{\mathcal{X},b}{\text{argmin}}\ \mathcal{L}(\mathcal{X}^{(t)}, b^{(t)}, \mathcal{Z}^{(t)}, \mathcal{A}^{(t)}), \\ \mathbf{Z}_{(k)}^{(t+1)} &= \underset{\mathbf{Z}_{(k)}}{\text{argmin}}\ \mathcal{L}(\mathcal{X}^{(t+1)}, b^{(t+1)}, \mathbf{Z}_{(k)}, \mathcal{A}^{(t)}) \quad (k=1,2,3), \\ \mathbf{A}_{(k)}^{(t+1)} &= \mathbf{A}_{(k)}^{(t)} + (\mathbf{X}_{(k)}^{(t+1)} - \mathbf{Z}_{(k)}^{(t+1)}) \quad (k=1,2,3). \end{aligned} \tag{3.5}$$

We update each of the 3 unfoldings of $\mathcal{Z}$ and $\mathcal{A}$ sequentially.

$\mathcal{X}$ **step:** This can be solved using Newton's method. The Hessian is diagonal without the smoothness term. With the smoothness term it is tridiagonal, so is still easily inverted.

$\mathcal{Z}$ **step:**

$$\mathbf{Z}_{(k)}^{(t+1)} = \text{prox}_{1/\eta}(\mathbf{X}_{(k)}^{(t+1)} + \mathbf{A}_{(k)}^{(t)}), \tag{3.6}$$

where

$$\text{prox}_{\theta}(\mathbf{M}) = \mathbf{U}\max(\mathbf{S} - \theta, 0)\mathbf{V}^{\top} \tag{3.7}$$

where $\mathbf{M} = \mathbf{USV}^{\top}$ is the SVD of $\mathbf{M}$.

This approach is outlined in [35], except we extend it with Poisson noise and the smoothness term. These extensions are trivial since they contribute to the $\mathcal{X}$-step. As pointed out in several papers, including [35], the sum of nuclear norms is not necessarily the best convex approximation of multilinear rank. Further, note that structure on $\mathbf{X}_{(2)}$ is imposed both locally (smoothness term) and globally (low rank term). Thus, there might be a concern with respect to oversmoothing in time. However, the local smoothness term is necessary when working with a Poisson noise model for spike train data.

3.5 Applications of the ADMM approach

As stated above, when applied to spike count M1 data the ADMM approach achieved similar results to HOSVD on the filtered M1 data. Gaussian distributions are limits of Poisson distributions as the number of events increases. So, for datasets with high spike rates, the benefits of a Poisson noise model are expected to be marginal. The M1 datasets had average firing rates of 22.7 and 25.9 spikes per second for monkeys A and B, respectively, corresponding to over 0.2 spikes per bin on average. Empirically, then, this level of activity is at least an upper bound for which the ADMM approach is likely to yield

marginal benefits. Many neural datasets, however, exhibit much sparser firing rate patterns and could benefit from the ADMM approach where the Poisson noise model is particularly appropriate.

Additionally, note that in order for the HOSVD approach to be applicable, the filtered spike trains from each trial must exhibit enough overlap so that the method can exploit any low-rank structure in $\mathbb{R}^T$ (above, we chose a conservative 10 ms Guassian filter). For datasets with sparse firing patterns, the requisite filter width may be too large for comfort, and we may wish to have more explicit control over modeling temporal structure. The ADMM approach offers this. In Eq. 3.3 our cost term penalizes the squared difference between nearby time points, corresponding to the assumption that $\mathcal{X}$ is modeled by a Gaussian process. Yet more sophisticated methods can be employed, such as modeling the temporal smoothness as a linear dynamical system [25]. In such a case, the $\mathcal{X}$-step would be more sophisticated, but one could easily rely on gradient descent methods. The $\mathcal{Z}$-step would remain unchanged.

A final potential benefit of the ADMM approach is the choice of nuclear norm in the cost function 3.3, which serves as a convex relaxation of matrix rank. Algorithmically, this allows for a fast and straightforward $\mathcal{Z}$-step via the proximal operator. For the practictioner, the benefit is that the nuclear norm alleviates the need to optimize over rank, replacing a costly hyperparameter grid search with a single choice of regularization parameter λ. We found that there was essentially no difference in choosing a single λ versus three separate $\lambda_1, \lambda_2, \lambda_3$ for each of the three unfoldings. A single λ still correctly identified low-rank structure across the three unfoldings, even when the ranks of these unfoldings differed.

Chapter 4

Mapping Motor Cortex to Muscles with Dynamic Transformations

4.1 Introduction

The analysis of M1 data in Chapter 2 revealed that M1 activity contains more "structure" than the corresponding EMG that M1 ultimately drives. Although not emphasized in previous chapters it is indeed the case that the dimensionality of M1 is higher than the dimensionality of EMG (across neurons and muscles, respectively). The preferred-mode analysis of Chapter 2 simply took this observation a step further, indicating that the extra "structure" in M1 is consistent with an autonomous linear dynamical system. This result provides a simple explanation for differences in dimensionality: in order for M1 to produce muscle commands it requires more dimensions of neural activity than its corresponding outputs, due to the constraints imposed by autonomous linear dynamical systems.

Nevertheless, the preferred mode analysis never related M1 and EMG activity directly. The richness of the dataset—M1 recorded alongside EMG—allows for analyses that relates the two signals, which is the focus of this chapter. By relating the two sets of signals, we can get a further sense of just

how *anatomically* distributed the dynamics of the sensory-to-behavior transformation are. In particular, we ask: do the intervening structures between M1 and muscles themselves play a role in the dynamic transformation? Or, is the relationship between M1 and muscles approximately static, suggesting that the bulk of dynamics occur in M1?

Anatomically, both situations are defensible. Motor cortex sends direct projections to spinal motorneurons. It is thus conceivable that the transformation from neural to muscle activity could be almost purely static. However, many cortico-spinal projections synapse on spinal interneurons, and it seems likely that the local circuitry of the spinal cord could contribute significant dynamics. In the same vein, many motor cortex neurons send projections to brainstem nuclei that may have their own internal dynamics, and that in turn project to the spinal cord.

4.2 Subspace identification

In what follows, we consider the linear system

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{4.1}$$

with $u \in \mathbb{R}^m$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^p$, and the system matrices A, B, C, D with appropriate dimensions. In this case, we let u be the vector of motor cortex activations (inputs) and y be the vector of EMG responses (outputs). Since u and y are both known (recorded), the system identification problem is as follows: For known sequences $u(t)$ and $y(t)$, determine the system matrices, as well as the state sequence $x(t)$.

To test the significance of dynamics in the M1-to-EMG transformation, we compare two models. The **dynamic model** sets $D = 0$, while the **static model** sets $A, B,$ and C to 0.

There are numerous ways to fit equation 4.1, indicated by the extensive literature on the problem (known as linear system identification) [37]. Even recently, it was shown that simple gradient descent efficiently converges to a solution [16], despite the fact that the problem is nonconvex and the state space model is overparameterized. Here, we focus on a simple algebraic approach known as subspace identification.

We proceed by deriving an algebraic relation between the input and output sequence. First, it is easy to derive a form for the state sequence:

$$x(t) = A^t x(0) + \sum_{i=0}^{t-1} A^{t-i-1} B u(i). \tag{4.2}$$

We can consider a batch of data points, for $t = 0$ to $t = s - 1$:

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(s-1) \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{s-1} \end{bmatrix}}_{\mathcal{O}_s} x(0) + \underbrace{\begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & & 0 \\ \vdots & & \ddots & \ddots & \\ CA^{s-2}B & CA^{s-3}B & \cdots & CB & D \end{bmatrix}}_{\mathcal{T}_s} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(s-1) \end{bmatrix}$$

This data equation can be easily derived from (4.2) and (4.1). Since the system is time-invariant, then the above equation applies for a batch of data from $t = 1$ to $t = s$, from $t = 2$ to $t = s + 1$, and so forth. Thus we can consider multiple data batches at once using a block Hankel matrix for y,

$$Y_{0,s,N} = \begin{bmatrix} y(0) & y(1) & \cdots & y(N-1) \\ y(1) & y(2) & \cdots & y(N) \\ \vdots & \vdots & \ddots & \vdots \\ y(s-1) & y(s) & \cdots & y(N+s-2) \end{bmatrix},$$

with $U_{0,s,N}$ defined similarly. The subscript 0 indicates the top-left entry, s

is the number of block-rows, and N is the number of columns. We can now compactly write our data equation as

$$Y_{0,s,N} = \mathcal{O}_s X_{0,N} + \mathcal{T}_s U_{0,s,N} \tag{4.3}$$

where

$$X_{0,N} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix}$$

In practice, s is a user parameter that is chosen to be larger than some guess for n, which is at this point unknown. And N is chosen to be as large as possible such that $N+s-2$ is the last recorded data point. Thus the Hankel matrices $Y_{0,s,N}$ and $U_{0,s,N}$ have many more columns than rows.

The idea of subspace identification is to isolate the term $\mathcal{O}_s X_{0,N}$ in Eq. (4.3), from which $\mathcal{O}_s$ can be inferred from the column space (the "subspace"), and subsequently we can identify the matrices A and C. To do so, we must remove the influence of input in Eq. (4.3). This can be accomplished by multiplying on the right by $\Pi^{\perp}_{U_{0,s,N}}$, a projection matrix that defines the orthogonal projection onto the column space of $U_{0,s,N}$. Since $U_{0,s,N}\Pi^{\perp}_{U_{0,s,N}} = 0$, we have

$$Y_{0,s,N}\Pi^{\perp}_{U_{0,s,N}} = \mathcal{O}_s X_{0,N}\Pi^{\perp}_{U_{0,s,N}}.$$

And finally, under general conditions, it can be shown that

$$\text{range}(Y_{0,s,N}\Pi^{\perp}_{U_{0,s,N}}) = \text{range}(\mathcal{O}_s)$$

(see [37] for a proof). We thus can determine $\mathcal{O}_s$ by the column space of $Y_{0,s,N}\Pi^{\perp}_{U_{0,s,N}}$ by computing its SVD. The number of significant singular values reveal the *order* of the system, n (i.e. the dimension of the state variable). The corresponding first n left singular vectors, U_n, then form our estimate for $\mathcal{O}_s$.

From here, $C = U_n(1:p,:)$, and A is the unique solution to

$$U_n(1:(s-1)p,:)A = U_n(p+1:sp,:).$$

Once A and C are known, the matrices B and D and the initial state $x(0)$ must be found. There are numerous methods for finding these, and they are all equivalent in the noise-free case, but perhaps the most principled approach is to notice the system is linear in the parameters B, D, and $x(0)$ once A and C are known, thus can be solved via least squares. The formulation of the least squares problem is somewhat complicated and can be found in [37].

In practice, the orthogonal projection is computed by a QR decomposition, which is computationally efficient. This particular subspace method is known as MOESP, or "Multivariable Output-Error State-sPace." The MOESP technique yields unbiased estimates of system parameters with white observation noise (i.e. a noise sequence added to $y(t)$). For colored output noise, or white innovation noise (i.e. a noise sequence added to $x(t)$), the method must be modified, which has led to the PO-MOESP technique. In PO-MOESP, we construct four block Hankel matrices: $U_{0,s,N}$, $U_{s,s,N}$, $Y_{0,s,N}$, and $Y_{s,s,N}$. Here, the oblique projection of $Y_{s,s,N}$ onto the row space of $\begin{bmatrix} U_{0,s,N} \\ Y_{0,s,N} \end{bmatrix}$ along the row space of $U_{s,s,N}$ removes the influence of both the input and noise (this projection is also calculated by a QR decomposition). The theory behind this is rather involved, and proofs that show this method yields unbiased estimates of system parameters can be found in [37].

The method must also be modified to handle multiple conditions. We cannot simply concatenate different conditions across time, as this would introduce discontinuities in the signals. One option is to identify each condition separately and then average. However, subspace identification techniques perform very poorly with short time signals thus it is better to consider all

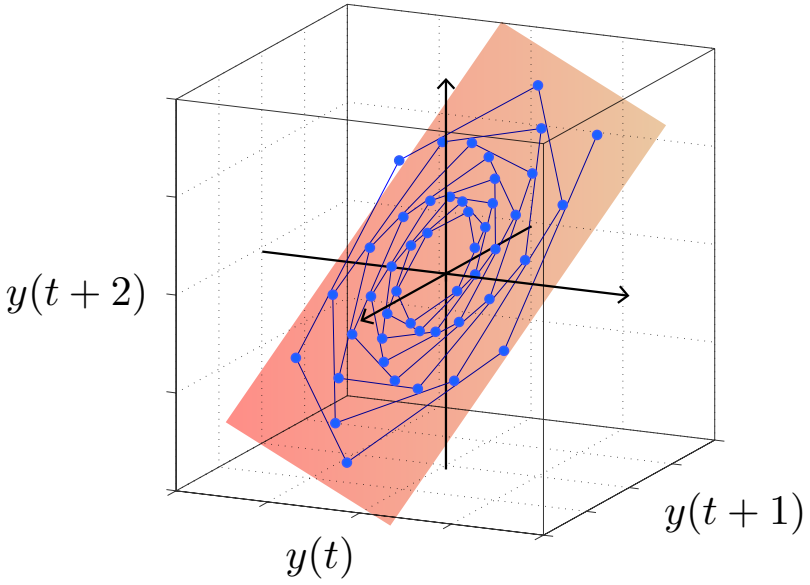

FIGURE 4.1: Schematic of subspace identification. Here, $p = 1$ (single output), $s = 3$ and $u = 0$. Each point is a vector in the block Hankel matrix $Y_{0,s,N}$. The 2-dimenaional subspace implies that $Y_{0,s,N}$ is rank 2, thus $n = 2$. The subspace itself is sufficient to determine parameters A and C. Different 2-dimensional subspaces would indicate different 2nd order systems. Figure replicated from [37]

data at once. We can simply concatenate the Hankel matrices from different conditions and proceed as usual:

$$Y_{0,s,N} = \left[Y^{(1)}_{0,s,N} \;\middle|\; Y^{(2)}_{0,s,N} \;\middle|\; \cdots \;\middle|\; Y^{(c)}_{0,s,N} \right],$$

where $Y^{(i)}_{0,s,N}$ is the Hankel matrix from the ith condition. We must also identify the initial states for each separate condition, adding to the number of free parameters of the model, and this can be done via the least squares method mentioned above.

4.3 Other system identification methods

As outlined in Chapter 1, the motivation for using a state-space representation of linear dynamical system is that it naturally captures shared-pole transformations between multidimensional inputs and multidimensional outputs.

In BMI applications this constraint is probably not necessary—an impulse response model, ARX model, or their nonlinear versions would be preferable. But in terms of scientific interpretation we wish to understand M1 as a population and its relationship with muscles as a "population." Or, in engineering parlance, we wish to model a MIMO (multi-input multi-output) dynamical system and state-space models do this naturally.

Intuitively, it is tempting to further suggest that state-space models are preferable since x gives us a proxy for actual neural activity (e.g. spinal) from "hidden" neurons. This interpretation should be cautioned. First, any change of basis of $\mathbb{R}^n$ corresponds to an equivalent dynamical system (up to input-output behavior). Thus, the components of x do not admit special interpretation. Second, impulse-response and ARX models contain (mostly) the same information as state-space models, and one can fit an impulse response and then **realize** a corresponding state-space description. Thus, there is nothing notable about the introduction of latent variables x. They are just the consequence of this particular parameterization of the dynamical system $\mathcal{D}$.

With that said, there is an interpretational advantage of state-space models that we do not exploit, but could be grounds for future work on the relationship between M1 and EMG. Ideally, we wish not just to fit EMG from M1 data, but to understand this transformation. In particular, the dynamics are determined by the poles of transfer function $H(z)$ (the eigenvalues of A) but different input-output pairs u_i, y_j will exhibit distinct zeros (the zeros of the numerator of $H(z)$), leading to distinct pole-zero cancellations or near-cancellations. Meaning, one muscle group may wish to preferentially utilize some modes over others (e.g. different frequency/phase preference), and these modes may be driven by a distinct set of inputs. A state-space formalism makes this relationship a bit more interpretable in the following way. We have a state-space model fit from data. Next, we parameterize A in

modal form, e.g.:

$$A_{\text{modal}} = \begin{bmatrix} \sigma_1 & \omega_1 & 0 & 0 & 0 \\ -\omega_1 & \sigma_1 & 0 & 0 & 0 \\ 0 & 0 & \sigma_2 & \omega_2 & 0 \\ 0 & 0 & -\omega_2 & \sigma_2 & 0 \\ 0 & 0 & 0 & 0 & \lambda_3 \end{bmatrix} \tag{4.4}$$

where the eigenvalues of A (in this example) are $\sigma_1 \pm i\omega_1, \sigma_2 \pm i\omega_2$ and $\lambda_3 \in \mathbb{R}$. Modal form is a representation of A like the diagonalization of A, but where conjugate-pairs are expanded into 2×2 blocks (This can also be defined for non-diagonalizable A in the obvious way via comparison to the Jordan normal form). The modal form is one of several parameterizations that are used in making the system more interpretable, easier to train, etc. (the companion form being the other main example).

We can put the system in modal form via a transformation of the state variable $\tilde{x} = Px$ for a specific nonsingular P. The corresponding matrices change via $A \mapsto PAP^{-1}$, $B \mapsto PB$ and $C \mapsto CP^{-1}$.

The resulting system admits direct interpretation, since the state variable $\hat{x}$ has interpretable components. The columns of B specify the coefficients for how much neuron i drives or "cares about" each of the modes of the system. Since the data u come with meaningful labels (e.g. the location of the neuron on the array, the depth of the electrode, the profile of its spikes), B might have structure that relates to the labels in some way. The rows of C specify the coefficients for how much muscle j is driven by each of the modes of the system. Since each output component is meaningfully labeled (trapezius, deltoid, etc.), the matrix C is interpretable in the context of those labels. A full understanding of the M1-to-EMG transformation would involve the understanding of how a particular subset of neurons drives a particular set of

dynamical modes, which drive a particular set of muscles.

4.4 Comparing dynamic with static

We wish to compare the efficacy of dynamic vs. static maps in the M1-to-EMG transformation. As emphasized in other chapters, one of the defining features of M1 as it relates to EMG is that it contains more structure—i.e. M1 is higher dimensional than EMG and the extra dimensions are not just noise. This makes model comparison fundamentally difficult: fitting EMG is easy, regardless of the model. This is because any model has a superset of patterns in M1 to draw upon—all of which are "EMG-like." Thus, both the static and dynamic models are already overdetermined, given the structure of inputs u.

In recent work, Kaufman et al. [19] decomposed M1 into mutually orthogonal subspaces—an output-null space that does not map to muscles and an output-potent spaces that does. More specifically, anything not in the output-null subspace is mapped to muscles, and its orthogonal complement, the output-potent subspace, defines the spaces where all variation is mapped to muscle variation. All other M1 states are a combination of vectors from the two. We could adopt their formalism and use data only from the output-potent subspace. This would be somewhat circular since the spaces themselves were defined via regression.

We therefore adopted the following approach. We preprocessed the inputs by using the top m principal components of neural activity and varied m as a measure of model complexity. Increasing m meant the system could draw from a larger set of input patterns. Since the inputs were principal components, each input was mutually orthogonal. Preprocessing inputs in this fashion is in any case standard procedure in system identification [37]. With m sufficiently large, both models should perform equally well. For smaller m, one might perform better than the other.

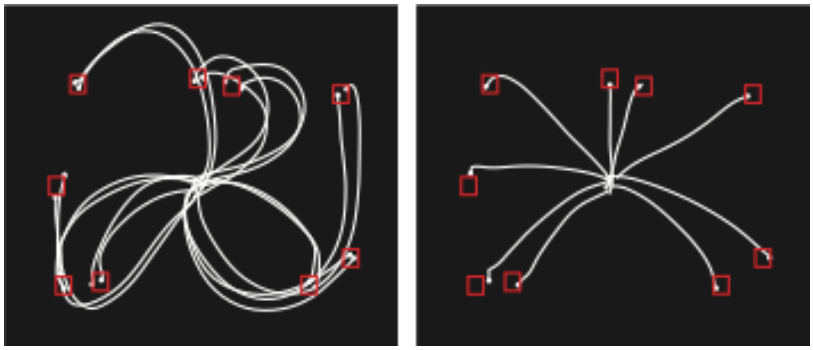

FIGURE 4.2: Train set (left) and test set (right).

Thus, for a given choice of m, does the static or dynamic model perform better? We fit the dynamic model via subspace identification outlined above and the static model via regularized regression.

Naturally, since both models are qualitatively different and contain different numbers of parameters, we compare them by assessing performance on test data. Data were trained on curved reaches. Test data included all the straight reaches (Figure X). Data was too limited to perform model selection over hyperparameters. We therefore confirmed that results held over different choices of L2 regularization constant in the static model. Further, we opted for a reasonable time shift of the inputs u to account for the transmission delay between M1 and EMG, though this value can be optimized. We did not apply a time shift to the inputs in the dynamical model.

4.5 Results

In monkey J we found that the dynamic model performed better than the static model (0.045 vs 0.07 mean-squared error on the test set). In monkey N, both models performed similarly well. The latter result may be due to the EMG recordings of monkey N, which tended to be less oscillatory and exhibited simple temporal structure. This could be a simple data limitation—a different sampling of muscle sites may have revealed richer signals—as opposed to particular behavioral patterns of that monkey.

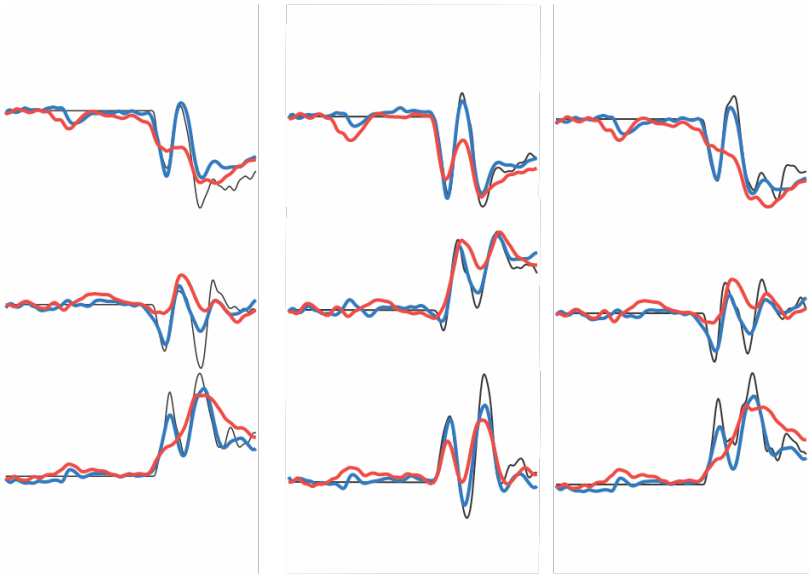

FIGURE 4.3: Example, not necessarily representative fit traces. Three trapezius muscles (columns 1, 2, 3) and three reach conditions (rows 1, 2, 3), the (processed) EMG data (black), prediction from the dynamic model (blue), and prediction from the static model (red). The static model traces sometimes matched the data reasonably well (e.g. in other muscle-condition examples not shown above), but were at times of the wrong phase (middle row), poorly accentuated (middle column), or absent of the main oscillatory frequency (four corners).

The dynamic models were of order $n = 6$, which roughly models a bottleneck between the higher-dimensional inputs and outputs. The modes had frequencies of 4 Hz, 2.3 Hz, and 0.6 Hz with corresponding time constants of 0.28, 0.11, and 0.14, respectively. The 0.6 Hz mode acts primarily as an exponential filter of the inputs, while the other two modes provided support for muscle oscillations not directly phase-aligned to the corresponding oscillations present in the inputs.

We emphasize: the frequency content of EMG signals is roughly the same as the frequency content of the neural signals. There is thus no reason *a priori* that a dynamic transformation is required. Why did the dynamic model perform better? The likely explanation is that the models had to generate outputs consistently across different conditions. A static model fit to each condition separately performed well, similar to the dynamic model, but each

corresponding B_c was different. When we insist that a single B is used for all conditions, it presumably sacrifices fit quality for some subsets of conditions. In other words, even though the frequency content of M1 data for condition c is sufficient to fit EMG data for condition c via B_c, the specified weights do not generalize across conditions. Alternatively, the dynamic model did not exhibit this problem. And this is the key result: the dynamical model did not outperform the static model because it generated frequency content not present in the inputs; it outperformed because a single set of parameters A, B, C worked across all conditions. From a model-fitting perspective, we might say the dynamic model was picking up the slack in some conditions vs others. Scientifically, we would argue that this is evidence of nontrivial dynamics in the M1-to-EMG transformation. This preliminary result was not sufficiently verified due to limited data, which we explain below.

4.6 Future work

EMG data were limited and were matched to a limited set of neural recordings. The array recordings utilized in Chapter 2 had no corresponding muscle datasets; only the smaller single-electrode datasets were recorded alongside muscles. With limited data, a clear distinction between dynamic and static transformations was difficult. Results were shown in only one monkey. Thus, the above results are not conclusive.

A simple continuation of this work might involve the following. We could fit data using the dynamic, static, and full (A, B, C, D nonzero) model, noting particular hyperparameter regimes where the full model relies more on the feedthrough term D vs relying more on the dynamic pathway via A, B, C. Further, we could opt for much more flexible models—e.g. nonlinear static and dynamic models—and rely on a common gradient descent method for fitting (as opposed to subspace identification, which only works for linear

models). Clearly, the transformation between M1 and EMG is nonlinear, but for brisk movements analyzed around a cross-condition mean a linear approximation is likely adequate and comes with the benefit of significantly more interpretable results. Further emphasis could be placed on different regularization techniques. More promising, however, would be a full interpretable model that leads to true understanding of the M1-to-EMG transformation. The (known) geometry of muscles and their relationships to each other, as specified by a particular submanifold of $\mathbb{R}^p$—the EMG configuration space—should be related to specific aspects of the dynamic model. Perhaps biomechanics impose that the deltoid and trapezius must coordinate in such a fashion that they receive the same control signal but at a specified phase offset. A dynamic model could potentially reveal such relationships and elucidate the role of spinal circuits in the motor control system.

Chapter 5

A Network Model for Motor Cortex

Here, we will continue to explore M1 activity in terms of an autonomous dynamical system, but through the use of nonlinear models. We focus on a question similar to the one posed in Chapter 2: We wish to find analyses that can identify whether signals are generated by a dynamical system or are the *output* of one (i.e. are input-driven). We will train general nonlinear dynamical systems—recurrent neural networks (RNNs)—to produce EMG signals from a novel "cycling" task (described below). We wish to find analyses that can correctly identify the RNN state trajectories as coming from an autonomous dynamical system, while classifying the EMG signals as non-dynamical. Applying such an analysis to M1 data could then validate the hypotheses from previous chapters, but now within the context of nonlinear systems.

Our approach is to focus on geometric properties of state trajectory curves that are informative of the underlying vector fields for which the trajectories are solutions. These trajectory properties include basic features from differential geometry, such as curvature and torsion, as well as "tangling," a global analog of curvature. The argument is that simple—in particular, smooth—vector fields are naturally and easily realizable by dynamical systems and necessarily produce trajectories of low curvature and low tangling. Trajectories with high curvature and high tangling could also be realized as solutions to a vector field, but that vector field would be overly complex and likely not

robust to noise. In this case a more plausible explanation is that the trajectores are not solutions to a dynamical system but are driven by external or upstream inputs. Indeed, empirically we find that M1 trajectories exhibit low tangling/curvature while EMG trajectories exhibit high tangling/curvature. Thus, we can already see that this story parallels that of Chatper 2 (compare Figure 4 c,d to Figure 6 c).

Finding a precise mathematical relationship between vector field "complexity," the corresponding geometric properties of samples of trajectories from that vector field, and M1 and EMG data, is beyond the scope of this chapter, in part due to the inherent difficulty of this problem. We opt for a slightly more empirical approach. We train a large family of RNNs parameterized by various regularization hyperparameters and instances of weight initializations. For each RNN, we calculate the aforementioned summary features of the state trajectories without strong assumptions of which will match those of M1, if any. We find that the tangling of RNN state trajectories very robustly match those of M1 but not EMG. Other properties like curvature and torsion are meaningfully related to regularization hyperparameters but not as robustly matched to M1.

RNNs are a reasonable proxy for biologically plausible parameterizations of vector fields of $\mathbb{R}^n$ since they are written as linear terms (corresponding to inputs from other neurons), followed by a bias vector input (corresponding to thresholds for each neuron), followed by an element-wise nonlinearity (corresponding to the activation nonlinearity of neurons). By constraining this family of vector fields further to those that produce EMG data as outputs, we obtain a family of RNNs that could reasonably model the putative vector fields underlying the M1 data. By sweeping a range of regularization hyperparameters, we explore a space of vector fields across a range of different geometric properties that nevertheless all produce EMG as output. It is thus nontrivial that all such RNNs exhibited low tangling, quantitatively similar

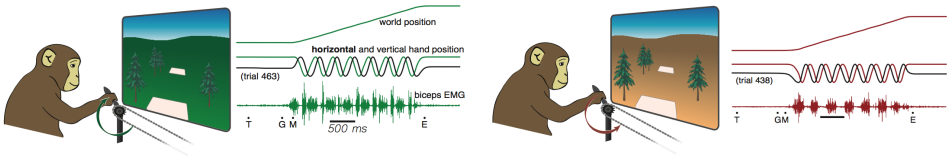

FIGURE 5.1: Cycling task. Visual cues indicate forward (left) or backward (right) pedaling.

to the M1 data. Ultimately, this supports the claim that M1 data is reasonably modeled as a dynamical system while EMG is modeled (and known to be) an output of one. That we arrived at this conclusion with qualitatively different datasets, through qualitatively different considerations—dimensionality (Chapter 2) and geometry (here)—is compelling.

5.0.1 Data

In this study, two rhesus macaques (C and D) were trained to navigate a virtual environment by grasping and cycling a hand pedal. Monkeys pedaled forward or backward to move forward in the environment. Visual cues indicated whether forward or backward pedaling corresponded to forward movement. The monkeys cycled for 7 revolutions before collecting a reward. Other sets of revolutions were part of the dataset, but we analyzed the 7-revolution data only. In addition to forward and backward pedaling, the cycling movements either started at the top or bottom of the hand pedal rotation, amounting to $C = 4$ conditions. The data consists of single neuron recordings from motor cortex and dorsal premotor cortex (109/103 neurons for monkey C/D). EMG recordings (29/35 recordings) were obtained from muscles in the arm, shoulder, and chest.

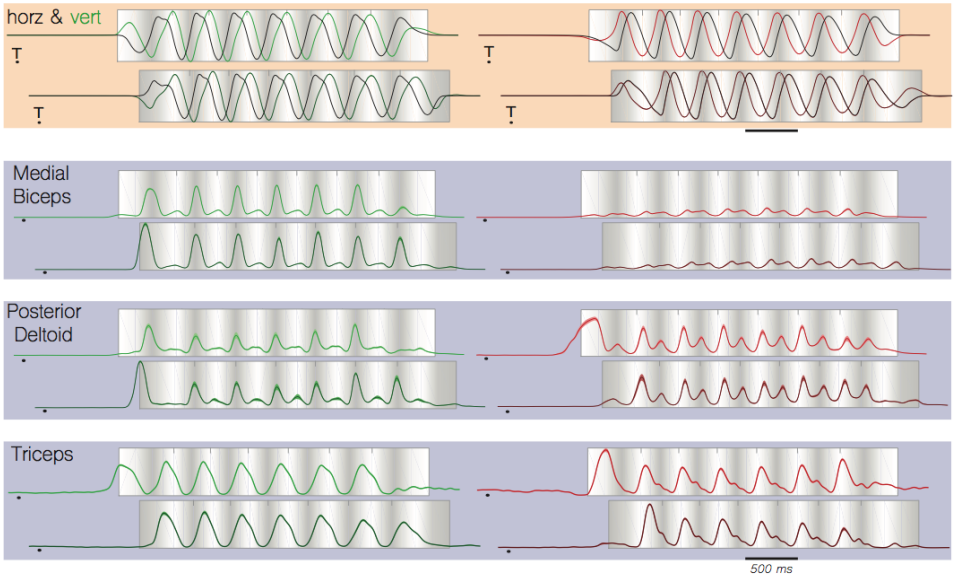

FIGURE 5.2: Example kinematic (top) and EMG (bottom) data.

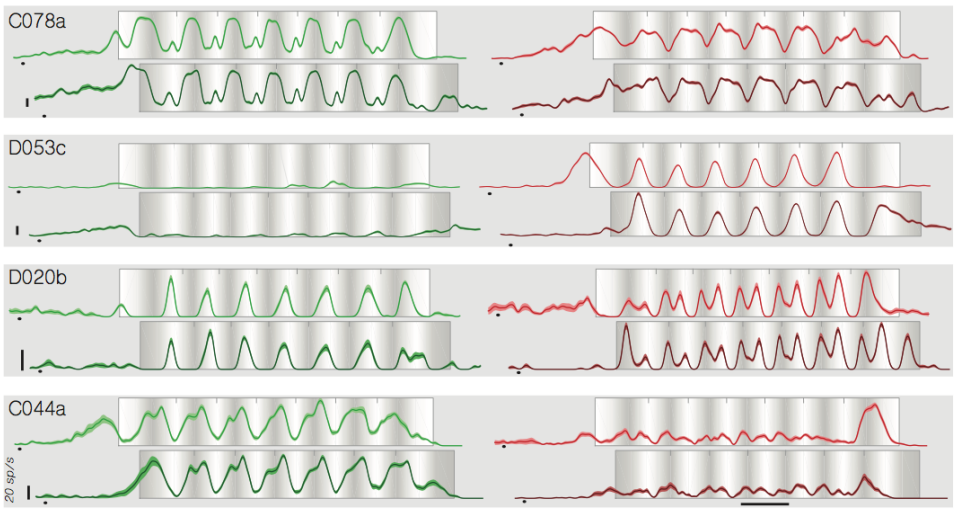

FIGURE 5.3: Example neural data.

5.0.2 Model

We use the nonlinear system,

$$x(t+1,c) = f(Ax(t,c) + Bu(t,c) + w(t,c)) \tag{5.1}$$

$$\hat{y}(t,c) = Cx(t,c) \tag{5.2}$$

where $w \sim N(0, \sigma_w)$, $f := \tanh$, and $x \in \mathbb{R}^n$. The conditions $c \in \{1,2,3,4\}$ correspond to "forward, top-start," "forward, bottom-start," "backward, top-start," and "backward, bottom-start," respectively. We set $y(t,c)$ to be the recorded muscle activity. We model the input $u \in \mathbb{R}^2$ as $u(t,c) = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ for $c = 1,2$ and $u(t,c) = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top$ for $c = 3,4$ (for all t in both cases). It is worth noting that the two different inputs correspond to two different vector fields of $\mathbb{R}^n$. Thus, the RNN trajectories taken across all conditions do not come from a single autonomous dynamical system but rather two.

We fit the parameters, A, B, C, and $x(0,c)$ by minimizing the loss function,

$$L = \sum_{t,c} \left[\frac{1}{2} \|\hat{y}(t,c) - y(t,c)\|_2^2 \right] + \frac{\lambda_A}{2} \|A\|_F^2 + \frac{\lambda_C}{2} \|C\|_F^2 + \sum_{t,c} \left[\frac{\lambda_x}{2} \|x(t,c)\|_2^2 \right] \tag{5.3}$$

We analyze the effect of regularization on the RNN. Within a broad range of different regularized RNNs, we found similar fit performance with significantly different geometric properties of the state variable trajectories.

Penalizing the magnitude of the entries of A simplifies the linearized dynamics of the system by allowing only a few modes that do not strongly decay. Penalizing the magnitude of the entries of C ensures that $y(t,c)$ is not dependent on weakly active neurons. Penalizing the neural activity $x(t,c,)$ ensures that neurons do not saturate the $\tanh$ nonlinearity. Despite these descriptions, the full effects of these regularization terms are not entirely understood.

The noise variance σ_w is also considered and varied as a hyperparameter alongside each λ_i. Noise injection is a standard regularization technique in the RNN literature.

There are a number of other standard and effective regularization techniques in the RNN literature that we omit since their scientific interpretation is not clear. These include dropout (on the output weights C) [34] and layer normalization [4].

The data $y(t, c)$ contains drift across the 7-cycle duration. We did not wish to overfit to this drift. Knowing that the brain is capable of cycling indefinitely (given an infinite supply of juice rewards), we wished to only study RNNs capable of producing not only the data $y(t, c)$, but also muscle activity of arbitrary cycling duration. We thus created 8 new conditions derived from the data. We created two canonical cycles. The first was the average of the middle 5 cycles. The second was the average of 4 cycles, half a phase offset from the middle 5. These canonical cycles were then concatenated for a total of 10 cycles each. A Butterworth filter was then applied to smooth the discontinuities from concatenation. The augmented data contained a total of 12 conditions (4 real, and 8 from the two canonical cycles).

5.0.3 Geometric analyses

Curves are often used as an elementary case study for differential geometry before one dives into the richer world of surfaces and its generalizations. The differential geometry of curves are easy to handle and well-understood. They key feature is **curvature**, which measures how much a curve deviates from being a straight line. Curvature is often defined as the inverse of the radius of the best-fit circle at a point on a curve. Generalized curvatures are also considered: the **torsion** of a curve measures how much a curve deviates locally from a plane. A curve in $\mathbb{R}^n$ with constant curvature and zero torsion

is a circle confined to a 2-dimensional subspace. Nonzero torsion indicates the curve must explore beyond a 2-dimensional subspace of $\mathbb{R}^n$. We use these geometric features to analyze RNN, M1, and EMG data.

Curvature

A curve is a vector-valued function:

$$\gamma : I \to \mathbb{R}^n \tag{5.4}$$

where I is a nonempty interval of $\mathbb{R}$, e.g. $I = [0, T]$.

The Frenet frame is a set of orthonormal vectors, denoted $e_1(t), \ldots, e_n(t)$. The first vector e_1 is tangent to the curve. The first and second define the plane of curvature: the plane in which the best-fit circle—the **osculating circle**—resides. The third vector defines the direction of torsion, etc.

To construct the Frenet frame, we apply Gram-Schmidt orthogonalization to the derivatives of $\gamma(t)$: $\gamma'(t), \gamma''(t), \ldots, \gamma^{(n)}(t)$.

Explicitly, we can write this as:

$$e_1(t) = \frac{\gamma'(t)}{\|\gamma'(t)\|} \tag{5.5}$$

$$e_i(t) = \frac{\overline{e_i}(t)}{\|\overline{e_i}(t)\|} \tag{5.6}$$

where

$$\overline{e_i}(t) = \gamma^{(i)}(t) - \sum_{j=1}^{i-1} \left(\gamma^{(i)}(t)^\top e_j(t)\right) e_j(t) \tag{5.7}$$

From here, we can calculate the generalized curvatures of γ, denoted $\chi_i(t)$:

$$\chi_i(t) = \frac{e_i'(t)^\top e_{i+1}(t)}{\|\gamma'(t)\|} \tag{5.8}$$

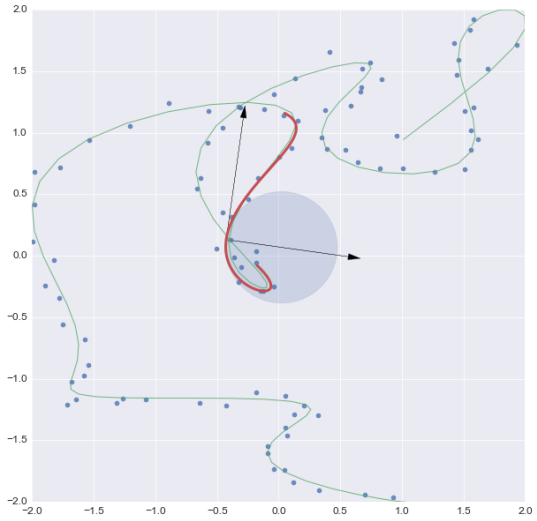

FIGURE 5.4: To determine the curvature of a curve (green) in $\mathbb{R}^n$ from noisy samples (blue dots) at a point, we select adjacent points and fit a polynomial approximation (red). From here, the Frenet frame (black) and osculating circle (blue) can be calculated explicitly.

Where $\chi_1(t)$ is the curvature of γ and $\chi_2(t)$ is the torsion of γ.

How do we calculate e_i and χ_i from discrete, possibly noisy samples of γ? Numerically, we can calculate the Frenet frame along with the generalized curvatures by locally approximating the curve $\gamma(t)$ by an order d polynomial at time t. Using $N > d$ data points nearby and including the data point at time t. From here, the values of $\gamma(t)$: $\gamma'(t), \gamma''(t), \ldots, \gamma^{(n)}(t)$ can be calculated symbolically using the derivatives of the polynomial (Figure 5.4).

To calculate $e_i'(t)$, we can make use of the identities:

$$\frac{d}{dt}\left(\frac{f(t)}{\|f(t)\|}\right) = \frac{f'(t)}{\sqrt{f(t)^\top f(t)}} + \frac{\left(f'(t)^\top f(t)\right) f(t)}{\sqrt{\left(f(t)^\top f(t)\right)^3}} \tag{5.9}$$

$$\frac{d}{dt}\left(\left(f(t)^\top g(t)\right) g(t)\right) = \left(f'(t)^\top g(t) + f(t)^\top g'(t)\right) g(t) + \left(f(t)^\top g(t)\right) g'(t) \tag{5.10}$$

And simply use our polynomial estimates of $\gamma'(t), \gamma''(t), \ldots$. That is, in

the above calculations of $e_i(t)$ and $\chi_i(t)$, we simply need our polynomial estimates of $\gamma(t)$ and its derivatives, leading to a simple numerical implementation of generalized curvature calculation.

Clearly, we can only estimate $e_i(t)$ and $\chi_i(t)$ up to the order of the polynomial d. There is an apparent trade-off between our need for estimating higher-order curvatures and the quality of those fits: a d too large would overfit noisy data and vastly overestimate curvature.

Note that the above procedure reproduces the more familiar formula for curvature in n dimensions:

$$\chi_1(t) = \kappa(t) = \sqrt{\frac{(\gamma'^\top \gamma')(\gamma''^\top \gamma'') - (\gamma'^\top \gamma'')^2}{(\gamma'^\top \gamma')^3}} \tag{5.11}$$

Tangling

We observe, soon, that EMG trajectories exhibit higher curvature than M1 trajectories. A plausible explanation is as follows. M1 must generate patterns that drive muscles, thus M1 activity is determined by two constraints. The first is that the patterns must resemble muscle activity such that they can be transformed to EMG via a downstream function. The second is that the patterns must be those that can be generated by neural circuit dynamics. A natural assumption is that the underlying vector field must be relatively smooth and thus produce trajectories of relatively low curvature. Testing this idea directly is impossible with current data given that we do not have access to M1's vector field, only to four trajectories, potentially all generated from a single vector field (or two, corresponding to the forward and backward experimental conditions). We thus need other tools to get the most out of the available data. This is the first motivation for the following analysis.

The second motivation is to note that curvature is local, and that this is necessary but not sufficient for trajectories arising from a smooth dynamical system. Curvature captures smoothness of an underlying vector field for a

given trajectory (along the direction of the trajectory), but only at a given point. A figure '8' is locally smooth with low curvature everywhere locally, but globally inconsistent with an underlying, 2-dimensional vector field due to the intersection. We thus generalize curvature to not just compare nearby time points but to compare a given time point to *all* other time points in the set of trajectories. This naturally leads to the definition of **tangling**, denoted as Q,

$$q(t,i) = \frac{\|\dot{x}(t) - \dot{x}(i)\|_2^2}{\|x(t) - x(i)\|_2^2 + \alpha \operatorname{var}(x(t))} \tag{5.12}$$

$$Q(t) = \max_i (q(t,i)) \tag{5.13}$$

where $\dot{x}(t)$ is the is finite difference, $\dot{x}(t) = (x(t+1) - x(t))/\Delta t$, and the trajectory $x(t) \in \mathbb{R}^n$ is taken over all time points of a given condition c, or as the concatenation of all conditions. The constant α scales the degree to which we weight nearby points versus distant points, and also ensures the denominator does not go to 0. We set $\alpha = 0.1$. Intuitively, tangling at time t is large if there exists another time point with similar state and dissimilar derivative (or difference).

It is clear that tangling, not curvature, is the right metric for testing the hypothesis above—whether a trajectory comes from a relatively smooth vector field. Both metrics reveal more information than either alone. The comparison of the two metrics reveals the degree to which tangling depends on nonlocal points as opposed to nearby points.

Robustness

We calculated two measures of RNN robustness. We calculated **structural robustness** by perturbing A by a random matrix: $A_p \leftarrow A + \Sigma$, where $\Sigma_{i,j} \sim N(0, \sigma_s)$ for all i, j. We simulated the RNN using A_p in place of A. For each σ_s, we ran 5 trials with different random draws of Σ and took the mean R^2.

We increased σ_s until the error between y and $\hat{y}$ dropped below $R^2 = 0.5$. Here, we set $\sigma_w = 0$ in Eq. 5.1 even if a positive σ_w was used during training.

Noise robustness was calculated similarly, except we increased σ_w in Eq. 5.1, while keeping the original learned A.

Simulations

We trained multiple RNNs to the EMG data of both monkeys (C and D). For each RNN, values of λ_A, λ_C, λ_x, and σ_w were drawn randomly from log uniform distributions, $\lambda_A \in [10^{-4}, 10^{-1}]$, $\lambda_C \in [10^{-6}, 10^{1}]$, $\lambda_x \in [10^{-4}, 10^{1}]$, and $\sigma_w \in [10^{-4}, 10^{-1}]$. Each RNN included $n = 100$ neurons. Additionally, each matrix of the RNN was initialized to a random orthonormal matrix. RNNs were trained using Tensorflow's Adam optimizer [1, 20]. We discarded RNNs that failed training ($R^2 < 0.5$). For each RNN, we then calculated its tangling, mean curvature, mean torsion, Euclidean path length, structural robustness, and noise robustness.

5.0.4 Results

Figure 5.5 shows the principal components of the activations of one example RNN (left) and the M1 data. Qualitatively, both the RNN and M1 exhibit similar behavior when compared with EMG (Figure 5.6)

Hyperparameter results are displayed in Figure 5.7. Notably, the tangling of each RNN (top row) depended weakly on λ_x and λ_A, but clustered near the tangling of M1, well below the tangling of EMG. Since the tangling metric depends on both time and condition, we summarized tangling with a single value by took the empirical cumulative distribution and evaluated the number of points larger than 0.9. The robustness of the networks most strongly depended on λ_x and σ_w. Naturally, λ_x produced shorter trajectories in terms of Euclidean path length. It is plausible that the corresponding decrease in

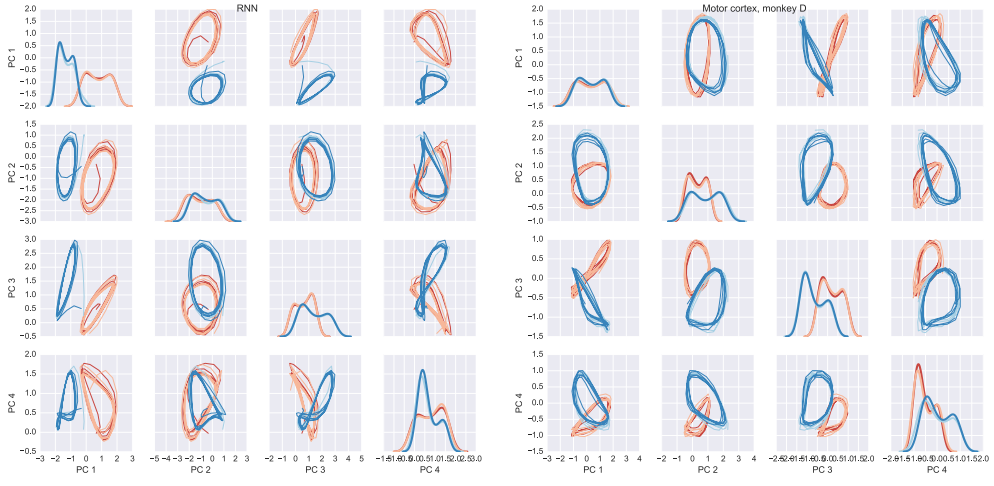

FIGURE 5.5: RNN (left) and M1 data (right). Top 4 principal components plotted against each other (off diagonals) and histograms of the data projected onto each of the four principal components (diagonals).

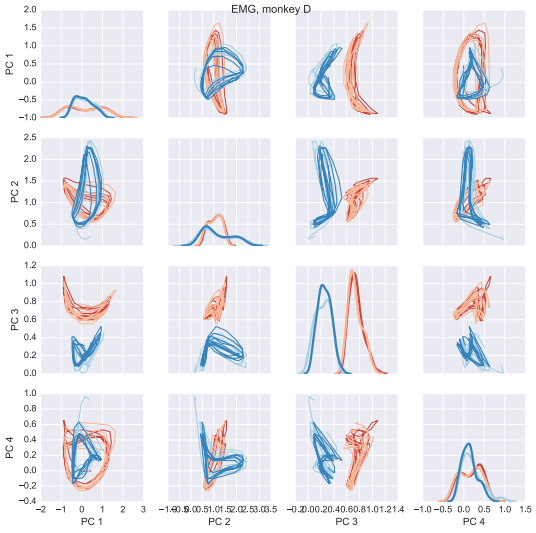

FIGURE 5.6: Top 4 principal components of EMG data.

robustness is due to the smaller scale at large λ_x. I.e. the cutoff values for σ_w and σ_s could be lower at large λ_x not due to inherent lack of robustness, but due to significantly different scaling of those values. Alternatively, it is plausible that large λ_x imposes linear solutions, thus are incapable of producing self-correcting limit cycles. Curvature appeared most strongly related to λ_A, which is sensible since large λ_A encourages simpler solutions. Yet a large λ_A did not seem to be strongly related to tangling. Finally, we note that in Figure 5.8, robustness did not seem to be strongly related to tangling, despite one of our initial hypotheses.

5.1 Discussion

Our primary result is that RNNs naturally recapitulate features of M1 data, most notably tangling, when trained solely on EMG data. This result is observed in the top row of Figure 5.7.

We initially predicted that tangling and robustness should be strongly and negatively correlated. Intuitively, smoother vector fields are expected to be less tangled, but this is not trivial to make precise. We suspect that this lack of correlation may be in part due to the different scales of the RNN activations. For example, large λ_x corresponds to lower firing rates of the neurons which can be compensated by larger weights in C to fit EMG, but require a different notion of scaling in σ_w for determining noise threshold in the robustness calculation. Further work may investigate the robustness relative to the scaling of activation determined by, say, the variance of firing rate across the population or even the path length of trajectories. Additionally, both curvature and tangling still measure features of the trajectories themselves and not the vector fields that generated them. Thus, a highly tangled trajectory can still be robust if at each point the Jacobian of the system at

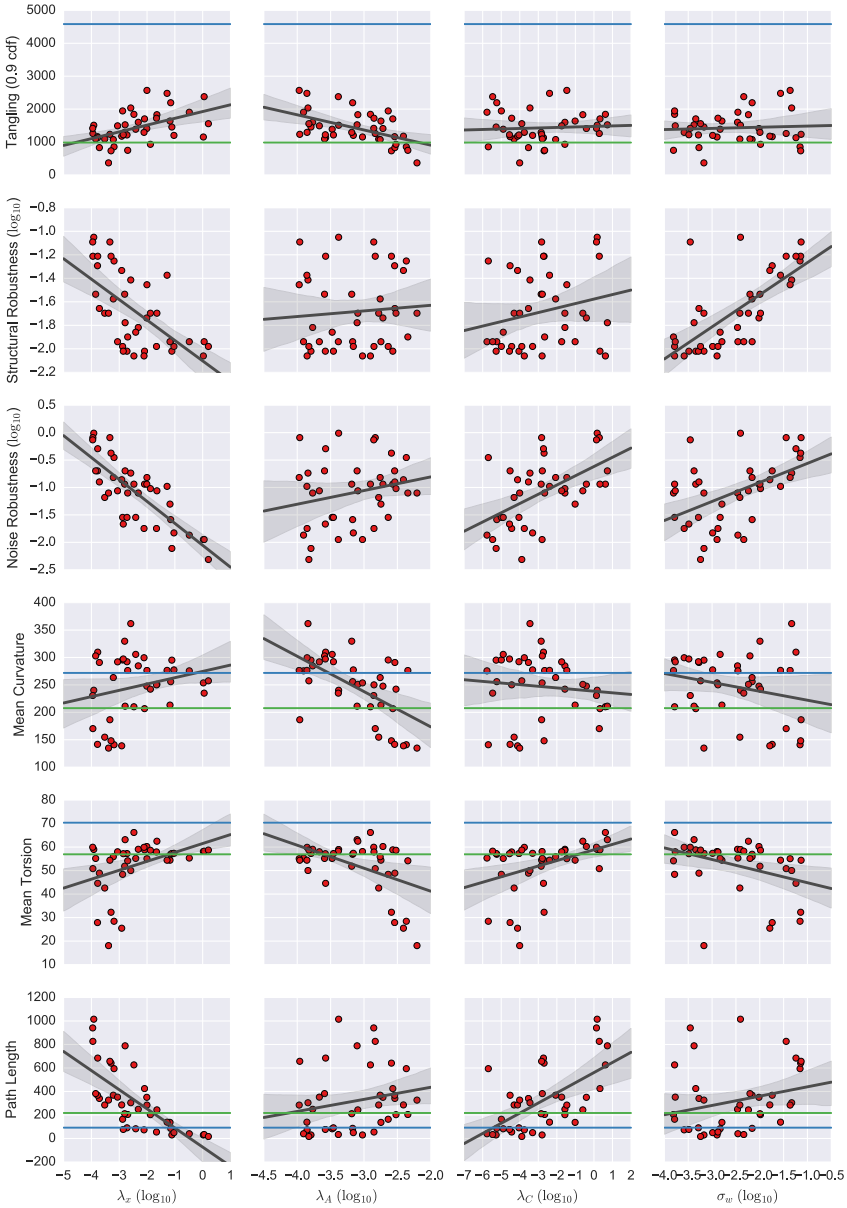

FIGURE 5.7: Influence of hyperparameters on various features of the RNN state trajectories. Each dot (red) corresponds to a particular choice of hyperparameters (columns). The green horizontal line corresponds to the corresponding value of the M1 dataset. The blue line corresponds to the corresponding value of the EMG dataset. Black lines are regression fits. Data is for monkey C.

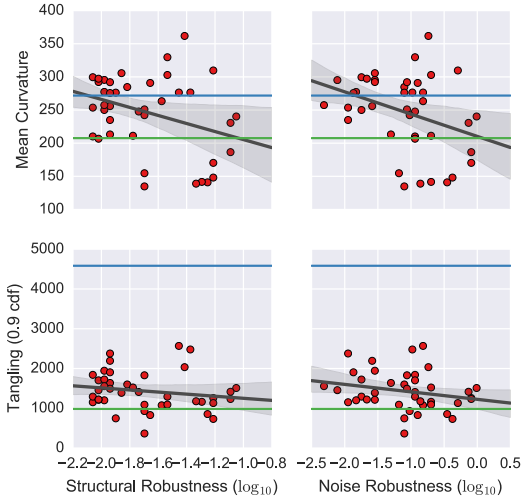

FIGURE 5.8: Lines and dots as in Figure 5.7. RNN robustness is not strongly correlated with tangling, but most RNNs captured the low tangling of M1. Robustness is negatively correlated with mean curvature, but mean curvature did not cluster near the empirical mean curvatures of M1 or EMG.

each point is strongly contracting, corresponding to a highly robust limit cycle. Additionally, our intuition for tangling in 3-dimensional systems would suggest a negative relationship between tangling and robustness, but this intuition need not apply in $\mathbb{R}^{100}$. Further work could nevertheless explore this relationship more carefully, controlling for features such as path length and dimensionality.

Nevertheless, the data presented itself with an interesting mathematical problem: investigate the degree to which a set of trajectories is likely generated by an underlying dynamical system, or is simply the output of one—i.e. is input-driven and thus not constrained to obey a flow field. The emprical observation that M1 and EMG differed in basic geometric features suggests a route to investigate this problem. One could almost always describe an externally-driven system with an underlying vector field (given that there are no actual intersections in the trajectories)—but such a vector field would be 1) overly complex and 2) not likely generalizable across conditions. In Chapter 2, we focused on the dimensionality considerations of the 2nd constraint—not sufficiently generalizable across conditions implies the

neuron-preferred structure of such systems. In this chapter, we are presented with data with an insufficient number of conditions (4) to investigate this structure, yet the geometric features of the data trajectories suggest a route to tackle the problem by focusing on the first constraint: underlying vector fields for autonomous dynamical systems should be relatively simple, or smooth, compared to those that one might try to fit to externally driven systems. Curvature is one such measure of smoothness, though we argued that tangling is a more appropriate measure for the problem at hand. The empirically lower tangling of M1 compared to EMG is suggestive of the result. The fact that various RNNs—which as ground truth are dynamical systems—recapitulated this result, while also matching, quantitatively, the tangling of M1, strengthens it.

Bibliography

[1] Martín Abadi et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[2] Misha B Ahrens and Florian Engert. "Large-scale imaging in small brains". In: *Current opinion in neurobiology* 32 (2015), pp. 78–86.

[3] Animashree Anandkumar et al. "Tensor decompositions for learning latent variable models." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 2773–2832.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[5] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine Learning* 3.1 (2011), pp. 1–122.

[6] Wieland Brendel, Ranulfo Romo, and Christian K Machens. "Demixed principal component analysis". In: *Advances in Neural Information Processing Systems*. 2011, pp. 2654–2662.

[7] Mark M Churchland et al. "Neural population dynamics during reaching". In: *Nature* 487.7405 (2012), pp. 51–56.

[8] Andrzej Cichocki et al. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[9] Andrzej Cichocki et al. "Tensor decompositions for signal processing applications: From two-way to multiway component analysis". In: *IEEE Signal Processing Magazine* 32.2 (2015), pp. 145–163.

[10] Pierre Comon. "Tensors: a brief introduction". In: *IEEE Signal Processing Magazine* 31.3 (2014), pp. 44–53.

[11] John P Cunningham and M Yu Byron. "Dimensionality reduction for large-scale neural recordings". In: *Nature neuroscience* 17.11 (2014), pp. 1500–1509.

[12] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "An introduction to independent component analysis". In: *Journal of chemometrics* 14.3 (2000), pp. 123–149.

[13] Gamaleldin F Elsayed et al. "Reorganization between preparatory and movement population responses in motor cortex". In: *Nature Communications* 7 (2016).

[14] Peiran Gao and Surya Ganguli. "On simplicity and complexity in the brave new world of large-scale neuroscience". In: *Current opinion in neurobiology* 32 (2015), pp. 148–155.

[15] Apostolos P Georgopoulos, Andrew B Schwartz, Ronald E Kettner, et al. "Neuronal population coding of movement direction". In: *Science* 233.4771 (1986), pp. 1416–1419.

[16] Moritz Hardt, Tengyu Ma, and Benjamin Recht. "Gradient Descent Learns Linear Dynamical Systems". In: *arXiv preprint arXiv:1609.05191* (2016).

[17] David H Hubel and Torsten N Wiesel. "Receptive fields of single neurones in the cat's striate cortex". In: *The Journal of physiology* 148.3 (1959), pp. 574–591.

[18] Saul Kato et al. "Global brain dynamics embed the motor command sequence of Caenorhabditis elegans". In: *Cell* 163.3 (2015), pp. 656–669.

[19] Matthew T Kaufman et al. "Cortical activity in the null space: permitting preparation without movement". In: *Nature neuroscience* 17.3 (2014), pp. 440–448.

[20] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[21] Dmitry Kobak et al. "Demixed principal component analysis of neural population data". In: *Elife* 5 (2016), e10989.

[22] Tamara G Kolda and Brett W Bader. "Tensor decompositions and applications". In: *SIAM review* 51.3 (2009), pp. 455–500.

[23] Lek-Heng Lim. "Tensors and hypermatrices". In: *Handbook of Linear Algebra, 2nd Ed., CRC Press, Boca Raton, FL* (2013), pp. 231–260.

[24] Christian K Machens, Ranulfo Romo, and Carlos D Brody. "Functional, but not anatomical, separation of "what" and "when" in prefrontal cortex". In: *Journal of Neuroscience* 30.1 (2010), pp. 350–360.

[25] Jakob H Macke et al. "Empirical models of spiking in neural populations". In: *Advances in neural information processing systems*. 2011, pp. 1350–1358.

[26] Cun Mu et al. "Square deal: Lower bounds and improved convex relaxations for tensor recovery". In: *Journal of Machine Learning Research* 1.1-48 (2014), p. 2.

[27] Meinard Müller. "Dynamic time warping". In: *Information retrieval for music and motion* (2007), pp. 69–84.

[28] Ivan V Oseledets. "Tensor-train decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317.

[29] Liqun Qi. "Hankel tensors: Associated Hankel matrices and Vandermonde decomposition". In: *arXiv preprint arXiv:1310.5470* (2013).

[30] Rajesh PN Rao and Dana H Ballard. "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects". In: *Nature neuroscience* 2.1 (1999), pp. 79–87.

[31] Robert Shapley, Michael Hawken, and Dajun Xing. "The dynamics of visual responses in the primary visual cortex". In: *Progress in brain research* 165 (2007), pp. 21–32.

[32] Matthew A Smith, Wyeth Bair, and J Anthony Movshon. "Dynamics of suppression in macaque primary visual cortex". In: *Journal of Neuroscience* 26.18 (2006), pp. 4826–4834.

[33] Laurent Sorber, Marc Van Barel, and Lieven De Lathauwer. "Structured data fusion". In: *IEEE Journal of Selected Topics in Signal Processing* 9.4 (2015), pp. 586–600.

[34] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[35] Ryota Tomioka and Taiji Suzuki. "Convex tensor decomposition via structured Schatten norm regularization". In: *Advances in neural information processing systems*. 2013, pp. 1331–1339.

[36] Madeleine Udell et al. "Generalized low rank models". In: *Foundations and Trends® in Machine Learning* 9.1 (2016), pp. 1–118.

[37] Michel Verhaegen and Vincent Verdult. *Filtering and system identification: a least squares approach*. Cambridge university press, 2007.

[38] Brian A Wandell, Serge O Dumoulin, and Alyssa A Brewer. "Visual field maps in human cortex". In: *Neuron* 56.2 (2007), pp. 366–383.